

Dottorato di Ricerca in Informatica
Università di Bologna, Padova, Venezia

Models and Types for Wide Area Computing

The calculus of Boxed Ambients

Silvia Crafa

15/01/2003

Coordinatore:
Prof. Özalp Babaoğlu

Tutore:
Prof. Gilberto Filè

Supervisori:
Prof. Michele Bugliesi
Dr. Giuseppe Castagna

*“Il vero discernimento va davvero in linea retta
o si giova di serpeggiamenti scaltri perfino dolorosi?
Eppure il lampo della buona intuizione
è come un salto da giocoliere,
non ondeggia nell’aria, scatta e arriva.
Così vorrei arrivare a pensare e a giudicare.”*

M.Bellonci, *Rinascimento privato*

Abstract

In recent years highly distributed networks have become a common platform for large-scale distributed programming. New programming paradigms are emerging to support mobility of code and computations, as well as to provide for an effective infrastructure for coordination and control of software modules dynamically loaded from the network.

In this thesis we study foundational models and programming languages designed to cope with critical issues of wide area computing, such as concurrency, reconfigurability, flexibility, heterogeneity and security. We develop an effective framework for building and analyzing wide area computation. In particular, by relying on strong typing as the basic principle, we develop type theoretic methods and tools that enable formal analyses of security guarantees appropriate for systems and applications on the global computing platform.

We present the calculus of *Boxed Ambients*, a process calculus derived from Cardelli and Gordon's Mobile Ambients, using it as a framework to study three basic aspects of security: access control, the analysis of information flows and the protection of agents from unreliable environments.

Acknowledgements

At the beginning of this dissertation I would like to thank all the people that friendly supported me during these four years of PhD studies.

First of all, I would like to thank my tutor Prof. Gilberto Filè that suggested me the idea of PhD, leading me to research.

I am then really indebted to my supervisors Michele Bugliesi and Giuseppe Castagna for having kindly dedicated so much time to supervise my studies, making my PhD so successful. Thanks to Michele for having shown me the job, teaching me to seriously devote to research, always pointing to my best. Thanks to Beppe for having received me in his laboratory in Paris, offering me the opportunity of learning and improving so much. And, above all, thanks a lot to both my supervisors since they gave me the fantastic chance of working side by side with them in a stimulating, exciting and pleasant atmosphere.

I would also like to thank my colleagues in Venice: Ombretta, Chiara, Damiano, Massimiliano, Moreno (special thanks for the technical support) and Valentino. Thanks for your affection and pleasantness; “I hope the ghost of unemployment will never haunt our room 13!”.

Thanks a lot to all my friends in Maison de l’Italie in Paris, that showed me how much Happy it could be any ordinary day. Thanks to my family, Apo, Mamy and Ste, for the constant and total support. Finally, thank you to Paolo for all that has not been work, may be not much but great.

Contents

Abstract	iv
Acknowledgements	v
1 Introduction	1
1.1 Thesis overview	5
1.2 Related Work	7
1.3 Remark	9
2 The calculus of Boxed Ambients	11
2.1 Introduction	11
2.1.1 Motivations for being <i>boxed</i>	13
2.1.2 Boxed Ambients: overview and main results	14
2.2 Boxed Ambients	16
2.3 Communication Channels	20
2.3.1 π -calculus channels	20
2.3.2 Parent-child channeled communication à la Seal Calculus	21
2.4 Typing Boxed Ambients	22
2.4.1 Types	23
2.4.2 Typing Rules	24
2.5 Mobile Ambients versus Boxed Ambients	27
2.5.1 From boxed to mobile ambients	28
2.5.2 From mobile to boxed ambients	29

2.6	Moded Typing	30
2.6.1	Moded Types	31
2.6.2	Subtyping	33
2.6.3	Moded Judgments and Typing Rules	34
2.6.4	Subject Reduction	38
2.7	Asynchronous communications	40
2.7.1	Asynchronous Boxed Ambient vs Mobile Ambients	42
2.7.2	Synchrony versus asynchrony: security trade-offs	44
2.8	Related work	45
2.9	Summary of Moded Type System	46
3	Access Control for Mobile Agents	49
3.1	Introduction	49
3.2	Mobile Ambients and Multilevel Security	50
3.2.1	Resource access control in multilevel security	51
3.2.2	Summary and Assessment	55
3.3	Boxed Ambients and Multilevel Security	56
3.4	Access Control by Typing	58
3.4.1	Access Control Types and Typing rules	60
3.4.2	Soundness of the Type System	63
3.5	Examples	64
3.6	Related work	72
3.7	Proofs of Subject Reduction and Type Soundness	73
3.8	Summary of Type System	93

4	Information Flow Security	97
4.1	Introduction	97
4.2	A Type System for Secure Information Flow	101
4.2.1	Types and Judgments	101
4.2.2	Typing Rules	104
4.2.3	Absence of Explicit Flows	110
4.3	Non-interference	112
4.3.1	Typed Equivalence	113
4.3.2	Discussion	116
4.4	Proof of non-interference	118
4.4.1	A Hardening Relation	118
4.4.2	A Labelled Transition System	120
4.4.3	Context and Activity Lemmas	121
4.4.4	Proof of non-interference	125
4.5	Related Work	130
4.6	Omitted Proofs	132
4.6.1	Proof of Harmony Theorem 4.4.3	132
4.6.2	Proof of Activity Theorem 4.4.9	134
4.6.3	Proof of Context Theorem 4.4.4	144
5	Controlling Interferences in Boxed Ambients	149
5.1	Introduction	149
5.2	The NBA Calculus	152
5.2.1	Reduction and Behavioural Semantics	153
5.3	LTS Semantics and Algebraic Theory	155
5.3.1	A characterization of barbed congruence	163
5.3.2	Algebraic Laws	173
5.4	The Type System	176
5.5	Examples	180
5.5.1	Encoding of Channels	180

5.5.2	NBA versus BA and SA	185
6	Secrecy in Open Networks	193
6.1	Introduction	193
6.2	The CBA calculus	199
6.2.1	Syntax	199
6.2.2	Semantics	200
6.3	Type System for partial typing	203
6.3.1	Type System	203
6.4	Secrecy	214
6.5	Examples	217
6.6	Encoding of Spi calculus	221
6.7	Conclusion and Related Work	223
7	Conclusions	225
A		227
References		229

Chapter 1

Introduction

Computer networks have rapidly evolved over the last years, and their role is shifting progressively from that of plain communication media to that of large-scale distributed platforms.

Wide-area networks typically consist of collections of independent domains, all different and separated by administrative boundaries (firewalls). Their structure is dynamic, in that new hosts and communication links can be added with no advance notice, hosts may disappear and later re-appear, possibly under a new name and address. In addition, some components of the network (e.g. laptops and PDA's) can move about, increasing the problems of discontinuous connectivity and dynamicity of network topology.

Programming applications for (and in) such networks requires abstraction mechanisms more flexible than those based on traditional models for distributed systems. Such models, e.g. RPC or client-server architectures, do not scale to networks with dynamic structure and unstable connectivity, and various authors have instead advocated *mobility* as a fundamental programming abstraction for modern applications. Programming techniques based on mobility are receiving increasing interest, and new technologies are being developed to support mobility in different forms, from mechanisms for *code mobility* (e.g. Java applets), to constructs for *agent mobility*.

Drawing on the π calculus [MPW92], many process calculi have been proposed as theoretical foundations for mobility and wide-area networks. The proliferation of such calculi aims at identifying what programming abstractions are best suited for the development

and the analysis of programs and systems. Most of these calculi share few, core principles: agent mobility (i.e. mobility of code, state and control), security management, coordination and control of resource usage.

Our approach is based on the calculus of Mobile Ambients (MA), proposed by Cardelli and Gordon [CG98] as a unifying foundation for mobility, security and communication. The basic notion is that of ambient - the process $n[P]$ - that represents a bounded place where multi-threaded computation happens. An ambient is an abstraction of both notions of agent and place. Like an agent, an ambient can move across places (ambients) where it can interact with other agents. Like a place, an ambient supports local undirected communication, and can receive messages from other places. Furthermore, ambients can be arbitrarily nested, forming a tree structure that models the hierarchical organization of administrative domains. Mobility within this structure is achieved by letting an agent exercise a capability **in** or **out**, associated with the name of the target ambient. For instance, the following process shows how an ambient n may move from the location a to the location b :

$$a[P \mid n[\text{out } a.\text{in } b.Q \mid Q']] \mid b[R] \quad \rightarrow\rightarrow \quad a[P] \mid b[R \mid n[Q \mid Q']]$$

Note that the move of n is not atomic, to reflect the structured nature of the network. The capabilities **in** and **out** provide for the dynamic reconfiguration of systems encompassed by mobility. A third capability, **open**, gives further reconfiguration power by allowing ambient boundaries to be dissolved. Finally, Mobile Ambients provide resources for local, asynchronous communication. These resources are similar to channels in the asynchronous π -calculus, except that MA channels have no name and the context where the communication happens is provided by the surrounding ambient, as in

$$n[\langle M \rangle \mid (x).P] \quad \rightarrow \quad n[P\{x := M\}]$$

where angle brackets represent outputs, round parentheses bind input variables and curly brackets represent substitution.

This core set of primitives is rather elegant, and makes Mobile Ambients a very expressive calculus for mobile computations. However, at the same time, the very choice of the primitives yield a certain lack of atomicity that would be desirable to encode common protocols for distributed computation. For instance, since communication is only local to

an ambient, a message exchange between two sibling agents A and B requires a specific protocol that consists of creating a third agent that carries the message from the A to B , where it is opened and its content read. However, being not atomic, such a protocol has a number of problems, and in particular requires the two agents to be immobile during the communication protocol in order to ensure that the message reaches its destination. More generally, ambient's local processes often contain interfering threads, and their execution may bring the inherent non-determinism of the calculus out of control: lack of atomicity has a formal counterpart in the algebraic theory of the calculus, which appears rather poor. This problem has been addressed in [LS00] by adding to the MA calculus movement cocapabilities, that are used by ambients to express their willingness to be entered/exited/opened.

A further source of problems with Mobile Ambients is related to security. Security is a major concern for computation in open networks, and is often considered a serious source of potential limitation to a widespread use of mobile code technologies. In highly distributed, open networks several attacks are possible. Attackers may sniff network traffic to access or modify application data in transit. A mobile agent, in turn, may try and impersonate another legitimate agent in order to receive sensitive data and gain access to private information. In addition, an agent may try several kinds of attacks against the host supporting its execution: for example, unregulated access to the file system may allow an application to install viruses or Trojan Horses. Dually, mobile agents must be protected from malicious hosts that may try to guess or corrupt their content.

The security model of Mobile Ambients is based on the possession of capabilities: permission to cross ambient boundaries or to access the content of an ambient is given by making the name available to the client willing to enter, exit or open. Names are thus viewed as passwords, and the fact that ambient names are unforgeable is the most basic security property. While this model is suggestive, and powerful for its simplicity, it entirely depends on the ability of the authorization mechanism based on naming to filter out undesired clients. In particular, the **open** capability appears to be very dangerous in that opening an ambient to exchange messages could grant malicious agents full access to all the resources local to the opening process:

$$\begin{aligned} a[Bad \mid \text{in } b.\langle M \rangle] \mid b[\text{open } a.(x).Q] &\rightarrow b[\text{open } a.(x).Q \mid a[Bad \mid \langle M \rangle]] \\ &\rightarrow b[(x).Q \mid Bad \mid \langle M \rangle] \rightarrow b[Q\{x := M\} \mid Bad] \end{aligned}$$

To counter these problems, we propose a variant of Mobile Ambients, that disallows ambient openings and uses different communication primitives for directly exchanging messages between two ambients without boundary dissolution. For instance, the communication between ambient a and b can be achieved by the following process,

$$\begin{aligned} a[Bad \mid \text{in } b.\langle M \rangle] \mid b[(x)^a Q] &\rightarrow b[(x)^a Q \mid a[Bad \mid \langle M \rangle]] \\ &\rightarrow b[Q\{x := M\} \mid a[Bad]] \end{aligned}$$

where the bad process Bad is confined within a .

The resulting calculus, called Boxed Ambients (BA), provides an effective framework for the study of security of mobile agents. We support this argument by showing that the new communication primitives enable a precise type-based analysis of security aspects for distributed computations. In particular, we study three classical security mechanisms: access control, information flow, and the protection of agents within untrusted networks. The first two intend to protect a trusted environment (a host) enforcing a specific security policy that rule out unauthorized operations and unsafe flows of information. The last one, instead, addresses the dual problem, that is to protect an agent from untrusted hosts/sites it visits when moving in the unreliable network.

Our approach to access control aims at providing guarantees of safety and authorization. Safety corresponds to building safeguards against misuse of data (in our model, against misuse of communication channels) leading to run-time failures. Authorization provides an insurance that access to resources is granted only to principals that have obtained appropriate permissions. Information flow analysis, on the other hand, aims at protecting confidentiality and integrity of data by preventing flows of sensitive information to non-authorized subjects. As in other approaches, our analysis of information flow provides for the detection of “implicit” flows of information, i.e. indirect ways of transmitting information (viz. *covert channels*): we complement traditional analysis with a study of the flows of information resulting from agent mobility. Finally, we propose a weak form of cryptography providing a mechanism to lock/unlock agent interactions based on the knowledge of a secret key. This mechanism is then used to protect agents moving through untrusted sites: any such agent, will first block all its interactions with the enclosing context and then move. The agent’s interactions with the context will then be re-enabled only when the agent enters a computational environment that knows its locking key.

1.1 Thesis overview

We propose the calculus of Boxed Ambients (BA), a foundational calculus for programming and analyzing distributed and mobile systems. It is a variant of Mobile Ambients (MA), that, compared to the original, provides a more effective framework for representing and programming mobile agents. The dissertation supports this argument showing that:

- ▷ BA's new communication primitives allow more effective interactions between agents, avoiding ad hoc protocols required in MA.
- ▷ By removing the *open* capability distinctive of MA, BA loses a lot of reconfiguration power, but it more naturally allows the study of different facets of security in distributed systems.
- ▷ BA enjoys rich and tractable type theory and algebraic theory.
- ▷ BA proposes an access protocol where incoming agents must be “authenticated” and registered, reflecting what happens in real computational domains.

Detailed summary

In Chapter 2 we present the calculus of Boxed Ambients (BA) and motivate the choice of the new primitives for communication. We complement the definition of the new calculus with a study of different type systems and a number of examples showing the expressive power of BA. In particular we show that the new model of communication enhances the flexibility of typed communications over Mobile Ambients, it provides new insight into the relationship between synchronous and asynchronous input-output. (Chapter 2 is based on article [BCC01a])

In Chapter 3 we start our study of security-specific issues proposing different formalization attempts for access control policies in Ambient Calculus, providing an in-depth analysis of the problems one encounters. As it turns out, common policies, such as those for mandatory access control (MAC) security, do not appear to have fully convincing interpretations in MA. Instead, the new semantics of communication of Boxed Ambients naturally leads to a direct formalization of classical notions of access control. We propose a type system that provides for static detection of violations of MAC policies in a multilevel security system and we discuss several examples coming from the literature on security issues. (Chapter 3 is based on article [BCC01b])

The study of security properties of Boxed Ambients continues in Chapter 4 where we address the subtler problem of information flow. Relying on non-interference as the basic principle, we develop a sound type system that provides static guarantees of absence of unwanted flow of information for well typed processes. The non-interference proof builds on the equational theory developed in [CG99a] by Cardelli and Gordon for MA, and relies on the choice of contextual equivalence as the underlying equivalence relation. (Chapter 4 is a personal elaboration of [CBC02])

One of the main problems with Boxed Ambients is that the expressivity of its communication model is achieved at the price of grave interferences that may lead to undesired forms of non-determinism. In Chapter 5 we solve this problem by refining the semantics of communication and introducing a new form of co-capabilities that control mobility. The resulting calculus, we call NBA, enjoys good theoretical properties, such as a rich and tractable algebraic theory and a type system that is simpler than those for BA. Furthermore, by means of examples and encodings we show that NBA remains expressive enough, and the expressive power we loose with respect to BA is essentially that directly related to communication interferences. (Chapter 5 is based on article [BCMS02])

The NBA calculus is then the basis for the study of secrecy properties of agents that move in unreliable networks. In Chapter 6 Boxed Ambients are extended with specific primitives that lead to a lower-level calculus that aims to stand at ambients as spi stands to π . The result is a model for protecting agents from the untrusted hosts/sites they visit when moving within open networks. The main idea is a mechanism to dynamically block/unblock the interactions between an agent and its external context, based on the knowledge of a secret key. This model can be seen as well as a first approach to cryptography in the context of mobile agents. We further present a type system for partial typing that statically detects leakages of secrets even in presence of untrusted components. The chapter ends with a number of examples and an encoding of the spi calculus. (Chapter 6 has been updated by [BCPS03])

Chapter 7 concludes the dissertation with a summary of the contributions and a discussion of possible future work.

1.2 Related Work

We already discussed the calculus of Mobile Ambients, that is the starting point of the thesis; in the following we review other four calculi, closely related to our, that have been proposed in the literature. A deeper comparison with related work can be found at the end of each chapter, where we take into account also technical details.

Distributed π -calculus: $D\pi$ [HR98, HR99] and [Sew98] $D\pi$ is a distributed variant of the π -calculus, where processes are located threads, and primitives are provided for threads to move among locations, and for creating new locations. Locations are places that include computational processes that may run in parallel, declare new names and communicate. Local communication, among processes located at the same site, happens exactly as in the π -calculus. Processes located at different sites also may communicate, but remote communication requires thread mobility.

Security management is performed by a type checking mechanism: mobile agents are type checked before being executed in order to ensure that they do not violate the access policy of the current site. The type system for $D\pi$ guarantees that channels are not misused and that resources are accessed by agents that have been granted to do so. Furthermore, in [RH99], the authors study a type system for $D\pi$ that deals with open networks, in which a subset of hosts (localities) may be untrusted. The structure of the type system is then enriched with (i) partial types that label some location as untrusted, and (ii) run-time type checking that enforces security restrictions for processes coming from untrusted locations.

In [Sew98] a variant of distributed π -calculus is introduced. The main differences are the possibility of nesting locations and the fact that (asynchronous) communication is possible not only locally, but also across the location tree structure. In addition, the capability-based type system distinguishes input/output capabilities for local and global communication channels. Via the typing of channel declarations, the programmer can then separate local and global communications.

Distributed join-calculus: $D\text{join}$ [FGL⁺96] The Join calculus is an asynchronous variant of the π -calculus where the operators for input, restriction and replication are combined into a single operator, called “definition”. A definition binds names and entirely specifies their communication behavior by describing how messages sent on these names

will be received. A definition consists of a set of linear patterns of messages associates to a guarded process; when messages running in parallel match a pattern, they are consumed and replaced by an instance of the guarded process.

The distributed join-calculus refines the basic calculus to explicitly model locality, mobility, remote communication and failures. Remote communication is asynchronous and happens in two phases. First, the message is forwarded to the location of its definition, then it can be consumed locally according to the join-pattern of the definition. Contrary to what happens in Mobile Ambients, join calculus hides the details of message routing.

In `Djoin` security is achieved by means of cryptographic primitives analogous to those of spi-calculus ([AG99]).

Seal calculus [VC99] The Seal calculus is a process calculus for secure distributed applications over large scale distributed open networks. In Seal, interactions are synchronous communications of a value or of a seal along named channels. Mobility is thus obtained by communicating a seal on a channel. Interactions can happens among processes in the same seal, or among processes in two seals that are in the parent-child relationship.

Security is addressed by a fine-grained access control mechanism. First, each seal controls all interactions of its children, both with the outside world and with one-another; second, each seal has total control over its name-space and therefore must determine the names of its children. In [VC99] channels are considered as resources and each channel belongs to one and only one seal. Specific syntactic constructs allow the owner of a channel to regulate remote access to it and thus to control both remote communication and mobility.

The design of Boxed Ambients shares with Seal the principles of mediation and locality. Mediation implies that remote communication between boxed ambients requires mobility or the intervention of the ambient's parent. Locality means that communication channels are local to ambients, and message exchanges result from explicit read and write requests on those channels. However, the two calculi are different in the underlying model of mobility: in BA moves are subjective, while in Seal they are objective; in Chapter 5 we discuss the benefits of an alternative mobility model for Boxed Ambient that is more closely related to that of Seal.

KLAIM kernel calculus [DNFP98, DNFP99, DNFPV00] KLAIM is a language that supports a programming paradigm where processes, like data, can be moved from one computing environment to another. It is a Linda-like language with multiple tuple spaces enriched with explicit information about the location of the nodes where processes and tuples are allocated.

Explicit localities enable the programmer to distribute and retrieve data and processes to and from the sites of a net and to structure the tuple space as multiple, located spaces. Moreover, localities, considered as first-order data, can be dynamically created and communicated over the network. Coordinators have complete control over changes of configuration of the network that may be due to addition/deletion of software components and sites, or to transmission of programs and sites references.

KLAIM comes equipped with a type system that statically checks access rights violations of mobile agents. Types are used to describe the intentions (read, write, execute,..) of processes in relation to the various localities. The type system is used to determine the operations that processes want to perform at each locality, to check whether they comply with the declared intentions and whether they have the necessary rights to perform the intended operations at the specific localities.

1.3 Remark

In addition to the work covered by this thesis, during my PhD I studied a different research topic, that is the type theory for object-oriented languages. In [BBC02] we study a type preserving and computationally adequate interpretation of a formal calculus of extensible objects into a higher order λ -calculus with records, polymorphic types, recursive types, and subtyping. The resulting interpretation provides theoretical foundations of object-oriented languages and their specific constructs.

In [BCC00, BCC01c] we describe a general model for embedding object-oriented constructs into calculi of mobile agents. In particular we study a typed instance of that model based on the calculus of Mobile Ambients.

Chapter 2

The calculus of Boxed Ambients

Boxed Ambients are a variant of Mobile Ambients based on a different choice of communication primitives. The new calculus drops the **open** capability and complements the constructs **in** and **out** for mobility with what we believe to be more effective mechanisms for ambient interaction and resource protection. The new model of communication is faithful to the principles of distribution and location-awareness of Mobile Ambients, it provides for more flexible typing of communications, and new insight into the relation between synchrony and asynchrony.

2.1 Introduction

There is a general agreement that calculi and programming languages for wide-area computing and mobile-code environments should be designed according to appropriate principles, among which distribution and location awareness are the most fundamental.

Cardelli and Gordon's Mobile Ambients [CG98] are one of the first, and currently one of the most successful implementations of these principles into a formal calculus. Their design is centered around four basic notions: location, mobility, communication by shared location and authorization to move based on acquisition of names and capabilities. The ability or inability to cross boundaries, which is conferred by the capabilities **in** and **out**, is at the core of the security model underlying MA. Permission to cross ambient boundaries is given by making the name available to the clients requesting access. Names are thus viewed as passwords, or cryptokeys: when embedded in a capability, an ambient name provides the pass that enables access to, or else the cryptokey that discloses the contents of that ambient. While MA's model of security is suggestive, and powerful for its simplicity,

it does not appear to be fully adequate for modeling realistic policies for resource access control. Security in MA entirely depends on the ability by the name-based authorization mechanism to filter out unwanted clients: an authorization breach could grant malicious agents full access to all the resources located inside the ambient boundary.

The starting point of this thesis was an assessment of security and resource access control in ambient-based calculi. Our analysis, detailed below and in Chapter 3, points out the shortcomings of MA as a formal basis for reasoning about these concepts.

To overcome these difficulties, we introduce the calculus of *Boxed Ambients*, a variant of MA based on a different set of primitives for communication. Boxed Ambients inherit from MA the primitives `in` and `out` for mobility, with the exact same semantics. Instead, they rely on a different model of communication, which results from dropping the `open` capability, and from introducing direct primitives for parent-child communication across ambient boundaries.

The new communication primitives fit nicely the design principles of Mobile Ambients, and complement the existing constructs for ambient mobility with finer-grained, and more effective, mechanisms for ambient interaction. As a result Boxed Ambients retain the computational flavor of Ambient Calculus, as well as the elegance of its formal presentation. In addition, they enhance the flexibility of typed communications over Mobile Ambients, and provide new insight into the relationship between synchronous and asynchronous input-output.

In this chapter we present the calculus of Boxed Ambients. We complement the definition of the calculus with a study of different type systems. A first type system provides standard safety guarantees for communication. A second type system enhances the typing of mobility and develops a new typing technique, based on different typing “modes” for processes, in which processes and their continuations may have different types while still preserving subject reduction. As an example of the expressive power of BA, in Section 2.3 we give the encoding of different forms of channeled communications, based either on π -calculus channels or on the Seal calculus’ located channels. In Section 2.7 we study an asynchronous variant of the calculus, and we discuss the relationship between the two forms of communication and their impact on mobility. The study of security properties of Boxed Ambients is postponed to the following chapters.

2.1.1 Motivations for being *boxed*

While fundamental in the Ambient Calculus to enable communication across ambient boundaries, an unrestricted use of the **open** capability appears to bring about serious security concerns in wide-area distributed applications.

Consider a scenario where a process P running on host h downloads an application program Q from some other host over the network. This situation can be modeled by the configuration $a[\text{in } h.Q] \mid h[P]$, where Q is included in the transport ambient a which is routed to h in response to the download request from P . As a result of a exercising the capability in h , the system evolves into the new configuration $h[a[Q] \mid P]$, where the download is completed. The application program Q may be running and computing within a , but as long as it is encapsulated into a , there is no way that P and Q can effectively interact. To allow for interactions, P will need to dissolve the transport ambient a . Now, dissolving a produces the new configuration $h[P \mid Q]$ where now P and Q are granted free access to each other's resources: the problem, of course, is that there is no way to tell what Q may do with them. An alternative, but essentially equivalent, solution to the above scenario, is to treat a as a *sandbox* and take the Java approach to security: P clones itself and enters the sandbox to interact with Q . Again, however, the kind of interaction between P and Q is not fully satisfactory: either they interact freely within a , or are isolated from each other.

Static or dynamic analysis of incoming code are often advocated as solutions to the above problem: incoming code must be statically checked and certified prior to being granted access to resources and sensitive data. Various papers explored this possibility, proposing control-flow analyses [NNHJ99, NN00, DLB00] and type systems [CGG99, DCS00, BC01b] for Mobile Ambients. The problem with these solutions is that they may not be always possible, or feasible, in practice. The source code of incoming software may be not available for analysis, or be otherwise inaccessible or too complex to ensure rigorous assessment of its behavior. This is not meant to dismiss the role of static analysis. To the contrary, it should be taken as a motivation to seek new design principles enabling a more effective use of static analysis. One such principle for Mobile Ambients, which we advocate and investigate in this chapter, is that ambient interaction should be controlled by finer-grained policies to prevent from unrestricted resource access while still providing

effective communication primitives.

2.1.2 Boxed Ambients: overview and main results

Boxed Ambients result from Cardelli and Gordon’s Mobile Ambients essentially by dropping the *open* capability while retaining the *in* and *out* capabilities for mobility. Disallowing *open* represents a rather drastic departure from the Ambient Calculus, and requires new primitives for process communication.

As in the Ambient Calculus, processes in the new calculus communicate via anonymous channels, inside ambients. This is a formal parsimony that simplifies the definition of the new calculus while not involving any loss of expressive power: in fact, named communication channels can be coded in faithful ways using the existing primitives. In addition, to compensate for the absence of *open*, Boxed Ambients are equipped with primitives for communication across ambient boundaries, between parent and children. Syntactically, this is obtained by means of tags specifying the *location* where the communication has to take place: for instance, $(x)^n P$ indicates an input from child ambient n , while $\langle M \rangle^\uparrow$ is an output to the parent ambient.

The choice of these primitives, and the resulting model of communication is inspired to Castagna and Vitek’s *Seal Calculus* [VC99], from which Boxed Ambients also inherit the two principles of *mediation* and *locality*. Mediation implies that remote communication, i.e. between sibling ambients, is not possible: it either requires mobility, or intervention by the ambients’ parent. Locality means that communication resources are *local* to ambients, and message exchanges result from explicit read and write requests on those resources. To exemplify, consider the following nested configuration:

$$n[(x)^p P \mid p[\langle M \rangle \mid (x)Q \mid q[\langle N \rangle^\uparrow]]]$$

Ambient n makes a downward request to read p ’s local value M , while ambient q makes an upward write request to communicate its value N to its parent. The downward input request $(x)^p P$ may only synchronize with the output $\langle M \rangle$ local to p . Instead, $(x)Q$ may non-deterministically synchronize with either output: of course, type safety requires that M and N be of the same type. Interestingly, however, exchanges of different types may take place within the same ambient without type confusion:

$$n[(x)^p P \mid (x)^q Q \mid p[\langle M \rangle] \mid q[\langle N \rangle]]$$

The two values M and N are local to p and q , respectively, and may very well have different types: there is no risk of type confusion, as $(x)^p P$ requests a read from p , while $(x)^q Q$ requests a read from q .

This flexibility of communication results from the combination of two design choices: directed input/output operations, and resource locality. In fact, these choices have other interesting consequences.

- ▷ They provide the calculus with fine-grained primitives for ambient interaction, and with clear notions of resource ownership and access request. Based on that, Boxed Ambients enable a rather direct formalization of classical security policies for resource protection and access control: this is not easy (if at all possible) with Mobile Ambients (see Chapter 3).
- ▷ They ease the design of type systems providing precise accounts of ambient behavior. As we show in § 2.4, a rather simple structure of types suffices for that purpose. Ambient and process types are defined simply as two-place constructors describing the types of exchanges that may take place locally, and with the enclosing context. Interestingly, this simple type structure is all that is needed to give a full account of ambient interaction. This is a consequence of (i) there being no way for ambients to communicate directly across more than one boundary, and (ii) communication being the only means for ambient to interact. Based on that, the typing of Boxed Ambients provides for more flexibility of communication and mobility than existing type systems for Mobile Ambients (see § 2.5).
- ▷ Finally, resource locality and directed input/output provide new insight into the relation between the synchronous and asynchronous models of communication. Specifically, the classic π -calculus relations between synchronous and asynchronous output, as stated by Boudol in [Bou92], no longer hold as a result of combining remote communications, resource locality and mobility. More precisely asynchronous output may *no longer* be viewed as a special case of synchronous output with null continuation, neither can it be encoded by structural equivalence, by stipulating that $\langle M \rangle P \equiv \langle M \rangle \mid P$. As we show (see § 2.7) these two solutions, which are equivalent in the π -calculus, have rather different consequences in Boxed Ambients.

2.2 Boxed Ambients

Syntax. The untyped syntax of the polyadic synchronous calculus is defined by the following productions:

<i>Expressions</i>	M	$::=$	m, n, \dots	names
		$ $	x, y, \dots	variables
		$ $	$\text{in } M$	enter M
		$ $	$\text{out } M$	exit M
		$ $	$M.M$	path
<i>Locations</i>	η	$::=$	M	names and variables
		$ $	\uparrow	enclosing ambient
		$ $	\star	local
<i>Processes</i>	P	$::=$	$\mathbf{0}$	stop
		$ $	$M.P$	action
		$ $	$(\nu n)P$	restriction
		$ $	$P \mid P$	composition
		$ $	$M[P]$	ambient
		$ $	$!P$	replication
		$ $	$(x_1, \dots, x_k)^\eta P$	patterned input, $k \geq 0$
		$ $	$\langle M_1, \dots, M_k \rangle^\eta P$	synchronous output, $k \geq 0$

Patterns and expressions are as in the polyadic Ambient Calculus, save that the capability **open** and the “empty path” ε are left out. As in the Ambient Calculus, the syntax allows the formation of meaningless process forms such as **in out** m or **(open** a) $[P]$: these terms may arise as a result of reduction, in the untyped calculus, but are ruled out by the type system. We use a number of notation conventions. We reserve m, n, p, q to range over ambient names, x, y, z over variables, and a, b, c over both. As usual we omit trailing dead processes, writing M for $M.\mathbf{0}$, and $\langle M \rangle$ for $\langle M \rangle \mathbf{0}$. We write $\langle \tilde{M} \rangle^\eta, (\tilde{x})^\eta$ for $\langle M_1, \dots, M_k \rangle^\eta$ and $(x_1, \dots, x_k)^\eta$. The empty tuple (resulting from $k = 0$) allow synchronization without exchange of values. Finally, the superscript \star denoting local communication, is omitted.

The set of free names and variables of a process is mostly defined as follows:

$$\begin{array}{ll}
fn(\star) &= \emptyset \\
fn(n) &= \{n\} \\
fn(x) &= \emptyset \\
fn(M.N) &= fn(N) \cup fn(M) \\
fn(P \mid Q) &= fn(P) \cup fn(Q) \\
fn(M[P]) &= fn(P) \cup fn(M) \\
fn((x_1, \dots, x_k)^\eta P) &= fn(P) \cup fn(\eta) \\
fn(\langle M_1, \dots, M_k \rangle^\eta P) &= fn(P) \cup fn(\eta) \cup fn(M_1) \cup \dots \cup fn(M_k) \\
fv(\star) &= \emptyset \\
fv(n) &= \emptyset \\
fv(x) &= \{x\} \\
fv(M.N) &= fv(N) \cup fv(M) \\
fv(P \mid Q) &= fv(P) \cup fv(Q) \\
fv(M[P]) &= fv(P) \cup fv(M) \\
fv((x_1, \dots, x_k)^\eta P) &= (fv(P) \setminus \{x_1, \dots, x_k\}) \cup fv(\eta) \\
fv(\langle M_1, \dots, M_k \rangle^\eta P) &= fv(P) \cup fv(\eta) \cup fv(M_1) \cup \dots \cup fv(M_k) \\
fn(\uparrow) &= \emptyset \\
fn(\mathbf{cap} M) &= fn(M) \\
fn(\mathbf{0}) &= \emptyset \\
fn(!P) &= fn(P) \\
fn((\nu n)P) &= fn(P) \setminus \{n\} \\
fn(M.P) &= fn(M) \cup fn(P) \\
fv(\uparrow) &= \emptyset \\
fv(\mathbf{cap} M) &= fv(M) \\
fv(\mathbf{0}) &= \emptyset \\
fv(!P) &= fv(P) \\
fv((\nu n)P) &= fv(P) \\
fv(M.P) &= fv(M) \cup fv(P)
\end{array}$$

As customary we use $P\{x := M\}$ to denote the term P in which M is replaced for every free occurrence of x . We also use $P\{x_i := M_i\}_{1 \leq i \leq k}$ to denote the simultaneous substitution of x_1, \dots, x_k and omit the subscript when clear from the context. The formal definition of these meta-notations is omitted as it is the same as for MA, with the only difference that substitutions apply also to locations.

The operational semantics is defined in terms of reduction and structural congruence. Structural Congruence is defined as a congruence relation that is a commutative monoid for $\mathbf{0}$ and \mid , that is

$$\begin{array}{ll}
P \equiv P & P \mid Q \equiv Q \mid P \\
P \equiv Q \Rightarrow Q \equiv P & (P \mid Q) \mid R \equiv P \mid (Q \mid R) \\
P \equiv Q, Q \equiv R \Rightarrow P \equiv R & P \mid \mathbf{0} \equiv P \\
P \equiv Q \Rightarrow (\nu n)P \equiv (\nu n)Q & P \equiv Q \Rightarrow P \mid R \equiv Q \mid R \\
P \equiv Q \Rightarrow !P \equiv !Q & P \equiv Q \Rightarrow n[P] \equiv n[Q] \\
P \equiv Q \Rightarrow M.P \equiv M.Q & P \equiv Q \Rightarrow (x)^\eta.P \equiv (x)^\eta.Q \\
P \equiv Q \Rightarrow \langle M \rangle^\eta.P \equiv \langle M \rangle^\eta.Q &
\end{array}$$

and it is closed under the following rules:

$$\begin{array}{ll}
(\text{Struct Res Dead}) & (\nu n)\mathbf{0} \equiv \mathbf{0} \\
(\text{Struct Res Res}) & (\nu n)(\nu m)P \equiv (\nu m)(\nu n)P \quad n \neq m \\
(\text{Struct Path Assoc}) & (M.M').P \equiv M.(M'.P) \\
(\text{Struct Res Par}) & (\nu n)(P \mid Q) \equiv P \mid (\nu n)Q \quad n \notin \text{fn}(P) \\
(\text{Struct Repl}) & !P \equiv !P \mid P \\
(\text{Struct Res Amb}) & (\nu n)a[P] \equiv a[(\nu n)P] \quad n \neq a
\end{array}$$

As usual, structural congruence is used to rearrange the process so that they can reduce:

$$(\text{Red Struct}) \quad P' \equiv P, P \rightarrow Q, Q \equiv Q' \Rightarrow P' \rightarrow Q'$$

Reduction is defined by the rules for mobility and communication given below plus the standard context reduction rules.

Mobility. Ambient mobility is governed by the following rules, inherited from MA:

$$\begin{array}{ll}
(\text{enter}) & n[\text{in } m.P \mid Q] \mid m[R] \rightarrow m[n[P \mid Q] \mid R] \\
(\text{exit}) & n[m[\text{out } n.P \mid Q] \mid R] \rightarrow m[P \mid Q] \mid n[R]
\end{array}$$

Communication. Communication can be local, as in Mobile Ambients, or across ambient boundaries, between parent and child.

$$\begin{array}{ll}
(\text{local}) & (\tilde{x})P \mid \langle \tilde{M} \rangle Q \rightarrow P\{\tilde{x} := \tilde{M}\} \mid Q \\
(\text{input } n) & (\tilde{x})^n P \mid n[\langle \tilde{M} \rangle Q \mid R] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R] \\
(\text{input } \uparrow) & \langle \tilde{M} \rangle P \mid n[(\tilde{x})^\uparrow Q \mid R] \rightarrow P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] \\
(\text{output } n) & \langle \tilde{M} \rangle^n P \mid n[(\tilde{x})Q \mid R] \rightarrow P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] \\
(\text{output } \uparrow) & (\tilde{x})P \mid n[\langle \tilde{M} \rangle^\uparrow Q \mid R] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R]
\end{array}$$

Contextual Rules As for mobile ambients, reduction can take place in every context, except prefixed processes:

$$\begin{array}{ll}
(\text{Red Par}) & P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R \\
(\text{Red Res}) & P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q \\
(\text{Red Amb}) & P \rightarrow Q \Rightarrow n[P] \rightarrow n[Q]
\end{array}$$

Four different reduction rules for parent-child communication may be thought of as redundant, especially because there are only two reducts. However, none of these reductions can be encoded using the others. Instead, introducing different *directions* for input/output is a key design choice that provides the calculus with precise notions of resource locality and resource access request. The input prefix $(x)^n$ can be seen as a request to read from the anonymous channel located into child ambient n . In fact, given the anonymous nature of channels, $(x)^n$ can equivalently be seen as an access to the ambient n . Dually, $\langle M \rangle^\dagger$ can be interpreted as write request to the parent ambient (equivalently, its local channel). Other alternatives would be possible to model intraboundary communications: in Chapter 5 we study a different communication mechanism that provides ambients with full control of the exchanges they may have with their children. However, parent-child communications introduced in this section enhance the expressivity of BA, thanks to the dynamic binding mechanism implicit in upward exchanges (i.e. in rules $(input \uparrow)$, $(output \uparrow)$)¹.

Based on these intuitions, the definition of reduction enables the study of properties related to standard resource access control policies, even in the absence of channels (see Section 2.5 and Chapter 3 for more details). Channels, on the other hand, can be encoded elegantly, as we show in the next section. Finally, the communication model of Boxed Ambients fits nicely the security model of MA which is based on authorization, and predicated in/out access to ambients on possession of appropriate passwords or cryptokeys.

As we noted, in the present formulation output is synchronous. For the time being, asynchronous output can either be considered as the special case of synchronous communication with null continuation, or else be accounted for by introducing the following rule of structural equivalence inspired by [Bou92]: $\langle M \rangle^\eta P \equiv \langle M \rangle^\eta \mid P$. As we shall discuss later on, these interpretations are equivalent, and both type sound, with “simple” types, that is, up to § 2.5. Instead, with “moded types” we introduce in § 2.6 they are different, and neither one satisfactory, as the former is too restrictive while the latter is unsound.

¹See Chapter 5 for a formal comparison of the two variants of BA that uses two different communication models

2.3 Communication Channels

In this section we investigate on the expressive power of BA showing that value exchange between Boxed Ambients can also take place on named channels. In particular, we first show how to define π -calculus channels; then, we refine the encoding to account for parent-child channeled communications.

2.3.1 π -calculus channels

In π -calculus, two processes communicating over a channel, say c , have the form $\bar{c}\langle a \rangle.P$ (output) and $c(x).Q$ (input). The output prefix $\bar{c}\langle a \rangle$ can be represented in BA with an ambient c that holds a local message $\langle a \rangle$. The input process $c(x).P$ is then represented in BA by the process $(x)^c P$ that reads the local message contained in the ambient c and then continues.

A compositional encoding of the asynchronous π -calculus is thus the following:

$$\begin{aligned} \langle \nu n \rangle P &= (\nu n) \langle P \rangle & \langle \mathbf{0} \rangle &= \mathbf{0} \\ \langle P \mid Q \rangle &= \langle P \rangle \mid \langle Q \rangle & \langle !P \rangle &= !\langle P \rangle \\ \langle \bar{c}\langle a \rangle \rangle &= c[\langle a \rangle] & \langle c(x).P \rangle &= (x)^c \langle P \rangle \end{aligned}$$

Furthermore, the synchronous output can be obtained from the asynchronous one, adding a private ambient that acknowledges the receipt of the message:

$$\langle \bar{c}\langle a \rangle.P \rangle = (\nu p) c[\langle a \rangle p[\text{out } c.\langle \rangle]] \mid ()^p \langle P \rangle \quad p \notin fn(P)$$

The communication along a channel c is then captured by the following sequence of reductions:

$$\begin{aligned} \langle \bar{c}\langle a \rangle.P \mid c(x).Q \rangle &= (\nu p) c[\langle a \rangle p[\text{out } c.\langle \rangle]] \mid ()^p \langle P \rangle \mid (x)^c \langle Q \rangle \\ &\rightarrow (\nu p) c[p[\text{out } c.\langle \rangle]] \mid ()^p \langle P \rangle \mid \langle Q \rangle\{x := a\} \\ &\rightarrow\rightarrow c[\mathbf{0}] \mid (\nu p)(p[\mathbf{0}]) \mid \langle P \rangle \mid \langle Q \rangle\{x := a\} \end{aligned}$$

At the end of the communication, besides the continuation $\langle P \rangle \mid \langle Q \rangle\{x := a\}$, two additional processes remain: $c[\mathbf{0}]$ and $(\nu p)(p[\mathbf{0}])$. The second process can be garbage collected in BA since any reasonable (observational) equivalence equates $(\nu p)(p[\mathbf{0}])$ and $\mathbf{0}$. On the other hand, the process $c[\mathbf{0}]$ is not equivalent to $\mathbf{0}$ since a BA process may want to enter/exit the ambient c . Then we have that for the encoding above, no soundness

property can be formally proved. Anyway, we postpone the definition of a sound encoding of π -calculus to Chapter 5, the aim of the present section simply being an example of the expressive power of BA.

2.3.2 Parent-child channeled communication à la Seal Calculus

Having looked at π -calculus channels, we now discuss an extension of the encoding that conforms with the notion of locality and directed input-output of the core calculus. The extension yields a notion of located channels and a set of communication protocols that are similar, in spirit, to those given as primitive in the Seal Calculus [VC99]. In the Seal Calculus, one can express output prefixes of the form $c^n\langle M \rangle$ requesting a write access on the channel c residing in ambient (or seal) n . Dually, the input prefix $c^\uparrow(x)$ denotes a read request on the channel c residing in the parent ambient. Upward output and downward input on local channels may be expressed in similar ways. All these communication protocols can be expressed in the core calculus of Boxed Ambients: below, we only consider only the asynchronous case (i.e. continuation-less outputs) and we give detailed descriptions of downward output and upward input.

The intended reduction of a downward output is:

$$c^n\langle M \rangle \mid n[c(x)P \mid Q] \rightarrow n[P\{x := M\} \mid Q].$$

The channel c is local to n , and the outer process requests a write access on c . There are several ways that the reduction can be captured with the existing constructs. Here, we describe an encoding that renders the locality of c . The channel c is represented as a buffer with an unbounded number of positions, and the local input/output prefixes as read/write access request to c :

$$c(x)P = c[!(x)\langle x \rangle] \mid (x)^cP \quad c\langle M \rangle = c[!(x)\langle x \rangle] \mid \langle M \rangle^c$$

Now, however, the write access $c^n\langle M \rangle$ cannot be represented directly as the local output $c\langle M \rangle$, because c is located into n . To capture the desired behavior we can rely on mobility:

$$c^n\langle M \rangle \triangleq (\nu p)p[\text{in } n.\text{in } c.\langle M \rangle^\uparrow]$$

The output M is encapsulated into a transport ambient p , which enters n and then c to deliver the message (the name of the transport ambient p must be fresh). Thus, the Seal Calculus process $c^n\langle M \rangle \mid n[c(x)P \mid Q]$ is encoded as follows:

$$(\nu p)p[\text{in } n.\text{in } c.\langle M \rangle^\uparrow] \mid n[c[!(x)\langle x \rangle] \mid (x)^cP \mid Q]$$

By a sequence of reductions, the process above evolves into

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

where the process $(\nu p)p[\mathbf{0}]$ can be garbage collected by any reasonable (observational) equivalence.

Remote inputs are slightly more complex, since the transport ambient must fetch the output and bring it back. The intended reduction is $c\langle M \rangle \mid n[c^\uparrow(x)P \mid Q] \rightarrow n[P\{x := M\} \mid Q]$.

Upward input from within a seal n is simulated in Boxed Ambients as

$$c^\uparrow(x)P \triangleq (\nu p)p[\text{out } \underline{n}.\text{in } c.(x)^\uparrow \text{out } c.\text{in } \underline{n}.\langle x \rangle] \mid (x)^p P$$

Note that the definition depends on the name n of the enclosing ambient. For a formal definition, it is enough to keep track of this information:

$$\begin{aligned} \langle c\langle M \rangle \rangle_n &= c[!(x)\langle x \rangle] \mid \langle M \rangle^c \\ \langle c^m\langle M \rangle \rangle_n &= (\nu p)p[\text{in } m.\text{in } c.\langle M \rangle^\uparrow] \\ \langle c^\uparrow\langle M \rangle \rangle_n &= (\nu p)p[\text{out } n.\text{in } c.\langle M \rangle^\uparrow] \\ \langle c(x)P \rangle_n &= c[!(x)\langle x \rangle] \mid (x)^c \langle P \rangle_n \\ \langle c^m(x)P \rangle_n &= (\nu p)p[\text{in } m.\text{in } c.(x)^\uparrow \text{out } c.\text{out } m.\langle x \rangle] \mid (x)^p \langle P \rangle_n \\ \langle c^\uparrow(x)P \rangle_n &= (\nu p)p[\text{out } n.\text{in } c.(x)^\uparrow \text{out } c.\text{in } n.\langle x \rangle] \mid (x)^p \langle P \rangle_n \\ \langle a[P] \rangle_n &= a[\langle P \rangle_a] \end{aligned}$$

2.4 Typing Boxed Ambients

As we stated at the outset, one of the goals in the design of Boxed Ambients was to enhance static reasoning on ambient and process behavior, by enabling focused and precise analyses while preserving the expressive power of the calculus. The definition of the type system, given in this section, proves that the design satisfies these requirements.

A rather simple structure of types suffices to provide precise accounts of process behavior. Ambient and process types are defined simply as two-place constructors describing the types of the exchanges that may take place locally and with the enclosing context. Interestingly, this simple type structure is all that is needed to give a full account of ambient interaction. This is a consequence of (i) there being no way for ambients to communicate directly across more than one boundary, and (ii) communication being the only means for ambient to interact.

2.4.1 Types

The structure of types is defined by the following productions.

$$\begin{array}{lll}
 \textit{Expression Types } W & ::= & \text{Amb}[E, F] \quad \text{ambient} \\
 & | & \text{Cap}[E] \quad \text{capability} \\
 & | & W_1 \times \dots \times W_k \quad \text{tuple, } k \geq 0
 \end{array}$$

$$\begin{array}{lll}
 \textit{Exchange Types } E, F & ::= & \text{shh} \quad \text{no exchange} \\
 & | & W \quad \text{exchange}
 \end{array}$$

$$\textit{Process Types } T ::= [E, F] \quad \text{composite exchange}$$

The type structure is superficially similar to that of companion type systems for the Ambient Calculus [CG99b, CGG99]. The interpretation, however, is different.

- $\text{Amb}[E, F]$ ambients that enclose processes of type $[E, F]$,
- $\text{Cap}[E]$ capabilities exercised within ambients with E upward exchanges,
- $[E, F]$ processes whose local and upward exchanges have types E and F , resp.

Notice that capability types are defined as one-place constructors, and disregard the local exchanges of the ambients where they are exercised. This is because exercising a capability within an ambient, say a , may only cause a to move, and the safety of ambient mobility may be established regardless of the ambient's local exchanges.

As for process types, a few examples help explain the intuition about composite exchange. We use a Church style typed syntax, in which all inputs and restrictions specify the type of the bound variable: more precisely we use $(\nu n : \text{Amb}[E, F])P$ and $(x_1 : W_1, \dots, x_k : W_k)^n P$ instead of $(\nu n)P$ and $(x_1, \dots, x_k)^n P$, respectively. We also use the notation $\tilde{x} : \tilde{W}$ instead of $x_1 : W_1, \dots, x_k : W_k$.

$(x : W)\langle x \rangle : [W, \text{shh}]$. W is exchanged (read and written) locally, and there is no upward exchange.

$(x : W)^\dagger \langle x \rangle^n : [\text{shh}, W]$. W is exchanged (i.e. read from) upwards, and then written to ambient n . There is no local exchange, hence the type shh as the first component of the process type. For the typing to be derivable, one needs $n : \text{Amb}[W, E]$ for some exchange type E .

$(x:W)^\dagger(y:W')(\langle x \rangle^n \mid \langle y \rangle) : [W', W]$. W is exchanged (read from) upwards, and then forwarded to ambient n , while W' is exchanged (read and written) locally. Again, for the typing to be derivable, one needs $n : \text{Amb}[W, E]$ for some exchange type E .

$(x:W)\langle x \rangle^\dagger : [W, W]$. W is read locally, and written upwards.

These simple examples give a flavor of the flexibility provided by the constructs for communication: like mobile ambients, boxed ambients are “places of conversation”, but unlike ambients they allow more than just one “topic” of conversation. This is made possible by the local nature of (anonymous) channels, and the consequent “directed” forms of input/output. Specifically, every ambient may exchange values of different types with any of its children, as long as the exchange is directed from the ambient to the children. Instead, upward communication is more constrained: all the children must agree on the (unique) type of exchange they may direct to their parent.

2.4.2 Typing Rules

The judgments of the type system have two forms: $\Gamma \vdash M : W$, read “*expression M has type W under Γ* ”, and $\Gamma \vdash P : T$, read “*process P has type T under Γ* ”, where as usual Γ is a type environment mapping names and variables into types. Accordingly we have two sets of typing rules, one for names and capabilities, one for processes. In addition, we introduce a subsumption rule for process types, based on the following definition of subtyping.

Definition 2.4.1 (*Subtyping*). We denote by \leq the smallest reflexive and transitive relation over exchange types such that $\text{shh} \leq E$ for every exchange type E . Exchange subtyping is lifted to process types as follows: $[E, F] \leq [E', F]$ if and only if $E \leq E'$. \square

Process subtyping is used in conjunction with subsumption, exchange subtyping is not. The intuition for exchange subtyping is that a (locally or upward) shh exchange is always type compatible with a situation in which some exchange is expected: this is useful in the typing of capabilities. As for process subtyping, the relation is quite shallow: it is “almost equality” as there is no subtyping *in depth*. It would be tempting to extend the subtyping relation so as to allow subtyping over upward exchanges, as well. However, as we explain later in this section, uses of this relation in conjunction with a subsumption rule for process types would not be sound.

Typing of Expressions

$$\begin{array}{c}
\text{(PROJECTION)} \\
\frac{\Gamma(a) = W}{\Gamma \vdash a : W}
\end{array}
\qquad
\begin{array}{c}
\text{(PATH)} \\
\frac{\Gamma \vdash M_1 : \text{Cap}[F] \quad \Gamma \vdash M_2 : \text{Cap}[F]}{\Gamma \vdash M_1.M_2 : \text{Cap}[F]}
\end{array}$$

$$\begin{array}{c}
\text{(IN)} \\
\frac{\Gamma \vdash M : \text{Amb}[F, E] \quad F' \leq F}{\Gamma \vdash \text{in } M : \text{Cap}[F']}
\end{array}
\qquad
\begin{array}{c}
\text{(OUT)} \\
\frac{\Gamma \vdash M : \text{Amb}[E, F] \quad F' \leq F}{\Gamma \vdash \text{out } M : \text{Cap}[F']}
\end{array}$$

The (PROJECTION) and (PATH) rules are standard. The rules (IN) and (OUT) define the constraints for safe ambient mobility, and explain why capability types are built around a single component. The intuition is as follows: take a capability, say $\text{in } n$ with $n : \text{Amb}[F, E]$, and suppose that this capability is exercised within ambient, say, m . If m has upward exchanges of type F' , then $\text{in } n : \text{Cap}[F']$. Now, for the move of m into n to be safe, one must ensure that the type F of the local exchanges of n be equal to the type F' of the upward exchanges of m . In fact, the typing can be slightly more flexible, for if m has no upward exchange, then $F' = \text{shh} \leq F$, and m may safely move into n . Dual reasoning applies to the (OUT) rule: the upward exchanges of the exiting ambient must have type \leq -compatible with the type of the upward exchanges of the ambient being exited.

Typing of Processes

$$\begin{array}{c}
\text{(DEAD)} \\
\frac{}{\Gamma \vdash \mathbf{0} : T}
\end{array}
\qquad
\begin{array}{c}
\text{(NEW)} \\
\frac{\Gamma, n : \text{Amb}[E, F] \vdash P : T}{\Gamma \vdash (\nu n : \text{Amb}[E, F])P : T}
\end{array}
\qquad
\begin{array}{c}
\text{(PARALLEL)} \\
\frac{\Gamma \vdash P : [E, F] \quad \Gamma \vdash Q : [E, F]}{\Gamma \vdash P \mid Q : [E, F]}
\end{array}$$

$$\begin{array}{c}
\text{(PREFIX)} \\
\frac{\Gamma \vdash M : \text{Cap}[F] \quad \Gamma \vdash P : [E, F]}{\Gamma \vdash M.P : [E, F]}
\end{array}
\qquad
\begin{array}{c}
\text{(AMB)} \\
\frac{\Gamma \vdash M : \text{Amb}[E, F] \quad \Gamma \vdash P : [E, F]}{\Gamma \vdash M[P] : [F, G]}
\end{array}$$

$$\begin{array}{c}
\text{(SUBSUM PROC)} \\
\frac{\Gamma \vdash P : T \quad T \leq T'}{\Gamma \vdash P : T'}
\end{array}
\qquad
\begin{array}{c}
\text{(REPLICATION)} \\
\frac{\Gamma \vdash P : [E, F]}{\Gamma \vdash !P : [E, F]}
\end{array}$$

(DEAD), (NEW), (PARALLEL), (REPLICATION) and the subsumption rule are standard². In the (PREFIX) rule, the typing of the capability M ensures, via the (IN), (OUT), and (PATH) rules introduced earlier, that each of the ambients being traversed as a result of exercising M have local exchanges of type compatible with the upward exchanges of the current ambient.

The rule (AMB) establishes the constraints that must be satisfied by P to be enclosed in M : specifically, the exchanges declared for M must have the same types E and F as the exchanges of P . In fact, P could be locally silent, and the typing of $M[P]$ be derivable from $\Gamma \vdash P : [\text{shh}, F]$ by subsumption. In addition, if $\Gamma \vdash M : \text{Amb}[E, \text{shh}]$, and $\Gamma \vdash P : [E, \text{shh}]$, then by (AMB) and subsumption one derives $\Gamma \vdash M[P] : \text{Amb}[F, G]$ for any F and G , as the rule imposes no constraint on the upward exchanges of the process $M[P]$.

In the following communication rules we write $\Gamma, \tilde{x} : \tilde{W} \vdash P : T$ as a shorthand for $\Gamma, x_1 : W_1, \dots, x_k : W_k \vdash P : T$.

$$\begin{array}{c} \text{(INPUT } \star) \\ \Gamma, \tilde{x} : \tilde{W} \vdash P : [W_1 \times \dots \times W_k, E] \\ \hline \Gamma \vdash (\tilde{x} : \tilde{W})P : [W_1 \times \dots \times W_k, E] \end{array} \qquad \begin{array}{c} \text{(OUTPUT } \star) \quad i = 1, \dots, k \\ \Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [W_1 \times \dots \times W_k, E] \\ \hline \Gamma \vdash \langle M_1, \dots, M_k \rangle P : [W_1 \times \dots \times W_k, E] \end{array}$$

$$\begin{array}{c} \text{(INPUT } M) \\ \Gamma \vdash M : \text{Amb}[W_1 \times \dots \times W_k, E] \quad \Gamma, \tilde{x} : \tilde{W} \vdash P : T \\ \hline \Gamma \vdash (\tilde{x} : \tilde{W})^M P : T \end{array}$$

$$\begin{array}{c} \text{(OUTPUT } M) \\ \Gamma \vdash M : \text{Amb}[W_1 \times \dots \times W_k, E] \quad \Gamma \vdash N_i : W_i \quad i = 1, \dots, k \quad \Gamma \vdash P : T \\ \hline \Gamma \vdash \langle N_1, \dots, N_k \rangle^M P : T \end{array}$$

$$\begin{array}{c} \text{(INPUT } \uparrow) \\ \Gamma, \tilde{x} : \tilde{W} \vdash P : [E, W_1 \times \dots \times W_k] \\ \hline \Gamma \vdash (\tilde{x} : \tilde{W})^\uparrow P : [E, W_1 \times \dots \times W_k] \end{array} \qquad \begin{array}{c} \text{(OUTPUT } \uparrow) \quad i = 1, \dots, k \\ \Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [E, W_1 \times \dots \times W_k] \\ \hline \Gamma \vdash \langle M_1, \dots, M_k \rangle^\uparrow P : [E, W_1 \times \dots \times W_k] \end{array}$$

²The reason why in (PARALLEL) and (REPLICATION) we used $[E, F]$ rather than T will become clear in Section 2.6

The rules for input/output are not surprising. In all cases, the type of the exchange must comply with the local exchange type of the target ambient, as expected. Also note that input/output exchanges with children, in the rules (INPUT M) and (OUTPUT M), do not impose any constraint on local and upward exchanges.

As we noted earlier, type soundness requires that subtyping between upward silent and upward non-silent processes be disallowed. To see why, consider for example allowing the relation $[E, \text{shh}] \leq [E, F]$, implying that upward-silent processes may be subsumed to non-silent processes with any upward exchange type F . While this form of subsumption seems reasonable, it is unsound in the presence of parallel composition. Consider the ambient $a[\text{in } b.\mathbf{0} \mid \langle M \rangle^\dagger P]$ with, say $M : W$ for some type W , and note that $\text{in } b.\mathbf{0}$ can be typed as $[\text{shh}, \text{shh}]$ regardless of the type of b . If the suggested subtyping were available, then the parallel composition could be typed as $[\text{shh}, W]$. However, if $b : \text{Amb}[W', F]$ for some $W' \neq W$, the ambient a could move into b and have unsound upward exchanges after the move. By forbidding subtyping on the upper component of process types, instead, the types that can be deduced for the process in $b.\mathbf{0}$ above may only be of the form $[E, W']$ or $[E, \text{shh}]$ for some exchange E .

The type system rules ensures that communication inside and across ambients never leads to type mismatches. The latter result is a consequence of the subject reduction property stated next.

Theorem 2.4.2 (Subject Reduction). *If $\Gamma \vdash P : T$ and $P \rightarrow Q$, then $\Gamma \vdash Q : T$.*

Proof. A corollary of Theorem 2.6.4. □

2.5 Mobile Ambients versus Boxed Ambients

Having defined the type system, we now look at the impact of typing on mobility and communication, and contrast it with mobility and communication in Mobile Ambients.

We already remarked that mobility is orthogonal to the local exchanges within ambients. Thus, the types of the local exchanges of an ambient do not affect the ambient's capability to move. On the other hand, the presence of upward exchanges does enforce somewhat severe constraints over ambient mobility. Specifically, ambients with upward exchanges of type W may only traverse ambients whose local exchanges have type W .

However, when we compare the flexibility of mobility and communication in Boxed Ambients versus the corresponding constructs provided by Mobile Ambients, we find that the two calculi are essentially equivalent. We study this relationship in the rest of this section. We start by sketching a translation of BA into MA. The existence of such a translation should not come as a surprise: by allowing ambient dissolution, the open capability is, at least in principle, powerful enough to code communication across boundaries (indeed, in Section 2.1 -and in Chapter 3- we argued that this ability to dissolve boundaries is too powerful and hard to control, and we have introduced communication across boundaries to dispense with it). However, when we look at the translation more closely, we find a number of subtle problems, for which we were able to find only partial solutions. As we shall see, a more satisfactory encoding can be found for the asynchronous version of BA (cf. Section 2.7).

2.5.1 From boxed to mobile ambients

The idea of the translation draws on Gonthier's coalescing encoding of the π -calculus from [CG99b]. We represent each BA ambient with a corresponding MA ambient, provide the latter with the local buffer ch : $\langle\langle n[P] \rangle\rangle = n[\text{ch}[\text{!open pk}] \mid \langle\langle P \rangle\rangle]$, where $\langle\langle P \rangle\rangle$ is the translation of P . The buffer opens a packet pk which is intended to carry input or output. The names ch and pk are “well-known”, i.e. they are global reserved names which are used uniformly³ for all ambients (and the top level). This yields a translation in which the buffer ch is used to implement all the local and non-local exchanges involving n in the source term. Specifically, a local input operation within an ambient n generates an input packet that enters the buffer associated with n , reads an input after having been opened, then creates a return packet that exits the buffer and continues with the rest of the process:

$$\langle\langle (x)P \rangle\rangle_n = (\nu k)(\text{pk}[\text{in ch.}(x)k[\text{out ch.}\langle\langle P \rangle\rangle_n] \mid \text{open } k])$$

The behavior of an output operation is captured in the same way. The idea carries over directly to the case of downward exchanges. It only requires a two-step move for

³This uniform use of the same name is problematic in extending the translation to the typed cases. The problem is easily solved, however, by choosing names indexed by the types of their local exchanges (cf. Section 2.7.1).

the packets: first into the ambient, then into associated buffer. The case of upward communication is similar. In this case, however, we need to know the name n of the enclosing ambient.

$$\llbracket (x)^\dagger P \rrbracket_n = (\nu k)(\text{pk}[\text{out } n.\text{in } \text{ch}.(x)k[\text{out } \text{ch}.\text{in } n.\llbracket P \rrbracket_n] \mid \text{open } k])$$

$$\llbracket (x)^m P \rrbracket_n = (\nu k)(\text{pk}[\text{in } m.\text{in } \text{ch}.(x)k[\text{out } \text{ch}.\text{out } m.\llbracket P \rrbracket_n] \mid \text{open } k])$$

A problem with this encoding is that the translation allows synchronizations that are not possible in the source term: specifically, the translation of $(x)^n P \mid n[n[\langle q \rangle^\dagger Q]]$ has a reduction to the translation of $P\{x := q\} \mid n[n[Q]]$.

We have not been able to find encodings that solve this problem: in fact, for this and other encodings we have investigated, the inherent non-determinism of the reductions for value exchanged, combined with a synchronous semantics, appear to require a choice operator to be modeled in a satisfactory way in MA. Of course we do not exclude that a satisfactory encoding for the synchronous calculus can be found. On the other hand, the problems we outlined do suggest that the new form of communication is computationally interesting in itself, irrespective of its import on security.

2.5.2 From mobile to boxed ambients

Because of the presence of **open** in MA, a translation of Mobile Ambients in our calculus appears problematic, if at all possible. Nevertheless, we may still argue that typed communication and mobility in MA are captured with essentially no loss of expressive power in BA. To see that, it is instructive to note that the type system of Section 2.4 section can be specialized to only allow upward-silent ambient types of the form $\text{Amb}[E, \text{shh}]$, thus effectively inhibiting all forms of upward exchanges (this follows from the format of the (AMB) rule). The specialized type system provides full flexibility for mobility, while still allowing flexible forms of communication. In particular:

- ▷ *Mobility for Boxed Ambients is as flexible as in/out-mobility for typed Mobile Ambients.* This follows by the format of the (IN) and (OUT) rules. Capabilities exercised within upward silent ambients have type $\text{Cap}[\text{shh}]$, and $\text{shh} \leq F$ for every F : consequently, upward silent ambients have full freedom of moving across ambient boundaries. Furthermore, since Boxed Ambients may not be opened, they may move

regardless of the local exchanges of the ambients they traverse. As a consequence, with the specialized type system, an ambient can move independently of its type, and of the type of its (intermediate and final) destinations.

- ▷ *Communication is as flexible as in the Ambient Calculus, even in the absence of upward exchanges.* “Upward silent” does not imply “non-communicating”: an upward-silent ambient may very well move to a target ambient a , and communicate with it by means of downward reads and writes by a itself. Indeed, an ambient may access all of its children’s anonymous channels as well as those of any incoming ambient, and all these exchanges may be of different types. In addition, the ambient may hold local exchanges of yet a different type. The encoding of channels given § 2.3.2 can also be used for encoding local exchanges of different types: the ambient $c[!(x:W)\langle x \rangle]$ can be viewed as a local channel c of type W , whose input output operators are $(x:W)^c$ and $\langle M \rangle^c$: the type system allows (encoded) channels of different types to be used in the same ambient.

Having illustrated the flexibility of the specialized type system, it is obvious that giving up upward exchanges is a problem: for instance, we would not be able to type-check pilot ambients, such as those used in the encoding of the channeled communications of § 2.3.2, whose function is to silently carry a process to a certain destination where the process eventually delivers its output to and/or receives input from its enclosing context. We solve the problem in the next section, where we study a refined type system that supports a more flexible, and type safe, integration of upward communication and mobility.

2.6 Moded Typing

The typing technique we develop in this section is based on a refinement of the observation we just made of the specialized type system, namely that ambients enclosing upward-silent processes may safely move across other ambients, regardless of the types of the latter. The new type system uses type modifiers to characterize the computation progress of processes, and in particular, to identify the silent and non-silent phases in the computation of the processes enclosed within ambients: based on that, it enhances the typing of mobility during the ambients’ silent phases.

2.6.1 Moded Types

The new type system is built around the classes of process and expression types of the previous section (henceforth *regular* types) extended by new *moded* types as defined below:

$$\text{Process Types} \quad T ::= [E, F] \mid [E, \bullet F] \mid [E, \circ F] \mid [E, \Delta F]$$

$$\text{Expression Types} \quad W ::= \text{Amb}[E, F] \mid \text{Amb}^\circ[E, F] \mid \text{Cap}[E] \mid W_1 \times \cdots \times W_n$$

Ambient types of the form $\text{Amb}[E, F]$ are exactly as in § 2.4, and their enclosed processes have “regular” process types $[E, F]$, deduced by the same rules. On the other hand, ambient types of the form $\text{Amb}^\circ[E, F]$ are associated with “taxi” ambients, whose enclosed processes are assigned moded types, according to the following intuitions:

$[E, \bullet W]$: upward silent processes with local exchanges of type E . The type W signals that processes with this type may be safely run in parallel with processes with upward exchanges of type W .

$[E, \circ W]$: processes with local exchanges of type E and upward exchanges of type W . The upward exchanges are temporarily inactive since the process is moving.

$[E, \Delta W]$: processes with local exchanges of type E and that, after performing upward exchanges of type W , evolve into processes of type $[E, \circ W]$ or $[E, \Delta W]$.

The syntax allows the formation of process types of the form $[E, \bullet \text{shh}]$, $[E, \circ \text{shh}]$ and $[E, \Delta \text{shh}]$: even though these types do not fit the above intuitions, and could safely be dispensed with, they are convenient in stating definitions and typing rules. To make sense of them, we stipulate that $\bullet \text{shh} = \circ \text{shh} = \Delta \text{shh} = \text{shh}$.

The following notation and conventions are assumed throughout: $^\mu F$ denotes any of the exchanges $\Delta F, \bullet F, \circ F$, while $^? F$ denotes either $^\mu F$ or F . When occurring in definitions and typing rules, the notations $^\mu F$ and $^? F$ are intended to be used uniformly (i.e. all the occurrences of μ and $?$ in a rule or in a definition denote the same symbol, unless otherwise stated).

The terms of the language are as before, with the only exception that the new ambient type Amb° can occur in the terms (it can be used to type bound names and variables). To exemplify moded types, consider the following process, where we assume $\Gamma \vdash M : W$.

$$(x : W') \langle x \rangle^m \mid \text{in } n. \langle M \rangle^\uparrow. \text{out } n : [W', \circ W].$$

The left component of this process does not have upward exchanges, hence it can be assigned the type $[W', \bullet W]$ provided, of course, that $m : \text{Amb}[W', E]$ for some E . On the other hand, the right component does have upward exchanges, but is currently silent because the output prefix is blocked by the move: thus in $n.\langle M \rangle^\uparrow.\text{out } n : [W', \circ W]$, provided that $n : \text{Amb}[W, W]$. The type $[W', \circ W]$ can also be assigned to the parallel composition which is, in fact, currently silent. Interestingly, the type $[W', \circ W]$ can *not* be assigned to the continuation process $\langle M \rangle^\uparrow.\text{out } n$ (nor to the parallel composition $(x:W')\langle x \rangle^m \mid \langle M \rangle^\uparrow.\text{out } n$), because, after consuming the capability in b , the upward exchanges of this process are active. At this stage, a legal type for the process is $[W', \Delta W]$, signaling, that after the upward exchange, the process enters again an upward-silent phase.

As the example shows, processes that are subject to moded typing may have different types at different stages of their computation. This does not break subject reduction, as it would seem, as reductions involving the consumption of capabilities only involve the ambients enclosing the capabilities being consumed: as a consequence, while the process enclosed in an ambient changes its type according to the process' progress, the type of the ambient itself is invariant through reduction.

The reader may wonder whether the new class of taxi ambients is really necessary, and why the same effect can not be obtained by solely relying on “regular” ambient types. The problem is that mobility may not be controlled by moded typing: in particular, moded types can not be employed to prevent non-silent ambients to exit their parent during the upward-silent phases of the latter. To see the problem, assume that ambient, say a , is currently silent and moving across ambients with local exchanges, say W . Also assume that a contains a non-silent ambient b with upward exchanges of type W' incompatible with W . As long as b is enclosed in a , its upward exchanges do not interfere with the local exchanges W of the ambients traversed by a . But if b exits a , then its upward exchanges may cause a type mismatch. In our system⁴, the problem is solved by providing guarantees that taxi ambients can only be exited by (regular or taxi) ambients which have no upward exchanges.

⁴A different solution would be possible by extending the calculus with co-capabilities à la *Safe Ambients* [LS00]. In that case, an ambient would be in a silent phase when its enclosed process does not perform upward exchanges and does not offer a *co-out* capability for nested ambients to exit.

2.6.2 Subtyping

The modes associated to the new class of process induce a richer subtype structure for process types.

Definition 2.6.1 (*Process Subtyping*).

Let \leq denote the same relation of exchange subtyping of Definition 2.4.1. Process subtyping is the smallest reflexive and transitive relation such that $[\mathbf{shh}, {}^\mu F] \leq [E, {}^\mu F]$ and in addition, satisfies the following diagram for all E and F . \square

$$\begin{array}{ccc}
 & [E, {}^\Delta F] & \\
 \nearrow & & \nwarrow \\
 [E, F] & & [E, {}^\circ F] \\
 \nwarrow & & \nearrow \\
 & [E, {}^\bullet F] &
 \end{array}$$

The intuition underlying process subtyping is as follows. As we said, the type $[-, {}^\bullet E]$ identifies upward-silent processes that move their enclosing ambient only through locations with local exchanges of type E . Clearly, any such process can always be considered as a process of type $[-, E]$ that is, as a process whose all upward exchanges are of type E and that moves the enclosing ambient only through locations with local communications of type E . In fact, it can also be considered as a process of type $[-, {}^\circ E]$, that is as a temporary upward-silent process that guarantees its enclosing ambient that whenever it performs an upward communication it will be in a context with local exchanges of type E . The two types $[-, E]$ and $[-, {}^\circ E]$ are incompatible, as processes of the first type may not be assumed to be (even temporary) upward-silent, while processes of the second type may move across ambients regardless of the types of the latter and therefore across ambients whose local exchanges are of a type different from E . Nevertheless, the two types have a common supertype $[-, {}^\Delta E]$, as this type identifies processes that may be currently upward-active, and whose enclosing ambients are guaranteed to reside in contexts with local exchanges of type E .

2.6.3 Moded Judgments and Typing Rules

The additional expressive power of the new type system results from a more flexible typing of capabilities, based on the modes associated with processes. Capabilities are typed in two modes: a “regular” mode, as in the type system of the previous section, and a “silent” mode in which some of the constraints on mobility can be lifted without consequences on safety.

The silent mode for capabilities is accounted for by a new form of judgment, denoted by $\Gamma \vdash^\circ M : \text{Cap}[E]$, which is useful when typing capability paths: if typed in silent mode, every intermediate move on the path may safely disregard the type of the ambient traversed along the move.

The new type system (see § 2.9 at the end of the chapter for the complete set of typing rules) includes all the typing rules from § 2.4.2 and new rules for deriving silent typings of capabilities and moded types for processes.

Typing for Expressions. The key rules that characterize moded ambients are those that govern mobility into and out from a moded ambient:

$$\begin{array}{c} \text{(IN } \circ) \\ \frac{\Gamma \vdash M : \text{Amb}^\circ[F, E] \quad F' \leq F}{\Gamma \vdash \text{in } M : \text{Cap}[F']} \end{array} \qquad \begin{array}{c} \text{(OUT } \circ) \\ \frac{\Gamma \vdash M : \text{Amb}^\circ[E, F]}{\Gamma \vdash \text{out } M : \text{Cap}[\text{shh}]} \end{array}$$

Note that while there is no constraint for entering a moded ambient as the rule (IN \circ) imposes exactly the same restrictions as the rule (IN)—the rule (OUT \circ) requires that if M is moded, then $\text{out } M$ can only be exercised in ambients that are upward silent.

The next rules are those that relate and differentiate \vdash° from \vdash .

$$\begin{array}{c} \text{(POLYCAP)} \\ \frac{\Gamma \vdash M : \text{Cap}[E]}{\Gamma \vdash^\circ M : \text{Cap}[E]} \end{array} \qquad \begin{array}{c} \text{(POLYPATH)} \\ \frac{\Gamma \vdash^\circ M_1 : \text{Cap}[E_1] \quad \Gamma \vdash^\circ M_2 : \text{Cap}[E_2]}{\Gamma \vdash^\circ M_1.M_2 : \text{Cap}[E_2]} \end{array}$$

By (POLYCAP) well typed capabilities type checks also under silent typing. In addition, for capability paths—that is, for sequences of **in** and **out** moves—we have the special, and more flexible rule (POLYPATH) stating that we may disregard intermediate steps, as no communication takes place during those steps: we only need to trace precise information on the last move on the path. This effectively corresponds to interpreting $\text{Cap}[E]$ as the type of capability paths whose *last* move requires upward exchanges of type E .

Silent typing of capabilities helps derive moded process types for prefixed processes as illustrated by the rules below⁵.

Typing of Processes

As we said, the new type system includes all the typing rules for processes in § 2.4.2. In addition, we have the following rules. We start with the typing of prefixes.

$$\begin{array}{c}
 \text{(PREFIX } \circ) \\
 \frac{\Gamma \vdash \circ M : \text{Cap}[G] \quad \Gamma \vdash P : [E, \circ F]}{\Gamma \vdash M.P : [E, \circ F]}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(PREFIX } \Delta) \\
 \frac{\Gamma \vdash \circ M : \text{Cap}[F] \quad \Gamma \vdash P : [E, \Delta F]}{\Gamma \vdash M.P : [E, \circ F]}
 \end{array}$$

$$\begin{array}{c}
 \text{(PREFIX } \bullet) \\
 \frac{\Gamma \vdash M : \text{Cap}[F] \quad \Gamma \vdash P : [E, \bullet F]}{\Gamma \vdash M.P : [E, \bullet F]}
 \end{array}$$

(PREFIX \circ) and (PREFIX Δ) state that prefixing a process P with a move capability always yields “moving” types, that is types with mode \circ . In particular, (PREFIX \circ) says that we may disregard the type of M (as long as M is a capability) if P is also a moving process.⁶ This rule has the same rationale as the (POLYPATH) rule above: both rules are necessary for subject congruence —specifically, for the congruence rule $(M_1.M_2).P \equiv M_1.(M_2.P)$. On the other hand, by (PREFIX Δ), the upward exchanges of M and P must be consistent (equal) when P is not moving. In other words, the *last* move of the prefix must be compatible with the upward exchanges that the process will have right after. Notice, to this regard, that by subsumption, (PREFIX Δ) also accounts for the case of prefixing a process P of type $[E, F]$.

The rule (PREFIX \bullet) types silent processes running in a context whose upward exchanges (if any) have type F . In this case, the type of the path M in the premise guarantees that P is type compatible with the local exchanges of the ambients hit on the move. Hence the typing of the capability must be “standard”, as in the (PREFIX) rule from § 2.4.

The next two rules apply to parallel compositions.

⁵ The reader may wonder why we introduced a new turnstile symbol rather than adding a mode to capabilities types, as in Cap° . Adding a moded capability type would indeed result in a slightly more expressive system (one which would allow “moded” paths to be communicated). On the other hand, the current solution has the advantage of requiring minimal changes to the syntax of expression types, thus to the typed syntax.

⁶This characterization is possible because our syntax does not include the empty path.

$$\begin{array}{c}
\text{(PARALLEL } \mu \text{ LEFT)} \\
\frac{\Gamma \vdash P : [E, {}^\mu W] \quad \Gamma \vdash Q : [E, {}^\bullet W]}{\Gamma \vdash P \mid Q : [E, {}^\mu W]}
\end{array}
\qquad
\begin{array}{c}
\text{(PARALLEL } \mu \text{ RIGHT)} \\
\frac{\Gamma \vdash P : [E, {}^\bullet W] \quad \Gamma \vdash Q : [E, {}^\mu W]}{\Gamma \vdash P \mid Q : [E, {}^\mu W]}
\end{array}$$

Two rules, and an appeal to subsumption, suffice to capture all cases. If P and Q are upward-silent (i.e. with upward exchanges ${}^\bullet W$), then $P \mid Q$ is also upward silent (with upward exchanges ${}^\bullet W$). $P \mid Q$ can be typed as moving (that is, with upward exchanges ${}^\circ W$), only when (i) either P or Q is moving and (ii) the other process is upward silent and type compatible with the exchanges of the moving process. The same reasoning applies when $P \mid Q : [E, {}^\Delta W]$, i.e. when $P \mid Q$ perform some upward exchange and then eventually move, hence the types $[E, {}^\Delta W]$ are derived with the same rules. We need two rules because we have to handle the two cases when the moving subprocess is P or Q .

The rules (DEAD) and (NEW) from § 2.4 handle also the cases for moded types (of course, save the fact that now T ranges over the extended class of process types). This is not true of the rule (REPL). In fact, if P and Q are both moving, then $P \mid Q$ may not be typed as moving, as either of the two could start its upward exchanges before the other. For this reason, there is no way to type a replicated process as a moving process: the only two possible types for a replicated process are a “regular” type (deduced by the rule REPL from § 2.4) or a silent type, as stated by the following new rule ⁷:

$$\begin{array}{c}
\text{(REPL } \bullet) \\
\frac{\Gamma \vdash P : [E, {}^\bullet F]}{\Gamma \vdash !P : [E, {}^\bullet F]}
\end{array}$$

For processes of the form $M[P]$, we need new rules. The rule (AMB) from § 2.4 is modified so that it now deduces an upward-silent type, compatible with all the other modes. Two new rules handle the case when M is a transport ambient, distinguishing the cases when the enclosed process is moving or not.

$$\begin{array}{c}
\text{(AMB)} \\
\frac{\Gamma \vdash M : \text{Amb}[E, F] \quad \Gamma \vdash P : [E, F]}{\Gamma \vdash M[P] : [F, {}^\bullet H]}
\end{array}
\qquad
\begin{array}{c}
\text{(AMB } \Delta) \\
\frac{\Gamma \vdash M : \text{Amb}^\circ[E, F] \quad \Gamma \vdash P : [E, {}^\Delta F]}{\Gamma \vdash M[P] : [F, {}^\bullet H]}
\end{array}$$

⁷This is due to the particular semantics of replication we use, which yields to an unrestrained generation of copies. It is clear that the use of guarded replication or call by need would make replication compatible with moded types (see also next section).

$$\begin{array}{c}
(\text{AMB } \circ) \\
\frac{\Gamma \vdash M : \text{Amb}^\circ[E, F] \quad \Gamma \vdash P : [E, {}^\circ F]}{\Gamma \vdash M[P] : [G, {}^\bullet H]}
\end{array}$$

In $(\text{AMB } \Delta)$ P is not moving, and the rule imposes type constraints equivalent to those imposed by the (AMB) rule: note, in fact, that the judgment $\Gamma \vdash P : [E, {}^\Delta F]$ could be derived by subsumption from $\Gamma \vdash P : [E, F]$. If, instead, P is moving, as in $(\text{AMB } \circ)$, its upward exchanges are blocked by the move, and we have freedom to chose the type of local exchanges of the process $M[P]$. Once again, subject reduction does not break if exercising the capability in P activates upward exchanges: $(\text{AMB } \Delta)$ can be used to type the reductum

We conclude with the rules for input-output.

$$\begin{array}{c}
(\text{INPUT } \star \mu) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [W_1 \times \cdots \times W_k, {}^\mu F]}{\Gamma \vdash (\tilde{x} : \tilde{W})P : [W_1 \times \cdots \times W_k, {}^\mu F]}
\end{array}
\qquad
\begin{array}{c}
(\text{OUTPUT } \star \mu) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [W_1 \times \cdots \times W_k, {}^\mu F]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle P : [W_1 \times \cdots \times W_k, {}^\mu F]}
\end{array}$$

$$\begin{array}{c}
(\text{INPUT } \uparrow \Delta) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [F, {}^\Delta W_1 \times \cdots \times W_k, {}^\bullet]}{\Gamma \vdash (\tilde{x} : \tilde{W})^\uparrow P : [F, {}^\Delta W_1 \times \cdots \times W_k, {}^\bullet]}
\end{array}
\qquad
\begin{array}{c}
(\text{OUTPUT } \uparrow \Delta) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [F, {}^\Delta W_1 \times \cdots \times W_k, {}^\bullet]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^\uparrow P : [F, {}^\Delta W_1 \times \cdots \times W_k, {}^\bullet]}
\end{array}$$

Local communications are not affected by modes: it is the mode of the continuation process that determines the moded type of the input/output process itself.

Upward exchanges have only non-moving types, for obvious reasons. The particular type they have—that is, either $[F, {}^\Delta W]$ or the more informative $[F, W]$ —depends on the type of their continuation. If their continuation is of type $[F, {}^\bullet W]$ or $[F, W]$, then the process—which is clearly not silent—can be typed as $[F, W]$. These cases are captured by the rule $(\text{INPUT/OUTPUT } \uparrow)$ of § 2.4 (together with subsumption for the case $[F, {}^\bullet W]$). If instead the continuation has type $[F, {}^\Delta W]$ or $[F, {}^\circ W]$, as in $(\text{INPUT/OUTPUT } \uparrow \Delta)$, we can just say that the process may eventually evolve into a moving process, hence the type $[F, {}^\Delta W]$ in the conclusion.

Finally, downward communications are not affected by whether the target ambient is moded or not. The rules from § 2.4 work just as well for the new system: two new rules,

with the same format, handle the case when target ambient is moded:

$$\frac{\text{(INPUT } M \circ) \quad \Gamma \vdash M : \mathbf{Amb}^\circ[W_1 \times \dots \times W_k, E] \quad \Gamma, \tilde{x} : \tilde{W} \vdash P : T}{\Gamma \vdash (\tilde{x} : \tilde{W})^M P : T}$$

$$\frac{\text{(OUTPUT } M \circ) \quad \Gamma \vdash M : \mathbf{Amb}^\circ[W_1 \times \dots \times W_k, E] \quad \Gamma \vdash N_i : W_i \quad i = 1, \dots, k \quad \Gamma \vdash P : T}{\Gamma \vdash \langle N_1, \dots, N_k \rangle^M P : T}$$

Note that in all output rules, the typing of the expression M being output is subject to “regular” typing. As a consequence, capability paths may be communicated only if well-typed under regular typing. This restriction could be lifted, had we employed moded capability types as suggested in § 2.6.3 (cf. footnote 5), but with no significant additional expressive power.

2.6.4 Subject Reduction

The subject reduction theorem for moded typing is proved following the standard technique, that we only sketch here (a detailed proof is in Section 3.7, where subject reduction is proved for a superset of the type system of moded types enriched with security annotations).

Lemma 2.6.2 (Substitution). *Let $\vdash^?$ denote either \vdash or \vdash° .*

- Assume $\Gamma, x : W \vdash^? M : W'$ and $\Gamma \vdash N : W$. Then $\Gamma \vdash^? M\{x := N\} : W'$.
- Assume $\Gamma, x : W \vdash P : T$ and $\Gamma \vdash N : W$. Then $\Gamma \vdash P\{x := N\} : T$.

Proof. Standard: by induction on the derivations of the two judgments $\Gamma, x : W \vdash^? M : W'$ and $\Gamma \vdash P : T$. □

Lemma 2.6.3 (Subject Congruence). *If $\Gamma \vdash P : T$ and $P \equiv Q$ then $\Gamma \vdash Q : T$.*

Proof. By simultaneous induction on the derivations of $P \equiv Q$ and $Q \equiv P$. □

Theorem 2.6.4 (Subject Reduction). *If $\Gamma \vdash P : T$ and $P \rightarrow Q$ then $\Gamma \vdash Q : T$.*

Proof. By induction on the derivation of $P \rightarrow Q$. □

We conclude this section with an example showing how moded typing help type-check the taxi ambients used in § 2.3.2 to encode communication on named channels à la Seal Calculus. In the typed case, a channel c is expressed by the ambient $c[!(x:W)\langle x \rangle]$ and the encoding is defined as follows:

$$\begin{aligned} \langle c^m \langle M \rangle \rangle_n &= (\nu p:\text{Amb}^\circ[\text{shh}, W])p[\text{in } m.\text{in } c.\langle M \rangle^\uparrow] \\ \langle c^\uparrow \langle M \rangle \rangle_n &= (\nu p:\text{Amb}^\circ[\text{shh}, W])p[\text{out } n.\text{in } c.\langle M \rangle^\uparrow] \\ \langle c^m(x:W)P \rangle_n &= (\nu p:\text{Amb}^\circ[W, W])p[\text{in } m.\text{in } c.(x:W)^\uparrow \text{out } c.\text{out } m.\langle x \rangle] \mid (x:W)^p \langle P \rangle_n \\ \langle c^\uparrow(x:W)P \rangle_n &= (\nu p:\text{Amb}^\circ[W, W])p[\text{out } n.\text{in } c.(x:W)^\uparrow \text{out } c.\text{in } n.\langle x \rangle] \mid (x:W)^p \langle P \rangle_n \end{aligned}$$

The definition is stated independently of the types of the two ambients m and n traversed by the ambient p : this flexibility is enabled by the typing of p as a taxi ambient.

To illustrate the type system, we give a type derivation for the case of downward input on a channel of type W , as in $c^m(x:W)P$, as representative. With no loss of generality we make the following assumptions: $\Gamma \vdash \langle P \rangle_n : [E, ?F]$, with Γ a type environment in which $m:\text{Amb}^?[G, H]$, $c:\text{Amb}[W, \text{shh}]$ and $p:\text{Amb}^\circ[W, W]$, and E, F, G, H are arbitrary exchange types. The fact that the ambient p is typed as a taxi ambient is essential for the typed encoding to type-check. This is shown by the following analysis that also illustrate the interplay between the modes \circ and Δ .

A type derivation for the encoding of downward input derives routinely from the judgment:

$$\Gamma \vdash p[\text{in } m.\text{in } c.(x:W)^\uparrow \text{out } c.\text{out } m.\langle x \rangle] : [E, \bullet F]$$

This judgment may be derived by the rule (AMB \circ), provided that the process enclosed in p can be typed with mode \circ , that is, if

$$\Gamma \vdash \text{in } m.\text{in } c.(x:W)^\uparrow \text{out } c.\text{out } m.\langle x \rangle : [W, \circ W].$$

This follows by (PREFIX \circ) from $\Gamma \vdash \text{in } m:\text{Cap}[G]$ and

$$\Gamma \vdash \text{in } c.(x:W)^\uparrow \text{out } c.\text{out } m.\langle x \rangle : [W, \circ W]$$

G is the type of the local exchanges in m , and moded typing allows G to be any type, irrespective of W . The last judgment follows again by (PREFIX Δ) from $\Gamma \vdash \text{in } c:\text{Cap}[W]$ and from

$$\Gamma \vdash (x:W)^\uparrow \text{out } c.\text{out } m.\langle x \rangle : [W, \Delta W],$$

This judgment can be derived by $(\text{INPUT} \uparrow \Delta)$ from

$$\Gamma, x:W \vdash \text{out } c.\text{out } m.\langle x \rangle : [W, {}^\Delta W].$$

Again, we rely on moded typing: the whole process type-checks since the move that precedes the upward output brings the ambient in an environment with the right exchange type. Deriving the last judgment is not difficult. From $\Gamma, x:W \vdash^\circ \text{out } m:\text{Cap}[H]$ and from $\Gamma, x:W \vdash \langle x \rangle : [W, {}^\circ W]$, we have $\Gamma, x:W \vdash \text{out } m.\langle x \rangle : [W, {}^\circ W]$. Now, from the last judgment and from $\Gamma, x:W \vdash^\circ \text{out } c:\text{Cap}[\text{shh}]$ an application of $(\text{PREFIX } \circ)$ yields $\Gamma, x:W \vdash \text{out } c.\text{out } m.\langle x \rangle : [W, {}^\circ W]$ as desired. To conclude, we can apply subsumption, based on the subtyping $[W, {}^\circ W] \leq [W, {}^\Delta W]$, and then $(\text{INPUT} \uparrow \Delta)$ to obtain the desired typing.

2.7 Asynchronous communications

In the previous section we studied various aspects of the synchronous version of the calculus. Now we look at different ways of introducing forms of asynchronous communication. Asynchrony in a calculus for mobile and distributed computation is motivated by widely agreed design principles and practical experience. As noted in [Car99], mobile and distributed computation can hardly rely on synchronous input-output as the only mechanism of communication. Also, experience with implementation of distributed calculi [BV02, FLA00] shows that the form of consensus required for synchronous communication is quite hard to implement in a distributed environment.

In § 2.2 we said that asynchronous communication can be recovered in our calculus in two possible ways: (i) either by coding it with synchronous output and null continuations, or (ii) by introducing the additional equivalence $\langle M \rangle^\eta P \equiv \langle M \rangle^\eta \mid P$. The first solution allows synchronous and asynchronous output to coexist. An asynchronous output-prefix $\langle M \rangle^\eta$ followed by a continuation P can be expressed in terms of synchronous output by the parallel composition $\langle M \rangle^\eta \mathbf{0} \mid P$. The second solution takes this idea to its extreme, and leads to a purely asynchronous calculus.

Neither alternative is entirely satisfactory. One problem with the first is that $\langle M \rangle^\eta P$ and $\langle M \rangle^\eta \mathbf{0} \mid P$ are only equivalent under the type system of § 2.4, not with moded types. In fact, for $\eta = \uparrow$, it is not difficult to find situations where $\langle M \rangle^\eta P$ is well-typed and

$\langle M \rangle^{\circ} \mathbf{0} \mid P$ is not (with moded typing)⁸. An immediate consequence of this observation is that the congruence law $\langle M \rangle^{\uparrow} P \equiv \langle M \rangle^{\uparrow} \mid P$ is not preserved by moded typing, hence the second alternative is not sound for the system of § 2.6.

A further reason for being unsatisfied with the first solution is that the use of null continuations to code asynchronous output has the effect of essentially defeating moded typing. Moded typing is possible, and effective, only along a single thread, while the coding of asynchronous output introduces parallel compositions and leaves no residual following an output. Notice, however, that the problem is not a consequence of moded typing and asynchrony being inherently incompatible. To see that, observe that in $\langle M \rangle^{\uparrow} P$ the continuation P could be typed with a mode independently of whether the prefix denotes synchronous or asynchronous output. All that matters for P to receive a (sound) “moving” type is that $\langle M \rangle$ gets delivered to the parent ambient before unleashing P : once delivered, whether or not $\langle M \rangle$ also synchronizes with local input is irrelevant.

Based on this observation, a smoother integration of asynchronous output and moded typing may be achieved by re-stating the congruence law as a reduction rule, and making it location-aware so that the output is delivered to the appropriate ambient.

Different formulations of the asynchronous version of the calculus are possible. A first solution, given below, is to replace the reductions (*output* n) and (*output* \uparrow) of § 2.2 with the reductions (*asynch output* n) and (*asynch output* \uparrow) below, and to introduce the new reduction (*asynch output* \star):

$$\begin{array}{ll}
 (\textit{asynch output } \star) & \langle \tilde{M} \rangle P \rightarrow \langle \tilde{M} \rangle \mid P \\
 (\textit{asynch output down}) & \langle \tilde{M} \rangle^n P \rightarrow \langle \tilde{M} \rangle^n \mid P \\
 (\textit{asynch output } n) & \langle \tilde{M} \rangle^n \mid n[(\tilde{x})P \mid Q] \rightarrow n[P\{\tilde{x} := \tilde{M}\} \mid Q] \\
 (\textit{asynch output } \uparrow) & n[\langle \tilde{M} \rangle^{\uparrow} P \mid Q] \rightarrow \langle \tilde{M} \rangle \mid n[P \mid Q]
 \end{array}$$

With these reductions, the problem with moded types is solved: an upward output followed by a move, as in $\langle N \rangle^{\uparrow} M.P$ may safely be typed with mode Δ (based on the mode \circ for $M.P$) irrespective of whether the output synchronizes or not. More generally, we may

⁸For example, take the following two processes, where we assume the primitive types *int* and *bool*; the first one is well typed, while the second one is not:

$$(x : \textit{int})P \mid b[\langle 5 \rangle^{\uparrow} \textit{in } a] \mid a[(x : \textit{bool})Q] \qquad (x : \textit{int})P \mid b[\langle 5 \rangle^{\uparrow} \mid \textit{in } a] \mid a[(x : \textit{bool})Q]$$

prove that subject reduction holds for this form of asynchronous reduction and the moded type system presented in the previous section: no further modification is needed.

A second possibility, is to combine synchrony and asynchrony. Cardelli [Car00], advocates that local exchanges can be synchronous, while remote communication ought to be asynchronous. This is a sound choice for our calculus: in fact, the reduction (*asynch output* \star) for local exchanges may be dispensed with, as local asynchronous output may be coded by $\langle M \rangle \mathbf{0} \mid P$ without affecting moded typing.

A third possibility is to replace the output rules of § 2.2 by the *asynch*-output rules defined above, and the input rules with the new reductions below:

$$\begin{array}{ll} (\textit{asynch input } \star) & (\tilde{x})P \mid \langle \tilde{M} \rangle \rightarrow P\{\tilde{x} := \tilde{M}\} \\ (\textit{asynch input } n) & (\tilde{x})^n P \mid n[\langle \tilde{M} \rangle \mid Q] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q] \\ (\textit{asynch input } \uparrow) & \langle \tilde{M} \rangle \mid n[(\tilde{x})^\uparrow P \mid Q] \rightarrow n[P\{\tilde{x} := \tilde{M}\} \mid Q] \end{array}$$

This solution is similar to the first one proposed above (essentially, it only adds new intermediate reduction steps). Nevertheless, it is interesting, as it suggests a novel interpretation of the process form $\langle M \rangle$ as a *memory cell*. Indeed, one may view $\langle M \rangle \mathbf{0}$ and $\langle M \rangle$ as denoting two very distinct processes, the former being a local output with a null continuation, the latter being a memory cell (more precisely a one-place buffer)

Taking this view, every communication becomes a two-step protocol and the reductions have new interpretations. To exemplify, (*asynch output* \star) describes how a writer process $\langle M \rangle P$ writes a memory cell $\langle M \rangle$ and then continues as P ; (*asynch input* \star) shows a reader that makes a destructive access to a memory cell $\langle M \rangle$. The same reasoning applies to downward and upward exchanges. As a result, memory cells, that is the output form $\langle M \rangle$, take the role of the resources of the calculus, which are bound to their location.

Whatever solution we choose in this section, they are all compatible with the moded typing of § 2.6 and, *a fortiori*, with the type system of § 2.4.

2.7.1 Asynchronous Boxed Ambient vs Mobile Ambients

We mentioned in Section 2.5 that the asynchronous semantics enables the definition of more robust translation of Boxed Ambients in MA. The translation is defined formally in the following. To ease the presentation, we restrict to a monadic version of BA in which only names (and not capabilities) can be exchanged. There is no fundamental difficulty

in extending the definition to the general case (some care is required to handle the typed exchange of capabilities).

The translation is typed: with respect to the untyped case, the main difference is in the use of a family of names ch_W and pk , indexed on types, with the implicit assumption that $\text{ch}_W, \text{pk}_W : \text{Amb}[W]$ for all type W . This indexing is required for typing, as each pair of names enables exchanges of the corresponding type.

Types:

$$\begin{aligned} \langle \text{Amb}[W, E] \rangle &= \text{Amb}[\langle W \rangle] & \langle \text{Amb}[\text{shh}, E] \rangle &= \text{Amb}[\text{shh}] \\ \langle [W, E] \rangle &= [\langle W \rangle] & \langle [\text{shh}, E] \rangle &= \text{shh} \end{aligned}$$

Type environments: $\langle x_1 : W_1, \dots, x_n : W_n \rangle = x_1 : \langle W_1 \rangle, \dots, x_n : \langle W_n \rangle$

Terms: assume $\Gamma \vdash n : \text{Amb}[W, W']$ and $\Gamma \vdash m : \text{Amb}[W'', E]$

$$\begin{aligned} \langle \Gamma \triangleright \langle q \rangle P \rangle_n &= (\nu k : \text{Amb}[\langle W \rangle]) \\ &\quad (\text{pk}_W[\langle q \rangle \mid \text{in } \text{ch}_W.k[\text{out } \text{ch}_W.\langle \Gamma \triangleright P \rangle_n] \mid \text{open } k]) \\ \langle \Gamma \triangleright (x : W) P \rangle_n &= (\nu k : \text{Amb}[\langle W \rangle]) \\ &\quad (\text{pk}_W[\text{in } \text{ch}_W.(x : \langle W \rangle)k[\text{out } \text{ch}_W.\langle \Gamma \triangleright P \rangle_n] \mid \text{open } k]) \\ \langle \Gamma \triangleright \langle q \rangle^m P \rangle_n &= (\nu k : \text{Amb}[\langle W \rangle]) \\ &\quad (\text{pk}_{W''}[\langle q \rangle \mid \text{in } m.\text{in } \text{ch}_{W''}.k[\text{out } \text{ch}_{W''}.\text{out } m\langle \Gamma \triangleright P \rangle_n] \mid \text{open } k]) \\ \langle \Gamma \triangleright (x : W'')^m P \rangle_n &= (\nu k : \text{Amb}[\langle W \rangle]) \\ &\quad (\text{pk}_{W''}[\text{in } m.\text{in } \text{ch}_{W''}.(x : \langle W'' \rangle)k[\text{out } \text{ch}_{W''}.\text{out } m\langle \Gamma \triangleright P \rangle_n] \mid \text{open } k]) \\ \langle \Gamma \triangleright \langle q \rangle^\uparrow P \rangle_n &= (\nu k : \text{Amb}[\langle W \rangle]) \\ &\quad (\text{pk}_{W'}[\langle q \rangle \mid \text{out } n.\text{in } \text{ch}_{W'}.k[\text{out } \text{ch}_{W'}.\text{in } n.\langle \Gamma \triangleright P \rangle_n] \mid \text{open } k]) \\ \langle \Gamma \triangleright (x : W')^\uparrow P \rangle_n &= (\nu k : \text{Amb}[\langle W \rangle]) \\ &\quad (\text{pk}_{W'}[\text{out } n.\text{in } \text{ch}_{W'}.(x : \langle W' \rangle)k[\text{out } \text{ch}_{W'}.\text{in } n.\langle \Gamma \triangleright P \rangle_n] \mid \text{open } k]) \\ \langle \Gamma \triangleright m[P] \rangle_n &= m[\text{ch}_{W''}[\text{!open } \text{pk}_{W''}] \mid \langle \Gamma \triangleright P \rangle_m] \\ \langle \Gamma \triangleright P \mid Q \rangle_n &= \langle \Gamma \triangleright P \rangle_n \mid \langle \Gamma \triangleright Q \rangle_n \\ \langle \Gamma \triangleright (\nu m : W) P \rangle_n &= (\nu m : \langle W \rangle) \langle \Gamma, m : W \triangleright P \rangle_n \\ \langle \Gamma \triangleright M.P \rangle_n &= M.\langle \Gamma \triangleright P \rangle_n \\ \langle \Gamma \triangleright ! P \rangle_n &= ! \langle \Gamma \triangleright P \rangle_n \end{aligned}$$

The translation has interesting properties. It is type preserving, namely: if $\Gamma \vdash P : [E, F]$ in BA, then one can show that $\langle \Gamma \rangle \vdash \langle \Gamma \triangleright P \rangle : \langle [E, F] \rangle$. Also, the translation

simulates the reductions of the source (asynchronous) calculus correctly. Unfortunately, there are still two remaining problems. First the translation is not fully compositional, as the complete encoding needs a further step, to add a buffer to the top level: $\langle\langle \Gamma \triangleright P \rangle\rangle = \langle\langle \Gamma \triangleright P \rangle\rangle_{\text{top}} \mid \text{ch}_W[\]$ where W is the type of the local exchanges of P . Secondly, the protocols that implement the exchanges across boundaries are not atomic, and hence subject to interferences. To make them atomic, one would need further hypotheses of the source term, akin to those encompassed by the notion of *single-threadedness* defined for Safe Ambients [LS00]. For instance, for the upward exchanges to be implemented correctly, we would need to assume that the ambient n exited at the start of the protocol does not move until the ambient k is back into n (hence the protocol is complete).

2.7.2 Synchrony versus asynchrony: security trade-offs

The choice of synchronous versus asynchronous communication has other consequences on the calculus, specifically, in terms of the security guarantees that can be made for it.

On one side, it is well known that synchronous communication generates hard-to-detect information flows based on synchronization. The synchronous version also has this problem. For example, in the system $a[Q \mid b[\langle M \rangle P]]$, the sub-ambient b , gets to know exactly when (and if) Q makes a downward read access to its contents. Therefore one bit of information flowed by a read access from the reader to the writer. This makes non-interference [GM82, FG97] quite hard to satisfy (The problem of information flow in Boxed Ambients is analyzed and solved in Chapter 4).

On the other hand, by adopting asynchronous communication we effectively give up *mediation*, that is, control over interaction between sibling ambients. With synchronous input-output no ambient can be “spoiled” with unexpected (and possibly unwanted) output by its enclosing or enclosed ambients. As an example, consider the system $a[(x:W)^b P \mid b[c[\langle M \rangle^\dagger \mid Q]]]$ which is typable in our system provided that $M:W$ and the b is declared of type $\text{Amb}[W, F]$ for some F . With synchronous reductions there is no way for the upward output in c and the downward input in a to synchronize. Instead, in the asynchronous case, the initial configuration would evolve into $a[(x:W)^b P \mid b[\langle M \rangle \mid c[Q]]]$; by a further reduction step the ambient a gets hold of the message $\langle M \rangle$ without any mediation of b .

Similarly, two siblings may establish a “covert channel”: $b[a[(x:W)^\dagger P] \mid c[\langle M \rangle^\dagger Q]]$ reduces in two steps into $b[a[P\{x := M\}] \mid c[Q]]$. Both situations result in security breaches, based on the presence of covert channels, that cannot be prevented by the primitives of the calculus, as it stands. One can, however, resort to types and type analysis to provide stronger security guarantees for the calculus (e.g. non-interference), and enhanced policies for resource access control. We discuss these aspects in the next two chapters.

2.8 Related work

Besides Mobile Ambients and Seals, whose relationships with Boxed Ambients have been discussed all along, the new calculus shares the same motivations, and is superficially similar to Sewell and Vitek’s Box- π [SV00]. The technical development, however, is entirely different. We do not provide direct mechanisms for constructing *wrappers*, rather we propose a new construct for ambient interaction in the attempt to provide easier-to-monitor communications. Also, our form of communication is anonymous, and based on a notion of locality which is absent in the Box- π Calculus. This latter choice has important consequences in the formalization of classic security models as we discuss in Chapter 3. Finally Box- π does not consider mobility which is a fundamental component of this work.

Our type system is clearly also related to other typing systems developed for Mobile Ambients. In [CG99b] types guarantees absence of type confusion for communications. The type systems of [CGG99] and [Zim00] provide control over ambients moves and opening. The powerful type discipline for Safe Ambients, presented in [LS00], add a finer control over ambient interactions and remove all *grave interference*, i.e. all non-deterministic choice between logical incompatible interactions. Relying on [CGG00a], in [MS02] authors introduce an accurate type system to control ambient mobility in the BA calculus. In particular, they introduce ambient types that consist of four components: the *group* to which the ambient type belongs, a component describing who can exit the ambient and where the ambient may reside, a component that characterizes ambient communications, and a set of markers that specify what the ambient name can be used for. This type system, that also contains a non-trivial form of subtyping both on mobility and on communication, allows a satisfactory control on boundary crossing and acquisition of capabilities.

All these approaches are orthogonal to the particular communication primitives. We

believe that similar typing disciplines and mobility types (without opening control, of course), can be adapted to Boxed Ambients to obtain similar strong results.

2.9 Summary of Moded Type System

In order to have a more compact set of rules, we use ${}^\mu F$ to denote any of the exchanges ${}^\Delta F$, ${}^\bullet F$, ${}^\circ F$, and use ${}^? F$ to denote either ${}^\mu F$ or F . Similarly we use $\mathbf{Amb}^?[E, F]$ to denote either $\mathbf{Amb}[E, F]$ or $\mathbf{Amb}^\circ[E, F]$. The use of such shorthands make it possible to express the rules of § 2.4, § 2.6 as instances of the same rule.

Expressions

$$\begin{array}{c}
\text{(PROJECTION)} \quad \frac{\Gamma(a) = W}{\Gamma \vdash a : W} \quad \text{(IN)} \quad \frac{\Gamma \vdash M : \mathbf{Amb}^?[E, F] \quad G \leq E}{\Gamma \vdash \text{in } M : \mathbf{Cap}[G]} \quad \text{(OUT)} \quad \frac{\Gamma \vdash M : \mathbf{Amb}[E, F] \quad G \leq F}{\Gamma \vdash \text{out } M : \mathbf{Cap}[G]} \\[10pt]
\text{(PATH)} \quad \frac{\Gamma \vdash M_1 : \mathbf{Cap}[E] \quad \Gamma \vdash M_2 : \mathbf{Cap}[E]}{\Gamma \vdash M_1.M_2 : \mathbf{Cap}[E]} \quad \text{(OUT } \circ) \quad \frac{\Gamma \vdash M : \mathbf{Amb}^\circ[E, F]}{\Gamma \vdash \text{out } M : \mathbf{Cap}[\text{shh}]} \\[10pt]
\text{(POLYPATH)} \quad \frac{\Gamma \vdash M_1 : \mathbf{Cap}[F] \quad \Gamma \vdash M_2 : \mathbf{Cap}[E]}{\Gamma \vdash M_1.M_2 : \mathbf{Cap}[E]} \quad \text{(POLYCAP)} \quad \frac{\Gamma \vdash M : \mathbf{Cap}[E]}{\Gamma \vdash M : \mathbf{Cap}[E]}
\end{array}$$

Processes

$$\begin{array}{c}
\text{(PREFIX)} \quad \frac{\Gamma \vdash M : \mathbf{Cap}[F] \quad \Gamma \vdash P : [E, F]}{\Gamma \vdash M.P : [E, F]} \quad \text{(PREFIX } \circ) \quad \frac{\Gamma \vdash M : \mathbf{Cap}[G] \quad \Gamma \vdash P : [E, {}^\circ F]}{\Gamma \vdash M.P : [E, {}^\circ F]} \\[10pt]
\text{(PREFIX } \Delta) \quad \frac{\Gamma \vdash M : \mathbf{Cap}[F] \quad \Gamma \vdash P : [E, {}^\Delta F]}{\Gamma \vdash M.P : [E, {}^\circ F]} \quad \text{(PREFIX } \bullet) \quad \frac{\Gamma \vdash M : \mathbf{Cap}[F] \quad \Gamma \vdash P : [E, {}^\bullet F]}{\Gamma \vdash M.P : [E, {}^\bullet F]}
\end{array}$$

$$\begin{array}{c}
\text{(PAR)} \\
\frac{\Gamma \vdash P : [E, F] \quad \Gamma \vdash Q : [E, F]}{\Gamma \vdash P \mid Q : [E, F]}
\end{array}
\quad
\begin{array}{c}
\text{(PAR } \mu) \\
\frac{\Gamma \vdash P : [E, {}^\mu F] \quad \Gamma \vdash Q : [E, {}^\bullet F]}{\Gamma \vdash P \mid Q, Q \mid P : [E, {}^\mu F]}
\end{array}$$

$$\begin{array}{c}
\text{(DEAD)} \\
\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0} : T}
\end{array}
\quad
\begin{array}{c}
\text{(NEW)} \\
\frac{\Gamma, n : \mathbf{Amb}^?[E, F] \vdash P : T}{\Gamma \vdash (\nu n : \mathbf{Amb}^?[E, F])P : T}
\end{array}
\quad
\begin{array}{c}
\text{(REPL } \bullet) \\
\frac{\Gamma \vdash P : [E, {}^\bullet F]}{\Gamma \vdash !P : [E, {}^\bullet F]}
\end{array}
\quad
\begin{array}{c}
\text{(REPL)} \\
\frac{\Gamma \vdash P : [E, F]}{\Gamma \vdash !P : [E, F]}
\end{array}$$

$$\begin{array}{c}
\text{(AMB)} \\
\frac{\Gamma \vdash M : \mathbf{Amb}[E, F] \quad \Gamma \vdash P : [E, F]}{\Gamma \vdash M[P] : [F, {}^\bullet H]}
\end{array}
\quad
\begin{array}{c}
\text{(SUBSUMPTION)} \\
\frac{\Gamma \vdash P : T \quad T \leq T'}{\Gamma \vdash P : T'}
\end{array}$$

$$\begin{array}{c}
\text{(AMB } \Delta) \\
\frac{\Gamma \vdash M : \mathbf{Amb}^\Delta[E, F] \quad \Gamma \vdash P : [E, {}^\Delta F]}{\Gamma \vdash M[P] : [F, {}^\bullet H]}
\end{array}
\quad
\begin{array}{c}
\text{(AMB } \circ) \\
\frac{\Gamma \vdash M : \mathbf{Amb}^\circ[E, F] \quad \Gamma \vdash P : [E, {}^\circ F]}{\Gamma \vdash M[P] : [G, {}^\bullet H]}
\end{array}$$

In the input rules below, we use the notation $\tilde{x} : \tilde{W}$ as a shorthand for $x_1 : W_1, \dots, x_k : W_k$.

$$\begin{array}{c}
\text{(INPUT } \star) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [W_1 \times \dots \times W_k, {}^? F]}{\Gamma \vdash (\tilde{x} : \tilde{W})P : [W_1 \times \dots \times W_k, {}^? F]}
\end{array}
\quad
\begin{array}{c}
\text{(OUTPUT } \star) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [W_1 \times \dots \times W_k, {}^? F]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle P : [W_1 \times \dots \times W_k, {}^? F]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } M) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [E, {}^\mu F] \quad \Gamma \vdash M : \mathbf{Amb}^?[W_1 \times \dots \times W_k, G]}{\Gamma \vdash (\tilde{x} : \tilde{W})^M P : [E, {}^\mu F]}
\end{array}$$

$$\begin{array}{c}
\text{(OUTPUT } M) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash N_i : W_i \quad \Gamma \vdash P : [E, {}^\mu F] \quad \Gamma \vdash M : \mathbf{Amb}^?[W_1 \times \dots \times W_k, G]}{\Gamma \vdash \langle N_1, \dots, N_k \rangle^M P : [E, {}^\mu F]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } \uparrow) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [F, W_1 \times \dots \times W_k]}{\Gamma \vdash (\tilde{x} : \tilde{W})^\uparrow P : [F, W_1 \times \dots \times W_k]}
\end{array}
\quad
\begin{array}{c}
\text{(INPUT } \uparrow \Delta) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [F, {}^\Delta W_1 \times \dots \times W_k]}{\Gamma \vdash (\tilde{x} : \tilde{W})^\uparrow P : [F, {}^\Delta W_1 \times \dots \times W_k]}
\end{array}$$

$$\begin{array}{c}
\text{(OUTPUT } \uparrow) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [F, W_1 \times \dots \times W_k]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^\uparrow P : [F, W_1 \times \dots \times W_k]}
\end{array}
\quad
\begin{array}{c}
\text{(OUTPUT } \uparrow \Delta) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [F, {}^\Delta (W_1 \times \dots \times W_k)]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^\uparrow P : [F, {}^\Delta (W_1 \times \dots \times W_k)]}
\end{array}$$

For the asynchronous version of the calculus we need two additional typing rule for the process forms $\langle M_1, \dots, M_k \rangle$ and $\langle M_1, \dots, M_k \rangle^M$.

$$\begin{array}{c} \text{(ASYNCH OUTPUT)} \\ \hline \Gamma \vdash M_i : W_i \quad i = 1, \dots, k \\ \hline \Gamma \vdash \langle M_1, \dots, M_k \rangle : [W_1 \times \dots \times W_k, {}^?F] \end{array}$$

$$\begin{array}{c} \text{(ASYNCH OUTPUT } M) \quad i = 1, \dots, k \\ \hline \Gamma \vdash N_i : W_i \quad \Gamma \vdash M : \text{Amb}^?[W_1 \times \dots \times W_k, G] \\ \hline \Gamma \vdash \langle N_1, \dots, N_k \rangle^M : [E, {}^\mu F] \end{array}$$

Chapter 3

Access Control for Mobile Agents

This chapter gives an assessment of security for Mobile Ambients, with specific focus on *mandatory access control* (MAC) policies in multilevel security systems. The first part of the chapter reports on different formalization attempts for MAC policies in the Ambient Calculus, and provides an in-depth analysis of the problems one encounters. As it turns out, MAC security does not appear to have fully convincing interpretations in the calculus. The second part proposes a solution to this *impasse*, based on Boxed Ambients. A type system for resource access control is defined, and the BA calculus is discussed and illustrated with several examples of resource management policies.

3.1 Introduction

Mobile computing relies on sharing of data and software resources among computing sites distributed across wide-area open networks. This sharing is successful inasmuch as it satisfies several criteria, including safety, e.g. execution of mobile code without failure, and security, e.g. protection of sites against malicious intruders and misuse of their computing resources.

This chapter provides an assessment of security in ambient based calculi, studying what (if any) new insight and challenges mobile code languages provide for well-established security models. The focus of this chapter is on *mandatory access control* policies (MAC) in multilevel security systems. In particular, the emphasis is on the specific aspects of MAC policies related to confidentiality and integrity, and their different implementations as *military* security (no read-up, no write-down) and *commercial* security (no read-up, no write-up).

The first part of the chapter (§ 3.2) is a survey of our attempts of formalizing MAC policies in Mobile Ambients; we point out the shortcomings of MA as a formal basis for reasoning about these concepts. In fact, the main problem comes far ahead the point where one starts the formalization, because the security concepts assumed as references do not appear to have any fully convincing interpretation in the calculus. The very meaning of basis notions such as “read access” and “write access” by subjects on objects, or even “ownership”, is somehow difficult to grasp and characterize when looked at from within the Ambient Calculus. As a consequence of these difficulties, one is led to the conclusion that Ambients lack adequate primitives to capture and characterize those security concepts. While our arguments are only informal, the analysis we detail in the first part of the chapter does provide convincing evidence in favor of our conclusion.

The second part of the chapter proposes a solution based on the calculus of *Boxed Ambients* introduced in Chapter 2. In this chapter we put the emphasis on the role of the new calculus for describing and expressing resource access control policies. In particular, we develop a type system that provides static detection of resource access violations of MAC policies in a multilevel security system. Finally, § 3.5 illustrates Boxed Ambients at work on several examples coming from the literature on security and related issues.

3.2 Mobile Ambients and Multilevel Security

Standard models of security for resource access control are built around *subjects* performing access requests on *objects* by *write* and *read* (in some models, also *append*, *execute*, and others) operations.

Multilevel security presupposes a lattice of security levels, and every subject and object is assigned a level in this lattice. Based on these levels, access to objects by subjects are classified as *read-up* (res. *read-down*) when a subject access by a read an object of higher (resp. lower) level, and similarly for write accesses. Relying on this classification, one may distinguish two security policies: *military* security, which forbids (both direct and indirect) read-up’s and write-down’s, and *commercial* security that forbids read-up’s and write-up’s. These notions cover also *indirect* accesses resulting from the composition of atomic operations: thus also the fact of writing into an object of any level a piece of information read (or just coming) from another object whose level is higher than the level

of the first object, is considered as a write-down (classic security handles these cases by the so-called \star -property [BP76, Gol99]¹).

As we pointed out in the previous chapters, the security model of Mobile Ambients is based on the idea that permission to cross ambient boundaries is given by making the names available to the clients willing to enter or exit. Entirely depending on the ability by the authorization mechanism to filter out undesired clients, this model appear not to be fully adequate for representing realistic policies for resource access control: an authorization breach could grant malicious agents full access to all resources located inside the ambient boundary. Clearly, one first have to identify what “resource access” is in the Ambient Calculus. Entering an ambient, or opening it are all good notions of access: in addition, there is of course communication. In facts, it is the interplay between communication and the primitives for ambient mobility which makes it difficult to reason about resource access in terms of classical security models. To motivate the point further, we discuss a simple concrete example.

3.2.1 Resource access control in multilevel security

Suppose we have a system consisting of a set of resources $\{r_1, \dots, r_n\}$ and an agent named a running program P , willing to access the resources available on the system. To control the access to the resources, one would typically refer to [DoD85] and set up a resource manager. In the Ambient Calculus the system under consideration can be represented as follows:

$$a[P] \mid m[r_1[\dots] \mid \dots \mid r_n[\dots] \mid R]$$

Here, m is the name of the resource manager and R is the associated process. To access, say, r_i , the agent a needs to know the name m to be able to move inside the resource manager. Assuming that a knows that name, the result of the move is the new system:

$$m[a[P] \mid r_1[\dots] \mid \dots \mid r_n[\dots] \mid R]$$

Looking at this configuration, it is clear that the process R does not have an active role in the system: given the primitive constructs of Mobile Ambients, there is indeed nothing

¹As a matter of fact, these references do not define precisely what a *write-down access* is; instead, they give a definition of *no-write down policy*.

R can do to enable or control the access, as the interaction between $a[P]$ and r_i may only result from autonomous actions by either the agent or the resource². The role of the ambient m is therefore reduced to the role of its name: it is simply the first password required for the access. Rather, it is each of the r_i 's that needs to include its own manager to control and enable the access to the content of the resource.

We can thus formulate the problem in simpler terms, and look directly at the case of the agent $a[P]$ willing to access a resource r as shown below:

Initial configuration: $a[P] \mid r[R \mid \langle M \rangle]$

R is the manager for r , and M is the content: for the purpose of the example we assume that the content is a value the agent is willing to read.

Having defined the problem, we now look at different ways to attack it in MA and discuss their implications in terms of the security models introduced above.

First solution: agent dissolution

A first solution is based on the following protocol proposed by [CG98]. In order for a to access r , a first enters r :

Enter: $r[R \mid \langle M \rangle \mid a[P]]$

Now, the idea of the protocol is that the manager R should be the process `!open p`, which unleashes authorized clients that entered the resource within a taxi ambient named p . In other words, the protocol requires the client to know the name of the resource, as well the name of the “port” p used for the access. The agent would thus first rename itself to p to comply with the rules of the protocol, and then enter: if the access to r is a read access, the agent will contain a reading process. Thus, after renaming, the new configuration would be as follows:

Renaming: $r[!open p \mid \langle M \rangle \mid p[(x)P]]$

Finally, the resource manager enables the read access, by opening p :

Read Access: $r[!open p \mid \langle M \rangle \mid p[(x)P]] \rightarrow r[!open p \mid \langle M \rangle \mid (x)P]$

²The use of *coactions*, as in Levi and Sangiorgi's *Safe Ambients* [LS00], would not help: the move of a into m would only restrict the move, by predicating it to the presence of the co-capability $\overline{\text{in}} m$ in R .

The protocol is elegant and robust: there are two passwords the agent needs to know, the resource name r and the name of the port p . There are, however, a number of unsatisfactory aspects to it.

A first reason for being unsatisfied with the protocol is that it is hardly realistic to assume that agents willing to read a value should be prepared to be dissolved. A second problem is that opening $p[P]$ may be upsetting to the resource manager, or else to the resource itself, because there is no telling what P might do once unleashed. For what we know, the contents of p could very well be the process $N.P$, with N a path of in or out capabilities. Unleashing this process inside r could thus result into r being carried away to possibly hostile locations, or otherwise being made unavailable to further clients requesting access to it.

Further problems arise when we try to classify the protocol according to the MAC security principles. As we noted, the action in the protocol that eventually enables the access to the resource is taken by the resource manager, which opens the incoming agent. In other words, it is the last step of the protocol that effectively determines the access, and since the process enclosed in p is an input process, it is classified as a read access (had p contained an output, this would have been a write access). In multilevel security, it would then be possible to further classify the access according to the security levels associated with r and p , and use that definition to enforce either the military or the commercial security policy.

However, while this form of classification is sensible for the protocol, it becomes rather artificial when applied to the primitives of the calculus. Indeed, saying that $\text{open } p \mid p[P]$ is a read (or write) access by P is rather counter-intuitive, as $p[P]$ undergoes the action rather than actively participating into it: it is very unlikely that we can rely on these notions of read and write access. The problem is that the protocol is entirely dependent on the effects of open , but when exercised to enable a read/write request, open exchanges the roles of the two participants in the request, as it is the subject, rather than the object, that is accessed (in fact, opened).

Second solution: resource dissolution.

The problem of the previous solution could be circumvented by a change of perspective. One could devise a different protocol where the active role of the subject is rendered by

a combination of open and input/output. Thus, for instance, the process $\text{open } r.(x)P$ could be interpreted, in the protocol, as a read request on r . This might work reasonably for read requests, even though the interpretation is not too convincing given that the access has also the side-effect of dissolving the resource. Even less convincing would be the interpretation of $\text{open } r.\langle M \rangle$ as a write access: after dissolving r the output $\langle M \rangle$ really has nothing to do with a write on r .

Third solution: agents and messengers.

To avoid indiscriminate dissolution upon access Cardelli and Gordon [CG98] suggest a different approach, based on a protocol in which agents use special ambients acting as messengers to communicate. The idea is to envisage two classes of messengers:

- *output messenger*: $o[M.\langle N \rangle]$, where M is a path to the location where the message N can be delivered
- *input messengers*: $i[M.(x)o[M^{-1}.\langle x \rangle]]$, where M is the path to the location where a value can be read. Once read, the messenger goes back to its original location (we informally use M^{-1} to denote the inverse path of M) where it delivers the value just read.

Thus, a read access would be encoded by a protocol based on the following initial configuration:

$$a[\text{open } o.(x)P \mid i[\text{out } a.\text{in } r.(x)o[\text{out } r.\text{in } a.\langle x \rangle]]] \mid r[!\text{open } i \mid \langle N \rangle]$$

The protocol still requires cooperation by the resource manager, which is required to open the input messenger. Also, looking at the primitive reductions, it would still be counter-intuitive to say that $\text{open } i \mid i[P]$ is a read access. However, if i could be identified as input-messenger within r , then the access classification would be more realistic.

The problem is that there is no way to syntactically tell messengers apart from ambients playing the role of “pure” agents, nor is there any way to syntactically detect “illegal” attempts to dissolve “pure” agents. Defining a notion of access, and attempting a syntactic classification would therefore still be problematic, if at all possible.

Types could be appealed to for more satisfactory solution. One could devise a type system to complement the syntax by enforcing a typed partition of ambients into agents

(i.e. ambients that cannot be dissolved) and messengers (as above). Based on the typed ambient classification and on an assignment of security levels, it would then be possible to classify access requests according to MAC policies. There would be only one remaining problem.

Consider the protocol structure and evolution. From the initial configuration:

$$a[P' \mid i[M.(x)o[M^{-1}.\langle x \rangle]]] \mid r[!\text{open } i \mid \langle N \rangle]$$

via a sequence of reductions the input messenger reaches its destination, it is opened there, and consumes N . At this stage, the structure of the system is:

$$a[P'] \mid r[!\text{open } i \mid o[M^{-1}.\langle N \rangle]]$$

This is the encoding of a write access by r to a . In other words, a read access by a includes a write access by r : if the former is, say, a read-up, then the latter is a write-down. In other words, the protocol has somehow the effect of merging read-up's and write-down's, and dually, write-up's and read-down's. Therefore, military security could still be accounted for with this approach, while commercial security could not.

3.2.2 Summary and Assessment

Although we do not see any significantly different approach to attack the problem, the survey of solutions we have given might still be incomplete, and there could be better solutions. As to the approaches we have presented, all of them provide only partial solution. Some of them appear artificial, since essential intuition is lost in the encoding of the protocol (§ 3.2.1, § 3.2.1), while in others, intuition is partially recovered but only at the expenses of failing to provide full account for both military and commercial security (§ 3.2.1).

Summarizing, we may certainly say that the Ambient Calculus *enables* resource access control, in that it provides constructs for encoding access protocols. Depending on the protocol, types may help define and check the desired security policies. On the other hand, the calculus *does not in itself support* these mechanisms and policies, as it does not provide built-in facilities to make it convenient or natural to reason about them. As we showed, the reasoning is possible at the level of access *protocols*, but when we look at the access *primitives* there appears to be no general principle to which one can steadily appeal.

Support for resource access control with Mobile Ambients appears to require different, finer-grained, constructs for ambient interaction and communication. The new constructs should be designed carefully, so as to complement the existing restrictions on ambient mobility based on authorization, without breaking them. In other words, the access to remote resources should still require mobility, hence authorization: local access, instead, could be made primitive.

To see how that can be accomplished, let us consider once more the protocol of § 3.2.1, based on messengers. We can re-state it equivalently as follows:

$$a[\text{in } r.i[\text{out } a.(x)o[\text{in } a.\langle x \rangle]] \mid \text{open } o.(x)\text{out } r.P \mid r[!\text{open } i \mid \langle M \rangle]$$

In other words, it is now the agent that is responsible for the moves needed to reach the resource, while the messenger just makes the in and out moves needed for the, now local, access. After the move of a into r , and of i out of a , the structure of the system (disregarding a and replication) is the following: $r[\text{open } i \mid \langle M \rangle \mid i[(x)Q]]$. This is where the read access takes place. Now, instead of coding it, via **open**, we can make it primitive, and do without **open**. If we denote with $(x)^\uparrow$ input from the enclosing ambient, the read access is simply: $r[\langle M \rangle \mid i[(x)^\uparrow P]]$. But then, the whole protocol can be simplified: $a[\text{in } r.(x)^\uparrow P \mid r[\langle M \rangle]]$. A choice of communication primitives based on this observation led us to the design of the Boxed Ambients calculus. The new primitives provide the calculus with what we believe to be more effective constructs for resource protection and access control, while at the same time retaining most of the computational flavor of MA, as well as the elegance of their formal presentation.

3.3 Boxed Ambients and Multilevel Security

In Chapter 2 we presented the calculus of Boxed Ambients arguing that it is a variant of Mobile Ambients whose finer-grained mechanisms for ambient interaction yields an effective framework for resource protection and access control. In particular, dropping the **open** capability, BA rely on a new communication model that can be summarized recalling the reduction rules for message exchanges:

$$\begin{array}{ll}
(local) & (\tilde{x})P \mid \langle \tilde{M} \rangle Q \rightarrow P\{\tilde{x} := \tilde{M}\} \mid Q \\
(input\ n) & (\tilde{x})^n P \mid n[\langle \tilde{M} \rangle Q \mid R] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R] \\
(input\ \uparrow) & \langle \tilde{M} \rangle P \mid n[(\tilde{x})^\uparrow Q \mid R] \rightarrow P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] \\
(output\ n) & \langle \tilde{M} \rangle^n P \mid n[(\tilde{x})Q \mid R] \rightarrow P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] \\
(output\ \uparrow) & (\tilde{x})P \mid n[\langle \tilde{M} \rangle^\uparrow Q \mid R] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R]
\end{array}$$

One of the main consequences of such a communication model, is that the communication primitives have a natural interpretation as access requests. To exemplify, rule *(input n)* represents an access in read mode to the channel located into the child ambient n . On the other hand, *(input \uparrow)* rule represents the ambient n reading from the channel located into the parent ambient. Dually, rule *(output n)* (resp. *(output \uparrow)*), can be seen as a write operation to the channel located into the child (resp. parent) ambient.

Relying on such a classification full and flexible support is now available for resource protection. An agent entering a resource needs not be opened there to enable the access: the resource manager can mediate and keep full control over the read and write requests made by the agent. If we take the resource access problem of § 3.2.1 we now have a fairly natural and elegant solution, in which we also find back a role for the resource manager m . Consider again the configuration

$$m[a[P] \mid r_1[\dots] \mid \dots \mid r_n[\dots] \mid R]$$

where now all ambients are boxed and a has entered the resource manager. We need not to include a manager in each resource, as R may act as a mediator. For instance, R could be defined as the parallel composition $R_1 \mid \dots \mid R_n$ where each R_i is the process $!(x)\langle x \rangle^{r_i}$, each waiting for upward output from a and forwarding it to the i th resource. Some of the R_i 's could be less generous with the agent, and ignore upward input from a to request read access on a instead: $!(x)^a\langle x \rangle^{r_i}$. Should any of the r_i 's be made non accessible, one would simply define $R_i = \mathbf{0}$.

Finally, in the next section we show how in the framework of Boxed Ambients multilevel security may be modeled by embedding security levels in types and using typing rules to enforce and verify *Mandatory* (system-wide) *Access Control* (MAC) policies.

3.4 Access Control by Typing

The resource access control framework we address is an instance of the standard Mandatory Access Control policies in multi-level security environments [BP76, Gol99]. The domain of security levels is assumed to be a lattice (Σ, \preceq) , whose elements are ranged over by ρ, σ, τ . Based on an assignment γ of security levels to subject and objects, one defines a *security policy* as a ternary boolean predicate \mathcal{P} on *subject levels*, *object levels*, and *access modes* $\mathcal{A}, \mathcal{B} \in \{\mathbf{w}, \mathbf{r}, \mathbf{rw}, -\}$ ³. Specifically, an access \mathcal{A} to an object o by a subject s is legal under \mathcal{P} if and only if $\mathcal{P}(\gamma(s), \gamma(o), \mathcal{A})$ holds true. Military security (no read-up, no write-down) and commercial security (no read-up, no write-up) can be enforced by the following security policies:

$$\begin{array}{ll}
 \mathcal{P}_{Mil}(\rho, \sigma, \mathbf{r}) & \triangleq \sigma \preceq \rho \\
 \mathcal{P}_{Mil}(\rho, \sigma, \mathbf{w}) & \triangleq \rho \preceq \sigma \\
 \mathcal{P}_{Mil}(\rho, \sigma, \mathbf{rw}) & \triangleq \sigma = \rho \\
 \mathcal{P}_{Mil}(\rho, \sigma, -) & \triangleq \text{true} \\
 \mathcal{P}_{Com}(\rho, \sigma, \mathbf{r}) & \triangleq \sigma \preceq \rho \\
 \mathcal{P}_{Com}(\rho, \sigma, \mathbf{w}) & \triangleq \sigma \preceq \rho \\
 \mathcal{P}_{Com}(\rho, \sigma, \mathbf{rw}) & \triangleq \sigma \preceq \rho \\
 \mathcal{P}_{Com}(\rho, \sigma, -) & \triangleq \text{true}
 \end{array}$$

In our framework processes take the role of subjects, while ambients take the role of objects, and we then rely on the notion of resource access we have assumed throughout, namely: $(x)^n P$ and $\langle M \rangle^n P$ represent processes (subjects) attempting to access the child n in read and write mode, respectively, whereas $\langle M \rangle^\dagger P$ and $(x)^\dagger P$ represent processes attempting to access their parent ambient, again in read and write mode.

We make these notions formal, and define the import of a security policy in the calculus, by introducing a tagged version of the reduction relation of section 2.2 in which any unauthorized access to an ambient results into a distinguished error-reduction.

The relation of tagged reduction, denoted $\rightarrow_{(\sigma, \Lambda)}$, is defined in terms of a security assignment Λ that associates names (and variables) to security levels, and a security level σ that identifies the clearance of the processes involved in the reduction. Based on that, we take reductions to a distinguished process term err as the formal counterpart of resource access violations.

³The mode “-”, denoting “no access”, is introduced for convenience, to make the notation uniform.

$$\begin{aligned}
(local) \quad & (\tilde{x}:\tilde{W})P \mid \langle \tilde{M} \rangle Q \rightarrow_{(\sigma, \Lambda)} P\{\tilde{x} := \tilde{M}\} \mid Q \\
(input \ n) \quad & (\tilde{x}:\tilde{W})^n P \mid n[\langle \tilde{M} \rangle Q \mid R] \rightarrow_{(\sigma, \Lambda)} \begin{cases} P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R] & \text{if } \mathcal{P}(\sigma, \Lambda(n), r) \\ \text{err} & \text{otherwise} \end{cases} \\
(input \ \uparrow) \quad & \langle \tilde{M} \rangle P \mid n[(\tilde{x}:\tilde{W})^\uparrow Q \mid R] \rightarrow_{(\sigma, \Lambda)} \begin{cases} P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] & \text{if } \mathcal{P}(\Lambda(n), \sigma, r) \\ \text{err} & \text{otherwise} \end{cases} \\
(output \ n) \quad & \langle \tilde{M} \rangle^n P \mid n[(\tilde{x}:\tilde{W})Q \mid R] \rightarrow_{(\sigma, \Lambda)} \begin{cases} P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] & \text{if } \mathcal{P}(\sigma, \Lambda(n), w) \\ \text{err} & \text{otherwise} \end{cases} \\
(output \ \uparrow) \quad & (\tilde{x}:\tilde{W})P \mid n[\langle \tilde{M} \rangle^\uparrow Q \mid R] \rightarrow_{(\sigma, \Lambda)} \begin{cases} P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R] & \text{if } \mathcal{P}(\Lambda(n), \sigma, w) \\ \text{err} & \text{otherwise} \end{cases} \\
(IN) \quad & n[\text{in } m.P \mid Q] \mid m[R] \rightarrow_{(\sigma, \Lambda)} m[n[P \mid Q] \mid R] \\
(OUT) \quad & n[m[\text{out } n.P \mid Q] \mid R] \rightarrow_{(\sigma, \Lambda)} m[P \mid Q] \mid n[R]
\end{aligned}$$

Notice that reductions to `err` only result from attempts to read or write on non-local resources. As such, the reduction relation does not account for errors resulting from type mismatches in any of the local or non-local exchanges of values. Also note, in the rules (INPUT η) and (OUTPUT η) that the clearance of a process enclosed in an ambient is determined by the security level associated with the ambient's name. This is consistent with the format of the rule (AMB) below, which is part of the inference rules that complete the definition of the tagged reduction relation.

$$\begin{aligned}
(NEW) \quad & \frac{P \rightarrow_{(\sigma, (\Lambda, n; \rho))} Q \quad A = \rho \text{Amb}^?[-, -]}{(\nu n:A)P \rightarrow_{(\sigma, \Lambda)} (\nu n:A)Q} & (AMB) \quad & \frac{P \rightarrow_{(\rho, \Lambda)} Q \quad \rho = \Lambda(n)}{n[P] \rightarrow_{(\sigma, \Lambda)} n[Q]} \\
(PAR) \quad & \frac{P \rightarrow_{(\sigma, \Lambda)} Q}{P \mid R \rightarrow_{(\sigma, \Lambda)} Q \mid R \quad , \quad R \mid P \rightarrow_{(\sigma, \Lambda)} R \mid Q} & (STRUCT) \quad & \frac{P' \equiv P \quad P \rightarrow_{(\sigma, \Lambda)} Q \quad Q \equiv Q'}{P' \rightarrow_{(\sigma, \Lambda)} Q'}
\end{aligned}$$

$$\begin{array}{c}
\text{(ERR STRUCT)} \\
\frac{Q \equiv P \quad P \rightarrow_{(\sigma, \Lambda)} \text{err}}{Q \rightarrow_{(\sigma, \Lambda)} \text{err}}
\end{array}
\qquad
\begin{array}{c}
\text{(ERR CTX)} \\
\frac{P \rightarrow_{(\sigma, \Lambda)} \text{err}}{\mathbf{E}(P) \rightarrow_{(\sigma, \Lambda)} \text{err}}
\end{array}$$

Where $\mathbf{E}(\)$ denotes an evaluation context defined as

$$\text{Evaluation Contexts } \mathbf{E}(\) ::= (\) \mid (\nu n)\mathbf{E}(\) \mid P \mid \mathbf{E}(\) \mid \mathbf{E}(\) \mid P \mid M[\mathbf{E}(\)]$$

Now we develop a type system that statically detects access violations. In particular, we prove that well typed processes do not evolve to `err`.

3.4.1 Access Control Types and Typing rules

The new classes of types extend the moded types of Section 2.6 with new tags specifying the security clearance of ambient names and capabilities. This additional information is enough to control the downward accesses to nested ambients, while additional machinery is needed to regulate the accesses to the enclosing context of a process. The new syntax of types is given below.

$$\begin{array}{lll}
\text{Ambient Types} & A & ::= \sigma\text{Amb}[E, F, \mathcal{A}] \mid \sigma\text{Amb}^\circ[E, F, \mathcal{A}] \\
\text{Expression Types} & W & ::= A \mid \sigma\text{Cap}[E, \mathcal{A}] \mid W_1 \times \cdots \times W_n \\
\text{Exchange Types} & E, F & ::= \text{shh} \mid W \\
\text{Process Types} & T & ::= [E, F, \mathcal{A}] \mid [E, {}^\mu F, \mathcal{A}]
\end{array}$$

The intended interpretation of the new types is a direct extension of the corresponding interpretation of the moded types introduced in Section 2.6. $\sigma\text{Amb}^\circ[E, F, \mathcal{A}]$ is the type of a (taxi) ambient that has clearance σ and carries processes whose local and upward exchanges are of type E and F : the upward exchanges have access mode \mathcal{A} . $\sigma\text{Cap}[F, \mathcal{A}]$ is the type of a capability that can be exercised within an ambient with clearance σ , upward exchanges of type F and access mode \mathcal{A} . $[E, {}^\mu F, \mathcal{A}]$ is the type of a process with local and upward exchanges of type, respectively E and F , and access mode \mathcal{A} . Note that process types are associated with two modes: the access mode \mathcal{A} defines the mode in which the process accesses the channel located in its parent ambient; the exchange mode μ defines whether the process is silent, moving or both. We do not explicitly associate security levels with process types: instead, we type check processes at a given security level, by

introducing judgments of the form $\Gamma \vdash_{\sigma} P : T$, where T is process type, and σ a security level.

The typing rules are collected in Section 3.8 at the end of this chapter: most of them are the direct generalization of the corresponding rules in Section 2.6, and so are most of the rules for processes, with the exceptions discussed below. We assume the following partial ordering on access modes: $- \leq \{r, w\} \leq rw$.

Typing of Capabilities.

$$\begin{array}{c}
 \text{(IN)} \\
 \frac{\Gamma \vdash M : \rho \text{Amb}^?[E, F, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathcal{A}) \quad G \leq E}{\Gamma \vdash \text{in } M : \sigma \text{Cap}[G, \mathcal{A}]}
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(OUT)} \\
 \frac{\Gamma \vdash M : \sigma \text{Amb}[E, F, \mathcal{B}] \quad G \leq F, \mathcal{A} \leq \mathcal{B}}{\Gamma \vdash \text{out } M : \rho \text{Cap}[G, \mathcal{A}]}
 \end{array}$$

In addition to the usual type safety constraints, the rules (IN) and (OUT) predicate the well-typing of an ambient move to the security policy under consideration. Specifically, an “in” reduction is well-typed only if the security levels of the two ambients involved in the move are compatible, given the upward accesses of entering ambient. Dually, an out move type checks only if the type of the upward exchanges of the exiting ambient are already encompassed by the upward component of the type of the exited ambient. The rule (OUT) specializes in the natural way to the case in which M is a taxi ambient (cf. Section 3.8).

Typing of Ambients. The typing rules for ambients define the clearance level at which the enclosed processes are typed: if P is enclosed into a σ -level ambient, then P is type-checked at clearance σ . In addition, the rules predicate well-typing to the security policy under consideration. A representative of these rules is the rule (AMB) given below:

$$\begin{array}{c}
 \text{(AMB)} \\
 \frac{\Gamma \vdash M : \sigma \text{Amb}[E, F, \mathcal{A}] \quad \Gamma \vdash_{\sigma} P : [E, F, \mathcal{A}] \quad \mathcal{P}(\sigma, \rho, \mathcal{A})}{\Gamma \vdash_{\rho} M[P] : [F, \bullet H, \mathcal{B}]}
 \end{array}$$

$$\begin{array}{c}
 \text{(AMB } \circ) \\
 \frac{\Gamma \vdash M : \sigma \text{Amb}^{\circ}[E, F, \mathcal{A}] \quad \Gamma \vdash_{\sigma} P : [E, {}^{\circ}F, \mathcal{A}]}{\Gamma \vdash_{\rho} M[P] : [G, \bullet H, \mathcal{B}]}
 \end{array}$$

Rule (AMB \circ) shows that the constraint imposed by the policy \mathcal{P} can safely be lifted for ambients whose enclosing processes are moving (or silent), as in that case there is no upward access to be checked.

Typing of input/output. The rules for local exchanges are straightforward: they enforce no resource access control, as processes are always granted access to their local resources. The rules for downward input/output relate the types of the input-output processes and their continuations, as in the type system of Section 2.6. In addition, they enforce the constraint that processes at clearance σ read only from (rule INPUT M) and write only to (rule OUTPUT M) ambients of clearance ρ compatible with σ according to the given security policy.

(INPUT M)

$$\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{\sigma} P : [E, {}^{\mu}F, \mathcal{A}] \quad \Gamma \vdash M : \rho \text{Amb}^? [W_1 \times \dots \times W_k, G, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathbf{r})}{\Gamma \vdash_{\sigma} (\tilde{x} : \tilde{W})^M P : [E, {}^{\mu}F, \mathcal{A}]}$$

(OUTPUT M) $i = 1, \dots, k$

$$\frac{\Gamma \vdash N_i : W_i \quad \Gamma \vdash_{\sigma} P : [E, {}^{\mu}F, \mathcal{A}] \quad \Gamma \vdash M : \rho \text{Amb}^? [W_1 \times \dots \times W_k, G, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathbf{w})}{\Gamma \vdash_{\sigma} \langle N_1, \dots, N_k \rangle^M P : [E, {}^{\mu}F, \mathcal{A}]}$$

As for the rules for upward exchanges, they do not impose any access control and just check that the access modes are correct: this is sound, as the upward accesses are already regulated by the rules for ambients, and by the rules governing mobility.

Subtyping and Subsumption. Subtyping over exchange and process types extends uniformly to the new set of types. As in the type systems of the previous sections, subtyping is only reflexive on capability and ambient types. Process subtyping, in turn, is the direct extension of the subtyping relation of Section 2.6, defined as follows:

$$\frac{[E, {}^{\mu_1}F] \leq_{2.6} [E', {}^{\mu_2}F]}{[E, {}^{\mu_1}F, \mathcal{A}] \leq [E', {}^{\mu_2}F, \mathcal{A}]}$$

where μ_i are (possibly empty) modes, and $\leq_{2.6}$ is the subtyping relation for processes defined in Section 2.6. This extension of the subtyping relation to the new types is rather weak, and it would be desirable to allow, for instance, subtype relationships such as

$[E, F, r] \leq [E, F, rw]$, distinctive of *descriptive* subtyping. Unfortunately, this and similar extensions are not sound. To see the problem with this form of subtyping, consider a system with military security and two security levels, \top and \perp with $\perp \prec \top$. Take then Γ to be a type environment such that $\Gamma(\ell) = \perp \text{Amb}[W, \text{shh}, r]$ and $\Gamma(h) = \top \text{Amb}[\text{shh}, W, rw]$. Under these assumptions, the judgment $\Gamma \vdash_{\top} \text{in } \ell.0 : [\text{shh}, W, r]$ is derivable by the type system of this section for any W . If, by subtyping, we upgrade the previous typing judgment to $\Gamma \vdash_{\top} \text{in } \ell.0 : [\text{shh}, W, rw]$, and take M of type W , the following judgment is derivable: $\Gamma \vdash_{\top} \text{in } \ell.0 \mid \langle M \rangle^{\uparrow} : [\text{shh}, W, rw]$. From this judgment, and from the assumption $\Gamma(h) = \top \text{Amb}[\text{shh}, W, rw]$, we then have $\Gamma \vdash_{\top} h[\text{in } \ell.0 \mid \langle M \rangle^{\uparrow}] : [W, F, \mathcal{A}]$, for any type F and mode \mathcal{A} . This typing is unsound, however, because h can move into ℓ and make a write access to the low-level ambient ℓ , thus violating the military security policy we had assumed. On the other hand, allowing the converse subtyping such as $[E, F, rw] \leq [E, F, w]$, would be again unsound. Consider the process $(x:W)^{\uparrow}Q$, the judgment $\Gamma \vdash_{\perp} (x:W)^{\uparrow}Q : [\text{shh}, W, rw]$ is derivable, then if we use subtyping we also have $\Gamma \vdash_{\perp} (x:W)^{\uparrow}Q : [\text{shh}, W, w]$, then $\Gamma \vdash_{\top} \ell[(x:W)^{\uparrow}Q] : T$ for some process type T and $\Gamma(\ell) = \perp \text{Amb}[\text{shh}, W, w]$. But this typing is unsound since the previous process performs a forbidden read-up operation.

3.4.2 Soundness of the Type System

The main purpose of the type system of this section is to statically detect access violations, with respect to the underlying security policy. As we state below (and prove in Section 3.7), the type system does provide these guarantees under the additional hypothesis that the security policy is *stable*, in the sense of the following definition.

Definition 3.4.1 (Stable Security Policies). We say that a security policy \mathcal{P} is stable if and only if it satisfies the following conditions

1. if $\mathcal{P}(\sigma, \rho, \mathcal{A})$ and $\mathcal{P}(\rho, \tau, \mathcal{A})$ then $\mathcal{P}(\sigma, \tau, \mathcal{A})$.
2. if $\mathcal{P}(\sigma, \rho, \mathcal{A})$ and $\mathcal{C} \leq \mathcal{A}$ then $\mathcal{P}(\sigma, \rho, \mathcal{C})$. □

Military and Commercial security, as defined in this section, are both examples of stable policies. The proof of the type soundness theorem makes the implicit assumption that the underlying security policy is stable.

Theorem 3.4.2 (Type Soundness). *Given a type environment Γ , say that a security assignment Λ is Γ -consistent if and only if for all $x \in \text{dom}(\Gamma)$, $\Gamma(x) = \sigma \text{Amb}^?[\dots]$ implies $\Lambda(x) = \sigma$. Now assume that $\Gamma \vdash_\sigma P : T$. Then for every Γ -consistent security assignment Λ , and process Q such that $P \rightarrow_{(\sigma, \Lambda)}^* Q$ we have $Q \not\rightarrow_{(\sigma, \Lambda)} \text{err}$.*

Proof. In Section 3.7. □

To exemplify the effects of typing, consider the following example from Section 2.7.2.

$$h[\ell[(x)^\uparrow P] \mid \ell'[\langle M \rangle^\uparrow Q]].$$

If we assume that the clearance of ambient h is strictly higher than the clearance of ℓ and ℓ' , then a type system based on a “no read-up” policy should reject the above process as ill-typed, because not secure. This is indeed the case for our type system: to see that, note that the process $\ell[(x)^\uparrow P]$ is type checked at the clearance of the enclosing ambient h , and the side condition to the (AMB Δ) rules fails to be satisfied under a “no read-up” policy. A similar reasoning shows that the unsafe process $h[\ell[\text{out } h.\text{in } h.(x:W)^\uparrow \mid \langle N \rangle] \mid \langle M \rangle] \mid (y)^\ell$ (with ℓ and h of clearance \perp and \top respectively) is ill-typed, as ℓ inside h performs a read up-operation. In particular, in order for P to type check, the subprocess in $h.(x:W)^\uparrow$ enclosed in ℓ can only be typed at level \perp with the process type $[E, {}^\mu W, \mathcal{A}]$, for $r \leq \mathcal{A}$. However, the judgment $\Gamma \vdash_\perp \text{in } h.(x:W)^\uparrow : [E, {}^\mu W, \mathcal{A}]$ must come from $\Gamma \vdash \text{in } h : \perp \text{Cap}[W, \mathcal{A}]$, which is not derivable since the clearance \perp of the ambient h is greater than the clearance of ℓ , contradicting the hypothesis of rule (IN).

We demonstrate the import of the type system in enforcing effective resource access control policies in the next section on several examples.

3.5 Examples

Wrappers in the boxed π -calculus

As a solution for the problem of resource protection and access control in wide-area networks, Sewell and Vitek [SV00] propose an extension of the π -calculus, known as the *boxed* π -calculus. Within this calculus, they develop a programming technique, based on *wrappers*, whereby untrusted code can be secured into an isolated *box*, and its interactions with

the enclosing environment filtered by a process, the *wrapper*, that only forwards legitimate messages between the boxed program and its enclosing environment via secured channels. The paradigmatic example of that work can be rephrased in our syntax as follows:

$$(\nu a, b) \left(a[P] \mid !(x)^a \langle x \rangle^b \mid b[Q] \right).$$

P and Q are arbitrary processes that are encapsulated in ambients (“named boxes” in [SV00] terminology) with private names a and b , and placed in parallel with a process that forwards messages from a to b . Notice that ambient boundaries prevent any direct interaction, and the name restrictions on a and b ensure that the only possible exchanges with the environment are filtered by the process $!(x)^a \langle x \rangle^b$. Thus, as in Boxed- π , we can rely on wrappers to provide interesting security guarantees: specifically, the above configuration prevents (i) Q from leaking secrets to P and (ii) P and Q from corrupting the environment.

With the type system of Section 3.4, we can provide further guarantees. If we define a to be a high-level ambient and b a low-level ambient, then the type system built over military security will detect any unwanted access from Q to P regardless of context that encloses $a[P]$ and $b[Q]$. In addition, military security may be employed in the type system also to detect any attempt by P and Q to access the environment: for that purpose, we simply need to typecheck the configuration at a clearance incomparable with the clearance of a and b .

Firewalls in Mobile and Safe Ambients

We now look at the protocol for firewall crossing defined in [CG99b] and refined in [LS00]. The protocol can be expressed in our calculus as follows:

$$\begin{aligned} \textit{Firewall} &= (\nu f)f[k[\textit{out } f.\langle \textit{in } f \rangle^a] \mid \dots] \\ \textit{Agent} &= a[\textit{in } k.(x)\textit{out } k.x.Q] \end{aligned}$$

The idea is to let the *Agent* a cross the *Firewall* f by means of a shared key k . In [CG99b], the key k is used as the name of a taxi ambient that drives the agent into the protocol and is then dissolved. Our coding follows the same idea, but implements it differently, relying on communication: a enters k from which it receives the capability $\textit{in } f$ which is exercised after $\textit{out } k$ to drive a into the firewall. Interestingly, the process $a[\textit{in } k.(x)\textit{out } k.x.Q]$,

where a capability is first read (locally) and then exercised at the same nesting level is well-typed in our system (this is not true of the type system of [CG99b]).

Having authenticated an incoming agent, the firewall may then provide other security guarantees. For example, we may want to ensure that processes inside the firewall can access the resources of the agent that crossed the firewall, but not the converse. This guarantee can be provided with commercial security, by the type assignments $f : \phi \text{Amb}[E, F, \mathcal{A}]$ and $k : \kappa \text{Amb}[\text{shh}, \text{shh}]$, where $\kappa \prec \phi$, E and F are appropriate types, and \mathcal{A} is an appropriate access right.

To illustrate the effect of these type assignments, consider a generic agent $a[P]$ entering the firewall, and assume that $a : \alpha \text{Amb}[G, H, \mathcal{B}]$. To cross the firewall, a must accept write requests from k : with commercial security, this is possible only if $\alpha \preceq \kappa$ and this, by transitivity, implies that $\alpha \prec \phi$. Now observe that commercial security prevents low-level ambients (such as a) contained in high-level ambients (such as f) from attempting upward exchanges. Then, $\alpha \prec \phi$ implies $H = \text{shh}$. In summary, the type assignment enforces on a a security level strictly smaller than the level of f , and this implies that any agent entering the firewall f cannot directly access to local resources of f , as desired.

The protocol we just discussed depends on the assumption that the firewall knows the name of the incoming agent. To overcome the problem, we may either assume that a itself is also a password which is part of the protocol, or devise new protocol that relies on two passwords, k and h , and structure the agent and the firewall as follows:

$$\begin{aligned} \text{Firewall} &= (\nu f)f[k[\text{out } f.\langle \text{in } f \rangle^h] \mid \dots] \\ \text{Agent} &= h[\text{in } k.(x)\text{out } k.\langle x \rangle^a \mid a[(x).\text{out } h.x.Q]] \end{aligned}$$

Trojan Horses

In [BC01b] a type system that statically detects Trojan horses in mobile ambients is defined. One of the motivating examples of the work is:

$$a[\text{in } c.P] \mid b[\text{in } a.\text{out } a.\text{in } d.Q] \mid c[R \mid d[S]]. \quad (3.1)$$

Here the ambient d contains confidential data that should be made available to ambients running within c but not to ambients that will enter c . The question is whether c should let

a enter or not. If $d \notin fn(P)$, then apparently a does not attempt to access d ; nevertheless the move must be forbidden since b can use a as a Trojan Horse to enter c and access d .

$$a[\text{in } c.P] \mid b[\text{in } a.\text{out } a.\text{in } d.Q] \mid c[R \mid d[S]] \rightarrow^* c[R \mid a[P] \mid d[S \mid b[Q]]]$$

The attack is detected in [BC01b] by means of a type system that traces the behavior of a , revealing the move of b into a and hence a chance for the attack. The work in [BC01b] also showed how to perform this verification when c runs in a possibly untyped (and, thus, hostile) context.

With the type system of Section 3.4, we can obtain the same effect by setting the clearance of d to a level that is incomparable to any security level that is defined outside c . This can be accomplished by taking the powerset of security labels $(2^L, \subseteq)$ over a suitable alphabet L as lattice of security levels. Based on that, it is possible to use standard π -calculus restrictions to dynamically define security levels⁴, as we suggest below:

$$(\nu \ell \in L)(\nu d : \{\ell\} \text{Amb}[E, \text{shh}, \mathcal{A}])c[R \mid d[S]]$$

Now, regardless of how and where the name d is communicated and whether c is in a well typed-context or not, by commercial security we can guarantee that only the processes that are already inside the ambient c can access information contained in d . Indeed, to produce the initial configuration described in (3.1), c must communicate to b the name d . But, unlike what happens in Mobile Ambients, revealing the name of an ambient does not imply granting access to its resources. For the latter a subject should also possess a compatible security level, but the restriction on the level makes it impossible.

Distributed language security

None of the examples we have presented so far rely on upward communication: thus the simple type system of Section 2.4, or even its restricted version without upward exchanges, would suffice. We conclude with an example in which moded types are used in a critical way: the example builds on an encoding of the toy distributed calculus of [CGG00a] whose syntax is defined below (for the typing rules see [CGG00a]).

⁴This hints at a formalization of *Discretionary Access Control* policies (DAC), if, in addition, one also allows security labels to be communicated over channels.

<i>Types</i>	W	$::=$	$Node$	type of nodes
			$ $ $Ch(W)$	type of channels
<i>Network</i>	Net	$::=$	$(\nu x:W)Net$	restriction
			$ $ $Net Net$	network composition
			$ $ $node\ n[Cro]$	node
<i>Crowd</i>	Cro	$::=$	$(\nu x:W)Cro$	restriction
			$ $ $Cro Cro$	crowd composition
			$ $ channel c	channel
			$ $ thread $[Th]$	thread
<i>Threads</i>	Th	$::=$	go $n.Th$	migration
			$ $ $\overline{c}(x)$	output to a channel
			$ $ $c(x).Th$	input from a channel
			$ $ fork $(Cro).Th$	fork a new crowd
			$ $ spawn $n[Cro].Th$	spawn a new node

To encode this calculus we seldom need the full power of Boxed Ambients, as upward silent ambients mostly suffice (upward communication is used only in one case, that is, for taxi ambients). Therefore, to ease the notation, we will write $\sigma\text{Amb}[E]$ for $\sigma\text{Amb}[E, \text{shh}, \mathcal{A}]$ and omit the security level σ when it is not relevant. Let also *Synch* denote the empty tuple type. The encoding is defined as follows (we assume that all types have the same security level).

Types :

$$\begin{aligned} \langle Node \rangle &= \text{Amb}[\text{shh}] \\ \langle Ch(W) \rangle &= \text{Amb}[\langle W \rangle] \end{aligned}$$

Net :

$$\begin{aligned} \langle (\nu x:Node)Net \rangle_{\Gamma} &= (\nu x:\langle Node \rangle)\langle Net \rangle_{\Gamma} \\ \langle (\nu x:Ch(W))Net \rangle_{\Gamma} &= (\nu x:\langle Ch(W) \rangle)\langle Net \rangle_{\Gamma, x:W} \\ \langle Net_1 | Net_2 \rangle_{\Gamma} &= \langle Net_1 \rangle_{\Gamma} | \langle Net_2 \rangle_{\Gamma} \\ \langle node\ n[Cro] \rangle_{\Gamma} &= n[\langle Cro \rangle_{\Gamma}^n] \end{aligned}$$

Crowd :

$$\begin{aligned}
\langle\langle \nu x:Node \rangle Cro \rangle_{\Gamma}^n &= (\nu x: \langle Node \rangle) \langle Cro \rangle_{\Gamma}^n \\
\langle\langle \nu x:Ch(W) \rangle Cro \rangle_{\Gamma}^n &= (\nu x: \langle Ch(W) \rangle) \langle Cro \rangle_{\Gamma, x:W}^n \\
\langle Cro_1 \mid Cro_2 \rangle_{\Gamma}^n &= \langle Cro_1 \rangle_{\Gamma}^n \mid \langle Cro_2 \rangle_{\Gamma}^n \\
\langle thread [Th] \rangle_{\Gamma}^n &= (\nu t: Amb[shh]) t [\langle Th \rangle_{\Gamma}^{n,t}] \\
\langle channel c \rangle_{\Gamma}^n &= c [! (x: \langle \Gamma(c) \rangle) \langle x \rangle]
\end{aligned}$$

Threads :

$$\begin{aligned}
\langle \underline{c}(x) \rangle_{\Gamma}^{n,t} &= (\nu w: Amb^{\circ}[shh, \langle \Gamma(c) \rangle, w]) w [out\ t.in\ c. \langle x \rangle^{\uparrow}] \\
\langle c(x).Th \rangle_{\Gamma}^{n,t} &= (\nu r: Amb^{\circ}[\langle \Gamma(c) \rangle, \langle \Gamma(c) \rangle, r]) (x: \langle \Gamma(c) \rangle)^r \langle Th \rangle_{\Gamma}^{n,t} \mid \\
&\quad r [out\ t.in\ c. (x: \langle \Gamma(c) \rangle)^{\uparrow}. out\ c.in\ t. \langle x \rangle] \\
\langle go\ m.Th \rangle_{\Gamma}^{n,t} &= out\ n.in\ m. \langle Th \rangle_{\Gamma}^{n,t} \\
\langle fork\ (c).Th \rangle_{\Gamma}^{n,t} &= (\nu s: Amb[Synch]) ()^s \langle Th \rangle_{\Gamma}^{n,t} \\
&\quad \mid c [out\ t. (! (x: \langle \Gamma(c) \rangle) \langle x \rangle) \mid s [out\ c.in\ t. \langle \rangle]] \\
\langle fork\ (Th').Th \rangle_{\Gamma}^{n,t} &= (\nu s: Amb[Synch]) ()^s \langle Th \rangle_{\Gamma}^{n,t} \\
&\quad \mid (\nu t': Amb[shh]) t' [out\ t. (\langle Th' \rangle_{\Gamma}^{n,t'}) \mid s [out\ t'.in\ t. \langle \rangle]] \\
\langle fork\ (Cro_1 \mid Cro_2).Th \rangle_{\Gamma}^{n,t} &= \langle fork\ (Cro_1).fork\ (Cro_2).Th \rangle_{\Gamma}^{n,t} \\
\langle fork\ ((\nu x:Ch(W))Cro).Th \rangle_{\Gamma}^{n,t} &= (\nu x: \langle Ch(W) \rangle) \langle fork\ (Cro).Th \rangle_{\Gamma, x:W}^{n,t} \\
\langle fork\ ((\nu x:Node)Cro).Th \rangle_{\Gamma}^{n,t} &= (\nu x: \langle Node \rangle) \langle fork\ (Cro).Th \rangle_{\Gamma}^{n,t} \\
\langle spawn\ m[Cro].Th \rangle_{\Gamma}^{n,t} &= (\nu s: Amb[Synch]) ()^s \langle Th \rangle_{\Gamma}^{n,t} \\
&\quad \mid m [out\ t.out\ n. (\langle Cro \rangle_{\Gamma}^m \mid s [out\ m.in\ n.in\ t. \langle \rangle])]
\end{aligned}$$

The encoding of the network is parametric in the type environment Γ that records the types of the values transported by channels (we need this to implement channels and their operations, as the parameter of an input channel is not typed in the source calculus). The encoding of a crowd is parametrized also by the current node n of the crowd; the encoding of thread expressions has as further parameter the name of the thread the expressions belong to. Nodes are silent ambients whose subambients encode either channels or threads. Thread migration is obtained by exiting the current node and entering the destination node. There are only three points in the encoding that are not straightforward. The first is in the encodings of fork and spawn which need a synchronization ambient s that triggers the continuation of the thread. The second point is in the encoding of forks which must be given by cases on the concerned crowd: since we do not have open we cannot make a whole

crowd exit the thread and unleash it in the node; instead, we make each single component of the crowd exit the thread individually. The third point is in the encoding of channel operations that need fresh transport ambients, called w for writer and r for reader, which exit the current thread, enter the channel at issue and synchronize (and, in the case of readers, bring the message back). It is interesting to notice that to type readers and writers we need moded typing: readers and writers are not upwardly silent and to travel between threads and channels they have to traverse nodes, which are locally silent. However, they typecheck since the type system can deduce that their upward communications may happen only when they are inside a channel (with local communications of the right type) but they cannot happen while they are in a node or a thread. Indeed it is quite important to see that even though the type of writers and readers is not upwardly silent, they occur in ambients (the threads) that are locally silent: the bodies of writers and readers for a channel c have types of the form $[E, {}^\circ\langle \Gamma(c) \rangle, \mathcal{A}]$, therefore we can apply the rule (AMB \circ) of Section 3.4 and deduce for $r[\dots]$ and $w[\dots]$ any process type we need (in particular the type of threads' bodies).

This translation looks far simpler than the encoding in Mobile Ambients defined in [CGG00a] since channels and their operations are more easily encoded and types are much closer to those of the original language.

We now consider different security policies by slightly modifying the encoding. A first simple policy is to endow channel types and threads with security levels: threads are subjects and channels objects. We can then modify our translation as follows (where $\gamma(n)$ denotes the security level associated to ambient n):

$$\begin{aligned}
\langle \sigma Ch(W) \rangle &= \sigma \text{Amb}[\langle W \rangle] \\
\langle \text{thread} \sigma[Th] \rangle_{\Gamma}^n &= (\nu t : \sigma \text{Amb}[\text{shh}]) \dots \\
\langle \underline{c}(x) \rangle_{\Gamma}^{n,t} &= (\nu w : \gamma(t) \text{Amb}^\circ[\text{shh}, \langle \Gamma(c) \rangle, w]) \dots \\
\langle c(x).Th \rangle_{\Gamma}^{n,t} &= (\nu r : \gamma(t) \text{Amb}^\circ[\langle \Gamma(c) \rangle, \langle \Gamma(c) \rangle, r]) \dots \\
\langle \text{fork } (Th').Th \rangle_{\Gamma}^{n,t} &= (\nu s : \text{Amb}[\text{Synch}]) (\nu t' : \gamma(t) \text{Amb}[\text{shh}]) \dots
\end{aligned}$$

Quite interestingly imposing either military or commercial security on boxed ambients respectively induces, via the translation above, the military or commercial security policy in the distributed language. That is, each of these security policies in the target language induces the same policy in the original language. Indeed any operation non compatible

with the security policy at issue would result in either the writer ambient w or the reader process r being ill-typed. As such it would be statically detected (on the term obtained from the translation).

A different possible policy is to endow node types with security levels. Threads have then the level of the node in which they are created. This is a very common situation in practice (where it usually appears under the form of a partition of the nodes in trusted and distrusted, that is, only two security levels). This can be obtained by slightly modifying the previous definition as follows:

$$\begin{aligned}\langle \sigma Node \rangle &= \sigma \text{Amb}[\text{shh}] \\ \langle \text{thread}[Th] \rangle_{\Gamma}^n &= (\nu t: \gamma(n) \text{Amb}[\text{shh}]) \dots\end{aligned}$$

Note that with the last policy a thread can move into a node of any level even though once there it can access only to resources (channels) its level allows it to. If we want to forbid threads to move into “incompatible” nodes (this is usually needed in two cases: when we partition nodes in reliable and not reliable and want sensible threads to be executed only on reliable nodes, or when we partition threads in trusted and distrusted and we want sensible nodes to execute only trusted threads) we can modify the translation so that moving threads notify their entrance to the entered node:

$$\begin{aligned}\langle \sigma Node \rangle &= \sigma \text{Amb}[\text{Synch}] \\ \langle node\ n[Cr] \rangle_{\Gamma} &= n[\langle Cr \rangle_{\Gamma}^n \mid !()] \\ \langle \text{thread}[Th] \rangle_{\Gamma}^n &= (\nu t: \gamma(n) \text{Amb}[\text{shh}, \text{Synch}, w]) \dots \\ \langle \text{fork}(Th').Th \rangle_{\Gamma}^{n,t} &= (\nu s: \text{Amb}[\text{Synch}]) (\nu t': \gamma(t) \text{Amb}[\text{shh}, \text{Synch}, w]) \dots \\ \langle \text{go}\ m.Th \rangle_{\Gamma}^{n,t} &= \text{out}\ n.\text{in}\ m.\langle \uparrow Th \rangle_{\Gamma}^{n,t}\end{aligned}$$

Any attempt to move a thread into a non compatible node will result into an ill-typed thread (containing the `go` instruction). In particular the use of commercial security for boxed ambient corresponds in the distributed language to ensure a node-protection policy since a node will run only threads of security level higher than or equal to its level (this means that a node runs only threads coming from nodes it trusts). Instead the use of the military security policy enforces a thread-protection policy since a thread will run only on nodes of security level higher than or equal to its level (this means that sensitive threads will be run only on reliable nodes).

3.6 Related work

The $D\pi$ Calculus. In [HR98] Hennessy and Riley discuss a type system for resource protection in the $D\pi$ -calculus, a distributed variant of π -calculus where processes are located, and may migrate across locations. In $D\pi$, communication occurs via named channels that are associated with read/write capabilities: the type system controls that processes accessing a resource possess the appropriate capability.

In our approach, instead, in order to classify an access as legal or illegal, the type system checks that the security levels of subject and object satisfy the constraints imposed by the security policy for that access. A further difference is that in $D\pi$ the topology of locations is completely flat, while in BA ambients may be nested at will: the interplay between the dynamic nesting structure determined by moves, and the dynamic binding of the parent location \uparrow for upward communication makes access control for BA more complex.

The Security Π calculus. In [HR00], Hennessy and Riley discuss the *security π -calculus*, a variant of the π -calculus in which processes are syntactically defined as running at a given security level, and whose type system ensures that low-level processes never gain access to high-level resources. In BA, instead, we assume that clearances are specified by types, and the security level associated to an ambient type represents the clearance of resources contained in that ambient, as well as the clearance of the agent it implements. Besides resource protection, in [HR00], the authors also investigate non-interference, trying to provide guarantees against implicit information flow from high levels to lower levels. To that end, they check that the clearance of values are compatible with clearance of channels along which they (the values) are exchanged. Furthermore, they show that a suitable notion of observational equivalence (*may test*) is soundly captured by the non-interference property checked by the type system. We postpone the study of information flow to Chapter 4; in its current version, the type system only checks the clearance of subjects against the clearance of objects, disregarding the clearance of the values.

Typing of Security for Mobile Ambients. In [CGG00a] the introduction of *group* names and the possibility of creating fresh group names give flexible ways to statically prevent unwanted propagation of names. This approach is orthogonal to the resource access control mechanisms we studied; we believe that the use of group names can be

adapted to Boxed Ambients to obtain similar results. A paper more directly related to ours is [DCS00], where ambient types are associated with security levels in ways similar to ours. The difference is that in [DCS00], security checks are over opening and moves, while in our work we focus on read and write operations. In [TZH02] authors study Denial of Service attacks in the context of Mobile Ambients. As a solution, they propose resource-controlled systems where no subsystems will ever require more resources than those available.

A final mention goes to [BC01b], [NNHJ99] and [NN00], where control and data flow analysis, rather than typing disciplines, are used to check security properties of Ambients.

Other Languages and Calculi. In the literature on language-based security, several papers deal with access control techniques and information flow. Two examples are [DNFP99] and [LR]. In [DNFP99], authors take a similar approach to that of Hennessy and Riley, based on a variant of Linda with multiple “tuple spaces”. In [LR], Leroy and Rouaix focus on integrity of typed applets via access control.

3.7 Proofs of Subject Reduction and Type Soundness

In the type system of Section 3.4, the existence of multiple typing rules for the same syntactic form causes a proliferation of typing derivations for the same judgment. The following lemma shows that we can focus attention on derivations of more regular shape without losing generality.

Lemma 3.7.1 (Typing of Processes).

Ambients Assume $\Gamma \vdash_\rho a[P] : T$. Then there exist E and F exchanges, and \mathcal{A} and \mathcal{B} access modes, such that $[E, \bullet F, \mathcal{B}] \leq T$ and $\Gamma \vdash_\rho a[P] : [E, \bullet F, \mathcal{B}]$ is derivable from the following assumptions, where H and K are arbitrary exchanges:

- (a₁) either $\Gamma \vdash a : \sigma \text{Amb}[H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : [H, E, \mathcal{A}]$ and $\mathcal{P}(\sigma, \rho, \mathcal{A})$
- (a₂) or $\Gamma \vdash a : \sigma \text{Amb}^\circ[H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : [H, \triangle E, \mathcal{A}]$ and $\mathcal{P}(\sigma, \rho, \mathcal{A})$
- (a₃) or $\Gamma \vdash a : \sigma \text{Amb}^\circ[H, K, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : [H, \circ K, \mathcal{A}]$

Parallel Composition Assume $\Gamma \vdash_\sigma P \mid Q : T$. Then there exist H and E (exchanges) and \mathcal{A} (access mode) such that $\Gamma \vdash_\sigma P \mid Q : [E, ?E, \mathcal{A}]$ with $[E, ?E, \mathcal{A}] \leq T$, and the last judgment is derivable from the following assumptions.

- (c₁) if $T \leq [H, E, \mathcal{A}]$ then $\Gamma \vdash_\sigma P : T$ and $\Gamma \vdash_\sigma Q : T$
- (c₂) if $T = [H, {}^\circ E, \mathcal{A}]$ then $\Gamma \vdash_\sigma P : [H, \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, {}^\circ E, \mathcal{A}]$
or $\Gamma \vdash_\sigma P : [H, {}^\circ E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, \bullet E, \mathcal{A}]$
- (c₃) if $T = [H, {}^\Delta E, \mathcal{A}]$ then $\Gamma \vdash_\sigma P : [H, \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, {}^\Delta E, \mathcal{A}]$
or $\Gamma \vdash_\sigma P : [H, {}^\Delta E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, \bullet E, \mathcal{A}]$
or $\Gamma \vdash_\sigma P : [H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, E, \mathcal{A}]$

Prefix Assume $\Gamma \vdash_\sigma M.P : T$. Then there exist H and E exchanges, and \mathcal{A} access mode such that $\Gamma \vdash_\sigma M.P : [H, {}^? E, \mathcal{A}]$ with $[H, {}^? E, \mathcal{A}] \leq T$, and the last judgment is derivable from the following assumptions, where G is any exchange, \mathcal{B} is any access mode, and ρ any security level:

- (p₁) if $T \leq [H, E, \mathcal{A}]$ then $\Gamma \vdash M : \sigma\text{Cap}[E, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : T$
- (p₂) if $[H, {}^\circ E, \mathcal{A}] \leq T$ then $\Gamma \vdash M : \rho\text{Cap}[G, \mathcal{B}]$ and $\Gamma \vdash_\sigma P : [H, {}^\circ E, \mathcal{A}]$
or $\Gamma \vdash M : \sigma\text{Cap}[E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, {}^\Delta E, \mathcal{A}]$

Input Assume $\Gamma \vdash_\sigma (\mathbf{x} : W)^\eta P : T$. Then there exist E, F, G, \mathcal{A} , and ρ such that:

- (i₁) if $\eta = \star$ then $\Gamma, \mathbf{x} : W \vdash_\sigma P : [W, {}^? E, \mathcal{A}] \leq T$.
- (i₂) if $\eta = M$ then $\Gamma, \mathbf{x} : W \vdash_\sigma P : [E, {}^\mu F, \mathcal{A}] \leq T$, $\Gamma \vdash M : \rho\text{Amb}^?[W, G, \mathcal{A}]$,
and $\mathcal{P}(\sigma, \rho, \mathbf{r})$.
- (i₃) if $\eta = \uparrow$ then $\Gamma, \mathbf{x} : W \vdash_\sigma P : [E, W, \mathcal{A}] \leq T$,
or $\Gamma, \mathbf{x} : W \vdash_\sigma P : [E, {}^\Delta W, \mathcal{A}] \leq T$, with $\mathbf{r} \leq \mathcal{A}$

Output Assume $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^\eta P : T$. Then there exist $E, F, G, W_1, \dots, W_k, \mathcal{A}$, and ρ such that:

- (o₁) if $\eta = \star$ then $\Gamma \vdash_\sigma P : [W_1 \times \dots \times W_k, {}^? E, \mathcal{A}] \leq T$ and $\Gamma \vdash M_i : W_i$.
- (o₂) if $\eta = N$ then $\Gamma \vdash_\sigma P : [E, {}^\mu F, \mathcal{A}] \leq T$, $\Gamma \vdash M_i : W_i$,
and $\Gamma \vdash N : \rho\text{Amb}^?[W_1 \times \dots \times W_k, G, \mathcal{A}]$, with $\mathcal{P}(\sigma, \rho, \mathbf{w})$.
- (o₃) if $\eta = \uparrow$ then $\Gamma \vdash_\sigma P : [E, W_1 \times \dots \times W_k, \mathcal{A}] \leq T$,
or $\Gamma \vdash_\sigma P : [E, {}^\Delta (W_1 \times \dots \times W_k), \mathcal{A}] \leq T$,
with $\Gamma \vdash M_i : W_i$ and $\mathbf{w} \leq \mathcal{A}$.

Proof. We need to show that we have captured all the possible cases.

Ambients: The judgment $\Gamma \vdash_\rho a[P] : T$ must have been derived by an application of one of the (AMB) rules followed by any number of subsumption steps. An inspection of the typing rules for ambients proves the claim.

Parallel Composition: The first part of the claim is obvious. The second part is proved as follows.

(c₁) $T \leq [H, E, \mathcal{A}]$ covers two cases, namely $T = [H, \bullet E, \mathcal{A}]$ or $T = [H, E, \mathcal{A}]$.

In the first case, $\Gamma \vdash_\sigma P \mid Q : T$ must have been derived by (PAR \bullet) from $\Gamma \vdash_\sigma P : [H', \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', \bullet E, \mathcal{A}]$, with $H' \leq H$, followed by one or more subsumption steps. Thus $\Gamma \vdash_\sigma P : [H, \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, \bullet E, \mathcal{A}]$ are also derivable, and from these judgments one derives the $\Gamma \vdash_\sigma P \mid Q : [H, \bullet E, \mathcal{A}]$ by (PAR \bullet).

In the second case, $\Gamma \vdash_\sigma P \mid Q : T$ must have been derived either from $\Gamma \vdash_\sigma P : [H', \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', \bullet E, \mathcal{A}]$ by (PAR \bullet) followed by subsumption (with $H' \leq H$), or from $\Gamma \vdash_\sigma P : [H', E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', E, \mathcal{A}]$, by (PAR) again followed by subsumption. In both cases $\Gamma \vdash_\sigma P : [H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, E, \mathcal{A}]$ are derivable, and from these judgments one derives $\Gamma \vdash_\sigma P \mid Q : [H, E, \mathcal{A}]$ by (PAR).

(c₂) $\Gamma \vdash_\sigma P \mid Q : T$ may have been derived by subsumption from $\Gamma \vdash_\sigma P \mid Q : [H', \bullet E, \mathcal{A}]$, with $H' \leq H$ and $\Gamma \vdash_\sigma P \mid Q : [H', \bullet E, \mathcal{A}]$ derived from $\Gamma \vdash_\sigma P : [H', \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', \bullet E, \mathcal{A}]$ by (PAR \bullet). From the last two judgments, by subsumption, one derives $\Gamma \vdash_\sigma P : [H, \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, \circ E, \mathcal{A}]$, from which $\Gamma \vdash_\sigma P \mid Q : T$ derives by (PAR \circ).

The only other possibility is that $\Gamma \vdash_\sigma P \mid Q : T$ has been derived by (PAR \circ) from $\Gamma \vdash_\sigma P : [H', \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', \circ E, \mathcal{A}]$, or from $\Gamma \vdash_\sigma P : [H', \circ E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', \bullet E, \mathcal{A}]$ (with $H' \leq H$) followed by one or more subsumption steps. As in the previous cases, the proof follows by observing that the subsumption steps can be permuted up to the premises of (PAR \circ) rule.

(c₃) $\Gamma \vdash_\sigma P \mid Q : T$ may have been derived from $\Gamma \vdash_\sigma P : [H', E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', E, \mathcal{A}]$ by (PAR), for $H' \leq H$, followed by one or more subsumption steps. From these two judgments, by subsumption one derives $\Gamma \vdash_\sigma P : [H, E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, E, \mathcal{A}]$. Then the desired judgment derives by (PAR) followed by subsumption.

Otherwise, $\Gamma \vdash_\sigma P \mid Q : T$ must have been derived by (PAR $\mu \in \{\bullet, \circ, \Delta\}$) from

$\Gamma \vdash_\sigma P : [H', {}^\mu E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', \bullet E, \mathcal{A}]$, or from $\Gamma \vdash_\sigma P : [H', \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H', {}^\mu E, \mathcal{A}]$, with $H' \leq H$, followed by one or more subsumption steps. In all these cases, by subsumption, one derives $\Gamma \vdash_\sigma P : [H, {}^\Delta E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, \bullet E, \mathcal{A}]$, or $\Gamma \vdash_\sigma P : [H, \bullet E, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [H, {}^\Delta E, \mathcal{A}]$, and then $\Gamma \vdash_\sigma P \mid Q : [H, {}^\Delta E, \mathcal{A}]$ by (PAR Δ).

Prefix: The first part of the claim is obvious. For the second part, first observe that the rules for prefixes never derive types of the form $[H, {}^\Delta E, \mathcal{A}]$. Then, the proof follows by an inspection of the typing rules for prefixes and by observing that any subsumption step based on a subtyping relation of the form $[H', E, \mathcal{A}] \leq [H, E, \mathcal{A}]$ permutes with each of the prefix rules.

Input/Output: By an inspection of the typing rules. \square

A second lemma proves a useful property of upward silent processes.

Lemma 3.7.2 (Upward silent processes). *If $\Gamma \vdash_\sigma P : [E, {}^\Delta H, \mathcal{A}]$, with $H = \text{shh}$, then $\Gamma \vdash_\sigma P : [E, \bullet H, \mathcal{A}]$.*

Proof. By induction on the derivation of $\Gamma \vdash_\sigma P : [E, {}^\Delta H, \mathcal{A}]$, and observing that P may not have either of the forms $(\tilde{x} : \tilde{W})^\dagger Q$ and $\langle M_1, \dots, M_k \rangle^\dagger Q$. \square

Lemma 3.7.3 (in moves preserve types). *If $\Gamma \vdash_\rho a[\text{in } b.P \mid Q] \mid b[R] : T$, then $\Gamma \vdash_\rho b[a[P \mid Q] \mid R] : T$.*

Proof. By Lemma 3.7.1, the judgment in the hypothesis must have been derived from $\Gamma \vdash_\rho b[R] : [E, \bullet F, \mathcal{A}]$ and $\Gamma \vdash_\rho a[\text{in } b.P \mid Q] : [E, \bullet F, \mathcal{A}]$, for suitable E and F exchanges and \mathcal{A} access mode such that $[E, \bullet F, \mathcal{A}] \leq T$. We first show that $\Gamma \vdash_\rho b[a[P \mid Q] \mid R] : [E, \bullet F, \mathcal{A}]$ is derivable.

The proof is a case analysis of the possible types of the sub-terms of $a[\text{in } b.P \mid Q]$ and $b[R]$, guided by Lemma 3.7.1. From $\Gamma \vdash_\rho b[R] : [E, \bullet F, \mathcal{A}]$, we know that $\Gamma \vdash b : \tau \text{Amb}^? [I, L, \mathcal{B}]$ and $\Gamma \vdash_\tau R : [I, {}^? L, \mathcal{B}]$ for any exchange type I , and suitable τ , L and \mathcal{B} . Now we look at $a[\text{in } b.P \mid Q]$ and the possible types of its components, as informed by Lemma 3.7.1, and show that $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ for every J .

By Lemma 3.7.1, we know that there exists $E' \leq E$ such that $\Gamma \vdash_\rho a[\text{in } b.P \mid Q] : [E', \bullet F, \mathcal{A}]$ is derivable by any of the sets of assumptions defined by cases **(a₁)** – **(a₃)**. We next consider those cases.

(a₁) In this case $\Gamma \vdash a : \sigma\text{Amb}[H, E', \mathcal{C}]$, and $\Gamma \vdash_\sigma \text{in } b.P \mid Q : [H, E', \mathcal{C}]$ (with $\mathcal{P}(\sigma, \rho, \mathcal{C})$, but we may disregard this hypothesis, as it follows by the proof below). Let $T = [H, E', \mathcal{C}]$: by **(c₁)** we know that $\Gamma \vdash_\sigma \text{in } b.P : T$ and $\Gamma \vdash_\sigma Q : T$, and by **(p₁)** that $\Gamma \vdash \text{in } b : \sigma\text{Cap}[E', \mathcal{C}]$ and $\Gamma \vdash_\sigma P : T$. From the former judgment and from $\Gamma \vdash b : \tau\text{Amb}^?[I, J, \mathcal{B}]$, an inspection of the rule **(IN)** shows that $E' \leq I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. From $\Gamma \vdash_\sigma P : T$ and from $\Gamma \vdash_\sigma Q : T$, one has $\Gamma \vdash_\sigma P \mid Q : T$ by **(PAR)**. From the last judgment and from $\Gamma \vdash a : \sigma\text{Amb}[H, E', \mathcal{C}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$, by **(AMB)**, one derives $\Gamma \vdash_\tau a[P \mid Q] : [E', \bullet J, \mathcal{B}]$. Then $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ by subsumption, as desired.

(a₂) In this case $\Gamma \vdash a : \sigma\text{Amb}^\circ[H, E', \mathcal{C}]$ and $\Gamma \vdash_\sigma \text{in } b.P \mid Q : [H, \Delta E', \mathcal{C}]$. By Lemma 3.7.1 we can assume that $\Gamma \vdash_\sigma \text{in } b.P \mid Q : [H, \Delta E', \mathcal{C}]$ derives from the three sets of assumptions defined by case **(c₃)**, which we consider below.

(c_{3.1}) $\Gamma \vdash_\sigma \text{in } b.P : [H, \bullet E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, \Delta E', \mathcal{C}]$. From $\Gamma \vdash_\sigma \text{in } b.P : [H, \bullet E', \mathcal{C}]$, by **(p₁)** we know that $\Gamma \vdash_\sigma P : [H, \bullet E', \mathcal{C}]$ and $\Gamma \vdash \text{in } b : \sigma\text{Cap}[E', \mathcal{C}]$. From the last judgment, and the typing of b an inspection of the rule **(IN)** shows that $E' \leq I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. From $\Gamma \vdash_\sigma P : [H, \bullet E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, \Delta E', \mathcal{C}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [H, \Delta E', \mathcal{C}]$ by **(PAR Δ)**. From the last judgment and from $\Gamma \vdash a : \sigma\text{Amb}^\circ[H, E', \mathcal{C}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$, one derives $\Gamma \vdash_\tau a[P \mid Q] : [E', \bullet J, \mathcal{B}]$ by **(AMB Δ)**. Now, $\Gamma \vdash_\sigma a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ derives by subsumption.

(c_{3.2}) $\Gamma \vdash_\sigma \text{in } b.P : [H, \Delta E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, \bullet E', \mathcal{C}]$. By Lemma 3.7.1, we may assume that $\Gamma \vdash_\sigma \text{in } b.P : [H, \Delta E', \mathcal{C}]$ has been derived from the two sets of assumptions defined by case **(p₂)**. First observe that for all exchanges F and access modes \mathcal{C} , $\Gamma \vdash \text{in } b : \sigma\text{Cap}[F, \mathcal{C}]$ implies $\Gamma \vdash \text{in } b : \sigma\text{Cap}[F, \mathcal{C}]$. Then we can reason as follows.

In **(p_{2.1})**, one has $\Gamma \vdash \text{in } b : \bar{\sigma}\text{Cap}[G, \mathcal{D}]$ and $\Gamma \vdash_\sigma P : [H, \circ E', \mathcal{C}]$, with no constraint on the relationship between the exchanges G and E' , the access modes \mathcal{C} and \mathcal{D} , and the security levels $\bar{\sigma}$ and σ . From $\Gamma \vdash_\sigma P : [H, \circ E', \mathcal{C}]$ and

$\Gamma \vdash_\sigma Q : [H, \bullet E', \mathcal{C}]$, one has $\Gamma \vdash_\sigma P \mid Q : [H, \circ E', \mathcal{C}]$ by (PAR \circ). Then, $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ derives directly by (AMB \circ) from $\Gamma \vdash_\sigma P \mid Q : [H, \circ E', \mathcal{C}]$ and $\Gamma \vdash a : \sigma \text{Amb}^\circ[H, E', \mathcal{C}]$.

In (p2.2) one has $\Gamma \vdash \text{in } b : \sigma \text{Cap}[E', \mathcal{C}]$ and $\Gamma \vdash_\sigma P : [H, \Delta E', \mathcal{C}]$. From the former judgment, an inspection of the rule (IN) shows that $E' \leq I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. From $\Gamma \vdash_\sigma P : [H, \Delta E', \mathcal{C}]$ and from $\Gamma \vdash_\sigma Q : [H, \bullet E', \mathcal{C}]$, one derives $\Gamma \vdash_\sigma P \mid Q : [H, \Delta E', \mathcal{C}]$ by (PAR Δ), and then $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ by (AMB Δ), which is applicable since $\mathcal{P}(\sigma, \tau, \mathcal{C})$, followed by subsumption.

(c3.3) $\Gamma \vdash_\sigma \text{in } b.P : [H, E', \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, E', \mathcal{C}]$. This case is similar to the case (a1) proved above, with the difference that now a is a taxi ambient. Reasoning as in that case, one derives $\Gamma \vdash_\sigma P \mid Q : [H, E', \mathcal{C}]$, and then $\Gamma \vdash_\sigma P \mid Q : [H, \Delta E', \mathcal{C}]$ by subsumption, for $E' \leq I$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$. Now, from the last judgment and from $\Gamma \vdash a : \sigma \text{Amb}^\circ[H, E', \mathcal{C}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{C})$ we derive $\Gamma \vdash_\tau a[P \mid Q] : [E', \bullet J, \mathcal{B}]$ by (AMB Δ) and then, by subsumption $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$.

(a3) In this case $\Gamma \vdash a : \sigma \text{Amb}^\circ[H, K, \mathcal{C}]$ and $\Gamma \vdash_\sigma \text{in } b.P \mid Q : [H, \circ K, \mathcal{C}]$. By Lemma 3.7.1 we can assume that $\Gamma \vdash_\sigma \text{in } b.P \mid Q : [H, \circ K, \mathcal{C}]$ has been derived from the two pairs of assumptions defined by case (c2), which we consider below.

(c2.1) $\Gamma \vdash_\sigma \text{in } b.P : [H, \bullet K, \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, \circ K, \mathcal{C}]$. By (p1) we know that $\Gamma \vdash \text{in } b : \sigma \text{Cap}[K, \mathcal{C}]$ and $\Gamma \vdash_\sigma P : [H, \bullet K, \mathcal{C}]$. Now, from $\Gamma \vdash_\sigma P : [H, \bullet K, \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, \circ K, \mathcal{C}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [H, \circ K, \mathcal{C}]$ by (PAR \circ). Now, from the last judgment and $\Gamma \vdash a : \text{Amb}^\circ[H, K, \mathcal{C}]$, one derives $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ directly by (AMB \circ).

(c2.2) $\Gamma \vdash_\sigma \text{in } b.P : [H, \circ K, \mathcal{C}]$ and $\Gamma \vdash_\sigma Q : [H, \bullet K, \mathcal{C}]$. The proof further splits in the two subcases defined by (p2) and proceeds as in case (c3.2) above, with E' replaced by K .

From the previous analysis we have $\Gamma \vdash_\tau a[P \mid Q] : [I, \bullet J, \mathcal{B}]$ for any J . From the hypothesis, we had inferred that $\Gamma \vdash_\tau R : [I, ?L, \mathcal{B}]$, for a suitable L . Choosing $J = L$, by the appropriate (PAR ?) rule, we then derive $\Gamma \vdash_\tau R \mid a[P \mid Q] : [I, ?L, \mathcal{B}]$. From the last judgment and from $\Gamma \vdash b : \tau \text{Amb}^?[I, L, \mathcal{B}]$ we conclude $\Gamma \vdash_\rho b[R \mid a[P \mid Q]] : [E, \bullet F, \mathcal{A}]$

using the appropriate (AMB ?) rule (the same rule used in the derivation of $\Gamma \vdash_\rho b[R] : [E, \bullet F, \mathcal{A}]$). \square

Lemma 3.7.4 (out moves preserve types). *If $\Gamma \vdash_\rho a[b[\text{out } a.P \mid Q] \mid R] : T$ then $\Gamma \vdash_\rho b[P \mid Q] \mid a[R] : T$.*

Proof. By Lemma 3.7.1, there exist E, F exchanges, \mathcal{B} access mode such that $[E, \bullet F, \mathcal{B}] \leq T$ and the judgment in the hypothesis has been derived from $\Gamma \vdash_\rho a[b[\text{out } a.P \mid Q] \mid R] : [E, \bullet F, \mathcal{B}]$. This implies that $\Gamma \vdash_\rho a[R] : [E, \bullet F, \mathcal{B}]$ is also derivable. To prove the lemma, we thus need to show that $\Gamma \vdash_\rho b[P \mid Q] : [E, \bullet F, \mathcal{B}]$.

We distinguish two cases, depending on the type of a , namely: $\Gamma \vdash a : \tau \text{Amb}^?[I, E, \mathcal{C}]$, or $\Gamma \vdash a : \tau \text{Amb}^\circ[I, K, \mathcal{C}]$ for some exchanges I and K , security level τ and access mode \mathcal{C} (note that Lemma 3.7.1 ensures that E can be chosen so that a has the indicated types). For each of the two cases, we look at $b[\text{out } a.P \mid Q] \mid R$ and at the possible types of its components, as informed by Lemma 3.7.1.

Case $\Gamma \vdash a : \tau \text{Amb}^?[I, E, \mathcal{C}]$. From $\Gamma \vdash_\rho a[b[\text{out } a.P \mid Q] \mid R] : [E, \bullet F, \mathcal{A}]$, by Lemma 3.7.1 ((a₁) and (a₂)) it follows that $\mathcal{P}(\tau, \rho, \mathcal{C})$ and $\Gamma \vdash_\tau b[\text{out } a.P \mid Q] \mid R : [I, \mu E, \mathcal{C}]$ for any I , and μ either absent or equal to Δ .

Now, by two applications of Lemma 3.7.1 (to the parallel composition, and then to the process in ambient form), it follows that there exists $H \leq I$ such that $\Gamma \vdash_\tau b[\text{out } a.P \mid Q] : [H, \bullet E, \mathcal{C}]$ is derivable by any of the three sets of assumptions defined by cases (a₁) – (a₃). We consider those cases below.

(a₁) In this case $\Gamma \vdash b : \sigma \text{Amb}[L, H, \mathcal{A}]$, and $\Gamma \vdash_\sigma \text{out } a.P \mid Q : T$ with $T = [L, H, \mathcal{A}]$ and $\mathcal{P}(\sigma, \tau, \mathcal{A})$. By (c₁) we know that $\Gamma \vdash_\sigma \text{out } a.P : T$ and $\Gamma \vdash_\sigma Q : T$. From the first judgment, by (p₁), one has $\Gamma \vdash \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : T$. An inspection of the typing rules shows that $\Gamma \vdash \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$ must have been derived by the rule (OUT). Then the typing of a is, in fact, $\Gamma \vdash a : \tau \text{Amb}[H, E, \mathcal{C}]$, and furthermore $H \leq E$ and $\mathcal{A} \leq \mathcal{C}$. Since \mathcal{P} is stable, by assumption, from $\mathcal{P}(\tau, \rho, \mathcal{C})$ and $\mathcal{A} \leq \mathcal{C}$ we have $\mathcal{P}(\tau, \rho, \mathcal{A})$: this, together with $\mathcal{P}(\sigma, \tau, \mathcal{A})$, implies $\mathcal{P}(\sigma, \rho, \mathcal{A})$. Now, from $\Gamma \vdash_\sigma P : T$ and $\Gamma \vdash_\sigma Q : T$ one has $\Gamma \vdash_\sigma P \mid Q : T$ by (PAR), and from this judgment and from $\Gamma \vdash b : \sigma \text{Amb}[L, H, \mathcal{A}]$ one derives $\Gamma \vdash_\rho b[P \mid Q] : [H, \bullet F, \mathcal{B}]$

by (AMB). Then $\Gamma \vdash_\rho b[P \mid Q] : [E, \bullet F, \mathcal{B}]$ derives by subsumption, given that $H \leq E$.

(a₂) In this case $\Gamma \vdash b : \sigma \text{Amb}^\circ[L, H, \mathcal{A}]$ and $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, \Delta H, \mathcal{A}]$, with $\mathcal{P}(\sigma, \tau, \mathcal{A})$. By Lemma 3.7.1 we can assume that $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, \Delta H, \mathcal{A}]$ derives from one of the three sets of assumptions defined by case (c₃). We consider these cases below.

(c_{3.1}) $\Gamma \vdash_\sigma \text{out } a.P : [L, \bullet H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, \Delta H, \mathcal{A}]$. From $\Gamma \vdash_\sigma \text{out } a.P : [L, \bullet H, \mathcal{A}]$, by (p₁) we know that $\Gamma \vdash_\sigma P : [L, \bullet H, \mathcal{A}]$ and $\Gamma \vdash \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$, with the last judgment derived by the rule (OUT). Thus the typing of a must be $\Gamma \vdash a : \tau \text{Amb}[I, E, \mathcal{C}]$, and moreover $H \leq E$ and $\mathcal{A} \leq \mathcal{C}$. Reasoning as in case (a₁) above, it follows that $\mathcal{P}(\sigma, \rho, \mathcal{A})$. From $\Gamma \vdash_\sigma P : [L, \bullet H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, \Delta H, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [L, \Delta H, \mathcal{A}]$ by (PAR Δ). From the last judgment and from $\Gamma \vdash b : \sigma \text{Amb}^\circ[L, H, \mathcal{A}]$, one derives by (AMB Δ) $\Gamma \vdash_\rho b[P \mid Q] : [H, \bullet J, \mathcal{B}]$ for any J and thus, in particular, for $J = F$. Now, $\Gamma \vdash_\rho b[P \mid Q] : [E, \bullet F, \mathcal{B}]$ derives by subsumption, as $H \leq E$.

(c_{3.2}) $\Gamma \vdash_\sigma \text{out } a.P : [L, \Delta H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, \bullet H, \mathcal{A}]$. By Lemma 3.7.1, we may assume that $\Gamma \vdash_\sigma \text{out } a.P : [L, \Delta H, \mathcal{A}]$ has been derived from the two pairs of assumptions defined by case (p₂).

In (p_{2.1}), one has $\Gamma \vdash^\circ \text{out } a : \bar{\sigma} \text{Cap}[G, \mathcal{D}]$ and $\Gamma \vdash_\sigma P : [L, {}^\circ H, \mathcal{A}]$, with G any exchange, \mathcal{D} any access mode, and $\bar{\sigma}$ any security level. From $\Gamma \vdash_\sigma P : [L, {}^\circ H, \mathcal{A}]$ and From $\Gamma \vdash_\sigma Q : [L, \bullet H, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [L, {}^\circ H, \mathcal{A}]$ by (PAR \circ). Then, we derive $\Gamma \vdash_\rho b[P \mid Q] : [E, \bullet F, \mathcal{B}]$ directly by (AMB \circ).

In (p_{2.2}), one has $\Gamma \vdash^\circ \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : [L, \Delta H, \mathcal{A}]$.

If the typing of a is $\Gamma \vdash a : \tau \text{Amb}[I, E, \mathcal{C}]$, then $\Gamma \vdash^\circ \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$ must have been derived from $\Gamma \vdash \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$, which implies that $H \leq E$ and $\mathcal{A} \leq \mathcal{C}$. Reasoning as in case (a₁), it follows again that $\mathcal{P}(\sigma, \rho, \mathcal{A})$. Then, from $\Gamma \vdash_\sigma P : [L, \Delta H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, \bullet H, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [L, \Delta H, \mathcal{A}]$ by (PAR Δ). From this last judgment and from $\Gamma \vdash b : \sigma \text{Amb}^\circ[L, H, \mathcal{A}]$, one derives $\Gamma \vdash_\rho b[P \mid Q] : [E, \bullet F, \mathcal{B}]$ by (AMB Δ) and subsumption with $H \leq E$. Instead, if the typing of a is $\Gamma \vdash a : \sigma \text{Amb}^\circ[H, E, \mathcal{A}]$ then the fact that $\Gamma \vdash^\circ \text{out } a : \sigma \text{Cap}[H, \mathcal{A}]$ is derivable implies $H = \text{shh}$. But then, from the

hypothesis $\Gamma \vdash_\sigma P : [L, {}^\Delta H, \mathcal{A}]$, by Lemma 3.7.2, it follows that also $\Gamma \vdash_\sigma P : [L, {}^\circ H, \mathcal{A}]$ is derivable. Then $\Gamma \vdash_\sigma P \mid Q : [L, {}^\circ H, \mathcal{A}]$ is derivable by (PAR \circ). Now $\Gamma \vdash_\rho b[P \mid Q] : [H, {}^\bullet F, \mathcal{B}]$ derives by (AMB \circ), and $\Gamma \vdash_\rho b[P \mid Q] : [E, {}^\bullet F, \mathcal{B}]$ by subsumption.

(**c_{3.3}**) $\Gamma \vdash_\sigma \text{out } a.P : [L, H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, H, \mathcal{A}]$. This case has the same hypothesis as the case (**a₁**) above, save that b is typed as a taxi ambient. Reasoning as in that case, one derives $\Gamma \vdash_\sigma P \mid Q : [L, H, \mathcal{A}]$, and then $\Gamma \vdash_\sigma P \mid Q : [L, {}^\Delta H, \mathcal{A}]$ by subsumption, with $H \leq E$. The proof proceeds as in (**a₁**): only, it uses (AMB Δ) instead of (AMB).

(**a₃**) In this case $\Gamma \vdash b : \sigma\text{Amb}^\circ[L, K, \mathcal{A}]$ and $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, {}^\circ K, \mathcal{A}]$. By Lemma 3.7.1 we can assume that $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, {}^\circ K, \mathcal{A}]$ has been derived from any of the two sets of assumptions defined by case (**c₂**), which we consider below.

(**c_{2.1}**) $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\bullet K, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\circ K, \mathcal{A}]$. By (**p₁**) we know that $\Gamma \vdash_\sigma P : [L, {}^\bullet K, \mathcal{A}]$. From $\Gamma \vdash_\sigma P : [L, {}^\bullet K, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\circ K, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [L, {}^\circ K, \mathcal{A}]$ by (PAR \circ). From the last judgment and $\Gamma \vdash b : \sigma\text{Amb}^\circ[L, K, \mathcal{A}]$, one derives $\Gamma \vdash_\rho b[P \mid Q] : [E, {}^\bullet F, \mathcal{B}]$ directly by (AMB \circ).

(**c_{2.2}**) $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\circ K, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\bullet K, \mathcal{A}]$. The proof further splits in the two subcases defined by (**p₂**) and proceeds as in case (**c_{3.2}**) above, with H replaced by K .

Case $\Gamma \vdash a : \tau\text{Amb}^\circ[I, K, \mathcal{A}]$. From the judgment $\Gamma \vdash_\rho a[b[\text{out } a.P \mid Q] \mid R] : [E, {}^\bullet F, \mathcal{A}]$, by Lemma 3.7.1 (**a₁**) and (**a₂**), and then (**c₁**) – (**c₃**), it follows that there exists $H \leq I$ such that $\Gamma \vdash_\tau b[\text{out } a.P \mid Q] : [H, {}^\bullet K, \mathcal{A}]$ is derivable by any of the three sets of assumptions defined by cases (**a₁**) – (**a₃**). We consider those cases below: as we shall see, most of them are vacuous, given the typing of a as a taxi ambient.

(**a₁**) In this case $\Gamma \vdash b : \sigma\text{Amb}[L, H, \mathcal{A}]$ and $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, H, \mathcal{A}]$. This is one of the vacuous cases. To see that, note that $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, H, \mathcal{A}]$ implies, by (**c₁**) and (**p₁**), that $\Gamma \vdash \text{out } a : \sigma\text{Cap}[H, \mathcal{A}]$ is derivable. An inspection of the typing rules shows that this is not possible, as we are currently assuming that

$\Gamma \vdash a : \sigma\text{Amb}^\circ[H, K, \mathcal{A}]$, and hence the only derivable judgments for $\text{out } a$ are of the form $\Gamma \vdash^\circ \text{out } a : \sigma\text{Cap}[\text{shh}, \mathcal{A}]$ for some mode \mathcal{A} .

- (a₂) In this case $\Gamma \vdash b : \sigma\text{Amb}^\circ[L, H, \mathcal{A}]$ and $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, {}^\Delta H, \mathcal{A}]$. By Lemma 3.7.1 we can assume that $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, {}^\Delta H, \mathcal{A}]$ derives from any of the three sets of assumptions defined by case (c₃). We consider these cases below.

In case (c_{3.1}) one has $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\bullet H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\Delta H, \mathcal{A}]$: this is another vacuous case, for the reason given above. Similarly, case (c_{3.3}) is vacuous as $\Gamma \vdash_\sigma \text{out } a.P : [L, H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, H, \mathcal{A}]$.

In case (c_{3.2}), $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\Delta H, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\bullet H, \mathcal{A}]$. By Lemma 3.7.1, we may assume that $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\Delta H, \mathcal{A}]$ has been derived from one of the two pairs of assumptions defined by case (p₂).

In (p_{2.1}), one has $\Gamma \vdash^\circ \text{out } a : \bar{\sigma}\text{Cap}[G, \mathcal{D}]$ and $\Gamma \vdash_\sigma P : [L, {}^\circ H, \mathcal{A}]$, for any G , \mathcal{D} , and $\bar{\sigma}$. From $\Gamma \vdash_\sigma P : [L, {}^\circ H, \mathcal{A}]$ and from $\Gamma \vdash_\sigma Q : [L, {}^\bullet H, \mathcal{A}]$ one derives $\Gamma \vdash_\sigma P \mid Q : [L, {}^\circ H, \mathcal{A}]$ by (PAR \circ). Then, $\Gamma \vdash_\rho b[P \mid Q] : [E, {}^\bullet F, \mathcal{B}]$ derives directly by (AMB \circ).

In (p_{2.2}), one has $\Gamma \vdash^\circ \text{out } a : \sigma\text{Cap}[H, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : [L, {}^\Delta H, \mathcal{A}]$. The typing of a and $\Gamma \vdash^\circ \text{out } a : \sigma\text{Cap}[H, \mathcal{A}]$ imply that $H = \text{shh}$. But then, from the hypothesis $\Gamma \vdash_\sigma P : [L, {}^\Delta H, \mathcal{A}]$, by Lemma 3.7.2, it follows that also $\Gamma \vdash_\sigma P : [L, {}^\circ H, \mathcal{A}]$ is derivable. Now, $\Gamma \vdash_\sigma P \mid Q : [L, {}^\circ H, \mathcal{A}]$ is derivable by (PAR \circ), and then $\Gamma \vdash_\rho b[P \mid Q] : [E, {}^\bullet F, \mathcal{B}]$ directly by (AMB \circ).

- (a₃) In this case $\Gamma \vdash b : \sigma\text{Amb}^\circ[L, J, \mathcal{A}]$ and $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, {}^\circ J, \mathcal{A}]$. By Lemma 3.7.1 we can assume that $\Gamma \vdash_\sigma \text{out } a.P \mid Q : [L, {}^\circ J, \mathcal{A}]$ has been derived from one of the two pairs of assumptions defined by case (c₂). Case (c_{2.1}) is vacuous, as it implies $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\bullet J, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\circ J, \mathcal{A}]$.

In case (c_{2.2}), $\Gamma \vdash_\sigma \text{out } a.P : [L, {}^\circ J, \mathcal{A}]$ and $\Gamma \vdash_\sigma Q : [L, {}^\bullet J, \mathcal{A}]$. The proof further splits into the two subcases defined by (p₂), namely: (i) $\Gamma \vdash^\circ \text{out } a : \bar{\sigma}\text{Cap}[G, \mathcal{D}]$ and $\Gamma \vdash_\sigma P : [L, {}^\circ J, \mathcal{A}]$, and (ii) $\Gamma \vdash^\circ \text{out } a : \sigma\text{Cap}[J, \mathcal{A}]$ and $\Gamma \vdash_\sigma P : [L, {}^\Delta J, \mathcal{A}]$. In the first subcase, the claim follows as in case (p_{2.1}) above, by a final application of (AMB \circ). In the second, $J = \text{shh}$ and the claim follows as in case (p_{2.2}). \square

Lemma 3.7.5 (Type Environments). *Let $\Gamma \vdash^? U : Z$ denote any of the judgments $\Gamma \vdash \circ M : W$, $\Gamma \vdash M : W$ or $\Gamma \vdash_\sigma P : T$.*

1. *If $\Gamma \vdash \diamond$ then $\Gamma' \vdash \diamond$ for every $\Gamma' \subseteq \Gamma$.*
2. *If $\Gamma \vdash^? U : Z$ then $\Gamma \vdash \diamond$.*
3. *If $\Gamma, x : W, \Gamma' \vdash^? U : Z$ and $x \notin \text{fn}(U)$ then $\Gamma, \Gamma' \vdash^? U : Z$.*
4. *If $\Gamma, \vdash^? U : Z$ and $\Gamma, \Gamma' \vdash \diamond$ then $\Gamma, \Gamma' \vdash^? U : Z$.*

Proof. By induction on the derivation of the first judgments in each of the hypotheses. \square

Lemma 3.7.6 (Substitution).

1. *Assume $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash^? M : W$. For all N_1, \dots, N_k , if for all $i \in \{1 \dots k\}$ $\Gamma, \Gamma' \vdash N_i : W_i$, then $\Gamma, \Gamma' \vdash^? M\{x_i := N_i\} : W$.*
2. *Assume $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash_\sigma P : T$. For all N_1, \dots, N_k , if for all $i \in \{1 \dots k\}$ $\Gamma, \Gamma' \vdash N_i : W_i$, then $\Gamma, \Gamma' \vdash_\sigma P\{x_i := N_i\} : T$.*

Proof. The proof is by induction on the derivations of first judgments in the hypotheses and a case analysis on the last applied rule. Most cases follow directly by the induction hypothesis: we give a proof of the representative cases.

1. **(Projection)** The hypothesis is $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash y : W$. We have two cases to consider. If $y = x_i$ for some $i \in \{1 \dots k\}$, then it must be the case that $W = W_i$ and the claim follows from the hypothesis $\Gamma, \Gamma' \vdash N_i : W_i$. If $y \notin \{x_1, \dots, x_k\}$, from $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash y : W$, by Lemma 3.7.5.3 we have $\Gamma, \Gamma' \vdash y : W$. This concludes the proof since $y = y\{x_i := N_i\}$.

(In) The hypothesis is $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash \text{in } M : \sigma\text{Cap}[G, \mathcal{B}]$, derived from $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash M : \rho\text{Amb}^?[F, E, \mathcal{A}]$ with $G \leq F$ and $\mathcal{P}(\sigma, \rho, \mathcal{B})$. By the induction hypothesis it follows that $\Gamma, \Gamma' \vdash M\{x_i := N_i\} : \rho\text{Amb}^?[F, E, \mathcal{A}]$. Then, the desired judgment derives by an application of the rule (In).

2. **(Prefix)** The judgment in the hypothesis is $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash_\sigma M.P : [E, F, \mathcal{A}]$, derived from $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash M : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash_\sigma P : [E, F, \mathcal{A}]$. By the induction hypothesis, $\Gamma, \Gamma' \vdash M\{x_i := N_i\} : \sigma\text{Cap}[F, \mathcal{A}]$ and

$\Gamma, \Gamma' \vdash_{\sigma} P\{x_i := N_i\} : [E, F, \mathcal{A}]$ are both derivable. Then, $\Gamma, \Gamma' \vdash_{\sigma} M\{x_i := N_i\}.P\{x_i := N_i\} : [E, F, \mathcal{A}]$ derives by (PREFIX).

(New) The hypothesis is $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash_{\sigma} (\nu y : \rho \text{Amb}^?[E, F, \mathcal{A}])P : T$, which comes from $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma', y : \rho \text{Amb}^?[E, F, \mathcal{A}] \vdash_{\sigma} P : T$. By Lemma 3.7.5.2, we know that $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma', y : \rho \text{Amb}^?[E, F, \mathcal{A}] \vdash \diamond$, hence that $y \notin \{x_1, \dots, x_k\}$. By induction hypothesis we have $\Gamma, \Gamma', y : \rho \text{Amb}^?[E, F, \mathcal{A}] \vdash_{\sigma} P\{x_i := N_i\} : T$. Now, from the last judgment, by (NEW), one derives $\Gamma, \Gamma' \vdash_{\sigma} (\nu y : \rho \text{Amb}^?[E, F, \mathcal{A}])(P\{x_i := N_i\}) : T$. This is the judgment we wished to derive, as $y \notin \{x_1, \dots, x_n\}$ implies that $(\nu y : \rho \text{Amb}^?[E, F, \mathcal{A}])(P\{x_i := N_i\})$ is equal to $((\nu y : \rho \text{Amb}^?[E, F, \mathcal{A}])P)\{x_i := N_i\}$.

(Output N) The hypothesis is $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash_{\sigma} \langle M \rangle^N P : T$. By Lemma 3.7.1.(o2) there exist E, F, G, W, \mathcal{A} , and ρ such that $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash N : \rho \text{Amb}^?[W, G, \mathcal{A}]$, $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash M : W$, $\Gamma, x_1 : W_1, \dots, x_k : W_k, \Gamma' \vdash_{\sigma} P : [E, {}^{\mu}F, \mathcal{A}] \leq T$, with $\mathcal{P}(\sigma, \rho, w)$. By the induction hypothesis $\Gamma, \Gamma' \vdash N\{x_i := N_i\} : \rho \text{Amb}^?[W, G, \mathcal{A}]$, and $\Gamma, \Gamma' \vdash M\{x_i := N_i\} : W$ and $\Gamma, \Gamma' \vdash_{\sigma} P\{x_i := N_i\} : [E, {}^{\mu}F, \mathcal{A}]$ are all derivable. Then, by (OUTPUT N) and subsumption one derives the desired judgment $\Gamma, \Gamma' \vdash \langle M\{x_i := N_i\} \rangle^{N\{x_i := N_i\}} P\{x_i := N_i\} : T$.

□

Lemma 3.7.7 (Synchronous exchange preserves types).

1. If $\Gamma \vdash_{\sigma} (x_1 : W_1, \dots, x_k : W_k)P \mid \langle M_1, \dots, M_k \rangle Q : T$, then $\Gamma \vdash_{\sigma} P\{x_i := M_i\} \mid Q : T$
2. If $\Gamma \vdash_{\sigma} (x_1 : W_1, \dots, x_k : W_k)^n P \mid n[\langle M_1, \dots, M_k \rangle Q \mid R] : T$,
then $\Gamma \vdash_{\sigma} P\{x_i := M_i\} \mid n[Q \mid R] : T$
3. If $\Gamma \vdash_{\sigma} \langle M_1, \dots, M_k \rangle P \mid n[(x_1 : W_1, \dots, x_k : W_k)^{\uparrow} Q \mid R] : T$,
then $\Gamma \vdash_{\sigma} P \mid n[Q\{x_i := M_i\} \mid R] : T$
4. If $\Gamma \vdash_{\sigma} \langle M_1, \dots, M_k \rangle^n P \mid n[(x_1 : W_1, \dots, x_k : W_k)Q \mid R] : T$,
then $\Gamma \vdash_{\sigma} P \mid n[Q\{x_i := M_i\} \mid R] : T$
5. If $\Gamma \vdash_{\sigma} (x_1 : W_1, \dots, x_k : W_k)P \mid n[\langle M_1, \dots, M_k \rangle^{\uparrow} Q \mid R] : T$,
then $\Gamma \vdash_{\sigma} P\{x_i := M_i\} \mid n[Q \mid R] : T$

Proof. We only give three cases as representative: 1 (local exchange), 2 (downward input) and 5 (upward output). The remaining cases are handled similarly.

1. By repeated applications of Lemma 3.7.1 (on the parallel composition, and then on the component processes), it follows that $T = [W_1 \times \cdots \times W_k, {}^?F, \mathcal{A}]$, for some access \mathcal{A} , and exchange type ${}^?F$. The judgment in the hypothesis must have been derived from $\Gamma, x_1 : W_1, \dots, x_k : W_k \vdash_\sigma P : [W_1 \times \cdots \times W_k, {}^{?1}F, \mathcal{A}]$, from $\Gamma \vdash_\sigma M_i : W_i$ ($i = 1..k$) and from $\Gamma \vdash_\sigma Q : [W_1 \times \cdots \times W_k, {}^{?2}F, \mathcal{A}]$ by a rule (PAR ${}^?_h$) where h is either 1 or 2. From the first two judgments, by Lemma 3.7.6, we know that $\Gamma \vdash_\sigma P\{x_i := M_i\} : [W_1 \times \cdots \times W_k, {}^{?1}F, \mathcal{A}]$. Then $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid Q : [W_1 \times \cdots \times W_k, {}^?F, \mathcal{A}]$ derives directly by the given (PAR ${}^?_h$) rule.
2. The judgment in the hypothesis must have been derived from $\Gamma \vdash_\sigma (x_1 : W_1, \dots, x_k : W_k)^n P : T'$ and $\Gamma \vdash_\sigma n[\langle M_1, \dots, M_k \rangle Q \mid R] : T''$ for appropriate T' and T'' . Let Ξ be the (partial) derivation from these two judgments to the judgment in the hypothesis.

From $\Gamma \vdash_\sigma (x_1 : W_1, \dots, x_k : W_k)^n P : T'$, by Lemma 3.7.1(**i**₂) and subsumption, we know that $\Gamma, x_1 : W_1, \dots, x_k : W_k \vdash_\sigma P : T'$, and $\Gamma \vdash n : \rho \text{Amb}^? [W_1 \times \cdots \times W_k, E, \mathcal{A}]$ for some E , ρ and \mathcal{A} (we also know that $\mathcal{P}(\sigma, \rho, r)$, but this is only useful in the proof of type soundness, not here) We take the case in which $\Gamma \vdash n : \rho \text{Amb}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ as representative: the remaining cases are similar, as the reasoning only depends on the local exchanges of the processes involved in the reduction.

From $\Gamma \vdash_\sigma n[\langle M_1, \dots, M_k \rangle Q \mid R] : T''$ and $\Gamma \vdash n : \rho \text{Amb}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$, by Lemma 3.7.1, we know that $\mathcal{P}(\rho, \sigma, \mathcal{A})$, and there must exist F such that $[E, \bullet F, \mathcal{A}] \leq T''$, and $\Gamma \vdash_\sigma n[\langle M_1, \dots, M_k \rangle Q \mid R] : [E, \bullet F, \mathcal{A}]$ comes from $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle Q \mid R : [W_1 \times \cdots \times W_k, E, \mathcal{A}]$. From the last judgment, by Lemma 3.7.1(**c**₃), we know that $\Gamma \vdash_\rho R : [W_1 \times \cdots \times W_k, E, \mathcal{A}]$ and $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle Q : [W_1 \times \cdots \times W_k, E, \mathcal{A}]$. From the last judgment, by Lemma 3.7.1(**o**₁), $\Gamma \vdash M_i : W_i$, and (by subsumption) also $\Gamma \vdash_\rho Q : [W_1 \times \cdots \times W_k, E, \mathcal{A}]$. From this judgment and $\Gamma \vdash_\rho R : [W_1 \times \cdots \times W_k, E, \mathcal{A}]$ we then derive $\Gamma \vdash_\rho Q \mid R : [W_1 \times \cdots \times W_k, E, \mathcal{A}]$ by (PAR). From the last judgment, from $\Gamma \vdash n : \rho \text{Amb}[W_1 \times \cdots \times W_k, E, \mathcal{A}]$ and $\mathcal{P}(\rho, \sigma, \mathcal{A})$ one derives $\Gamma \vdash_\sigma n[Q \mid R] : [E, \bullet F, \mathcal{A}]$ by (AMB), and then $\Gamma \vdash_\sigma n[Q \mid R] : T''$ by sub-

sumption.

From $\Gamma \vdash M_i : W_i$ and $\Gamma, x_1 : W_1, \dots, x_k : W_k \vdash_\sigma P : T'$ which we had derived above, by Lemma 3.7.6, $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$. Finally, from $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$ and $\Gamma \vdash_\sigma n[Q \mid R] : T''$, the judgment $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid n[Q \mid R] : T$ derives by the same steps used in Ξ .

5. The judgment in the hypothesis must have been derived from $\Gamma \vdash_\sigma (x_1 : W_1, \dots, x_k : W_k)P : T'$, and $\Gamma \vdash_\sigma n[\langle M_1, \dots, M_k \rangle^\dagger Q \mid R] : T''$ for suitable T' and T'' . Let Ξ be the (partial) derivation from these two judgments to the judgment in the hypothesis

From the first judgment, by Lemma 3.7.1.(i₁), there exists E such that $[W_1 \times \dots \times W_k, {}^?E, \mathcal{A}] \leq T'$ and $\Gamma, x_1 : W_1, \dots, x_k : W_k \vdash_\sigma P : [W_1 \times \dots \times W_k, {}^?E, \mathcal{A}]$.

From $\Gamma \vdash_\sigma n[\langle M_1, \dots, M_k \rangle^\dagger Q \mid R] : T''$ and the judgment we just derived, by Lemma 3.7.1.(a₁) – (a₃), there must exist $H \leq W_1 \times \dots \times W_k$ such that $[H, \bullet E, \mathcal{A}] \leq T''$ and $\Gamma \vdash_\sigma n[\langle M_1, \dots, M_k \rangle^\dagger Q \mid R] : [H, \bullet E, \mathcal{A}]$ is derived from either of the sets of hypotheses defined by the cases (a₁) and (a₂). The case (a₃) may be dispensed with, as it implies that a derivation exists for the judgment $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^\dagger Q \mid R : [I, {}^\circ K, \mathcal{B}]$ (for some I, K and \mathcal{B}), while such derivation does not exist: if the judgment in question were derivable, then by Lemma 3.7.1.(c₂), either $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^\dagger Q : [I, {}^\circ K, \mathcal{B}]$ or $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^\dagger Q : [I, \bullet K, \mathcal{B}]$, would be derivable, contradicting Lemma 3.7.1.(o₃).

The cases (a₁) and (a₂) are similar: we prove the second, which is more complex, and leave the first to the reader. The hypotheses are $\Gamma \vdash n : \rho \text{Amb}^\circ [I, H, \mathcal{B}]$ and $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle^\dagger Q \mid R : [I, {}^\Delta H, \mathcal{B}]$. From the last judgment, by Lemma 3.7.1.(c₃) (and the reasoning we just made about case (a₃)), it follows that $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle^\dagger Q : T^*$ is derivable with T^* such that $[I, H, \mathcal{B}] \leq T^*$. This, by Lemma 3.7.1.(o₃) implies that $H \neq \text{shh}$: from this, since $H \leq W_1 \times \dots \times W_k$, it follows that $H = W_1 \times \dots \times W_k$. Lemma 3.7.1.(o₃) applied to $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle^\dagger Q : T^*$ also implies $\Gamma \vdash M_i : W_i$ and $\Gamma \vdash_\rho Q : T^*$. Then $\Gamma \vdash_\rho Q \mid R : [I, {}^\Delta H, \mathcal{B}]$ derives by the same steps that derived $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle^\dagger Q \mid R : [I, {}^\Delta H, \mathcal{B}]$ from $\Gamma \vdash_\rho \langle M_1, \dots, M_k \rangle^\dagger Q : T^*$.

Now from $\Gamma \vdash M_i : W_i$ and from $\Gamma, x_1 : W_1, \dots, x_k : W_k \vdash_\rho P : [W_1 \times \dots \times W_k, {}^?E, \mathcal{A}]$, which we had derived above, by Lemma 3.7.6, we have $\Gamma \vdash_\sigma P\{x_i := M_i\} : [W_1 \times \dots \times W_k, {}^?E, \mathcal{A}]$, and then $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$ by subsumption. We are ready

to conclude: from $\Gamma \vdash n : \rho\text{Amb}^\circ[I, H, \mathcal{B}]$ and $\Gamma \vdash_\rho Q \mid R : [I, {}^\Delta H, \mathcal{B}]$ we derive $\Gamma \vdash_\sigma n[Q \mid R] : [H, {}^\bullet E, \mathcal{A}]$, and then $\Gamma \vdash_\sigma n[Q \mid R] : T''$. From the last judgment and from $\Gamma \vdash_\sigma P\{x_i := M_i\} : T'$, one derives $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid n[Q \mid R] : T$ by the steps used in Ξ , \square

Lemma 3.7.8 (Asynchronous exchange preserves types).

1. If $\Gamma \vdash_\sigma (x_1 : W_1, \dots, x_k : W_k)P \mid \langle M_1, \dots, M_k \rangle : T$, then $\Gamma \vdash_\sigma P\{x_i := M_i\} : T$
2. If $\Gamma \vdash_\sigma (x_1 : W_1, \dots, x_k : W_k)^n P \mid n[\langle M_1, \dots, M_k \rangle \mid Q] : T$
then $\Gamma \vdash_\sigma P\{x_i := M_i\} \mid n[Q] : T$
3. If $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle \mid n[(x_1 : W_1, \dots, x_k : W_k)^\dagger P \mid Q] : T$
then $\Gamma \vdash_\sigma n[P\{x_i := M_i\} \mid Q] : T$
4. If $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle P : T$ then $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle \mid P : T$
5. If $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^n P : T$ then $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^n \mid P : T$
6. If $\Gamma \vdash_\sigma P \mid n[\langle M_1, \dots, M_k \rangle^\dagger Q \mid R] : T$ then $\Gamma \vdash_\sigma P \mid \langle M_1, \dots, M_k \rangle \mid n[Q \mid R] : T$
7. If $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle^n \mid n[(x_1, \dots, x_k)P \mid Q] : T$ then $\Gamma \vdash_\sigma n[P\{x_i := M_i\} \mid Q] : T$

Proof. The cases 1, 2, 3 and follow by the corresponding cases of Lemma 3.7.7, by (i) choosing $\mathbf{0}$ as the continuation of the output process, and (ii) observing that $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle : T$ if and only if $\Gamma \vdash_\sigma \langle M_1, \dots, M_k \rangle \mathbf{0} : T$. In case 4, the typing of the redex implies, by Lemma 3.7.1(\mathbf{o}_1), that $\Gamma \vdash M_i : W_i$ and $\Gamma \vdash_\sigma P : T$ are derivable. Then the typing of the reduct derives by (ASYNCH OUTPUT) and the appropriate rule for parallel composition. Proof of 5 is similar. In case 6 the proof is similar to the corresponding case of Lemma 3.7.7 (in fact, the proof is simpler, and follows without appealing to Lemma 3.7.6). \square

Lemma 3.7.9 (Subject Congruence).

1. If $\Gamma \vdash_\sigma P : T$ and $P \equiv Q$ then $\Gamma \vdash_\sigma Q : T$.
2. If $\Gamma \vdash_\sigma P : T$ and $Q \equiv P$ then $\Gamma \vdash_\sigma Q : T$.

Proof. By simultaneous induction on the depths of the derivations of $P \equiv Q$ and $Q \equiv P$.

1. If $\Gamma \vdash_\sigma P : T$ and $P \equiv Q$ then $\Gamma \vdash_\sigma Q : T$.

(Struct Refl) The hypothesis is $P \equiv P$, and the proof follows directly from the assumption $\Gamma \vdash_\sigma P : T$.

(Struct Symm) Then $P \equiv Q$ derives from $Q \equiv P$. $\Gamma \vdash_\sigma Q : T$ follows by induction hypothesis **(2)**.

(Struct Trans) Then $P \equiv Q$ derives from $P \equiv R$ and $R \equiv Q$ for some R . From $P \equiv R$ and $\Gamma \vdash_\sigma P : T$, by induction hypothesis **(1)** $\Gamma \vdash_\sigma R : T$. From the last judgment, and from $R \equiv Q$, again by induction hypothesis **(1)** $\Gamma \vdash_\sigma Q : T$.

(Struct Par Assoc) We prove this case as representative of **(Struct Par Dead)** and **(Struct Par Comm)**. The hypotheses are $P \mid (Q \mid R) \equiv (P \mid Q) \mid R$ and $\Gamma \vdash_\sigma P \mid (Q \mid R) : T$. By Lemma 3.7.1, there exist E and F such that $[E, ?F, \mathcal{A}] \leq T$ and $\Gamma \vdash_\sigma P \mid (Q \mid R) : [E, ?F, \mathcal{A}]$. Also, we may safely focus on derivations from the assumptions defined by cases **(c₁)**–**(c₃)**.

(c₁) $T \leq [E, F, \mathcal{A}]$: By two further applications of Lemma 3.7.1.**(c₁)**, we know that the judgment in the hypothesis is derivable from $\Gamma \vdash_\sigma P : T$, $\Gamma \vdash_\sigma Q : T$ and $\Gamma \vdash_\sigma R : T$. Then the judgment $\Gamma \vdash_\sigma P \mid (Q \mid R) : T$ follows by two applications of **(PAR)**.

(c₂) $T = [E, {}^\circ F, \mathcal{A}]$. By two applications of Lemma 3.7.1.**(c₂)**, we can focus on the following cases:

- $\Gamma \vdash_\sigma P : [E, {}^\circ F, \mathcal{A}]$, $\Gamma \vdash_\sigma Q : [E, \bullet F, \mathcal{A}]$ and $\Gamma \vdash_\sigma R : [E, \bullet F, \mathcal{A}]$. To derive the judgment $\Gamma \vdash_\sigma (P \mid Q) \mid R : T$, apply **(PAR \circ)** twice and then subsumption.
- $\Gamma \vdash_\sigma P : [E, \bullet F, \mathcal{A}]$, $\Gamma \vdash_\sigma Q : [E, {}^\circ F, \mathcal{A}]$ and $\Gamma \vdash_\sigma R : [E, \bullet F, \mathcal{A}]$. As above, apply **(PAR \circ)** twice and then subsumption.
- $\Gamma \vdash_\sigma P : [E, \bullet F, \mathcal{A}]$, $\Gamma \vdash_\sigma Q : [E, \bullet F, \mathcal{A}]$ and $\Gamma \vdash_\sigma R : [E, {}^\circ F, \mathcal{A}]$. Apply **(PAR \bullet)** and then **(PAR \circ)**, followed by subsumption.

(c₃) $T = [E, {}^\Delta F, \mathcal{A}]$. The proof is similar to the previous case, with more sub-cases to consider, but no further difficulty.

(Struct Res Dead) The hypothesis is $(\nu x:W)\mathbf{0} \equiv \mathbf{0}$. The judgment $\Gamma \vdash_\sigma (\nu x:W)\mathbf{0} : T$ must come from $\Gamma, x:W \vdash_\sigma \mathbf{0} : T'$, for $T' \leq T$ (and $W = \sigma \text{Amb}^?[E, F, \mathcal{A}]$ for

some E, F and \mathcal{A}). Thus $\Gamma, x:W \vdash \diamond$ and hence Lemma 3.7.1.3 and 3.7.1.2 yield $\Gamma \vdash \diamond$. The judgment $\Gamma \vdash_{\sigma} \mathbf{0} : T$ derives now by (DEAD).

(Struct Res Res) The hypothesis is $(\nu x:W)(\nu y:W')P \equiv (\nu y:W')(\nu x:W)P$, and $\Gamma \vdash_{\sigma} (\nu x:W)(\nu y:W')P : T$ must have been derived from $\Gamma, x:W, y:W' \vdash_{\sigma} P : T'$, with $T \leq T$. From the last judgment, by two applications of the rule (NEW), followed by subsumption, we derive $\Gamma \vdash_{\sigma} (\nu y:W')(\nu x:W)P : T$ as desired.

(Struct Res Par) The hypothesis is $(\nu x:W)(P \mid Q) \equiv P \mid (\nu x:W)Q$ with $x \notin \text{fn}(P)$. The judgment $\Gamma \vdash_{\sigma} (\nu x:W)(P \mid Q) : T$ must have been derived from $\Gamma, x:W \vdash_{\sigma} (P \mid Q) : T^*$ with $T^* \leq T$. The last judgment, in turn, must have been derived from $\Gamma, x:W \vdash_{\sigma} P : T'$ and $\Gamma, x:W \vdash_{\sigma} Q : T''$ for suitable T' and T'' . From the first of the two judgments, by the Lemma 3.7.5.3 one deduces $\Gamma \vdash_{\sigma} P : T'$. From the second, by (NEW) $\Gamma \vdash_{\sigma} (\nu x:W)Q : T''$. Now $\Gamma \vdash_{\sigma} P \mid (\nu x:W)Q : T$ by the suitable rule for parallel composition.

(Struct Res Amb) The hypothesis is $(\nu x:W)M[P] \equiv M[(\nu x:W)P]$ with $x \notin \text{fn}(M)$. The judgment $\Gamma \vdash_{\sigma} (\nu x:W)M[P] : T$ must have been derived from $\Gamma, x:W \vdash M : \rho\text{Amb}^?[E, F, \mathcal{A}]$ and $\Gamma, x:W \vdash_{\rho} P : T'$ for a suitable T' . Let Ξ be the typing derivation from these two judgments to the judgment in the hypothesis. Since $x \notin \text{fn}(M)$, Lemma 3.7.1.3 applied to the first judgment implies that $\Gamma \vdash M : \rho\text{Amb}^?[E, F, \mathcal{A}]$. From the second judgment $\Gamma \vdash_{\rho} (\nu x:W)P : T'$ derives by (NEW). Now $\Gamma \vdash_{\sigma} M[(\nu x:W)P] : T$ derives by the same steps used in Ξ .

(Struct Path Assoc) The hypotheses are $(M.M').P \equiv M.(M'.P)$ and $\Gamma \vdash_{\sigma} (M.M').P : T$. We have to distinguish two cases depending on the type T , as informed by Lemma 3.7.1:

(p₁) $T \leq [E, F, \mathcal{A}]$: in this case $\Gamma \vdash_{\sigma} P : T$ and $\Gamma \vdash M.M' : \sigma\text{Cap}[F, \mathcal{A}]$. The last judgment must have been derived from $\Gamma \vdash M' : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma \vdash M' : \sigma\text{Cap}[F, \mathcal{A}]$ by the rule (PATH). Then, by two applications of (PREFIX), we have that $\Gamma \vdash_{\sigma} M.(M'.P) : T$ is derivable, as desired.

(p₂) $T = [E, {}^{\circ}F, \mathcal{A}]$ or $T = [E, {}^{\Delta}F, \mathcal{A}]$. We have the following subcases:

- $\Gamma \vdash_{\circ} M.M' : \rho\text{Cap}[G, \mathcal{B}]$ and $\Gamma \vdash_{\sigma} P : [E, {}^{\circ}F, \mathcal{A}]$. The first judgment must have been derived by (POLYPATH) from $\Gamma \vdash_{\circ} M : \tau\text{Cap}[H, \mathcal{C}]$, for some H ,

- \mathcal{C} , and τ , and from $\Gamma \vdash_{\circ} M' : \rho\text{Cap}[G, \mathcal{B}]$. Then $\Gamma \vdash_{\sigma} M.(M'.P) : [E, {}^{\circ}F, \mathcal{A}]$ derives by two applications of (PREFIX \circ).
- $\Gamma \vdash_{\circ} M.M' : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} P : [E, {}^{\Delta}F, \mathcal{A}]$. The first judgment must have been derived by (POLYPATH) from $\Gamma \vdash_{\circ} M : \rho\text{Cap}[H, \mathcal{B}]$ and $\Gamma \vdash_{\circ} M' : \sigma\text{Cap}[F, \mathcal{A}]$. Then $\Gamma \vdash_{\sigma} M.(M'.P) : [E, {}^{\circ}F, \mathcal{A}]$ derives by (PREFIX Δ) and (PREFIX \circ).

A final subsumption step derives $\Gamma \vdash_{\sigma} M.(M'.P) : [E, {}^{\Delta}F, \mathcal{A}]$, in both cases.

(Struct Repl) The hypothesis is $!P \equiv P \mid !P$. The judgment $\Gamma \vdash_{\sigma} !P : T$ must have been derived from $\Gamma \vdash_{\sigma} P : T'$ with $T' \leq [E, F, \mathcal{A}] \leq T$. If $T' = [E, F, \mathcal{A}]$, one derives $\Gamma \vdash_{\sigma} !P : T'$ by (REPL) and then $\Gamma \vdash_{\sigma} P \mid !P : T$ by (PAR) followed by subsumption. If $T' = [E, {}^{\bullet}F, \mathcal{A}]$, one derives $\Gamma \vdash_{\sigma} !P : T'$ by (REPL \bullet) and then $\Gamma \vdash_{\sigma} P \mid !P : T$ by (PAR \bullet) followed by subsumption.

(Struct Cong Par) The hypothesis is $P \mid R \equiv Q \mid R$ derived from $P \equiv Q$. The judgment $\Gamma \vdash_{\sigma} P \mid R : T$ must have been derived from $\Gamma \vdash_{\sigma} P : T'$ and $\Gamma \vdash_{\sigma} R : T''$ for suitable T' and T'' . Let Ξ the partial derivation from these two judgments to $\Gamma \vdash_{\sigma} P \mid R : T$. By the induction hypothesis (1) on $P \equiv Q$, one has $\Gamma \vdash_{\sigma} Q : T'$. Then $\Gamma \vdash_{\sigma} Q \mid R : T$ derives by the same steps used in Ξ .

The remaining congruence cases, namely **(Struct Cong Action)**, **(Struct Cong Agent)**, **(Struct Cong Input)**, **(Struct Cong Output)**, **(Struct Cong New)**, **(Struct Cong Repl)** are all proved similarly to the previous case.

2. If $\Gamma \vdash_{\sigma} P : T$ and $Q \equiv P$ then $\Gamma \vdash_{\sigma} Q : T$.

The proof follows by the same analysis of case 1. In case **(Struct Symm)**, we use the induction hypothesis (1), in place of (2). In all the remaining cases, we use the induction hypothesis (2) in place of (1). The only nontrivial cases are **(Struct Par Assoc)**, which is symmetric to the corresponding subcase of 1, and **(Struct Path Assoc)**, which we give below.

(Struct Path Assoc) The hypotheses are $M.(M'.P) \equiv (M.M').P$ and $\Gamma \vdash_{\sigma} M.(M'.P) : T$. We have to distinguish two cases depending on the type T , as informed by Lemma 3.7.1:

- (p1) $T \leq [E, F, \mathcal{A}]$: in this case $\Gamma \vdash M : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} M'.P : T$. By

Lemma 3.7.1.(p₁), applied to the judgment $\Gamma \vdash_{\sigma} M'.P : T$, it follows that $\Gamma \vdash M' : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} P : T$. Then $\Gamma \vdash M.M' : \sigma\text{Cap}[F, \mathcal{A}]$ derives from (PATH), and $\Gamma \vdash_{\sigma} (M.M').P : T$ by (PREFIX).

(p₂) $T = [E, {}^{\circ}F, \mathcal{A}]$ or $T = [E, {}^{\Delta}F, \mathcal{A}]$. We have two sub-cases.

In the first sub-case, one has $\Gamma \vdash^{\circ} M : \rho\text{Cap}[G, \mathcal{B}]$ and $\Gamma \vdash M'.P : [E, {}^{\circ}F, \mathcal{A}]$. By Lemma 3.7.1.(p₁), applied to the judgment $\Gamma \vdash_{\sigma} M'.P : [E, {}^{\circ}F, \mathcal{A}]$, it follows that either $\Gamma \vdash^{\circ} M' : \tau\text{Cap}[H, \mathcal{C}]$ and $\Gamma \vdash_{\sigma} P : [E, {}^{\circ}F, \mathcal{A}]$, or $\Gamma \vdash^{\circ} M' : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} P' : [E, {}^{\Delta}F, \mathcal{A}]$. In the first case apply (POLYPATH) to derive $\Gamma \vdash M.M' : \tau\text{Cap}[H, \mathcal{C}]$, and then (PREFIX \circ). In the second case, apply (POLYPATH) to derive $\Gamma \vdash M.M' : \sigma\text{Cap}[F, \mathcal{A}]$, and then (PREFIX Δ).

In the second sub-case, $\Gamma \vdash^{\circ} M : \sigma\text{Cap}[F, \mathcal{A}]$ and $\Gamma \vdash_{\sigma} M'.P : [E, {}^{\Delta}F, \mathcal{A}]$, and the proof is similar to the one just given.

□

Theorem 3.7.10 (Type preservation for tagged reduction). *Given a type environment Γ , say that a security assignment Λ is Γ -consistent if and only if for all $x \in \text{dom}(\Gamma)$, $\Gamma(x) = \sigma\text{Amb}^?[\dots]$ implies $\Lambda(x) = \sigma$. Now assume that $\Gamma \vdash_{\sigma} P : T$. Then for every Γ -consistent assignment Λ and process Q such that $P \rightarrow_{(\sigma, \Lambda)} Q$, we have $\Gamma \vdash_{\sigma} Q : T$.*

Proof. By induction on the derivation of $P \rightarrow_{(\sigma, \Lambda)} Q$. The cases of top-level move reductions follow by Lemmas 3.7.3, 3.7.4. The cases of communication follows by Lemma 3.7.7. As for the inductive steps, the proof is guided by the inductive cases of definition of the tagged reduction (cf. Section 3.4):

- case (PAR) follows by the inductive hypothesis and an application of the appropriate (PAR ?) rule
- case (STRUCT) follows by the induction hypothesis and Lemma 3.7.9.
- case (NEW) follows again by the induction hypothesis, as the side condition to the rule guarantees that the assignment $\Lambda, (x : \rho)$ is $(\Gamma, x : A)$ -consistent.
- case (AMB) also follows by the induction hypothesis, as Λ being Γ -consistent implies that $\Gamma \vdash_{\sigma} a[P] : T$ depends on $\Gamma \vdash_{\Lambda(a)} P : T'$ for a suitable T' .

□

Theorem 3.7.11 (Type Soundness). *If $\Gamma \vdash_\sigma P : T$, then for every Γ -consistent security assignment, and process Q such that $P \rightarrow_{(\sigma, \Lambda)}^* Q$ we have $Q \not\rightarrow_{(\sigma, \Lambda)} \text{err}$.*

Proof. We first show that $P \not\rightarrow_{(\sigma, \Lambda)} \text{err}$: to show that, it is more convenient to prove the contrapositive, namely that $P \rightarrow_{(\sigma, \Lambda)} \text{err}$ implies $\Gamma \not\vdash_\sigma P : T$. The proof proceeds by induction on the derivation of $P \rightarrow_{(\sigma, \Lambda)} \text{err}$.

- For the basis of induction, we assume, for the purpose of contradiction, that $P \rightarrow_{(\sigma, \Lambda)} \text{err}$ and $\Gamma \vdash_\sigma P : T$. Then we have four possible cases: we give the proof of the case (OUTPUT n) and (OUTPUT \uparrow) as representative.

In case (OUTPUT n) the redex is $\langle \tilde{M} \rangle^n P \mid n[(\tilde{x}:\tilde{W})Q \mid R]$, and the hypothesis $\langle M \rangle^n P \mid n[(\tilde{x}:\tilde{W})Q \mid R] \rightarrow_{(\sigma, \Lambda)} \text{err}$ implies that $\neg \mathcal{P}(\sigma, \Lambda(n), \mathbf{w})$. Since Λ is Γ -compatible, it must be the case that $\Gamma \vdash \Lambda(n) \text{Amb}^?[W, ?E, \mathcal{A}]$ for suitable W , E , and \mathcal{A} . From our hypothesis $\Gamma \vdash_\sigma \langle M \rangle^n P : T$, by Lemma 3.7.1.(o₂), we know that $\mathcal{P}(\sigma, \Lambda(n), \mathbf{w})$: contradiction.

In case (OUTPUT \uparrow) the redex has the form $(\tilde{x}:\tilde{W})P \mid n[\langle \tilde{M} \rangle^\uparrow Q \mid R]$ and the reduction to err implies that $\neg \mathcal{P}(\Lambda(n), \sigma, \mathbf{w})$. Since Λ is Γ -compatible, it must be the case that $\Gamma \vdash \Lambda(n) \text{Amb}^?[-, -]$ for suitable types and access modes. From our hypothesis $\Gamma \vdash_\sigma (\tilde{x}:\tilde{W})P \mid n[\langle M \rangle^\uparrow Q \mid R] : T$ by repeated applications of Lemma 3.7.1, it follows that the derivation depends on the side-condition $\mathcal{P}(\Lambda(n), \sigma, \mathbf{w})$ to the (AMB) or the (AMB Δ) rule yielding the sought contradiction (that the judgment is derived by either of these rules, and not by (AMB \circ) follows by the same reasoning as in case 5 of Lemma 3.7.7).

- For the inductive steps we have to consider several cases. If $P \rightarrow_{(\sigma, \Lambda)} \text{err}$ by (ERR STRUCT), then there exists Q such that $P \equiv Q$ and $Q \rightarrow_{(\sigma, \Lambda)} \text{err}$. By the induction hypothesis, this implies $\Gamma \not\vdash_\sigma Q : T$, and then proof follows by Lemma 3.7.9. The cases in which $P \rightarrow_{(\sigma, \Lambda)} \text{err}$ by any of the contextual reductions follow directly by the induction hypothesis, noting that a process term is well-typed if and only if so are all of its sub-terms.

The proof of the theorem follows now by subject reduction. From $\Gamma \vdash_\sigma P : T$ and $P \rightarrow_{(\sigma, \Lambda)}^* Q$, by Theorem 3.7.10 we know that $\Gamma \vdash_\sigma Q : T$, and we have just proved that this implies that $Q \not\rightarrow_{(\sigma, \Lambda)} \text{err}$. \square

3.8 Summary of Type System

We list the complete set of rules for the type systems described in Sections 3.4 . In order to have a more compact set of rules, we use ${}^\mu F$ to denote any of the exchanges ${}^\Delta F, {}^\bullet F, {}^\circ F$, and use ${}^? F$ to denote either ${}^\mu F$ or F . Similarly we use $\text{Amb}^?[E, F, \mathcal{A}]$ to denote either $\text{Amb}[E, F, \mathcal{A}]$ or $\text{Amb}^\circ[E, F, \mathcal{A}]$.

Expressions

(PROJECTION)	(POLYCAP)	(POLYPATH)
$\frac{\Gamma(a) = W}{\Gamma \vdash a : W}$	$\frac{\Gamma \vdash M : \sigma\text{Cap}[E, \mathcal{A}]}{\Gamma \vdash M : \sigma\text{Cap}[E, \mathcal{A}]}$	$\frac{\Gamma \vdash M_1 : \rho\text{Cap}[F, \mathcal{A}] \quad \Gamma \vdash M_2 : \sigma\text{Cap}[E, \mathcal{B}]}{\Gamma \vdash M_1.M_2 : \sigma\text{Cap}[E, \mathcal{B}]}$
(PATH)	(OUT)	
$\frac{\Gamma \vdash M_1 : \sigma\text{Cap}[E, \mathcal{A}] \quad \Gamma \vdash M_2 : \sigma\text{Cap}[E, \mathcal{A}]}{\Gamma \vdash M_1.M_2 : \sigma\text{Cap}[E, \mathcal{A}]}$	$\frac{\Gamma \vdash M : \sigma\text{Amb}[E, F, \mathcal{B}] \quad G \leq F, \mathcal{A} \leq \mathcal{B}}{\Gamma \vdash \text{out } M : \rho\text{Cap}[G, \mathcal{A}]}$	
(OUT \circ)	(IN)	
$\frac{\Gamma \vdash M : \sigma\text{Amb}^\circ[E, F, \mathcal{B}]}{\Gamma \vdash \text{out } M : \rho\text{Cap}[\text{shh}, \mathcal{A}]}$	$\frac{\Gamma \vdash M : \rho\text{Amb}^?[E, F, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathcal{A}) \quad G \leq E}{\Gamma \vdash \text{in } M : \sigma\text{Cap}[G, \mathcal{A}]}$	

Processes

(PREFIX)	(PREFIX \circ)	
$\frac{\Gamma \vdash M : \sigma\text{Cap}[F, \mathcal{A}] \quad \Gamma \vdash_\sigma P : [E, F, \mathcal{A}]}{\Gamma \vdash_\sigma M.P : [E, F, \mathcal{A}]}$	$\frac{\Gamma \vdash M : \rho\text{Cap}[G, \mathcal{B}] \quad \Gamma \vdash_\sigma P : [E, {}^\circ F, \mathcal{A}]}{\Gamma \vdash_\sigma M.P : [E, {}^\circ F, \mathcal{A}]}$	
(PREFIX Δ)	(PREFIX \bullet)	
$\frac{\Gamma \vdash M : \sigma\text{Cap}[F, \mathcal{A}] \quad \Gamma \vdash_\sigma P : [E, {}^\Delta F, \mathcal{A}]}{\Gamma \vdash_\sigma M.P : [E, {}^\circ F, \mathcal{A}]}$	$\frac{\Gamma \vdash M : \sigma\text{Cap}[F, \mathcal{A}] \quad \Gamma \vdash_\sigma P : [E, {}^\bullet F, \mathcal{A}]}{\Gamma \vdash_\sigma M.P : [E, {}^\bullet F, \mathcal{A}]}$	
(PAR)	(PAR μ)	
$\frac{\Gamma \vdash_\sigma P : [E, F, \mathcal{A}] \quad \Gamma \vdash_\sigma Q : [E, F, \mathcal{A}]}{\Gamma \vdash_\sigma P \mid Q : [E, F, \mathcal{A}]}$	$\frac{\Gamma \vdash_\sigma P : [E, {}^\mu F, \mathcal{A}] \quad \Gamma \vdash_\sigma Q : [E, {}^\bullet F, \mathcal{A}]}{\Gamma \vdash_\sigma P \mid Q, Q \mid P : [E, {}^\mu F, \mathcal{A}]}$	
(DEAD)	(NEW)	(REPL \bullet)
$\frac{\Gamma \vdash \diamond}{\Gamma \vdash_\sigma \mathbf{0} : T}$	$\frac{\Gamma, n : \rho\text{Amb}^?[E, F] \vdash_\sigma P : T}{\Gamma \vdash_\sigma (\nu n : \rho\text{Amb}^?[E, F])P : T}$	$\frac{\Gamma \vdash_\sigma P : [E, {}^\bullet F, \mathcal{A}]}{\Gamma \vdash_\sigma !P : [E, {}^\bullet F, \mathcal{A}]}$

$$\begin{array}{c}
\text{(REPL)} \\
\frac{\Gamma \vdash_{\sigma} P : [E, F, \mathcal{A}]}{\Gamma \vdash_{\sigma} !P : [E, F, \mathcal{A}]}
\end{array}
\qquad
\begin{array}{c}
\text{(SUBSUMPTION)} \\
\frac{\Gamma \vdash_{\sigma} P : T \quad T \leq T'}{\Gamma \vdash_{\sigma} P : T'}
\end{array}$$

$$\begin{array}{c}
\text{(AMB)} \\
\frac{\Gamma \vdash M : \sigma \text{Amb}[E, F, \mathcal{A}] \quad \Gamma \vdash_{\sigma} P : [E, F, \mathcal{A}] \quad \mathcal{P}(\sigma, \rho, \mathcal{A})}{\Gamma \vdash_{\rho} M[P] : [F, \bullet H, \mathcal{B}]}
\end{array}$$

$$\begin{array}{c}
\text{(AMB } \Delta) \\
\frac{\Gamma \vdash M : \sigma \text{Amb}^{\circ}[E, F, \mathcal{A}] \quad \Gamma \vdash_{\sigma} P : [E, \Delta F, \mathcal{A}] \quad \mathcal{P}(\sigma, \rho, \mathcal{A})}{\Gamma \vdash_{\rho} M[P] : [F, \bullet H, \mathcal{B}]}
\end{array}$$

$$\begin{array}{c}
\text{(AMB } \circ) \\
\frac{\Gamma \vdash M : \sigma \text{Amb}^{\circ}[E, F, \mathcal{A}] \quad \Gamma \vdash_{\sigma} P : [E, \circ F, \mathcal{A}]}{\Gamma \vdash_{\rho} M[P] : [G, \bullet H, \mathcal{B}]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } \star) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{\sigma} P : [W_1 \times \dots \times W_k, ?F, \mathcal{A}]}{\Gamma \vdash_{\sigma} (\tilde{x} : \tilde{W})P : [W_1 \times \dots \times W_k, ?F, \mathcal{A}]}
\end{array}
\qquad
\begin{array}{c}
\text{(OUTPUT } \star \quad i = 1, \dots, k) \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_{\sigma} P : [W_1 \times \dots \times W_k, ?F, \mathcal{A}]}{\Gamma \vdash_{\sigma} \langle M_1, \dots, M_k \rangle P : [W_1 \times \dots \times W_k, ?F, \mathcal{A}]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } M) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{\sigma} P : [E, {}^{\mu}F, \mathcal{A}] \quad \Gamma \vdash M : \rho \text{Amb}^?[W_1 \times \dots \times W_k, G, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathbf{r})}{\Gamma \vdash_{\sigma} (\tilde{x} : \tilde{W})^M P : [E, {}^{\mu}F, \mathcal{A}]}
\end{array}$$

$$\begin{array}{c}
\text{(OUTPUT } M) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash N_i : W_i \quad \Gamma \vdash_{\sigma} P : [E, {}^{\mu}F, \mathcal{A}] \quad \Gamma \vdash M : \rho \text{Amb}^?[W_1 \times \dots \times W_k, G, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, \mathbf{w})}{\Gamma \vdash_{\sigma} \langle N_1, \dots, N_k \rangle^M P : [E, {}^{\mu}F, \mathcal{A}]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } \uparrow) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{\sigma} P : [F, W_1 \times \dots \times W_k, \mathcal{A}] \quad \mathbf{r} \leq \mathcal{A}}{\Gamma \vdash_{\sigma} (\tilde{x} : \tilde{W})^{\uparrow} P : [F, W_1 \times \dots \times W_k, \mathcal{A}]}
\end{array}
\qquad
\begin{array}{c}
\text{(INPUT } \uparrow \Delta) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{\sigma} P : [F, \Delta W_1 \times \dots \times W_k, \mathcal{A}] \quad \mathbf{r} \leq \mathcal{A}}{\Gamma \vdash_{\sigma} (\tilde{x} : \tilde{W})^{\uparrow} P : [F, \Delta W_1 \times \dots \times W_k, \mathcal{A}]}
\end{array}$$

$$\begin{array}{c}
(\text{OUTPUT } \uparrow) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash_{\sigma} M_i : W_i \quad \Gamma \vdash P : [F, W_1 \times \dots \times W_k, \mathcal{A}] \quad w \leq \mathcal{A}}{\Gamma \vdash_{\sigma} \langle M_1, \dots, M_k \rangle^{\uparrow} P : [F, W_1 \times \dots \times W_k, \mathcal{A}]} \\
\\
(\text{OUTPUT } \uparrow \Delta) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_{\sigma} P : [F, {}^{\Delta}(W_1 \times \dots \times W_k), \mathcal{A}] \quad w \leq \mathcal{A}}{\Gamma \vdash_{\sigma} \langle M_1, \dots, M_k \rangle^{\uparrow} P : [F, {}^{\Delta}(W_1 \times \dots \times W_k), \mathcal{A}]}
\end{array}$$

For the asynchronous version of the calculus we need two additional typing rule for the process forms $\langle M_1, \dots, M_k \rangle$ and $\langle M_1, \dots, M_k \rangle^M$.

$$\begin{array}{c}
(\text{ASYNCH OUTPUT}) \\
\frac{\Gamma \vdash M_i : W_i \quad i = 1, \dots, k}{\Gamma \vdash_{\sigma} \langle M_1, \dots, M_k \rangle : [W_1 \times \dots \times W_k, {}^?F, \mathcal{A}]} \\
\\
(\text{OUTPUT } M) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash N_i : W_i \quad \Gamma \vdash M : \rho \text{Amb}^?[W_1 \times \dots \times W_k, G, \mathcal{B}] \quad \mathcal{P}(\sigma, \rho, w)}{\Gamma \vdash_{\sigma} \langle N_1, \dots, N_k \rangle^M : [E, {}^{\mu}F, \mathcal{A}]}
\end{array}$$

Chapter 4

Information Flow Security

In this chapter we study the problem of secure information flow for Boxed Ambients in terms of non-interference. We develop a sound type system that provides static guarantees of absence of unwanted flow of information for well-typed processes. Non-interference is stated, and proved, in terms of a typed notion of contextual equivalence for Boxed Ambients that akin to the corresponding equivalence defined for Mobile Ambients.

4.1 Introduction

The type system we defined in Chapter 3 was targeted at resource access control, and specifically designed to protect resources, viz. channels, from undesired uses by unauthorized clients. Here, we change perspective, and focus on a different analysis that targets information flow. To motivate the change in perspective, consider the following example, where ℓ is a “low-level” ambient and h a “high-level” one:

$$\ell[(x)^h P \mid h[\langle M \rangle Q \mid R]] \quad (4.1)$$

In the system of Chapter 3 the attempt by the process enclosed in the low-level ambient ℓ to read from h is classified as a *read-up*, and therefore rejected as “insecure” by both military and commercial security, regardless of the information content of M . On the other hand, if the expression M is public, i.e., low-level, there is no reason for disallowing the read access: a piece of data is flowing from high to low, but the flow is “secure” as the piece of data carries a “low” information content.

The goal of the present chapter is to provide static safeguards against “unsafe” flow of information. A first, rather intuitive, notion of information flow may directly be related to

the flow of data: a system is “secure” if no high-level data flows from high-level to low-level principals. This form of secure information flow is easily accounted for, and enforced, by a static control over the transport layer used for data communication, viz channels. If we classify data and channels according to their security levels, absence of this form of “explicit” flow of information can be guaranteed by requiring that:

- (★) High-level data be only communicated along high-level channels, and high-level channels be only located within high-level subjects.

A subtler, and more interesting, notion of information flow security is related to the presence of implicit flow of information, resulting from indirect ways of transmitting information (namely, covert channels) via system-wide side effects. To illustrate, consider the following specialization of the system (4.1) above, where P is “low level”:

$$\ell[(x)^h \langle N \rangle \mid h[\langle M \rangle]] \mid (x)^\ell P \quad (4.2)$$

Assuming that M and N are low-level values, there is no direct flow in this system. However, a covert channel is established between P and the ambient h , as P is unleashed by an exchange that depends on the presence of the high ambient h , and the very presence (or absence) of a high-level ambient can be assimilated to a bit of high-level information that, in the system in question, flows downwards.

As a more ‘concrete’ example, consider the following process that implements a server that progressively distributes a token to incoming users:

$$\text{Server}[h[\langle 1 \rangle] \mid !(n)^h \langle n \rangle^{\text{usr}} h[\langle n+1 \rangle]] \mid \text{usr}[\text{in Server}.(n)P]$$

In the above process a user that enters twice (or more) in the server can easily get to know how many tokens have been requested in the meanwhile; it can also get to know how many tokens are available in total or how often the service is updated (that is, the number of tokens starts again from 1). An information flow then arises from the server manager to the user. The main problem with this example is in the implementation of the process that sends a token to the user: sending a public token indeed depends on a higher level action, that is the management of the counter ambient h .

Another subtle example of implicit information flow is represented by an agent a that simply navigates a file system to reach a precise file f on which it has access rights. The

structure of the file system can be modified by the system manager, then the fact that a can no more reach file f , tells a that something has happened at 'high level'. Again, the result is that a bit of high information has flown downwards via a system-wide side effect.

Information Flow Security and Non-interference

Defining what is exactly meant by (implicit) information flow can be hard (perhaps impossible), and various authors have instead relied on *non-interference*, a concept of easier formalization which implies absence of flow.

The notion of non-interference was first proposed by Goguen and Meseguer [GM82] for deterministic state machines. The idea is to determine whether in a given system the “inputs” of high level subjects (or “users”) may influence, i.e. interfere with, the “outputs” of low level subjects. If the latter are invariant on the former, then the system is decreed interference free.

Non-interference was later [FG95] reformulated in a CCS-like process calculus as the so-called *Non Deducibility on Composition* (NDC) property, which implies that low-level observers are insensitive to the presence of high-level components (sources) in the system. Here we take the same approach, and rephrase the NDC property to capture ambient-based specific aspects of computation, namely, locality and mobility.

Overview

Our technique for providing guarantees of non-interference in BA is based on static typing. We rely on types both to formalize the notions of high and low data and processes, and to define the relation of process equivalence underlying the definition of non-interference.

The type system is based on ambient and capability types akin to those employed in companion type systems for MA. In addition, the types of our type system carry security annotations that define the security clearance of values and processes. Processes have the clearance of their enclosing ambient, while values (i.e. capabilities and names) are assigned security levels as follows: names have the security level associated to their type, while capabilities are decreed “low-level” data, based on the observation that capabilities do not disclose their target ambient names, and hence provide rather limited control over such names.

Having partitioned data and processes into “high” and “low”, we then single out the set \mathcal{H} of *high level sources* as the set of all processes that can only produce high-level “inputs”, where an “input” corresponds to the presence, at top level, either of a communication, or an ambient, or a mobility action.

As the next step of our formalization, we introduce a relation of behavioral equivalence to compare processes. This relation is a typed version of the equivalence relation introduced in [CG99a] for MA: a *contextual equivalence* that equates two processes if and only if they admit the same elementary observations whenever they are inserted inside any arbitrary, but well-typed, enclosing context. Our *observability predicate* is akin to the one studied in [CG99a], but refined to capture the core form of interaction between Boxed Ambients, namely, the ability for an ambient to exchange values along its local channel. We thus say that a process P exhibits a name n if P (reduces, in any number of steps, to a process that) contains an ambient n that may accept interactions with the external environment, that is if P (or any of the processes it reduces to) is structurally equivalent to $(\nu m_1) \dots (\nu m_k)(n[\langle M \rangle P' \mid Q' \mid Q''])$ where $n \notin \{m_1, \dots, m_k\}$. Even though this notion of observation is specifically focused on communication, ambient mobility is still observed, indirectly, via its consequences on upward communications, as the following example illustrates (the presence of a high level ambient h triggers the upward communication of the low-level ambient ℓ):

$$\ell[\text{in } h.\text{out } h.\langle M \rangle] \mid h[]$$

Finally, we introduce the notion of contextual equivalence induced by a *low level observation*: two (well-typed) processes P and Q are equivalent, $P \cong_L Q$, if whenever they are inserted inside an arbitrary (well-typed) context, they exhibit the same *low level* names. Based on that, we can phrase the NDC property of [FG95] for BA as:

$$(\star\star) \quad \text{A process } P \text{ is } \textit{interference free} \text{ if and only if } \forall H \in \mathcal{H} \quad P \mid H \cong_L P.$$

As in [FG95] non interference of P is checked only against high-level sources that appear in parallel with P . This is rather natural in that context, since the topology of CCS processes is completely flat. On the contrary, in BA ambients may be nested arbitrarily and, consequently, a high-level process interacting with P may also (i) enclose P or (ii) be enclosed within P . It would then appear that our definition of non-interference should be generalized to capture these additional cases. Indeed, the definition, as given, does address

these cases as ambients running in parallel may nest arbitrarily as a result of mobility.

The Chapter ends with a proof that well-typed processes are interference free, in the sense we just outlined. The non-interference proof builds on the technical tools developed in [CG99a] by Cardelli and Gordon for MA, adapting them to BA, and relies critically on the choice of contextual equivalence as the underlying equivalence relation. In fact, as we discuss in Section 4.3, our present results do not extend to finer equivalence relationships, such as barbed congruence [MS92].

4.2 A Type System for Secure Information Flow

In this section we enrich the simple type system of BA (cf. §2.4) so as to provide static safeguards against insecure flow of information in the evolution of well-types processes. As a matter of simplicity we do not consider moded typing, whose aim was to enhance the typing of mobility. Nevertheless the technique developed in this section can be adapted also to moded types and their more flexible type system.

We presuppose a complete lattice of security levels (Σ, \leq) , and let ρ, σ, δ range over security levels. We then partition the elements of this lattice into two classes, “high” and “low”, as formalized in the following definition.

Definition 4.2.1 (Low and High levels). Let (Σ, \leq) be a complete lattice of security levels. A *security classification* is a partition of Σ into two non-empty sets L and H , with L downward closed. Based on this classification, we then define the following order:
 $\rho \preceq \sigma \triangleq (\rho \in L \vee \sigma \in H)$ □

4.2.1 Types and Judgments

The types E of exchanges, and the types of processes are defined as in Section 2.4. The types of expressions are redefined as follows:

<i>Expression Types</i>	$W ::=$	$\sigma \text{Amb}[E, F]$	ambient
		$\sigma \text{ucap}[E]$	unsafe capability
		$\sigma \text{scap}[E]$	safe capability

Each ambient type is annotated with a security level that defines the clearance of the ambient names with that type. Capability types also have an associated security level,

and are partitioned into safe and unsafe. In particular, ucap is the type of dangerous capabilities, those that are potential sources of flow of information: the typing rules will ensure that such capabilities may only be exercised within high-level ambients. Capability types are also annotated with a security level: while the annotation of ambient types is used to *assign* a security level to an ambient, the annotations of capability types are used to *record* the security of the actions performed by a process¹. The intuition is that in $\sigma\mathsf{Cap}[E]$ (with $\mathsf{Cap} \in \{\mathsf{ucap}, \mathsf{scap}\}$), σ is the greatest lower bound of (the security levels of) the capabilities on a path. This is formalized by the following “cap-type” composition:

$$\begin{aligned} \triangleright \sigma\mathsf{scap}[E] \cdot \delta\mathsf{scap}[E] &= (\sigma \sqcap \delta)\mathsf{scap}[E] \\ \triangleright \sigma\mathsf{ucap}[E] \cdot \delta\mathsf{ucap}[E] &= \sigma\mathsf{scap}[E] \cdot \delta\mathsf{ucap}[E] = \delta\mathsf{ucap}[E] \cdot \sigma\mathsf{scap}[E] = (\sigma \sqcap \delta)\mathsf{ucap}[E] \end{aligned}$$

where \sqcap is relative to the order \preceq introduced in Definition 4.2.1.

The next step is to determine the security clearance of the values that are exchanged in a process communication. This is formalized by the following *level* function $\Lambda : \mathsf{Exchange Types} \rightarrow \mathsf{Security Levels}$, where $\mathsf{Cap} \in \{\mathsf{ucap}, \mathsf{scap}\}$, and \perp is the bottom element in the lattice of security levels.

$$\begin{aligned} \Lambda(\sigma\mathsf{Amb}[E, F]) &= \sigma \\ \Lambda(W_1 \times \cdots \times W_n) &= \sqcup\{\Lambda(W_1), \dots, \Lambda(W_n)\} \\ \Lambda(\mathsf{shh}) = \Lambda(\sigma\mathsf{Cap}[E]) &= \perp \end{aligned}$$

As we anticipated in the Introduction, we are thus stipulating that capabilities should always be considered “low-level” values, as passing a capability does not disclose the name occurring in the capability. Notice, furthermore, that the type of a capability does trace the level of the target ambient: this information is needed to detect flows of information resulting from exercising (as opposed to exchanging) the capability in question.

The type system is defined in terms of the following classes of judgments.

¹Note that in this chapter, the role of σ in the capability types is different from that in Section 3.4. In Chapter 3 in fact, $\sigma\mathsf{Cap}[E, \mathcal{A}]$ was the type of a capability that could be exercised within an ambient with clearance σ . In this chapter, instead, σ records the level of the ambient traversed.

$\Gamma \vdash \diamond$	Well-formed Type Environment
$\Gamma \vdash E$	Well-formed Exchange Type
$\Gamma \vdash_{\sigma} [E, F]$	Well-formed Process Type at level σ
$\Gamma \vdash M : W$	Well-typed Expression
$\Gamma \vdash_{(\sigma, \rho)} P : [E, F]$	Well-typed Process

The judgments for well-formed (exchange and process) types are functional to enforce a safe flow of data along the (anonymous) communication channels inside and across ambient boundaries. In the judgment for well-typed processes, we use two annotations on the turnstile, with the following intended meaning: σ is the clearance of the ambient enclosing P (if any), while ρ is the lower-bound on the clearance of the actions encountered so far, and it helps define the clearance at which P should type-check. To understand the rationale of the typing rules, consider the following examples (as usual, ℓ denotes a low-level, while h and h_1 are high-level).

- ▷ the process $\ell[(x)^h\langle M \rangle]$ is not safe, because the observable message M is enabled as a result of ℓ exchanging a value with the high-level subambient h . Observing a local communication on ℓ (trying to read it) may thus reveal the presence of the high level ambient h within ℓ . The very same reasoning shows that, instead, $\ell[(x)^h\mathbf{0} \mid \langle M \rangle]$ is a secure process.

Flows of information may arise from subtler combinations of high-level and low-level actions. In particular, such actions need not occur sequentially as suggested by the example above. An implicit flow of information, may also arise as a result of running two parallel threads:

- ▷ the process $\ell_1[(x)^h() \mid \langle \rangle^{\uparrow}] \mid ()\ell[P]$ is not secure because the ambient ℓ is observable at top level if ℓ_1 reads from the high level ambient h . In particular, we have that the three threads are “linked” by the local and upward synchronizations, thus determining a causal dependency, and hence an implicit flow of information.

Both the previous examples, show that “secure” processes should satisfy a very basic invariant, namely that “actions” following a high-level synchronization (like $(x)^h$, independently from the level of the piece of information x read from h) must not be available

for further low-level-context interactions. However, processes of the form $(x)^h \langle x \rangle^{h_1}$ are safe also within a low level ambient. This explains the role of ρ in the typing judgment of processes. When prefixing a process P with an “action” (where action means capability, communication, and top level presence of an ambient), that action should have clearance not lower than ρ . In other words, ρ should be non-decreasing as a well-typed process progresses. However, this condition is not sufficient by itself.

▷ consider the process $P = h[\langle M \rangle \mid \ell[(x)^\uparrow \text{out } h.\mathbf{0}]]$, where a low-level ambient ℓ first reads upward, and then exits from the high-level location h . In this case, the very presence, at top level, of the ambient ℓ represents a public (low-level) information that depends in a private (high-level) one. This is a problem, as the ability to test the presence of ℓ at top level may, implicitly, reveal the presence of h to any low-level observer. More importantly, even the simpler process $P' = \ell[(x)^\uparrow \text{out } h.\mathbf{0}]$ is an unsafe process. To see the problem, and phrase it in terms of non-interference, we may encode the observer as the context: $\mathbf{C}() = \ell_1[\text{in } \ell.\text{out } \ell.\langle N \rangle] \mid h[()]$. Now, taking $H = \langle M \rangle$, a routine check verifies that the context distinguishes P from $P \mid H$.

This last example shows that low-level ambients exiting high-level locations may potentially disclose secret information about that high-level location. This suggests (i) that the `out` capability should be deemed unsafe when the target ambient is high-level, and (ii) that only high-level ambients should be allowed to exercise such capability.

4.2.2 Typing Rules

Environment and Type Formation

As we anticipated, the rules for well-formed types provide safeguards against explicit flows, in that they guarantee that σ -level values only circulate over channels (or ambients) with higher clearance. Thanks to the locality principle of Boxed Ambients (each ambient contains a local anonymous channel, cf. §2.2), this is obtained requiring the clearance σ of an ambient to be an upper bound of the clearance of its local exchanges. Such a constraint is made explicit in rule (TYPE LOW AMB), while it is implicit in rule (TYPE HIGH AMB).

$\frac{}{\emptyset \vdash \diamond}$	$\frac{\Gamma \vdash T \quad a \notin \text{Dom}(\Gamma)}{\Gamma, a : W \vdash \diamond}$	$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \text{shh}}$	$\frac{\Gamma \vdash E}{\Gamma \vdash \sigma\text{Cap}[E]}$
$\frac{\sigma \in \mathbf{L} \quad \Gamma \vdash E, F \quad \Lambda(E) \preceq \sigma}{\Gamma \vdash \sigma\text{Amb}[E, F]}$	$\frac{\sigma \in \mathbf{H} \quad \Gamma \vdash E, F \quad \Lambda(F) \in \mathbf{H}}{\Gamma \vdash \sigma\text{Amb}[E, F]}$	$\frac{\Gamma \vdash E, F \quad \Lambda(E) \preceq \sigma}{\Gamma \vdash_{\sigma} [E, F]}$	

Rule (TYPE HIGH AMB) has an additional constraint enforcing a high level ambient to upward exchange only high level values. Consider in fact the process $P = \ell[(x : W_L)\langle M \rangle]$, where W_L is a low level expression type. There is no reason for considering P an unsafe process, on the other hand the local read contained into ℓ may synchronize with a high level ambient $H = h[\text{in } \ell.\langle N \rangle^{\uparrow}]$, as long as N is a low level value of type W_L . Even if ℓ cannot know whether it synchronizes with a low or a high level ambient, we have that an external observer notices the difference between the behavior of P and $P \mid H$, that is, an interference is pointed out. As we said, nothing is wrong with P , the unsafe action yielding an observable interaction between high and low level is caused by the low upward output contained in h . Therefore, rule (TYPE HIGH AMB) is used to discard such subtle malicious flows.

On the other hand, local channels contained within high level ambients need not to be high level. Consider the process $P = h[\langle M \rangle Q \mid \ell[(x : W)^{\uparrow} P]]$; independently of M 's level, the flow of information is hidden within the high ambient h , thus, thanks to well typedness condition (see typing rules for `out`), it is not observable.

Subtyping and Subsumption

The relation of subtyping coincides with the one defined for the system of Section 2.4. The rule of subsumption requires the target type to be well-formed to enable type promotion.

$\frac{E \in \{\text{shh}, E'\}}{[E, F] \leq [E', F']}$	$\frac{\Gamma \vdash_{(\sigma, \rho)} P : [E, F] \quad [E, F] \leq [E', F'] \quad \Gamma \vdash_{\sigma} [E', F']}{\Gamma \vdash_{(\sigma, \rho)} P : [E', F']}$
---	--

Expressions

As suggested by the last example of § 4.2.1, a capability `out` n should be considered unsafe if n is high-level. On the other hand, an `in` capability may safely be exercised by any ambient (a low-level ambient ℓ entering a high-level ambient h may create a flow of information, but only if ℓ were allowed to eventually exit h).

$$\begin{array}{c}
\text{(PROJECTION)} \quad \frac{\Gamma, a : W, \Gamma' \vdash \diamond}{\Gamma, a : W, \Gamma' \vdash a : W} \quad \text{(PATH)} \quad \frac{\Gamma \vdash M_1 : \sigma_1 \text{Cap}[E] \quad \Gamma \vdash M_2 : \sigma_2 \text{Cap}[E] \quad (\text{Cap} \in \{\text{sCap}, \text{uCap}\})}{\Gamma \vdash M_1.M_2 : \sigma_1 \text{Cap}[E] \cdot \sigma_2 \text{Cap}[E]} \\
\\
\text{(SAFE-OUT)} \quad \frac{\Gamma \vdash M : \sigma \text{Amb}[E, F] \quad F' \leq F \quad \sigma \notin \mathbf{H}}{\Gamma \vdash \text{out } M : \sigma \text{sCap}[F']} \quad \text{(UNSAFE-OUT)} \quad \frac{\Gamma \vdash M : \sigma \text{Amb}[E, F] \quad F' \leq F \quad \sigma \in \mathbf{H}}{\Gamma \vdash \text{out } M : \sigma \text{uCap}[F']} \\
\\
\text{(IN)} \quad \frac{\Gamma \vdash M : \sigma \text{Amb}[F, E] \quad F' \leq F}{\Gamma \vdash \text{in } M : \sigma \text{sCap}[F']}
\end{array}$$

Processes

For the rules that follow, we define $\text{Safe}(\sigma, \rho, \delta) \triangleq (\sigma \in \mathbf{H}) \vee (\rho \preceq \delta)$: intuitively, a process P is safe either (i) if it is contained within an high level ambient, or (ii) if the clearances of the ‘actions’ performed by P do not decrease as P progresses.

$$\begin{array}{c}
\text{(SAFE-PREFIX)} \quad \frac{\Gamma \vdash M : \delta \text{sCap}[F] \quad \Gamma \vdash_{(\sigma, \rho)} P : [E, F] \quad \text{Safe}(\sigma, \rho, \delta)}{\Gamma \vdash_{(\sigma, \rho)} M.P : [E, F]} \\
\\
\text{(UNSAFE-PREFIX)} \quad \frac{\Gamma \vdash M : \delta \text{uCap}[F] \quad \Gamma \vdash_{(\sigma, \rho)} P : [E, F] \quad (\sigma \in \mathbf{H})}{\Gamma \vdash_{(\sigma, \rho)} M.P : [E, F]}
\end{array}$$

Safe prefixes are lower-bounded by ρ , following the previous intuition. As an example, the process $(x)^h \text{out } \ell$ is well-typed only at level $\sigma \in \mathbf{H}$, as it represents a low-level action that depends from (as it follows) a high level one. Instead, unsafe prefixes may only be

exercised within high-level ambients: this prevents low-level ambients from escaping from high-level contexts. Notice that mobility does not affect the lower bound ρ : this is safe and leaves a certain freedom to move (e.g. the path in $h.in\ l$ can be executed also at level $\sigma \in L$).

The following four rules are standard, and should be self-explanatory.

$$\begin{array}{c}
\text{(PAR)} \\
\frac{\Gamma \vdash_{(\sigma, \rho)} P : [E, F] \quad \Gamma \vdash_{(\sigma, \rho)} Q : [E, F]}{\Gamma \vdash_{(\sigma, \rho)} P \mid Q : [E, F]}
\end{array}
\qquad
\begin{array}{c}
\text{(DEAD)} \\
\frac{\Gamma \vdash_{\sigma} [E, F]}{\Gamma \vdash_{(\sigma, \rho)} \mathbf{0} : [E, F]}
\end{array}$$

$$\begin{array}{c}
\text{(NEW)} \\
\frac{\Gamma, n : \tau \mathbf{Amb}[G, H] \vdash_{(\sigma, \rho)} P : [E, F]}{\Gamma \vdash_{(\sigma, \rho)} (\nu n : \tau \mathbf{Amb}[G, H])P : [E, F]}
\end{array}
\qquad
\begin{array}{c}
\text{(REPL)} \\
\frac{\Gamma \vdash_{(\sigma, \rho)} P : [E, F]}{\Gamma \vdash_{(\sigma, \rho)} !P : [E, F]}
\end{array}$$

The (AMB) rule implements the idea that an ambient is viewed as an “action”. That is why the rule needs the hypothesis $\mathbf{Safe}(\sigma, \rho, \delta)$ as in rule (SAFE-PREFIX). Furthermore, the process enclosed in M is typed at level δ (the clearance of M) and with ρ initially set to the bottom security level.

$$\begin{array}{c}
\text{(AMB)} \\
\frac{\Gamma \vdash M : \delta \mathbf{Amb}[E, F] \quad \Gamma \vdash_{(\delta, \perp)} P : [E, F] \quad \Gamma \vdash_{\sigma} [F, G] \quad \mathbf{Safe}(\sigma, \rho, \delta)}{\Gamma \vdash_{(\sigma, \rho)} M[P] : [F, G]}
\end{array}$$

We finally come to the rules for communication, which test the predicate \mathbf{Safe} in ways similar to the rules for prefixes. In addition, exchanging a value affects the lower bound ρ in the typing of the continuation process P . Thus, when typed at level $\sigma \in L$, a process may safely communicate with a high level subambient, provided that all the subsequent actions are high-level. Thus, for instance, the processes $\ell[(x)^h\langle x \rangle^{h_1}]$, $\ell[(x:\mathbf{H})^h\langle x \rangle^{\uparrow}]$ and $\ell[\langle \ell_1 \rangle \langle M \rangle^h]$ are well typed, while $\ell[\langle M \rangle^h\langle \ell_1 \rangle]$ is not.

$$\begin{array}{c}
\text{(INPUT)} \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{(\sigma, \Lambda(W_1 \times \dots \times W_k))} P : [W_1 \times \dots \times W_k, E] \quad \mathbf{Safe}(\sigma, \rho, \Lambda(W_1 \times \dots \times W_k))}{\Gamma \vdash_{(\sigma, \rho)} (\tilde{x} : \tilde{W})P : [W_1 \times \dots \times W_k, E]}
\end{array}$$

$$\begin{array}{c}
\text{(OUTPUT)} \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_{(\sigma, \Lambda(W_1 \times \dots \times W_k))} P : [W_1 \times \dots \times W_k, E] \quad \mathbf{Safe}(\sigma, \rho, \Lambda(W_1 \times \dots \times W_k))}{\Gamma \vdash_{(\sigma, \rho)} \langle M_1, \dots, M_k \rangle P : [W_1 \times \dots \times W_k, E]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } M) \\
\frac{\Gamma \vdash M : \delta \text{Amb}[W_1 \times \cdots \times W_k, G] \quad \Gamma, \tilde{x} : \tilde{W} \vdash_{(\sigma, \delta)} P : [E, F] \quad \text{Safe}(\sigma, \rho, \delta)}{\Gamma \vdash_{(\sigma, \rho)} (\tilde{x} : \tilde{W})^M P : [E, F]} \\
\\
\text{(OUTPUT } N) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash N : \delta \text{Amb}[W_1 \times \cdots \times W_k, G] \quad \Gamma \vdash M_i : W_i \quad \Gamma \vdash_{(\sigma, \delta)} P : [E, F] \quad \text{Safe}(\sigma, \rho, \delta)}{\Gamma \vdash_{(\sigma, \rho)} \langle M_1, \dots, M_k \rangle^N P : [E, F]} \\
\\
\text{(INPUT } \uparrow) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash_{(\sigma, \Lambda(W_1 \times \cdots \times W_k))} P : [E, W_1 \times \cdots \times W_k] \quad \text{Safe}(\sigma, \rho, \Lambda(W_1 \times \cdots \times W_k))}{\Gamma \vdash_{(\sigma, \rho)} (\tilde{x} : \tilde{W})^\uparrow P : [E, W_1 \times \cdots \times W_k]} \\
\\
\text{(OUTPUT } \uparrow) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash_{(\sigma, \Lambda(W_1 \times \cdots \times W_k))} P : [E, W_1 \times \cdots \times W_k] \quad \text{Safe}(\sigma, \rho, \Lambda(W_1 \times \cdots \times W_k))}{\Gamma \vdash_{(\sigma, \rho)} \langle M_1, \dots, M_k \rangle^\uparrow P : [E, W_1 \times \cdots \times W_k]}
\end{array}$$

As usual, the correctness of the type system is guaranteed by the subject reduction property, whose proof follows the standard technique.

Lemma 4.2.2.

1. Strengthening

Assume $\Gamma \vdash_{(\sigma, \rho)} P : [E, F]$. Then $\Gamma \vdash_{(\sigma, \rho')} P : [E, F]$ for every $\rho' \preceq \rho$.

2. Subject Congruence

(a) If $\Gamma \vdash_{(\sigma, \rho)} P : T$ and $P \equiv Q$ then $\Gamma \vdash_{(\sigma, \rho)} Q : T$.

(b) If $\Gamma \vdash_{(\sigma, \rho)} P : T$ and $Q \equiv P$ then $\Gamma \vdash_{(\sigma, \rho)} Q : T$.

3. Substitution

(a) Assume $\Gamma, x : W, \Gamma' \vdash M : W'$. For any N , if $\Gamma, \Gamma' \vdash N : W$, then $\Gamma, \Gamma' \vdash M\{x := N\} : W'$.

(b) Assume $\Gamma, x : W, \Gamma' \vdash_{(\sigma, \rho)} P : T$. For any N , if $\Gamma, \Gamma' \vdash N : W$, then $\Gamma, \Gamma' \vdash_{(\sigma, \rho)} P\{x := N\} : T$.

Proof. Standard □

Proposition 4.2.3 (Subject Reduction).

If $\Gamma \vdash_{(\sigma, \rho)} P : [E, F]$ and $P \rightarrow Q$ then $\Gamma \vdash_{(\sigma, \rho)} Q : [E, F]$. \square

Proof. By induction of the derivation $P \rightarrow Q$. In the base case, we distinguish the possible cases of top-level reduction.

CASE ENTER. The redex is of the form $a[\text{in } b.P \mid Q] \mid b[R]$, and the judgment in the hypothesis must have been derived by (PAR) followed by (SUB) from $\Gamma \vdash_{(\sigma, \rho)} a[\text{in } b.P \mid Q] : [E_a, F]$ and $\Gamma \vdash_{(\sigma, \rho)} b[R] : [E_b, F]$, with $E_i \leq E$, $i \in \{a, b\}$. From the latter judgment, we know that $\Gamma \vdash_{(\delta_b, \perp)} R : [G, E_b]$, from which it follows that $\Gamma \vdash_{\delta_b} [G, E_b]$, with $\Gamma(b) = \delta_b \text{Amb}[G, E_b]$ and $\text{Safe}(\sigma, \rho, \delta_b)$. From the former judgment, we know that $\Gamma \vdash_{(\delta_a, \perp)} \text{in } b.P \mid Q : [I, E_a]$ and $\Gamma \vdash_{\delta_a} [I, E_a]$ with $\Gamma(a) = \delta_a \text{Amb}[I, E_a]$ and $\text{Safe}(\sigma, \rho, \delta_a)$.

From $\Gamma \vdash_{(\delta_a, \perp)} \text{in } b.P \mid Q : [I, E_a]$, an inspection of the rule (SAFE PREFIX) shows that $\Gamma \vdash_{(\delta_a, \perp)} P \mid Q : [I, E_a]$ is derivable for $E_a \leq G$. Since $\perp \preceq \delta_a$, one has $\text{Safe}(\delta_b, \perp, \delta_a)$, and we have just observed that $\Gamma \vdash_{\delta_b} [G, E_b]$ is derivable. Thus, by (AMB) and Subsumption one has $\Gamma \vdash_{(\delta_b, \perp)} a[P \mid Q] : [G, E_b]$. From the last judgment, and from $\Gamma \vdash_{(\delta_b, \perp)} R : [G, E_b]$, by (PAR) $\Gamma \vdash_{(\delta_b, \perp)} a[P \mid Q] \mid R : [G, E_b]$, then $\Gamma \vdash_{(\sigma, \rho)} b[a[P \mid Q] \mid R] : [E_b, F]$ and we conclude by subsumption.

CASE EXIT. The redex is of the form $a[b[\text{out } a.P \mid Q] \mid R]$, and the judgment in the hypothesis must have been derived by (PAR) and (AMB) from $\Gamma \vdash_{(\delta_a, \perp)} b[\text{out } a.P \mid Q] : [G, E_a]$ and $\Gamma \vdash_{(\delta_a, \perp)} R : [G, E_a]$, with $E_a \leq E$ and $\Gamma(a) = \delta_a \text{Amb}[G, E_a]$ such that $\text{Safe}(\sigma, \rho, \delta_a)$. A routine analysis shows that $\Gamma \vdash_{(\sigma, \rho)} a[R] : [E, F]$ is derivable. On the other hand, $\Gamma \vdash_{(\delta_a, \perp)} b[\text{out } a.P \mid Q] : [G, E_a]$ must have been derived from $\Gamma \vdash_{(\delta_b, \perp)} \text{out } a.P \mid Q : [E_b, F_b]$, with $E_b \leq G$ and $\Gamma(b) = \delta_b \text{Amb}[E_b, F_b]$ such that $\text{Safe}(\sigma, \rho, \delta_b)$. We distinguish the two possible subcases:

- $\Gamma \vdash \text{out } a : \delta_a \text{scap}[F_b]$. Then $\delta_a \notin \mathbf{H}$, and $\Gamma \vdash_{(\delta_b, \perp)} \text{out } a.P : [E_b, F_b]$ must have been derived by (SAFE PREFIX) with $F_b \leq E_a$. An inspection of the typing rules shows that $\Gamma \vdash_{(\delta_b, \perp)} P \mid Q : [E_b, F_b]$ also is derivable. Now, from $\delta_a \notin \mathbf{H}$ and $\text{Safe}(\sigma, \rho, \delta_a)$ we know that $\sigma \in \mathbf{H}$ or $\rho \in \mathbf{L}$. In both cases $\text{Safe}(\sigma, \rho, \delta_b)$, and hence, together with $F_b \leq E_a$ and $E_a \leq E$, $\Gamma \vdash_{(\sigma, \rho)} b[P \mid Q] : [E, F]$. The claim follows now by (PAR) from the last judgment and from $\Gamma \vdash_{(\sigma, \rho)} a[R] : [E, F]$.

- $\Gamma \vdash \text{out } a : \delta_a \text{ucap}[F_b]$. Then $\delta_a \in \mathbf{H}$, and $\Gamma \vdash_{(\delta_b, \perp)} \text{out } a.P \mid Q : [E_b, F_b]$ must have been derived by (UNSAFE-PREFIX) with $F_b \leq E_a$. Now the side-condition to that rule requires that $\delta_b \in \mathbf{H}$, and the previous analysis shows that $\Gamma \vdash_{(\sigma, \rho)} b[P \mid Q] : [E, F]$, as $\delta_b \in \mathbf{H}$ implies $\text{Safe}(\sigma, \rho, \delta_b)$ and $F_b \leq E$. The claim follows then by (PAR).

CASES (LOCAL), (INPUT n), (INPUT \uparrow), (OUTPUT n) AND (OUTPUT \uparrow). We give the proof for the case (LOCAL) as representative. In this case the redex is $(\tilde{x}:\tilde{W})P \mid \langle \tilde{M} \rangle Q$. The judgment in the hypothesis is $\Gamma \vdash_{(\sigma, \rho)} (\tilde{x}:\tilde{W})P \mid \langle \tilde{M} \rangle P : [W_1 \times \dots \times W_k, E]$ for some exchange E , derived from $\Gamma, \tilde{x}:\tilde{W} \vdash_{(\sigma, \Lambda(W_1 \times \dots \times W_k))} P : [W_1 \times \dots \times W_k, E]$, and from $\Gamma \vdash M_i : W_i$ for $i = 1, \dots, k$ and $\Gamma \vdash_{(\sigma, \Lambda(W_1 \times \dots \times W_k))} Q : [W_1 \times \dots \times W_k, E]$ with $\text{Safe}(\sigma, \rho, \Lambda(W_1 \times \dots \times W_k))$. By Lemma 4.2.2(1), we know that $\Gamma, \tilde{x}:\tilde{W} \vdash_{(\sigma, \rho)} P : [W_1 \times \dots \times W_k, E]$, and $\Gamma \vdash_{(\sigma, \rho)} Q : [W_1 \times \dots \times W_k, E]$. By Lemma 4.2.2(3), $\Gamma \vdash_{(\sigma, \rho)} P\{x_i := M_i\} : [W_1 \times \dots \times W_k, E]$, and the proof follows by (PAR).

For the inductive steps, the proof follows by a case analysis of the possible contextual reductions.

(CASE PAR) The reduction is $P \mid Q \rightarrow P' \mid Q$ derived from $P \rightarrow P'$. The judgment in the hypothesis, $\Gamma \vdash_{(\sigma, \rho)} P \mid Q : [E, F]$ must depend on two judgments $\Gamma \vdash_{(\sigma, \rho)} P : T_1$ and $\Gamma \vdash_{(\sigma, \rho)} Q : T_2$ for suitable T_1, T_2 . By the induction hypothesis, $\Gamma \vdash_{(\sigma, \rho)} P' : T_1$ and $\Gamma \vdash_{(\sigma, \rho)} P \mid Q : [E, F]$ derives by subsumption and (PAR).

(CASE STRUCT) follows by the induction hypothesis and Lemma 4.2.2(2), while (CASE NEW) follows directly by the induction hypothesis.

(CASE AMB.) The reduction is $a[P] \rightarrow a[Q]$, derived from $P \rightarrow Q$, and the judgment $\Gamma \vdash_{(\sigma, \rho)} a[P] : [E, F]$, in the hypothesis, must depend on $\Gamma \vdash_{(\delta_a, \perp)} P : T$ for a suitable T , with $\delta_a = \Lambda(\Gamma(a))$. By the inductive hypothesis, one has $\Gamma \vdash_{(\delta_a, \perp)} Q : T$. From this $\Gamma \vdash_{(\sigma, \rho)} a[Q] : [E, F]$ derives from $\Gamma \vdash_{(\delta_a, \perp)} Q : T$ by the same steps that derived $\Gamma \vdash_{(\sigma, \rho)} a[P] : [E, F]$ from $\Gamma \vdash_{(\delta_a, \perp)} P : T$.

□

4.2.3 Absence of Explicit Flows

In this subsection we define explicit information flows using a strong form of observation predicate; we then prove that the type system detects, and prevents, all unsafe forms of explicit flow, in the sense of property (\star) in the Introduction (cf. page 98).

The following definition presupposes the existence of a function γ from names and capabilities to their associated security clearance.

Definition 4.2.4. Let P be a process, we write $\gamma \triangleright P \downarrow_\rho^\sigma$ if either

- (i) $P \equiv (\nu \tilde{m})(n[\langle M \rangle Q \mid Q'] \mid Q'')$ with $n \notin \{\tilde{m}\}$, $\gamma(n) = \rho$ and $\gamma(M) = \sigma$, or
- (ii) $P \equiv (\nu \tilde{m})(n[(x : W)Q \mid Q'] \mid Q'')$ with $n \notin \{\tilde{m}\}$, $\gamma(n) = \rho$ and $\Lambda(W) = \sigma$. \square

Note that this definition of observation is entirely functional to the definition of explicit information flows. In the next section we will define a more general notion of observation (see Definition 4.3.3) over which we build the (typed) observational equivalence used to analyze the process behavior. Let \Longrightarrow be the reflexive and transitive closure of \rightarrow .

Definition 4.2.5 (Explicit Information Flow). Let γ be a function from values to security clearances. A process P contains an explicit information flow if there exists a process Q and a context $\mathbf{C}()$ such that $P \Longrightarrow \mathbf{C}(Q)$, where $\gamma \triangleright Q \downarrow_\rho^\sigma$ and $\rho \prec \sigma$. \square

Let Γ be a typing environment. We say that the function γ is Γ -compatible if for all $M \in \text{Dom}(\Gamma) \cap \text{Dom}(\gamma)$ we have $\gamma(M) = \Lambda(\Gamma(M))$. In the following we write $\Gamma \vdash P : T$ instead of $\Gamma \vdash_{(\sigma, \rho)} P : T$ when σ and ρ are not relevant.

Lemma 4.2.6. Let $\Gamma \vdash P : T$ and $\gamma \triangleright P \downarrow_\rho^\sigma$ where γ is Γ -compatible. Then $\sigma \preceq \rho$.

Proof. By induction on the structure of P . There are two basic cases:

- (i) $P \equiv n[\langle M \rangle Q_1 \mid Q_2]$ with $\gamma(n) = \rho$ and $\gamma(M) = \sigma$. From the hypothesis $\Gamma \vdash P : T$, the fact that γ is Γ -compatible, we have that $\Gamma \vdash n : \rho \text{Amb}[E, F]$, thus $\Gamma \vdash \rho \text{Amb}[E, F]$. By type formation rules, from the last judgment we have that $\Lambda(E) \preceq \rho$. Now, from $\Gamma \vdash P : T$ we also have that $\Gamma \vdash M : E$, then since $\gamma(M) = \sigma$ and γ is Γ -compatible, we have that $\Lambda(E) = \sigma$, thus $\sigma \preceq \rho$ as desired.
- (ii) $P \equiv n[(x : W)Q_1 \mid Q_2]$ with $\gamma(n) = \rho$ and $\Lambda(W) = \sigma$. From the hypothesis $\Gamma \vdash P : T$, the fact that γ is Γ -compatible, we have that $\Gamma \vdash n : \rho \text{Amb}[E, F]$, thus $\Gamma \vdash \rho \text{Amb}[E, F]$. By type formation rules, from the last judgment we have that $\Lambda(E) \preceq \rho$. Now, from $\Gamma \vdash P : T$ we also have that $E = W$, then since $\Lambda(W) = \sigma$, we have that $\sigma \preceq \rho$ as desired.

Then there are two inductive cases such that $\gamma \triangleright P \downarrow_\rho^\sigma$:

1. $P \equiv P_1 \mid P_2$. From $\gamma \triangleright P \downarrow_\rho^\sigma$, since P is a parallel composition, we may assume that $\gamma \triangleright P_1 \downarrow_\rho^\sigma$. From $\Gamma \vdash P : T$ we have $\Gamma \vdash P_1 : T$, then by induction we have $\sigma \preceq \rho$.
2. $P \equiv (\nu m : W)P_1$. Since m is bound, we may assume $m \notin \text{Dom}(\gamma)$. From $\gamma \triangleright P \downarrow_\rho^\sigma$, we then have that $(\gamma, m : \Lambda(W)) \triangleright P_1 \downarrow_\rho^\sigma$. From $\Gamma \vdash P : T$ we have that $\Gamma, m : W \vdash P_1 : T$. Then, since $(\gamma, m : \Lambda(W))$ is $(\Gamma, m : W)$ -compatible, by induction we have $\sigma \preceq \rho$. \square

Theorem 4.2.7 (Absence of Explicit Flows). *If $\Gamma \vdash P : T$ then P is free from explicit information flows. That is, for every process Q and every context $\mathbf{C}()$ such that $P \implies \mathbf{C}(Q)$, and $\gamma \triangleright Q \downarrow_\rho^\sigma$, we have $\sigma \preceq \rho$, with γ Δ -compatible function, where Δ is such that $\Delta \vdash Q : T'$ for some process type T' .*

Proof. By contradiction. Let Q and $\mathbf{C}()$ be such that $P \implies \mathbf{C}(Q)$ with $\gamma \triangleright Q \downarrow_\rho^\sigma$ and $\rho \prec \sigma$. Then from the hypothesis $\Gamma \vdash P : T$, by subject reduction we have that $\Gamma \vdash \mathbf{C}(Q) : T$, then there exists a process type T' and a typing environment Γ' such that $\Gamma, \Gamma' \vdash Q : T'$. Let γ be a (Γ, Γ') -compatible function, then by Lemma 4.2.6 we have $\sigma \preceq \rho$, hence we have the desired contradiction. \square

4.3 Non-interference

We start introducing the notion of ‘high-level sources’, in terms of which we then state our NDC-based definition of non-interference.

Definition 4.3.1 (High-level Sources). A process H is a *high-level source* if and only if $\Gamma \vdash_{(\sigma, \rho)} H : T$, for some security levels σ and ρ , and H is built according to the following grammar:

$$\begin{aligned}
 H ::= & (x : W_H)^\eta H \mid \langle M_H \rangle^\eta H \mid (x : W)^{M_H} H \mid \langle M \rangle^{N_H} H & \eta \in \{\star, \uparrow\} \\
 & \text{in } M_H.H \mid \text{out } M_H.H \mid M_H[P] \mid H \mid H \mid (\nu n : W)H \mid \mathbf{0} \mid !H
 \end{aligned}$$

where M_H are expressions such that $\Lambda(\Gamma(M_H)) \in H$, and W_H are types such that $\Lambda(W_H) \in H$. \square

Accordingly, high-level sources are well-typed processes that may only engage ‘high’ top-level interactions with any context in which they are inserted. This is true of all the top-level ambient occurrences and of processes in prefixed form in H by virtue of the grammar above. Furthermore, Definition 4.3.1 guarantees that if a process interacts (at

any nesting level) with a high level source, then either that interaction is safe, or it is a non-observable flow of information. In particular, this comes from the fact that well-typedness condition ensures that no low-level ambient may escape its enclosing high-level contexts.

Notation: We henceforth write $\Gamma \Vdash P : T$ to indicate that P is a high-level source in Γ . Also, we write $\Gamma \vdash P : [E, F]$ and $\Gamma \vdash P : ok$ when σ, ρ and/or $[E, F]$ are not relevant.

4.3.1 Typed Equivalence

Next, we introduce a *typed* notion of process equivalence. The equivalence is typed as we compare only processes with the same types, and inserted in contexts that respect their typing. We formalize these notions below following [SW01].

A *context* $\mathbf{C}()$ is a process term with just one hole $()$. We denote with $\mathbf{C}(P)$ the process resulting from replacing the hole with P in $\mathbf{C}()$. Note that variables and names that are free in P may become bound in $\mathbf{C}(P)$. Thus we do not identify contexts up to renaming of bound variables and names.

Definition 4.3.2 (($\Gamma/\Delta, T$) Context). Let Γ and Δ be type environments and T a process type. $\mathbf{C}()$ is a $(\Gamma/\Delta, T)$ -context if $\Gamma \vdash_{(\sigma, \rho)} \mathbf{C}() : ok$ with $\sigma \in L$, is derivable in the type system of Section 4.2 enriched with the a rule that derives $\Theta \vdash () : T$ for all Θ extending Δ . \square

Intuitively, a $(\Gamma/\Delta, T)$ -context is a context whose hole, of type T , is in the scope of the binders recorded in Δ , and whose free names and variables are contained in Γ . Furthermore, the context $\mathbf{C}()$ must be typed at low level, that is the clearance of external observers.

Definition 4.3.3 (Barbs). Define $P \downarrow_n \triangleq P \equiv (\nu \vec{m})(n[\langle M \rangle P' \mid Q'] \mid Q'') \ n \notin \{\vec{m}\}$. A process P exhibits the name n , written $P \Downarrow_n$ iff there exists Q such that $P \Longrightarrow Q$ and $Q \downarrow_n$, where \Longrightarrow is the reflexive and transitive closure of \rightarrow . \square

Now we can define our notion of ‘low’ typed equivalence, relative to an underlying security classification into ‘low’ and ‘high’ levels. Based on that we then have our definition of the non-interference.

Definition 4.3.4 (Typed observational equivalence and Non-interference). Assume $\Delta \vdash P : T$ and $\Delta \vdash Q : T$. The two processes are equivalent in Δ , written $\Delta \triangleright P \cong_L Q$ if and only if for all $(\Gamma/\Delta, T)$ -context $\mathbf{C}()$ with $\mathbf{C}(P)$ and $\mathbf{C}(Q)$ closed, for all n with $\Lambda(\Gamma(n)) \in L$, $\mathbf{C}(P) \Downarrow_n \Leftrightarrow \mathbf{C}(Q) \Downarrow_n$ \square

As we said in Section 4.1, our notion of observational equivalence is based on an observability predicate that refines the one introduced in [CG99a] for Mobile Ambients. Our definition of barbs is specific to BA, in that it reflects the core form of interaction between boxed ambients. On the other hand the following proposition shows that the contextual equivalence induced by our barbs is equivalent to that induced by barbs used in [CG99a]. We start with a couple of lemmas.

Lemma 4.3.5.

Let $\mathbf{C}_1() = \ell[\text{in } n.\text{out } n.\langle M \rangle] \mid ()$. Then $P \Downarrow_n^{MA}$ if and only if $\mathbf{C}_1(P) \Downarrow_\ell$ for $\ell \notin \text{fn}(P)$.

Proof. Left-to-right direction is immediate. For right-to-left assume $\mathbf{C}_1(P) \Downarrow_\ell$ with $\ell \notin \text{fn}(P)$. Then it must be $P \Longrightarrow (\nu \vec{m})(n[P'] \mid P'')$ with $n \notin \{\vec{m}\}$, that is $P \Downarrow_n^{MA}$. \square

Lemma 4.3.6.

Let $\mathbf{C}_2() = (x)^n \ell[0] \mid ()$. Then $P \Downarrow_n$ if and only if $\mathbf{C}_2(P) \Downarrow_\ell^{MA}$ for $\ell \notin \text{fn}(P)$.

Proof. Left-to-right direction is immediate. For right-to-left assume $\mathbf{C}_2(P) \Downarrow_\ell^{MA}$ with $\ell \notin \text{fn}(P)$. Then it must be $P \Longrightarrow (\nu \vec{m})(n[\langle M \rangle P'] \mid P'')$ with $n \notin \{\vec{m}\}$, that is $P \Downarrow_n$. \square

Proposition 4.3.7. Let $P \Downarrow_n^{MA} \triangleq P \equiv (\nu \vec{m})(n[P'] \mid P'')$ $n \notin \{\vec{m}\}$ and $P \Downarrow_n^{MA} \triangleq P \Longrightarrow Q \Downarrow_n^{MA}$. Let \cong_L^{MA} be the typed observational equivalence induced by \Downarrow_n^{MA} . Then $\Delta \triangleright P \cong_L Q$ if and only if $\Delta \triangleright P \cong_L^{MA} Q$.

Proof. We start proving $\cong_L \subseteq \cong_L^{MA}$. Assume $\Delta \triangleright P \cong_L Q$, and let $\mathbf{C}()$ be a $(\Gamma/\Delta, T)$ context, $\mathbf{C}(P), \mathbf{C}(Q)$ closed, and let n be a name such that $\Lambda(\Gamma(n)) \in L$; let be $\mathbf{C}(P) \Downarrow_n^{MA}$, we have to prove $\mathbf{C}(Q) \Downarrow_n^{MA}$. From $\mathbf{C}(P) \Downarrow_n^{MA}$, by Lemma 4.3.5, we have $\mathbf{C}_1(\mathbf{C}(P)) \Downarrow_\ell$, where we can choose ℓ such that $\Lambda(\Gamma(\ell)) \in L$. Now, since n and ℓ are low level names, then we have that $\mathbf{C}_1(\mathbf{C}())$ is a $(\Gamma/\Delta, T)$ context, $\mathbf{C}_1(\mathbf{C}(P)), \mathbf{C}_1(\mathbf{C}(Q))$ closed. Then from $\Delta \triangleright P \cong_L Q$ and $\mathbf{C}_1(\mathbf{C}(P)) \Downarrow_\ell$, we have $\mathbf{C}_1(\mathbf{C}(Q)) \Downarrow_\ell$, and again by Lemma 4.3.5, $\mathbf{C}(Q) \Downarrow_n^{MA}$. With a similar argument we have $\mathbf{C}(Q) \Downarrow_n^{MA} \Rightarrow \mathbf{C}(P) \Downarrow_n^{MA}$, thus $\Delta \triangleright P \cong_L^{MA} Q$. The proof of $\cong_L^{MA} \subseteq \cong_L$ follows similarly, applying Lemma 4.3.6.

□

Definition 4.3.8 (Non-interference). Let Σ be a security lattice and P a process. Given a security classification of Σ such that $\Delta \vdash P : T$, P is *secure* for that classification iff $\Delta \triangleright P \cong_L P \mid H$ for all H such that $\Delta \Vdash H : T$. P is *interference-free* if it is secure for all security classifications of Σ . □

We conclude with the main result, a theorem that states that the type system guarantees non-interference for well-typed processes.

Theorem 4.3.9 (Non-interference).

Given any security classification, if $\Delta \vdash P : T$ and $\Delta \Vdash H : T$, then $\Delta \triangleright P \cong_L P \mid H$.

Proof. In Section 4.4 □

Notice that the theorem is stated, and proved, only in reference to well-typed contexts. Accordingly, the non-interference analysis it addresses corresponds to verifying the ‘internal’ security of a system rather than its security with respect to external attackers. A stronger security property is studied in Chapter 6, where we rely on a weak form of cryptography and partial typing to protect trusted systems from untrusted, external, processes.

The non-interference proof draws on the technical tools developed by Cardelli and Gordon for Mobile Ambients [CG99a], adapting them to our Boxed Ambients. The non-interference result derives from a lemma that shows that high-level sources are low-level equivalent to the inactive process. More precisely, we show that for every context $\mathbf{C}()$ and high-level source H if $\mathbf{C}(H)$ is well-typed then it is indistinguishable at low level from $\mathbf{C}(\mathbf{0})$. Proving this result requires a characterization of all the possible interactions between a process and the surrounding context. We start in § 4.4.1, by introducing a hardening relation, that is functional to § 4.4.2, where we introduce an alternative (but equivalent) semantics for the calculus based on a labeled transition system. As in [CG99a], the labeled transition system allows an analysis of the reductions of a process based on its syntactic structure. Relying upon the labeled transition system, in § 4.4.3 we derive an “activity” lemma that describes how a process may evolve when enclosed within an evaluation context, and a “context” lemma that shows that process equivalence may be tested in evaluation contexts rather than in generic contexts. Relying on this characterizations of reduction and behavior, the proof of our main lemma derives from

the invariants satisfied by well-typed processes. Finally, in § 4.4.4, we prove the non-interference theorem.

4.3.2 Discussion

The type system we have defined to derive the non-interference proof is admittedly somewhat restrictive. While this is unfortunate, the discipline we impose on interactions between high and low-level processes has effects comparable to those found in existing type systems for secure information-flow in simpler process calculi [HR00], and multi-threaded languages [VSI96, VS97, BC01a] (cf. Section 4.5 for a detailed comparison).

Also, even though well-typed processes are constrained in the actions they may perform, the type system still allows non-trivial forms of interaction between high and low levels, both in terms of mobility, and of value exchanges.

The following table contains some example of safe interactions between high and low ambients that are allowed by the type system (if also Q_i are safe processes).

Communications	$\left[\begin{array}{l} \ell[(x:W)^h \langle x \rangle^{h_1} Q_1 \mid h[Q_2] \mid h_1[Q_3]] \\ \text{and similar down exchanges in } \ell \\ h[(x:W_L)^\ell Q_3 \mid \ell[\langle M \rangle Q_1 \mid Q_2]] \\ \text{and similar local exchanges in } \ell \\ h[\langle M \rangle P \mid \ell[(x:W)^\uparrow Q_1 \mid Q_2]] \\ \text{and similar up exchanges in } \ell \end{array} \right]$
Mobility	$\left[\begin{array}{l} \text{high ambients freely move} \\ \text{low ambients do not exit high ambients} \end{array} \right]$

As an example, consider an ambient ℓ that wants to communicate a value M to a high ambient h . ℓ cannot enter h without being trapped, but a different possibility is

$$\ell[\ell_1[\text{out } \ell.\text{in } h.\langle M \rangle^\uparrow] \mid P] \mid h[(x)Q]$$

where a taxi ambient ℓ_1 exits ℓ and carries M to h .

Figure 4.1 shows the legal flow of information for a well-typed composition of the two processes P and H , when H is a high-level source.

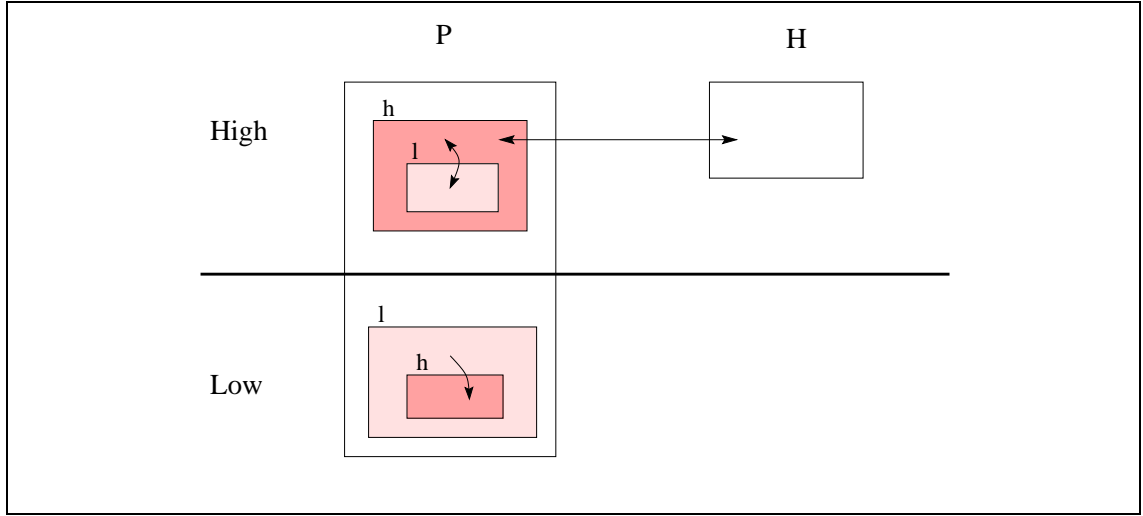


Figure 4.1: Flows of information of $P \mid H$

In particular, the flows enabled by the type system are (i) those from H to the high sub-processes of P (and vice versa), (ii) those from low to high subprocesses in P , and (iii) those from the high-level components of P to those low components of P that are not observable since they are shielded by high-level ambients. As a result, a low-level observer may thus observe only flows of information between low-level components of P and low level components of the surrounding context.

Also note that high-level information can be freely exchanged between the high and low level processes of P , as long as the latter are nested within high-level ambients. This is because the type system ensures that these low sub-processes are confined within high-level ambients.

As a further remark, we note that the proof of non-interference would not go through in the presence of finer equivalence relations such as barbed congruence, bisimulation or must testing. To see the problem with barbed congruence, consider defining \approx_L as the barbed congruence relation induced by our observability predicate $P \downarrow_n$. Then, take the processes $P = \ell[\langle M \rangle \mid \text{in } h.0]$ and the high-level process $H = h[]$. Now take the context $\mathbf{C}() = ()$, and observe that $\mathbf{C}(P \mid H) \rightarrow R = h[\ell[\langle M \rangle]]$, while there exists no process R' such that $\mathbf{C}(P) \rightarrow R'$ and $R \approx_L R'$.

4.4 Proof of non-interference

The main result of this section is the proof of Non-interference theorem. In order to prove it we introduce a number of useful relations and theorems that help prove contextual equivalence. In particular, we introduce a labelled transition system built over a hardening relation, that allows an analysis of the reductions of a process based on its syntactic structure. We then derive an “activity lemma” that describes how a process interacts with the enclosing context, and a “context lemma” stating that in order to prove contextual equivalence we can consider only a limited set of contexts.

To ease the notation, throughout this section the type annotation on restricted names and input variables are omitted unless relevant to the context. Furthermore, proofs omitted from this section can be found at the end of the chapter, in Section 4.6.

4.4.1 A Hardening Relation

Following [CG99a], we define a hardening relation for Boxed Ambients, that is a relation that explicitly identifies the top-level sub-processes of a process that may be involved in a reduction. The hardening relation will be useful in defining an alternative operational semantics based on a labelled transition system.

The hardening relation takes the form $P \succ (\nu \tilde{p})\langle P' \rangle P''$ where $(\nu \tilde{p})\langle P' \rangle P''$ is called a *concretion*, P' is the *prime* of the concretion, and P'' is the *residue* of concretion. Both P' and P'' are in the scope of the restricted names \tilde{p} . Intuitively, a process P which may have many top-level subprocesses, may harden to a concretion that singles out a prime subprocess P' , leaving behind the residue P'' . Concretions are defined below, following [CG99a]:

Concretions

$$\begin{aligned}
 C, D &::= (\nu \tilde{p})\langle M.P \rangle Q && \text{action, } M \in \{\text{in } n, \text{out } n\} \\
 &(\nu \tilde{p})\langle n[P] \rangle Q && \text{ambient} \\
 &(\nu \tilde{p})\langle (x:W)^\eta P \rangle Q && \text{input } \eta \in \{\star, \uparrow, n\} \\
 &(\nu \tilde{p})\langle \langle M \rangle^\eta P \rangle Q && \text{output } \eta \in \{\star, \uparrow, n\}
 \end{aligned}$$

The order of bound names $(\nu \tilde{p})$ in a concretion $(\nu \tilde{p})\langle P' \rangle P''$ does not matter and restricted names can be renamed consistently. If $\{\tilde{p}\} = \emptyset$, we write the concretion as $(\nu)\langle P' \rangle P''$.

Before defining the hardening relation we introduce the notation $\overline{(\nu n)}C$ for restricting a concretion.

Definition 4.4.1 (Restricting a concretion).

Let C be $(\nu \tilde{p})\langle P_1 \rangle P_2$ and $n \notin \{\tilde{p}\}$, then:

1. If $n \in fn(P_1)$ then

- (a) If $P_1 = m[P'_1]$, $m \neq n$, $n \notin fn(P_2)$, then $\overline{(\nu n)}C \triangleq (\nu \tilde{p})\langle m[(\nu n)P'_1] \rangle P_2$
- (b) Otherwise, $\overline{(\nu n)}C \triangleq (\nu n, \tilde{p})\langle P_1 \rangle P_2$

2. If $n \notin fn(P_1)$, then $\overline{(\nu n)}C \triangleq (\nu \tilde{p})\langle P_1 \rangle (\nu n)P_2$

□

Hardening relation is then defined by the following.

$$\begin{array}{ccc}
 \text{(HARDEN ACTION)} & \text{(HARDEN .)} & \text{(HARDEN AMB)} \\
 \frac{M \in \{\text{in } n, \text{out } n\}}{M.P \succ (\nu)\langle M.P \rangle \mathbf{0}} & \frac{M.(N.P) \succ C}{(M.N).P \succ C} & \frac{}{n[P] \succ (\nu)\langle n[P] \rangle \mathbf{0}}
 \end{array}$$

$$\begin{array}{cc}
 \text{(HARDEN INPUT } \eta) & \text{(HARDEN OUTPUT } \eta) \\
 \frac{}{(x)^\eta P \succ (\nu)\langle (x)^\eta P \rangle \mathbf{0}} & \frac{}{\langle M \rangle^\eta P \succ (\nu)\langle \langle M \rangle^\eta P \rangle \mathbf{0}}
 \end{array}$$

$$\begin{array}{cc}
 \text{(HARDEN PAR 1)} & \text{(HARDEN PAR 2)} \\
 \frac{P \succ (\nu \tilde{p})\langle P' \rangle P'' \quad \tilde{p} \cap fn(Q) = \emptyset}{P \mid Q \succ (\nu \tilde{p})\langle P' \rangle (P'' \mid Q)} & \frac{Q \succ (\nu \tilde{q})\langle Q' \rangle Q'' \quad \tilde{q} \cap fn(P) = \emptyset}{P \mid Q \succ (\nu \tilde{q})\langle Q' \rangle (P \mid Q'')}
 \end{array}$$

$$\begin{array}{cc}
 \text{(HARDEN REPL)} & \text{(HARDEN RES)} \\
 \frac{P \succ (\nu \tilde{p})\langle P' \rangle P''}{!P \succ (\nu \tilde{p})\langle P' \rangle (P'' \mid !P)} & \frac{P \succ C}{(\nu n)P \succ \overline{(\nu n)}C}
 \end{array}$$

The following proposition relates hardening and structural congruence.

Proposition 4.4.2 (Properties of Hardening).

1. If $P \succ (\nu \tilde{p})\langle P' \rangle P''$, then $P \equiv (\nu \tilde{p})(P' \mid P'')$.
2. If $P \equiv Q$ and $Q \succ (\nu \tilde{r})\langle Q' \rangle Q''$, then $P \succ (\nu \tilde{r})\langle P' \rangle P''$ for P and P' such that $P' \equiv Q'$, $P'' \equiv Q''$.

□

4.4.2 A Labelled Transition System

In this subsection we present an alternative semantics for BA, based on a labelled transition system. The main advantage of the LTS is that it allows an analysis of the possible evolutions of a process in terms of its syntactic structure. The transition system we use is built upon the following set of labels:

$$\text{Labels} \quad \alpha ::= \tau \mid \text{in } n \mid \text{out } n \mid (M) \mid (M)^\dagger$$

The transitions $P \xrightarrow{\text{in } n} Q$ and $P \xrightarrow{\text{out } n} Q$ mean that the process P has a top level process that exercise the capability in n and out n . The transition $P \xrightarrow{(M)} Q$ (resp. $P \xrightarrow{(M)^\dagger} Q$) means that the process P has a top level process that reads locally (resp. upward) the value M . Finally, the transition $P \xrightarrow{\tau} Q$ means that the process P evolves in one step to Q . The labelled transitions are defined by the following rules.

$$\begin{array}{c}
\begin{array}{c}
(\text{TRANS INPUT UP}) \\
\frac{P \succ (\nu \tilde{p}) \langle (x)^\dagger P' \rangle P'' \quad fn(M) \cap \tilde{p} = \emptyset}{P \xrightarrow{(M)^\dagger} (\nu \tilde{p}) (P' \{x := M\} \mid P'')}
\end{array}
\qquad
\begin{array}{c}
(\text{TRANS INPUT LOCAL}) \\
\frac{P \succ (\nu \tilde{p}) \langle (x) P' \rangle P'' \quad fn(M) \cap \tilde{p} = \emptyset}{P \xrightarrow{(M)} (\nu \tilde{p}) (P' \{x := M\} \mid P'')}
\end{array}
\\[10pt]
\begin{array}{c}
(\text{TRANS CAP}) \\
\frac{P \succ (\nu \tilde{p}) \langle M.P' \rangle P'' \quad fn(M) \cap \tilde{p} = \emptyset}{P \xrightarrow{M} (\nu \tilde{p}) (P' \mid P'')}
\end{array}
\qquad
\begin{array}{c}
(\text{TRANS AMB}) \\
\frac{P \succ (\nu \tilde{p}) \langle n[Q] \rangle R \quad Q \xrightarrow{\tau} Q'}{P \xrightarrow{\tau} (\nu \tilde{p}) (n[Q'] \mid R)}
\end{array}
\\[10pt]
\begin{array}{c}
(\tau\text{-IN}) \text{ (where } fn(n[Q]) \cap \tilde{r} = \emptyset) \\
\frac{P \succ (\nu \tilde{p}) \langle n[Q] \rangle R \quad Q \xrightarrow{\text{in } m} Q' \quad R \succ (\nu \tilde{r}) \langle m[R'] \rangle R''}{P \xrightarrow{\tau} (\nu \tilde{p}, \tilde{r}) (m[n[Q']] \mid R' \mid R'')}
\end{array}
\\[10pt]
\begin{array}{c}
(\tau\text{-OUT}) \text{ (where } n \notin \{\tilde{q}\}) \\
\frac{P \succ (\nu \tilde{p}) \langle n[Q] \rangle P' \quad Q \succ (\nu \tilde{q}) \langle m[R] \rangle Q' \quad R \xrightarrow{\text{out } n} R'}{P \xrightarrow{\tau} (\nu \tilde{p}) ((\nu \tilde{q}) (m[R'] \mid n[Q']) \mid P')}
\end{array}
\\[10pt]
\begin{array}{c}
(\tau\text{-I/O}) \\
\frac{P \succ (\nu \tilde{p}) \langle \langle M \rangle P' \rangle P'' \quad P'' \xrightarrow{(M)} R}{P \xrightarrow{\tau} (\nu \tilde{p}) (P' \mid R)}
\end{array}
\end{array}$$

$$\begin{array}{c}
(\tau\text{-INPUT } n) \text{ (where } \{\tilde{q}\} \cap \text{fn}(P'') = \{\tilde{r}\} \cap \text{fn}(n[P']) = \emptyset) \\
P \succ (\nu \tilde{p}) \langle n[P'] \rangle P'' \quad P' \succ (\nu \tilde{q}) \langle \langle M \rangle Q' \rangle Q'' \quad P'' \succ (\nu \tilde{r}) \langle (x)^n R' \rangle R'' \\
\hline
P \xrightarrow{\tau} (\nu \tilde{p}, \tilde{q}) (n[Q' \mid Q''] \mid (\nu \tilde{r}) (R' \{x := M\} \mid R''))
\end{array}$$

$$\begin{array}{c}
(\tau\text{-INPUT } \uparrow) \text{ (where } \text{fn}(\langle M \rangle P') \cap \tilde{q} = \emptyset) \\
P \succ (\nu \tilde{p}) \langle \langle M \rangle P' \rangle P'' \quad P'' \succ (\nu \tilde{q}) \langle n[R] \rangle R' \quad R \xrightarrow{(M)^\uparrow} R'' \\
\hline
P \xrightarrow{\tau} (\nu \tilde{p}) (P' \mid (\nu \tilde{q}) (n[R''] \mid R'))
\end{array}$$

$$\begin{array}{c}
(\tau\text{-OUTPUT } n) \text{ (where } \text{fn}(\langle M \rangle^n P') \cap \tilde{q} = \emptyset) \\
P \succ (\nu \tilde{p}) \langle \langle M \rangle^n P' \rangle P'' \quad P'' \succ (\nu \tilde{q}) \langle n[R] \rangle R' \quad R \xrightarrow{(M)} R'' \\
\hline
P \xrightarrow{\tau} (\nu \tilde{p}) (P' \mid (\nu \tilde{q}) (n[R''] \mid R'))
\end{array}$$

$$\begin{array}{c}
(\tau\text{-OUTPUT } \uparrow) \text{ (where } \{\tilde{q}\} \cap \text{fn}(P'') = \emptyset) \\
P \succ (\nu \tilde{p}) \langle n[P'] \rangle P'' \quad P' \succ (\nu \tilde{q}) \langle \langle M \rangle^\uparrow Q' \rangle Q'' \quad P'' \xrightarrow{(M)} R \\
\hline
P \xrightarrow{\tau} (\nu \tilde{p}, \tilde{q}) (n[Q' \mid Q''] \mid R)
\end{array}$$

Theorem 4.4.3, below, proves that τ -transitions and reductions are the same, up to structural congruence.

Theorem 4.4.3 (Harmony). *If $P \xrightarrow{\tau} P'$ then $P \rightarrow P'$. If $P \rightarrow P'$ then $P \xrightarrow{\tau} \equiv P'$. \square*

4.4.3 Context and Activity Lemmas

Before proving non-interference, we state two important lemmas that are useful to establish contextual equivalence. The first is a context lemma, that implies that we may consider only a limited set of contexts when proving contextual equivalences, namely, evaluation contexts.

$$\textit{Evaluation Contexts} \quad E ::= - \mid (\nu n)E \mid P \mid E \mid E \mid Q \mid n[E]$$

In an evaluation context there is exactly one variable “ $-$ ”. We denote with $E\{P\}$ the process resulting from substituting the variable with P in E : differently from contexts, names restricted in an evaluation context E are renamed in $E\{P\}$ to avoid capture of free names of P . The different semantics of the two classes of contexts is emphasized by using

a different notation for evaluation contexts ($E\{P\}$) and contexts ($\mathbf{C}(P)$). We extend the notion of structural equivalence and reduction for evaluation contexts as follows:

- $E \equiv E' \triangleq E\{P\} \equiv E'\{P\}$ for every process P .
- $E \rightarrow E' \triangleq E\{P\} \rightarrow E'\{P\}$ for every process P .

Our context theorem is the equivalent of the corresponding theorem for Mobile Ambients in [CG99a]. In addition, we need to restrict to “compatible substitutions”, in the following sense. A *substitution* ϑ is a mapping from variables to terms such that $\vartheta(x) = M$ implies $fv(M) = \emptyset$. A substitution is Γ -compatible is if $\text{Dom}(\vartheta) \subseteq \text{Dom}(\Gamma)$ and $\Gamma(x) = W$ implies $\Gamma \vdash \vartheta(x) : W$. Let $\Delta|_{\vartheta}$ be the type environment $\Delta \setminus \text{Dom}(\vartheta)$.

Theorem 4.4.4 (Context). *For all P, Q s.t. $\Delta \vdash P : T$ and $\Delta \vdash Q : T$, $\Delta \triangleright P \cong_L Q$ if and only if for all Δ -compatible substitutions ϑ s.t. $\text{Dom}(\vartheta) = fv(P) \cup fv(Q)$, for all closed $(\Gamma/\Delta|_{\vartheta}, T)$ evaluation contexts E , and for all n such that $\Lambda(\Gamma(n)) \in L$ one has $E\{P\vartheta\} \Downarrow_n \iff E\{Q\vartheta\} \Downarrow_n$.* \square

In order to use the previous theorem, we need to analyze judgments of the form $E\{P\} \Downarrow_n$ and $E\{P\} \rightarrow Q$. These analyses are formalized by the following two results, whose proofs are in Appendix 4.6. The first result states that there are two ways in which $E\{P\} \Downarrow_n$ can arise: either the process P exhibits the name by itself, or the evaluation context E exhibits the name n by itself. The second result is an activity lemma showing how a reduction $E\{P\} \rightarrow Q$ may arise.

Before giving the formal statement of these two lemmas, we informally explain the core of the proof. The main idea is to show that for every context $\mathbf{C}()$ and high level source H , if $\mathbf{C}(H)$ is well typed then it is indistinguishable at low level from $\mathbf{C}(\mathbf{0})$. Using theorem 4.4.4, we know that we can restrict to evaluation contexts. The main lemma (lemma 4.4.16) shows that the low level observation of a process $E\{H\}$ does not depend on any interaction between the high level source H and the evaluation context $E\{-\}$. The crux of the proof is to show that if in $E\{H\}$ there is an interaction between H and $E\{-\}$, this interaction actually takes place between H and a high level subprocess H' of $E\{-\}$. Then $E\{H\} \equiv E'\{H', H\}$ where H and H' are not necessarily contiguous; therefore we need to show that the low level observation of $E'\{H', H\}$ is equal to that of $E'\{\mathbf{0}, \mathbf{0}\}$. This explains why lemma 4.4.16 has a more general statement that considers a multihole evaluation context filled with a number of high level sources.

Evaluation contexts with k holes are written $E\{-_1, \dots, -_k\}$. We write $E\{P_1, \dots, P_k\}$ for the process resulting from the simultaneous, capture free, substitution of processes P_1, \dots, P_k for the k variables $-_i$ contained in E . The definition of $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole (evaluation) contexts is the following generalization of Definition 4.3.2:

Definition 4.4.5 ($(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole Context). Let Γ and Δ_i be type environments and T_i process types, $i = 1, \dots, k$. $\mathbf{C}()$ is a $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole context if Δ_i for $i = 1, \dots, k$ are pairwise compatible type environments and $\Gamma \vdash_{(\sigma, \rho)} \mathbf{C}() : ok$ with $\sigma \in L$, is derivable in the type system of Section 4.2 enriched with the k rules that derive $\Theta_i \vdash -_i : T_i$ for all Θ_i extending Δ_i . \square

Proposition 4.4.6. *If $E\{P_1, \dots, P_k\} \downarrow_n$, then either*

1. $E\{Q_1, \dots, Q_k\} \downarrow_n$ for all Q_1, \dots, Q_k , or
2. $\exists i$ such that $P_i \downarrow_n$ and for all Q , $Q \downarrow_n \Rightarrow E\{P_1, \dots, P_{i-1}, Q, P_{i+1}, \dots, P_k\} \downarrow_n$ \square

In order to prove Theorem 4.4.9, we adopt the following notation.

Definition 4.4.7 (Interaction between evaluation contexts and processes). Let E be a 1-hole evaluation context. Let $E \bullet P \rightsquigarrow R$ if and only if there are E' , 1-hole evaluation context, and \tilde{r} with $\{\tilde{r}\} \cap fn(P) = \emptyset$, and one of the following holds:

- (**Inter In**) $E \equiv (\nu \tilde{r})E'\{m[- \mid R'] \mid n[R'']\}$, $P \xrightarrow{\text{in } n} P'$ and $R \equiv (\nu \tilde{r})E'\{n[m[P' \mid R'] \mid R'']\}$
- (**Inter Out**) $E \equiv (\nu \tilde{r})E'\{n[m[- \mid R'] \mid R'']\}$, $P \xrightarrow{\text{out } n} P'$ and $R \equiv (\nu \tilde{r})E'\{n[R''] \mid m[P' \mid R']\}$
- (**Inter Input**) $E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle R'\}$, $P \xrightarrow{(M)} P'$ and $R \equiv (\nu \tilde{r})E'\{P' \mid R'\}$
- (**Inter Output**) $E \equiv (\nu \tilde{r})E'\{- \mid (x)R'\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle Q \rangle P'$
and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid Q \mid R'\{x := M\})\}$ with $\tilde{p} \cap fn(R') = \emptyset$.
- (**Inter Input n**) $E \equiv (\nu \tilde{r})E'\{- \mid n[\langle M \rangle Q' \mid Q'']\}$, $P \succ (\nu \tilde{p})\langle (x)^n P' \rangle P''$
and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P'\{x := M\} \mid P'' \mid n[Q' \mid Q''])\}$ with $\{\tilde{p}\} \cap fn(n[\langle M \rangle Q' \mid Q'']) = \emptyset$.
- (**Inter Output from n**) $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid (x)^n R''\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle P' \rangle P''$,
 $\{\tilde{p}\} \cap (fn(R') \cup fn((x)^n R'')) = \emptyset$. and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P' \mid P'' \mid R'] \mid R''\{x := M\})\}$
- (**Inter Output upwd**) $E \equiv (\nu \tilde{r})E'\{- \mid n[(x)^\dagger Q' \mid Q'']\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle P' \rangle P''$,
 $\tilde{p} \cap fn(n[Q' \mid Q'']) = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid P'' \mid n[Q'\{x := M\} \mid Q''])\}$
- (**Inter Input \uparrow from n**) $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid \langle M \rangle R''\}$, $P \xrightarrow{(M)^\dagger} P'$
and $R \equiv (\nu \tilde{r})E'\{n[P' \mid R'] \mid R''\}$

- (Inter Output n)** $E \equiv (\nu \tilde{r})E'\{- \mid n[(x)Q' \mid Q'']\}, P \succ (\nu \tilde{p})\langle \langle M \rangle^n P' \rangle P''$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid P'' \mid n[Q'\{x := M\} \mid Q''])\}$ with $\tilde{p} \cap \text{fn}(n[Q' \mid Q'']) = \emptyset$.
- (Inter Input from n)** $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid \langle M \rangle^n R''\}, P \xrightarrow{(M)} P'$
 and $R \equiv (\nu \tilde{r})E'\{n[P' \mid R'] \mid R''\}$
- (Inter Input upwd)** $E \equiv (\nu \tilde{r})E'\{- \mid n[\langle M \rangle^\uparrow Q' \mid Q'']\}, P \xrightarrow{(M)} P'$
 and $R \equiv (\nu \tilde{r})E'\{P' \mid n[Q' \mid Q'']\}$.
- (Inter Output \uparrow from n)** $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid (x)R''\}, P \succ (\nu \tilde{p})\langle \langle M \rangle^\uparrow P' \rangle P''$,
 $\{\tilde{p}\} \cap (\text{fn}(R') \cup \text{fn}(R'') \cup \{n\}) = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P' \mid P'' \mid R'] \mid R''\{x := M\})\}$
- (Inter Amb)** $P \succ (\nu \tilde{p})\langle n[Q] \rangle P'$ and one of the following:
- (1) $Q \xrightarrow{\text{in } m} Q', E \equiv (\nu \tilde{r})E'\{- \mid m[R']\}, \tilde{p} \cap \text{fn}(m[R']) = \emptyset$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid m[n[Q'] \mid R'])\}$
 - (2) $Q \xrightarrow{\text{out } m} Q', E \equiv (\nu \tilde{r})E'\{m[- \mid R']\}, m \notin \tilde{p}$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[Q'] \mid m[R' \mid P'])\}$
 - (3) $E \equiv (\nu \tilde{r})E'\{m[R' \mid \text{in } n.R''] \mid -\}, \tilde{p} \cap \text{fn}(m[R' \mid \text{in } n.R'']) = \emptyset$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[Q \mid m[R' \mid R'']] \mid P')\}$
 - (4) $Q \succ (\nu \tilde{q})\langle \langle M \rangle Q' \rangle Q'', E \equiv (\nu \tilde{r})E'\{- \mid (x)^n R'\}, \{\tilde{p}\} \cap (\{n\} \cup \text{fn}(R')) = \emptyset$,
 $\{\tilde{q}\} \cap (\{n\} \cup \text{fn}(P' \mid R')) = \emptyset$,
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p}, \tilde{q})(R'\{x := M\} \mid n[Q' \mid Q''] \mid P')\}$
 - (5) $Q \xrightarrow{(M)} Q', E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle^n R'\}, \tilde{p} \cap \text{fn}(\langle M \rangle^n R') = \emptyset$.
 and $R \equiv (\nu \tilde{r})E'\{R' \mid (\nu \tilde{p})(n[Q'] \mid P')\}$
 - (6) $Q \succ (\nu \tilde{q})\langle \langle M \rangle^\uparrow Q' \rangle Q'', E \equiv (\nu \tilde{r})E'\{- \mid (x)R'\}, \{\tilde{p}\} \cap (\{n\} \cup \text{fn}(R')) = \emptyset$,
 $\{\tilde{q}\} \cap (\{n\} \cup \text{fn}(P' \mid R')) = \emptyset$,
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p}, \tilde{q})(R'\{x := M\} \mid n[Q' \mid Q''] \mid P')\}$
 - (7) $Q \xrightarrow{(M)^\uparrow} Q', E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle R'\}, \tilde{p} \cap \text{fn}(\langle M \rangle R') = \emptyset$.
 and $R \equiv (\nu \tilde{r})E'\{R' \mid (\nu \tilde{p})(n[Q'] \mid P')\}$ □

Definition 4.4.8 (Interaction between two holes). Let E be a 2-hole evaluation context. Let $E \bullet (P_1, P_2) \rightsquigarrow R$ if and only if there are E' , 2-hole evaluation context, and \tilde{r} with $\{\tilde{r}\} \cap (\text{fn}(P_1) \cup \text{fn}(P_2)) = \emptyset$, and one of the following holds:

- (Inter In)** $E \equiv (\nu \tilde{r})E'\{m[-_1 \mid R'] \mid -_2\}, P_1 \xrightarrow{\text{in } n} P'_1, P_2 \succ (\nu \tilde{p})\langle n[P'_2] \rangle P'_2$,
 $\text{fn}(m[P'_1 \mid R']) \cap \{\tilde{p}\} = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[m[P'_1 \mid R'] \mid P'_2] \mid P'_2)\}$
- (Inter Output from n)** $E \equiv (\nu \tilde{r})E'\{n[-_1 \mid R'] \mid -_2\}, P_1 \succ (\nu \tilde{p})\langle \langle M \rangle P'_1 \rangle P'_1$,
 $P_2 \succ (\nu \tilde{q})\langle (x)^n P'_2 \rangle P'_2$, $\{\tilde{p}\} \cap (\text{fn}(R') \cup \text{fn}(P_2)) = \emptyset$, $\{\tilde{q}\} \cap \text{fn}(M) = \emptyset$ and
 $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid P'_2 \mid R'] \mid (\nu \tilde{q})(P'_2\{x := M\} \mid P'_2))\}$

- (**Inter Input** \uparrow **from n**) $E \equiv (\nu \tilde{r})E'\{n[-1 \mid R'] \mid -2\}, P_1 \xrightarrow{(M)^\dagger} P'_1, P_2 \succ (\nu \tilde{p})\langle\langle M \rangle P'_2\rangle P''_2,$
 $fn(n[P_1 \mid R']) \cap \{\tilde{p}\} = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid R'] \mid P'_2 \mid P''_2)\}$
- (**Inter Input from n**) $E \equiv (\nu \tilde{r})E'\{n[-1 \mid R'] \mid -2\}, P_1 \xrightarrow{(M)} P'_1, P_2 \succ (\nu \tilde{p})\langle\langle M \rangle^n P'_2\rangle P''_2,$
 $fn(n[P_1 \mid R']) \cap \{\tilde{p}\} = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid R'] \mid P'_2 \mid P''_2)\}$
- (**Inter Output** \uparrow **from n**) $E \equiv (\nu \tilde{r})E'\{n[-1 \mid R'] \mid -2\}, P_1 \succ (\nu \tilde{p})\langle\langle M \rangle^\dagger P'_1\rangle P'_1, P_2 \xrightarrow{(M)} P'_2$
 $\{\tilde{p}\} \cap (fn(R') \cup fn(P_2) \cup \{n\}) = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid P'_1 \mid R'] \mid P'_2)\}$ \square

The Activity theorem states that if $E\{P_1, \dots, P_k\} \rightarrow R$, then either (1) one of the processes P_i reduces by itself, or (2) the evaluation context E reduces by itself, or (3) there is an interaction between two process P_i, P_j , or (4)-(5) there is an interaction between the evaluation context and one or two processes P_i, P_j .

Theorem 4.4.9 (Activity). $E\{P_1, \dots, P_k\} \rightarrow R$ if and only if:

- (**Act Proc**) $\exists i$ such that $P_i \rightarrow P'_i$ with $R \equiv E\{P_1, \dots, P'_i, \dots, P_k\}$, or
- (**Act Ctx**) $E \rightarrow E'$ with $R \equiv E'\{P_1, \dots, P_k\}$, or
- (**Act Holes**) $\exists i, j$ and a $(k-1)$ -hole evaluation context E' such that
 $E\{P_1, \dots, P_k\} \equiv E'\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}$, and there is a reduction $P_i \mid P_j \rightarrow \bar{P}$ with
 $R \equiv E'\{P_1, \dots, \bar{P}, \dots, P_k\}$.
- (**Act Inter one hole**) $\exists i$ such that $E\{P_1, \dots, -, \dots, P_k\} \bullet P_i \rightsquigarrow R$.
- (**Act Inter two holes**) $\exists i, j$ such that $E\{P_1, \dots, P_{i-1}, -, P_{i+1}, \dots, P_{j-1}, -, P_{j+1}, \dots, P_k\} \bullet$
 $(P_i, P_j) \rightsquigarrow R$.

\square

4.4.4 Proof of non-interference

We start with a few preliminary results on the type system, showing some useful invariants that are guaranteed for well-typed processes.

Proposition 4.4.10 (High Processes and Low Barbs). Assume $\Gamma \Vdash H : T$ for some T . Then $H \downarrow_n \Rightarrow n \in \mathbf{H}$.

Proof. By definition, $H \downarrow_n$ if and only if $H \equiv (\nu \tilde{p})n[\langle M \rangle P \mid Q] \mid P'$ and $n \notin \{\tilde{p}\}$. By Definition 4.3.1 we have that $\Lambda(\Gamma(n)) \in \mathbf{H}$. \square

Lemma 4.4.11 (\Vdash versus \vdash). *If $\Gamma \Vdash_{(\sigma, \rho)} P : T$ then $\Gamma \vdash_{(\sigma, \rho)} P : T$.*

Proof. Immediate: well-formed types of \Vdash derivations are also well-formed for the corresponding \vdash -derivations. \square

Lemma 4.4.12.

1. *Let E be a closed $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole evaluation context and $E \rightarrow E'$. Then E' is also a closed $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole evaluation context.*
2. *Let P, Q be two processes such that $P \equiv Q$ and $P \Downarrow_n$ for some name n . Then $Q \Downarrow_n$ by a reduction sequence of the same length as $P \Downarrow_n$.*
3. *Let $\Gamma \Vdash m[P] : T$ be a derivable judgment. Let be $P \xrightarrow{\alpha} P'$, then $\Gamma \Vdash m[P'] : T$ is also a derivable judgment.*
4. *Let be $\Gamma \Vdash P : T$ and $P \xrightarrow{\text{in } n} P'$, then $\Lambda(\Gamma(n)) \in \mathbf{H}$.*
5. *Let be $\Gamma \Vdash_{(\sigma, \rho)} P : T$ and $P \xrightarrow{\text{out } n} P'$, then $\Lambda(\Gamma(n)) \in \mathbf{H}$ and $\sigma \in \mathbf{H}$.*
6. *Let be $\Gamma \Vdash P : T$ and $P \xrightarrow{(M)} P'$, then $\Lambda(\Gamma(M)) \in \mathbf{H}$.*

Proof. Immediate \square

Lemma 4.4.13. *Assume $\Delta_i \vdash P_i : T_i, i = 1, \dots, k$. For every closed $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole evaluation context E , and for all n one has $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_n \Rightarrow E\{P_1, \dots, P_k\} \Downarrow_n$.*

Proof. Take any name n . The proof is by induction on the number of reductions. For the base case, assume $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_n$. Since $\mathbf{0} \not\Downarrow_n$, by Proposition 4.4.6 we know that $E\{P_1, \dots, P_k\} \Downarrow_n$ for all P_1, \dots, P_k . For the inductive case, assume $E\{\mathbf{0}, \dots, \mathbf{0}\} \rightarrow R$ and $R \Downarrow_n$. By Theorem 4.4.9, it must be the case that $R \equiv E'\{\mathbf{0}, \dots, \mathbf{0}\}$ with $E \rightarrow E'$. By Proposition 4.2.3 (Subject Reduction), E' is a $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole evaluation context, and by Lemma 4.4.12(1) it is closed. Furthermore, by Lemma 4.4.12(2) $E'\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_n$ by a reduction sequence of the same length as $R \Downarrow_n$. Thus, by induction hypothesis, we know that $E'\{P_1, \dots, P_k\} \Downarrow_n$. We are done, since $E\{P_1, \dots, P_k\} \rightarrow E'\{P_1, \dots, P_k\}$. \square

Corollary 4.4.14. *Assume $\Delta_i \vdash H_i : T_i, i = 1, \dots, k$ with $\text{fv}(H_i) = \emptyset$. For every closed $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole evaluation context E , and for all n , one has $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_n \Rightarrow E\{H_1, \dots, H_k\} \Downarrow_n$.* \square

Lemma 4.4.15.

1. If $\Gamma \Vdash P : T$ and $P \rightarrow P'$, then $\Gamma \Vdash P' : T$.
2. If $\Gamma \Vdash (x:W)^n P$ and $\Gamma \vdash M : W$, then $\Gamma \Vdash P\{x := M\}$.
3. Let $\sigma \in \mathbf{L}$ and $\rho \in \mathbf{H}$, then $\Gamma \vdash_{(\sigma, \rho)} P$ if and only if $\Gamma \Vdash_{(\sigma, \rho)} P$.

Proof. By induction on the judgment in hypothesis. □

Lemma 4.4.16. Let $\Delta_i \Vdash H_i : T_i$, $i = 1, \dots, k$ with $fv(H_i) = \emptyset$. Let E be a closed $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ k -hole evaluation context. For all n s.t. $\Lambda(\Gamma(n)) \in \mathbf{L}$, one has $E\{H_1, \dots, H_k\} \Downarrow_n \Rightarrow E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_n$

Proof. Consider any ℓ with $\Lambda(\Gamma(\ell)) \in \mathbf{L}$. The proof is by induction on the length of the derivation $E\{H_1, \dots, H_k\} \Downarrow_\ell$. For the base case, assume $E\{H_1, \dots, H_k\} \Downarrow_\ell$. By Proposition 4.4.10 we know that $H_i \Downarrow_\ell$. Then, by Proposition 4.4.6, $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$. For the inductive case, assume $E\{H_1, \dots, H_k\} \rightarrow R$ and $R \Downarrow_\ell$. By Theorem 4.4.9, one of the following cases applies.

(Act Proc) $\exists i$ such that $H_i \rightarrow H'_i$ with $R \equiv E\{H_1, \dots, H'_i, \dots, H_k\}$. By Lemma 4.4.15 (1), we know that $\Delta_i \Vdash H'_i : T_i$. By Induction hypothesis, $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$.

(Act Ctx) $E \rightarrow E'$ with $R \equiv E'\{H_1, \dots, H_k\}$. By Proposition 4.2.3 (Subject Reduction) and Lemma 4.4.12(1) E' is a closed k -hole evaluation context, and $E'\{H_1, \dots, H_k\} \Downarrow_\ell$ by a reduction sequence of the same length as $R \Downarrow_\ell$. By Induction hypothesis, $E'\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$ and then $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$ as $E\{\mathbf{0}, \dots, \mathbf{0}\} \rightarrow E'\{\mathbf{0}, \dots, \mathbf{0}\}$.

(Act Holes) $\exists i, j$ and a $(k-1)$ -hole evaluation context E' such that $E\{H_1, \dots, H_k\} \equiv E'\{H_1, \dots, (H_i \mid H_j), \dots, H_k\}$, $H_i \mid H_j \rightarrow \overline{H}$ and $R \equiv E'\{H_1, \dots, \overline{H}, \dots, H_k\}$. Since E is a $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ evaluation context, we have that Δ_i and Δ_j are compatible type environments, and $\Delta_i \cup \Delta_j \Vdash H_i \mid H_j$ is a derivable judgment. Then by Lemma 4.4.15 we have $\Delta_i \cup \Delta_j \Vdash \overline{H}$, then by induction we have $E'\{\mathbf{0}, \dots, \mathbf{0}\}_{k-1} \Downarrow_\ell$, thus $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$ as desired.

(Act Inter one hole) There exist i such that $E\{H_1, \dots, -, \dots, H_k\} \bullet H_i \rightsquigarrow R$. Then there exists a 1-hole evaluation context E' and \tilde{r} , with $\tilde{r} \cap fn(H_i) = \emptyset$, and one of several sub-cases applies.

(Inter In) $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu \tilde{r} : \tilde{U})E'\{m[- \mid R'] \mid n[R'']\}$, $H_i \xrightarrow{\text{in } n} H'$, and $R \equiv (\nu \tilde{r} : \tilde{U})E'\{n[m[H' \mid R'] \mid R'']\}$. By Lemma 4.4.12(4) it follows that $\Lambda(\Delta_i(n)) \in \mathbf{H}$. Since E is a $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ (evaluation) context, one has $\Lambda(\Gamma(n)) \in \mathbf{H}$. Then it is easy to see that $\Gamma \Vdash n[R''] : T'$ for some T' . Hence also $\Gamma \Vdash n[m[H' \mid R'] \mid R''] : T'$. Then $\exists E_2, E'_2, E''_2$ multihole evaluation contexts such that $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu \tilde{r} : \tilde{U})$
 $E_2\{H_1, \dots, H_{i-1}, (m[- \mid E'_2\{H_{i+1}, \dots, H_p\}] \mid n[E''_2\{H_{p+1}, \dots, H_q\}]), H_{q+1}, \dots, H_k\}$
and $R \equiv (\nu \tilde{r} : \tilde{U})E_2\{H_1, \dots, H_{i-1}, (n[m[H' \mid R'] \mid R'']), H_{q+1}, \dots, H_k\}$. By induction we have $(\nu \tilde{r} : \tilde{U})E_2\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, then by Lemma 4.4.13,
 $(\nu \tilde{r} : \tilde{U})E_2\{\mathbf{0}, \dots, \mathbf{0}, (m[\mathbf{0} \mid E'_2\{\mathbf{0}, \dots, \mathbf{0}\}] \mid n[E''_2\{\mathbf{0}, \dots, \mathbf{0}\}]), \mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, that is exactly $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$ as desired.

(Inter Out) $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu \tilde{r})E'\{n[m[- \mid R'] \mid R'']\}$, $H_i \xrightarrow{\text{out } n} H'$ and $R \equiv (\nu \tilde{r})E'\{n[R''] \mid m[H' \mid R']\}$. By Lemma 4.4.12(5), $\Lambda(\Gamma(n)), \Lambda(\Gamma(m)) \in \mathbf{H}$. Then it is easy to see that $\Gamma \Vdash n[R''] \mid m[H' \mid R'] : T'$ for some T' . Then $\exists E_2, E'_2, E''_2$ multihole evaluation contexts s. t. $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu \tilde{r})$
 $E_2\{H_1, \dots, H_{i-1}, (n[m[- \mid E'_2\{H_{i+1}, \dots, H_p\}] \mid E''_2\{H_{p+1}, \dots, H_q\}]), H_{q+1}, \dots, H_k\}$
and $R \equiv (\nu \tilde{r})E_2\{H_1, \dots, H_{i-1}, (n[R''] \mid m[H' \mid R']), H_{q+1}, \dots, H_k\}$. By induction we have $(\nu \tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, then by Lemma 4.4.13,
 $(\nu \tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}, (n[m[\mathbf{0} \mid E'_2\{\mathbf{0}, \dots, \mathbf{0}\}] \mid E''_2\{\mathbf{0}, \dots, \mathbf{0}\}]), \mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, that is exactly $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$ as desired.

(Inter Input) $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle R'\}$, $H_i \xrightarrow{(M)} H'$ and $R \equiv (\nu \tilde{r})E'\{H' \mid R'\}$. By Lemma 4.4.12(6), $\Lambda(\Gamma(M)) \in \mathbf{H}$. Then we have to distinguish two subcases:

1. $\Gamma \vdash_{(\sigma, \rho)} \langle M \rangle R'$ with $\rho \in \mathbf{H}$ and $\sigma \in \mathbf{L}$. By Lemma 4.4.15(3), we know $\Gamma \Vdash \langle M \rangle R'$, that implies $\Gamma \Vdash H' \mid R'$. Then $\exists E_2, E'_2$ multihole evaluation contexts such that $E\{H_1, \dots, -, \dots, H_k\} \equiv$
 $(\nu \tilde{r})E_2\{H_1, \dots, H_{i-1}, (- \mid \langle M \rangle E'_2\{H_{i+1}, \dots, H_q\}), H_{q+1}, \dots, H_k\}$ and
 $R \equiv (\nu \tilde{r})E_2\{H_1, \dots, H_{i-1}, (H' \mid E'_2\{H_{i+1}, \dots, H_q\}), H_{q+1}, \dots, H_k\}$. By induction we have $(\nu \tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, then by Lemma 4.4.13, we conclude
 $(\nu \tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}, (\mathbf{0} \mid \langle M \rangle E'_2\{\mathbf{0}, \dots, \mathbf{0}\}), \mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, that is $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$.
2. $\Gamma \vdash_{(\sigma, \rho)} \langle M \rangle R'$ with $\rho \in \mathbf{H}$ and $\sigma \in \mathbf{H}$. Since E is a $(\Gamma/\Delta_1, \dots, \Delta_k, T_1, \dots, T_k)$ evaluation context, from Def.4.3.2 we have $\Gamma \vdash_{(\sigma', \rho')} E\{\} : ok$ with $\sigma' \in$

L. Then $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu\tilde{r})E'_1\{h[(\nu\tilde{q})(- \mid \langle M \rangle R' \mid R'')]\}$ with $\Lambda(\Gamma(h)) \in \mathbf{H}$, and $R \equiv (\nu\tilde{r})E'_1\{h[(\nu\tilde{q})(H' \mid R' \mid R'')]\}$. From $\Lambda(\Gamma(h)) \in \mathbf{H}$ we have $\Gamma \Vdash h[(\nu\tilde{q})(H' \mid R' \mid R'')]$. Then $\exists E_2, E'_2$ multihole evaluation contexts such that $E\{H_1, \dots, -, \dots, H_k\} \equiv (\nu\tilde{r})E_2\{H_1, \dots, H_{i-1}, (h[(\nu\tilde{q})(- \mid \langle M \rangle E'_2\{H_{i+1}, \dots, H_p\} \mid E''_2\{H_{p+1}, \dots, H_q\})]), H_{q+1}, \dots, H_k\}$ and $R \equiv (\nu\tilde{r})E_2\{H_1, \dots, H_{i-1}, (h[(\nu\tilde{q})(H' \mid E'_2\{H_{i+1}, \dots, H_p\} \mid E''_2\{H_{p+1}, \dots, H_q\})]), H_{q+1}, \dots, H_k\}$. By induction $(\nu\tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, then by Lemma 4.4.13, $(\nu\tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}, (h[(\nu\tilde{q})(\mathbf{0} \mid \langle M \rangle E'_2\{H_{i+1}, \dots, H_p\} \mid E''_2\{H_{p+1}, \dots, H_q\})]), \mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, that is exactly $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$ as desired.

The remaining cases of (Act Inter one hole) follow similarly.

(Act Inter two holes) There exist i, j such that $E\{\dots, -1, \dots, -2, \dots\} \bullet (H_i, H_j) \rightsquigarrow R$. Then there exist a 2-hole evaluation context E' and \tilde{r} , with $\tilde{r} \cap (fn(H_i) \cup fn(H_j)) = \emptyset$, and one of several sub-cases applies.

(Inter In) $E\{\dots, -1, \dots, -2, \dots\} \equiv (\nu\tilde{r})E'\{m[-1 \mid R'] \mid -2\}$, $H_i \xrightarrow{\text{in } n} H'_i$, $H_j \succ (\nu\tilde{p})\langle n[H'] \rangle H''$ and $R \equiv (\nu\tilde{r})E'\{n[m[H'_i \mid R'] \mid H'] \mid H''\}$. From $\Delta_j \Vdash H_j$ we have $\Lambda(\Gamma(n)) \in \mathbf{H}$ and $\Delta_j \Vdash H''$, thus $\Delta_j \Vdash n[m[H'_i \mid R'] \mid H'] \mid H''$. Then $\exists E_2, E'_2$ multihole evaluation contexts such that $E\{\dots, -1, \dots, -2\} \equiv (\nu\tilde{r})E_2\{H_1, \dots, H_{i-1}, (m[-1 \mid E'_2\{H_{i+1}, \dots, H_{j-1}\}] \mid -2), H_{j+1}, \dots, H_k\}$ and $R \equiv (\nu\tilde{r})E_2\{H_1, \dots, H_{i-1}, (n[m[H'_i \mid E'_2\{H_{i+1}, \dots, H_{j-1}\}]] \mid H'] \mid H''), H_{j+1}, \dots, H_k\}$. By induction $(\nu\tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, then by Lemma 4.4.13, $(\nu\tilde{r})E_2\{\mathbf{0}, \dots, \mathbf{0}, (m[\mathbf{0} \mid E'_2\{\mathbf{0}, \dots, \mathbf{0}\}] \mid \mathbf{0}), \mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$, that is $E\{\mathbf{0}, \dots, \mathbf{0}\} \Downarrow_\ell$.

The remaining cases of (Act Inter two holes) follow similarly.

□

Corollary 4.4.17. Assume $\Delta \Vdash H : T$. For all Δ -compatible ϑ with $\text{Dom}(\vartheta) = \text{fv}(H)$, for every closed $(\Gamma/\Delta|_{\vartheta}, T)$ evaluation context E , and for all n with $\Lambda(\Gamma(n)) \in \mathbf{L}$, one has $E\{H\vartheta\} \Downarrow_n \Leftrightarrow E\{\mathbf{0}\} \Downarrow_n$.

Proof. By Lemma 4.2.23 (Substitution), and the hypothesis of ϑ being Δ -compatible, it follows that $\Delta|_{\vartheta} \Vdash H\vartheta : T$. Then the proof follows by Corollary 4.4.14 and Proposition 4.4.16. □

Theorem 4.4.18 (High Processes). *Assume $\Delta \Vdash H : T$. For every $(\Gamma/\Delta, T)$ context $\mathbf{C}()$ with $\mathbf{C}(H)$ and $\mathbf{C}(\mathbf{0})$ closed, and for all n with $\Lambda(\Gamma(n)) \in \mathbf{L}$, one has $\mathbf{C}(H) \Downarrow_n \Leftrightarrow \mathbf{C}(\mathbf{0}) \Downarrow_n$*

Proof. Apply the Context theorem (Theorem 4.4.4), and the previous corollary. \square

Proof of Theorem 4.3.9. Assume $\Delta \vdash P : T$. We want to prove that for every H such that $\Delta \vdash H : T$ one has $\Delta \triangleright P \cong_{\mathbf{L}} P \mid H$. Let then $\mathbf{C}()$ be any $(\Gamma/\Delta, T)$ context with $\mathbf{C}(P)$ and $\mathbf{C}(P \mid H)$ closed. Now, define $\mathbf{C}'() = \mathbf{C}(P \mid -)$. Then $\mathbf{C}'()$ is also a $(\Gamma/\Delta, T)$ context with $\mathbf{C}'(H)$ and $\mathbf{C}'(\mathbf{0})$ closed. By Theorem 4.4.18 we have that for all n with $\Lambda(\Gamma(n)) \in \mathbf{L}$, $\mathbf{C}'(H) = \mathbf{C}(P \mid H) \Downarrow_n \Leftrightarrow \mathbf{C}'(\mathbf{0}) \equiv \mathbf{C}(P) \Downarrow_n$.

4.5 Related Work

Volpano and Smith [VSI96, VS97, VS98], and recently Boudol and Castellani [BC01a] study type-based techniques to enforce non-interference in multi-threaded imperative languages. In their approach explicit flow is prevented by imposing constraints on variable assignments, while additional restrictions on conditional commands and while-loops rule out implicit flow. In [BC01a] the authors point out that introducing parallelism may cause new problems, since information flow may be “disguised as control flow”, and a program may observe (and be influenced by) the behavior of other concurrent components in the course of their execution. The problem is solved in [VS98, BC01a] by relying on a form of asynchrony, whereby consulting the value of a high-level variable must not be followed by an assignment to a low variable. In BA we have a similar problem even though in a different setting, and our solution follows the same rationale, by imposing a non-decreasing clearance on the sequence of ‘actions’ performed by a process.

More directly related to ours are the type systems for π -calculus by Honda et al [HVG00] and for the *security π calculus* by Hennessy and Riely [HR00]. In [HVG00], the authors propose to use the informal principle of causal dependency to understand safety of information flow in various programming language, and develop a type system based on a behavioral notion of types to capture causality of actions. Our approach is similar, as it also draws on the principle of causal dependency, but our framework appears to be more complex, as in BA processes may interact both via communications and mobility.

In [HR00], security levels are attached to processes and to capabilities for reading/writing to channels, and a 'no read-up/no write-down' security policy is enforced by typing. To prove non-interference, further restrictions must be imposed, namely high-level processes must not evolve in low-level ones and the calculus must be asynchronous. Under these hypotheses, the authors show that well-typed asynchronous processes are interference free, where non-interference is defined in a way similar to ours, based on *may test* equivalence. Our type system enforces similar restrictions on the value exchanges between high and low processes, and corresponding restrictions on mobility. Unlike [HR00], our result holds true for the synchronous case as well. In [Hen00], Hennessy has developed an enhanced type system for the security π calculus for which non-interference can be proved also with respect to *must test* equivalence.

In [SV00] Sewell and Vitek introduce *box- π* , a process calculus that provides mechanisms for composing (partially trusted) software components and for enforcing information flow security policies. Their approach is based on a colored semantics, which annotates output processes with the sets of principals that have affected them (the processes) in the past; then the security properties are stated in terms of a colored lts. Finally, they introduce a type system that statically captures causal flows. As such, the characterization of information flow security is based on a causal model, rather than on non-interference as in our approach. Further important differences are the asynchronous semantics of *box- π* (as opposed to the synchronous semantics of BA) and our treatment of mobility and nested topology. A more in-depth comparison between the two approaches deserves to be made.

No type-based study of non-interference appears to have been conducted on ambient-based calculi. A number of papers have instead dealt with other aspects of security ([CGG00a],[LS00],[DCS00]) as we discussed in the previous chapters.

Other approaches based on type systems [BC01b] and control-flow analyses have also been applied [NNHJ99, NN00] to analyze different security properties of (various dialects of) mobile ambients. In particular Braghin et al. [BCF02] study 'explicit' information flow security in the scenario of pure Mobile Ambients by defining a control-flow analysis to detect security breaches arising as confidential data moving outside any boundary protection.

4.6 Omitted Proofs

4.6.1 Proof of Harmony Theorem 4.4.3

Lemma 4.6.1 (Properties of Hardening).

1. If $P \succ (\nu \tilde{p})(P')P''$, then $\tilde{p} \subseteq \text{fn}(P')$ and the names \tilde{p} are pairwise distinct.
2. If $P \succ (\nu \tilde{p})(P')P''$, then $P \equiv (\nu \tilde{p})(P' \mid P'')$.
3. If $P \equiv Q$ and $Q \succ (\nu \tilde{r})(Q')Q''$, then $P \succ (\nu \tilde{r})(P')P''$ for P and P' such that $P' \equiv Q', P'' \equiv Q''$.

Proof. of 1. and 2. by induction on the derivation of $P \succ (\nu \tilde{p})(P')P''$; almost identical to that in [CG99a]. Proof of 3. by induction on the derivation of $P \equiv Q$; almost identical to that in [CG99a]. \square

Lemma 4.6.2.

1. If $P \xrightarrow{M} P'$ then $P \equiv (\nu \tilde{p})(P_1 \mid M.P_2)$ with $P' \equiv (\nu \tilde{p})(P_1 \mid P_2)$ and $\text{fn}(M) \cap \{\tilde{p}\} = \emptyset$.
2. If $P \xrightarrow{(M)^\eta} P'$ with $\eta \in \{\uparrow, \star\}$ then $P \equiv (\nu \tilde{p})(P_1 \mid (x)^\eta P_2)$ with $P' \equiv (\nu \tilde{p})(P_1 \mid P_2\{x := M\})$ and $\text{fn}(M) \cap \{\tilde{p}\} = \emptyset$.

Proof. Immediate. \square

Lemma 4.6.3. If $P \equiv Q$ then $\text{fn}(P) = \text{fn}(Q)$ and $\text{fv}(P) = \text{fv}(Q)$. \square

Lemma 4.6.4.

1. If $P \equiv Q$ and $Q \xrightarrow{\alpha} Q'$ then there exists P' such that $P \xrightarrow{\alpha} P'$ and $P' \equiv Q'$.
2. If $P \xrightarrow{\alpha} P'$ and $n \notin \text{fn}(\alpha)$, then there is Q with $(\nu n)P \xrightarrow{\alpha} Q$ and $Q \equiv (\nu n)P'$.
3. If $(\nu n)P \xrightarrow{\alpha} Q$ then $n \notin \text{fn}(\alpha)$ and there is P' with $P \xrightarrow{\alpha} P'$ and $Q \equiv (\nu n)P'$.

Proof. of 2. and 3. are almost identical to that in [CG99a]. Proof of 1. is by induction on $Q \xrightarrow{\alpha} Q'$; we give just a couple of cases as representative, the others are similar.

(Trans input up) In this case we have $Q \succ (\nu \tilde{r})(x)^\uparrow Q_1 \setminus Q_2$ and $Q' = (\nu \tilde{r})(Q_1\{x := M\} \mid Q_2)$, with $\text{fn}(M) \cap \{\tilde{r}\} = \emptyset$. Then by Lemma 4.6.1(3) there exists P_1, P_2 such that $P \succ$

$(\nu \tilde{r})(\langle x \rangle^\dagger P_1) P_2$ with $P_1 \equiv Q_1$, $P_2 \equiv Q_2$. Now, by (TRANS INPUT UP) $P \xrightarrow{(M)^\dagger} (\nu \tilde{r})(P_1 \{x := M\} \mid P_2) = P'$ and $P' \equiv Q'$ as desired.

(Trans in) In this case we have $Q \succ (\nu \tilde{p})(\langle n[Q_1] \rangle Q_2)$, $Q_1 \xrightarrow{\text{in } m} Q'_1$, $Q_2 \succ (\nu \tilde{r})(\langle m[Q'_2] \rangle Q''_2)$ and $Q' = (\nu \tilde{p}, \tilde{r})(m[n[Q'_1] \mid Q'_2] \mid Q''_2)$, with $fn(n[Q_1]) \cap \{\tilde{r}\} = \emptyset$. Then by Lemma 4.6.1(3) there exists P_1, P_2 such that $P \succ (\nu \tilde{p})(\langle n[P_1] \rangle P_2)$ with $P_1 \equiv Q_1$, $P_2 \equiv Q_2$. From $P_1 \equiv Q_1$, by induction there exist P'_1 s.t. $P_1 \xrightarrow{\text{in } m} P'_1$, $P'_1 \equiv Q'_1$. Now, again by Lemma 4.6.1(3), $P_2 \succ (\nu \tilde{r})(\langle m[P'_2] \rangle P''_2)$ with $P'_2 \equiv Q'_2$ and $P''_2 \equiv Q''_2$. Form $fn(n[Q_1]) \cap \{\tilde{r}\} = \emptyset$, by Lemma 4.6.3, we have $fn(n[P_1]) \cap \{\tilde{r}\} = \emptyset$, and by (TRANS IN) we conclude $P \xrightarrow{\tau} (\nu \tilde{p}, \tilde{r})(m[n[P'_1] \mid P'_2] \mid P''_2) = P'$ and $P' \equiv Q'$ as desired.

□

Proof of Theorem 4.4.3 [Harmony] $P \rightarrow Q$ if and only if $P \xrightarrow{\tau} \equiv Q$.

Right-to-left direction: if $P \xrightarrow{\tau} P'$ then $P \rightarrow P'$. The proof is by induction on the derivation of $P \xrightarrow{\tau} P'$ and a case analysis on the last rule applied. (τ IN) The hypothesis is $P \succ (\nu \tilde{p})(\langle n[Q] \rangle R)$, $Q \xrightarrow{\text{in } m} Q'$, $R \succ (\nu \tilde{r})(\langle m[R'] \rangle R'')$ with $fn(n[Q]) \cap \{\tilde{r}\} = \emptyset$, and $P' = (\nu \tilde{p}, \tilde{r})(m[n[Q'] \mid R'] \mid R'')$. By lemma 4.6.1(2) we have $P \equiv (\nu \tilde{p})(n[Q] \mid R)$ and $R \equiv (\nu \tilde{r})(m[R'] \mid R'')$. Now, by Lemma 4.6.2(1) we also have $Q \equiv (\nu \tilde{q})(Q_1 \mid \text{in } m.Q_2)$ and $Q' \equiv (\nu \tilde{q})(Q_1 \mid Q_2)$, $m \notin \{\tilde{q}\}$. Then:

$$\begin{aligned}
P &\equiv (\nu \tilde{p})(n[(\nu \tilde{q})(Q_1 \mid \text{in } m.Q_2)] \mid (\nu \tilde{r})(m[R'] \mid R'')) \\
&\equiv (\nu \tilde{p}, \tilde{r})(n[(\nu \tilde{q})(Q_1 \mid \text{in } m.Q_2)] \mid m[R'] \mid R'') \quad \text{since } fn(n[Q]) \cap \{\tilde{r}\} = \emptyset \\
&\quad \tilde{q} \text{ are bound, then assume } fn(n[R']) \cap \{\tilde{q}\} = \emptyset \\
&\equiv (\nu \tilde{p}, \tilde{r})((\nu \tilde{q})(n[Q_1 \mid \text{in } m.Q_2] \mid m[R']) \mid R'') \\
&\rightarrow (\nu \tilde{p}, \tilde{r})(m[n[(\nu \tilde{q})(Q_1 \mid Q_2)] \mid R'] \mid R'') \\
&\equiv (\nu \tilde{p}, \tilde{r})(m[n[Q'] \mid R'] \mid R'') \\
&\equiv P'
\end{aligned}$$

and we conclude by (Red Struct).

(τ INPUT \uparrow) The hypothesis is $P \succ (\nu \tilde{p})(\langle M \rangle P') P''$, $P'' \succ (\nu \tilde{q})(\langle n[R] \rangle R')$, $R \xrightarrow{(M)^\dagger} R''$, with $fn(\langle M \rangle P') \cap \{\tilde{q}\} = \emptyset$, and $P' = (\nu \tilde{p})(P' \mid (\nu \tilde{q})(n[R''] \mid R'))$. By lemma 4.6.1(2) and Lemma 4.6.2(2), $fn(M) \cap \{\tilde{r}\} = \emptyset$, $P \equiv (\nu \tilde{p})(\langle M \rangle P' \mid (\nu \tilde{q})(n[(\nu \tilde{r})(R_1 \mid (x)^\dagger R_2)] \mid R'))$

and $R'' = (\nu \tilde{r})(R_1 \mid R_2\{x := M\})$. Then:

$$\begin{aligned}
P &\equiv (\nu \tilde{p}, \tilde{q})(\langle M \rangle P' \mid n[(\nu \tilde{r})(R_1 \mid (x)^\dagger R_2)] \mid R') \quad \text{since } fn(\langle M \rangle P') \cap \{\tilde{q}\} = \emptyset \\
&\quad fn(M) \cap \{\tilde{r}\} = \emptyset \text{ and since } \tilde{r} \text{ are bound, assume } (fn(P') \cup \{n\}) \cap \{\tilde{r}\} = \emptyset \\
&\equiv (\nu \tilde{p}, \tilde{q})(\langle M \rangle P' \mid n[R_1 \mid (x)^\dagger R_2] \mid R') \\
&\rightarrow (\nu \tilde{p}, \tilde{q})(\langle M \rangle P' \mid n[R_1 \mid R_2\{x := M\}]) \mid R') \\
&\equiv (\nu \tilde{p}, \tilde{q})(P' \mid n[R'']) \mid R') \\
&\equiv P'
\end{aligned}$$

and we conclude by (Red Struct).

(TRANS AMB) We have $P \succ (\nu \tilde{p})(n[P_1] \mid P_2)$, $P_1 \xrightarrow{\tau} P'_1$ and $P' = (\nu \tilde{p})(n[P'_1] \mid P_2)$. By lemma 4.6.1(2) we have $P \equiv (\nu \tilde{p})(n[P_1] \mid P_2)$. By inductive hypothesis $P_1 \rightarrow P'_1$. Thus by context rules of reduction plus (Red Struct), we have $P \rightarrow P'$ as desired.

The other cases follow similarly.

Left-to-right direction: if $P \rightarrow Q$ then $P \xrightarrow{\tau} \equiv Q$. The proof is by induction on the derivation of $P \rightarrow Q$ and a case analysis on the last rule applied. We show only a number of cases, the other follow similarly.

(RED IN) $P = a[\text{in } b.P_1 \mid P_2] \mid b[P_3]$, $Q = b[P_3 \mid a[P_1 \mid P_2]]$. Then we can easily calculate that $P \succ (\nu)(a[\text{in } b.P_1 \mid P_2] \mid b[P_3])$, $(\text{in } b.P_1 \mid P_2) \xrightarrow{\text{in } b} (P_1 \mid P_2)$ and $b[P_3] \succ (\nu)(b[P_3])\mathbf{0}$. Now by (TRANS IN) we have $P \xrightarrow{\tau} P' = b[a[P_1 \mid P_2] \mid P_3] \mid \mathbf{0}$, and we conclude since $P' \equiv Q$.

(RED RES) $P = (\nu n)P_1$, $Q = (\nu n)P'_1$ and $P_1 \rightarrow P'_1$. By induction we have that $P_1 \xrightarrow{\tau} R \equiv P'_1$, thus by Lemma 4.6.4(2) we conclude $P \xrightarrow{\tau} \equiv (\nu n)P'_1$.

(RED STRUCT) $P \equiv P'$, $P' \rightarrow Q'$ and $Q' \equiv Q$. By induction we have $P' \xrightarrow{\tau} R \equiv Q' \equiv Q$. Using the transitivity of \equiv and Lemma 4.6.4(1) we conclude $P \xrightarrow{\tau} \equiv Q$.

4.6.2 Proof of Activity Theorem 4.4.9

We begin by extending the structural congruence, hardening, and reduction relations to multihole evaluation contexts as follows:

▷ Let $E \equiv E'$ hold if and only if $E\{P_1, \dots, P_k\} \equiv E'\{P_1, \dots, P_k\}$ for all $P_i, i = 1, \dots, k$.

- ▷ Let $E \succ (\nu \tilde{p}) \langle n[E'] \rangle E''$ hold if and only if

$$E\{P_1, \dots, P_k\}_k \succ (\nu \tilde{p}) \langle n[E'\{P_1, \dots, P_l\}_l] \rangle E''\{P_{l+1}, \dots, P_k\}_{k-l}$$
 for all P_i such that $\{\tilde{p}\} \cap fn(P_i) = \emptyset, i = 1, \dots, k$.
- ▷ Let $E \succ (\nu \tilde{p}) \langle E' \rangle E''$ hold if and only if

$$E\{P_1, \dots, P_k\}_k \succ (\nu \tilde{p}) \langle E'\{P_1, \dots, P_l\}_l \rangle E''\{P_{l+1}, \dots, P_k\}_{k-l}$$
 for all P_i such that $\{\tilde{p}\} \cap fn(P_i) = \emptyset, i = 1, \dots, k$.
- ▷ Let $E \rightarrow E'$ hold if and only if, for all $P_i, i = 1, \dots, k, E\{P_1, \dots, P_k\}_k \rightarrow E'\{P_1, \dots, P_k\}_k$.

Lemma 4.6.5. *If $E\{P_1, \dots, P_k\} \succ (\nu \tilde{p}) \langle Q_1 \rangle Q_2$ then either:*

1. $E \succ (\nu \tilde{p}) \langle n[E'] \rangle Q_2$ and $Q_1 = n[E'\{P_1, \dots, P_k\}]$, or
2. $E \succ (\nu \tilde{p}) \langle Q_1 \rangle E'$ and $Q_2 = E'\{P_1, \dots, P_k\}$, or
3. $E \succ (\nu \tilde{p}) \langle n[E'] \rangle E''$ and $Q_1 = n[E'\{P_1, \dots, P_l\}_l]$, $Q_2 = E''\{P_{l+1}, \dots, P_k\}_{k-l}$, or
4. $\exists i$ and a $(k-1)$ -holes evaluation context E'_{k-1} s.t. $P_i \succ (\nu \tilde{p}) \langle Q_1 \rangle P'$, $E \equiv - \mid E'_{k-1}$, $Q_2 \equiv P' \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}$, and $\{\tilde{p}\} \cap fn(E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}) = \emptyset$.

Proof. By induction on the derivation of $E\{P\} \succ (\nu \tilde{p}) \langle P_1 \rangle P_2$. □

Lemma 4.6.6. *If $E\{P_1, \dots, P_k\} \succ (\nu \tilde{p}) \langle n[Q_1] \rangle Q_2$ then either:*

1. $E \equiv (\nu \tilde{p})(n[E'] \mid Q_2)$ and $Q_1 = E'\{P_1, \dots, P_k\}$, or
2. $E \equiv (\nu \tilde{p})(n[Q_1] \mid E')$ and $Q_2 = E'\{P_1, \dots, P_k\}$, or
3. $E \equiv (\nu \tilde{p})(n[E'] \mid E'')$ and $Q_1 = E'\{P_1, \dots, P_l\}_l$, $Q_2 = E''\{P_{l+1}, \dots, P_k\}_{k-l}$, or
4. $\exists i$ and a $(k-1)$ -holes evaluation context E'_{k-1} such that $P_i \succ (\nu \tilde{p}) \langle n[Q_1] \rangle P'$, $E \equiv - \mid E'_{k-1}$, $Q_2 \equiv P' \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}$, and $\{\tilde{p}\} \cap fn(E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}) = \emptyset$.

Proof. Corollary of Lemma 4.6.5. □

Lemma 4.6.7. *If $E\{P_1, \dots, P_k\} \xrightarrow{M} R$ for $M \in \{\text{in } n, \text{out } n\}$ then either:*

1. $E \equiv (\nu \tilde{r})(M.R' \mid E')$, $R \equiv (\nu \tilde{r})(R' \mid E'\{P_1, \dots, P_k\})$,
 $\{\tilde{r}\} \cap (\{n\} \cup fn(P_i)) = \emptyset, i = 1, \dots, k$, or
2. $\exists i$ and a $(k-1)$ -holes evaluation context E'_{k-1} s.t. $P_i \xrightarrow{M} P'_i$, $E \equiv - \mid E'_{k-1}$, and $R \equiv P'_i \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}$.

Proof. Immediate. \square

Lemma 4.6.8. *If $E\{P_1, \dots, P_k\} \xrightarrow{(M)} R$ for then either:*

1. $E \equiv (\nu \tilde{r})(\langle x \rangle R' \mid E')$, $R \equiv (\nu \tilde{r})(R'\{x := M\} \mid E'\{P_1, \dots, P_k\})$, and $\{\tilde{r}\} \cap (fn(M) \cup fn(P_i)) = \emptyset$, or
2. $\exists i$ and a $(k-1)$ -holes evaluation context E'_{k-1} s.t. $P_i \xrightarrow{(M)} P'_i$, $E \equiv - \mid E'_{k-1}$, and $R \equiv P'_i \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}$.

Proof. Immediate. \square

In order to prove Theorem 4.4.9, we adopt the following notation.

Definition 4.6.9 (Interaction between evaluation contexts and processes). Let E be a single hole evaluation context. Let $E \bullet P \rightsquigarrow R$ if and only if there are E' , single hole evaluation context, and \tilde{r} with $\{\tilde{r}\} \cap fn(P) = \emptyset$, and one of the following holds:

- (**Inter In**) $E \equiv (\nu \tilde{r})E'\{m[- \mid R'] \mid n[R'']\}$, $P \xrightarrow{\text{in } n} P'$ and $R \equiv (\nu \tilde{r})E'\{n[m[P' \mid R'] \mid R'']\}$
- (**Inter Out**) $E \equiv (\nu \tilde{r})E'\{n[m[- \mid R'] \mid R'']\}$, $P \xrightarrow{\text{out } n} P'$ and $R \equiv (\nu \tilde{r})E'\{n[R''] \mid m[P' \mid R']\}$
- (**Inter Input**) $E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle R'\}$, $P \xrightarrow{(M)} P'$ and $R \equiv (\nu \tilde{r})E'\{P' \mid R'\}$
- (**Inter Output**) $E \equiv (\nu \tilde{r})E'\{- \mid \langle x \rangle R'\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle Q \rangle P'$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid Q \mid R'\{x := M\})\}$ with $\tilde{p} \cap fn(R') = \emptyset$.
- (**Inter Input n**) $E \equiv (\nu \tilde{r})E'\{- \mid n[\langle M \rangle Q' \mid Q'']\}$, $P \succ (\nu \tilde{p})\langle \langle x \rangle^n P' \rangle P''$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P'\{x := M\} \mid P'' \mid n[Q' \mid Q''])\}$ with $\{\tilde{p}\} \cap fn(n[\langle M \rangle Q' \mid Q'']) = \emptyset$.
- (**Inter Output from n**) $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid \langle x \rangle^n R''\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle P' \rangle P''$, $\{\tilde{p}\} \cap (fn(R') \cup fn(\langle x \rangle^n R'')) = \emptyset$. and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P' \mid P'' \mid R'] \mid R''\{x := M\})\}$
- (**Inter Output upwd**) $E \equiv (\nu \tilde{r})E'\{- \mid n[\langle x \rangle^\dagger Q' \mid Q'']\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle P' \rangle P''$, $\tilde{p} \cap fn(n[Q' \mid Q'']) = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid P'' \mid n[Q'\{x := M\} \mid Q''])\}$
- (**Inter Input \uparrow from n**) $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid \langle M \rangle R''\}$, $P \xrightarrow{(M)^\dagger} P'$ and $R \equiv (\nu \tilde{r})E'\{n[P' \mid R'] \mid R''\}$
- (**Inter Output n**) $E \equiv (\nu \tilde{r})E'\{- \mid n[\langle x \rangle Q' \mid Q'']\}$, $P \succ (\nu \tilde{p})\langle \langle M \rangle^n P' \rangle P''$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid P'' \mid n[Q'\{x := M\} \mid Q''])\}$ with $\tilde{p} \cap fn(n[Q' \mid Q'']) = \emptyset$.

- (Inter Input from n)** $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid \langle M \rangle^n R''\}, P \xrightarrow{(M)} P'$
 and $R \equiv (\nu \tilde{r})E'\{n[P' \mid R'] \mid R''\}$
- (Inter Input upwd)** $E \equiv (\nu \tilde{r})E'\{- \mid n[\langle M \rangle^\dagger Q' \mid Q'']\}, P \xrightarrow{(M)} P'$
 and $R \equiv (\nu \tilde{r})E'\{P' \mid n[Q' \mid Q'']\}$.
- (Inter Output \uparrow from n)** $E \equiv (\nu \tilde{r})E'\{n[- \mid R'] \mid (x)R''\}, P \succ (\nu \tilde{p})\langle \langle M \rangle^\dagger P' \rangle P'',$
 $\{\tilde{p}\} \cap (fn(R') \cup fn(R'') \cup \{n\}) = \emptyset$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P' \mid P'' \mid R'] \mid R''\{x := M\})\}$
- (Inter Amb)** $P \succ (\nu \tilde{p})\langle n[Q] \rangle P'$ and one of the following:
- (1) $Q \xrightarrow{\text{in } m} Q', E \equiv (\nu \tilde{r})E'\{- \mid m[R']\}, \tilde{p} \cap fn(m[R']) = \emptyset$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(P' \mid m[n[Q'] \mid R'])\}$
 - (2) $Q \xrightarrow{\text{out } m} Q', E \equiv (\nu \tilde{r})E'\{m[- \mid R']\}, m \notin \tilde{p}$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[Q'] \mid m[R' \mid P'])\}$
 - (3) $E \equiv (\nu \tilde{r})E'\{m[R' \mid \text{in } n.R''] \mid -\}, \tilde{p} \cap fn(m[R' \mid \text{in } n.R'']) = \emptyset$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[Q \mid m[R' \mid R'']] \mid P')\}$
 - (4) $Q \succ (\nu \tilde{q})\langle \langle M \rangle Q' \rangle Q'', E \equiv (\nu \tilde{r})E'\{- \mid (x)^n R'\}, \{\tilde{p}\} \cap (\{n\} \cup fn(R')) = \emptyset,$
 $\{\tilde{q}\} \cap (\{n\} \cup fn(P' \mid R')) = \emptyset,$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p}, \tilde{q})(R'\{x := M\} \mid n[Q' \mid Q''] \mid P')\}$
 - (5) $Q \xrightarrow{(M)} Q', E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle^n R'\}, \tilde{p} \cap fn(\langle M \rangle^n R') = \emptyset.$
 and $R \equiv (\nu \tilde{r})E'\{R' \mid (\nu \tilde{p})(n[Q'] \mid P')\}$
 - (6) $Q \succ (\nu \tilde{q})\langle \langle M \rangle^\dagger Q' \rangle Q'', E \equiv (\nu \tilde{r})E'\{- \mid (x)R'\}, \{\tilde{p}\} \cap (\{n\} \cup fn(R')) = \emptyset,$
 $\{\tilde{q}\} \cap (\{n\} \cup fn(P' \mid R')) = \emptyset,$
 and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p}, \tilde{q})(R'\{x := M\} \mid n[Q' \mid Q''] \mid P')\}$
 - (7) $Q \xrightarrow{(M)^\dagger} Q', E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle R'\}, \tilde{p} \cap fn(\langle M \rangle R') = \emptyset.$
 and $R \equiv (\nu \tilde{r})E'\{R' \mid (\nu \tilde{p})(n[Q'] \mid P')\}$ □

Definition 4.6.10 (Interaction between two holes). Let E be a two holes evaluation context. Let $E \bullet (P_1, P_2) \rightsquigarrow R$ if and only if there are E' , two holes evaluation context, and \tilde{r} with $\{\tilde{r}\} \cap (fn(P_1) \cup fn(P_2)) = \emptyset$, and one of the following holds:

- (Inter In)** $E \equiv (\nu \tilde{r})E'\{m[-_1 \mid R'] \mid -_2\}, P_1 \xrightarrow{\text{in } n} P'_1, P_2 \succ (\nu \tilde{p})\langle n[P'_2] \rangle P''_2,$
 $fn(m[P_1 \mid R']) \cap \{\tilde{p}\} = \emptyset$ and $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[m[P'_1 \mid R'] \mid P'_2 \mid P''_2])\}$
- (Inter Output from n)** $E \equiv (\nu \tilde{r})E'\{n[-_1 \mid R'] \mid -_2\}, P_1 \succ (\nu \tilde{p})\langle \langle M \rangle P'_1 \rangle P''_1, P_2 \succ (\nu \tilde{q})\langle (x)^n P'_2 \rangle P''_2$
 $\{\tilde{p}\} \cap (fn(R') \cup fn(P_2)) = \emptyset, \{\tilde{q}\} \cap fn(M) = \emptyset$ and
 $R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid P'_2 \mid R'] \mid (\nu \tilde{q})(P'_2\{x := M\} \mid P''_2))\}$

$$\begin{aligned}
(\text{Inter Input } \uparrow \text{ from } n) \quad E &\equiv (\nu \tilde{r})E'\{n[-1 \mid R'] \mid -2\}, P_1 \xrightarrow{(M)^\dagger} P'_1, P_2 \succ (\nu \tilde{p})\langle\langle M \rangle P'_2\rangle P''_2, \\
&\quad fn(n[P_1 \mid R']) \cap \{\tilde{p}\} = \emptyset \text{ and } R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid R'] \mid P'_2 \mid P''_2)\} \\
(\text{Inter Input from } n) \quad E &\equiv (\nu \tilde{r})E'\{n[-1 \mid R'] \mid -2\}, P_1 \xrightarrow{(M)} P'_1, P_2 \succ (\nu \tilde{p})\langle\langle M \rangle^n P'_2\rangle P''_2, \\
&\quad fn(n[P_1 \mid R']) \cap \{\tilde{p}\} = \emptyset \text{ and } R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid R'] \mid P'_2 \mid P''_2)\} \\
(\text{Inter Output } \uparrow \text{ from } n) \quad E &\equiv (\nu \tilde{r})E'\{n[-1 \mid R'] \mid -2\}, P_1 \succ (\nu \tilde{p})\langle\langle M \rangle^\dagger P'_1\rangle P''_1, P_2 \xrightarrow{(M)} P'_2 \\
&\quad \{\tilde{p}\} \cap (fn(R') \cup fn(P_2) \cup \{n\}) = \emptyset \text{ and } R \equiv (\nu \tilde{r})E'\{(\nu \tilde{p})(n[P'_1 \mid P''_1 \mid R'] \mid P'_2)\} \quad \square
\end{aligned}$$

The following lemmas about the $E \bullet P \rightsquigarrow R$ notation may easily be checked.

Lemma 4.6.11.

1. If $E \bullet P \rightsquigarrow R$ and $R \equiv R'$ then $E \bullet P \rightsquigarrow R'$.
2. If $E \bullet P \rightsquigarrow R$ then $E'\{E\} \bullet P \rightsquigarrow E'\{R\}$.
3. If $E \bullet P \rightsquigarrow R$ and $n \notin fn(P)$ then $(\nu n)E \bullet P \rightsquigarrow (\nu n)R$. \square

Proposition 4.6.12. If $E\{P_1, \dots, P_k\} \xrightarrow{\tau} R$ then one of the following holds:

1. $\exists i$ s.t. there is a reduction $P_i \rightarrow P'_i$ with $R \equiv E\{P_1, \dots, P'_i, \dots, P_k\}$, or
2. there is a reduction $E \rightarrow E'$ with $R \equiv E'\{P_1, \dots, P_k\}$, or
3. $\exists i, j$ and a $(k-1)$ -holes evaluation context E'_{k-1} such that $E\{P_1, \dots, P_k\}_k \equiv E'\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}_{k-1}$, and there is a reduction $P_i \mid P_j \rightarrow \overline{P}$ with $R \equiv E'\{P_1, \dots, \overline{P}, \dots, P_k\}_{k-1}$.
4. $\exists i$ such that $E\{P_1, \dots, -, \dots, P_k\}_1 \bullet P_i \rightsquigarrow R$.
5. $\exists i, j$ such that $E\{P_1, \dots, P_{i-1}, -1, P_{i+1}, \dots, P_{j-1}, -2, P_{j+1}, \dots, P_k\} \bullet (P_i, P_j) \rightsquigarrow R$.

Proof. by induction on the derivation of $E\{P_1, \dots, P_k\} \xrightarrow{\tau} R$ and a case analysis on the last rule applied.

(TRANS AMB) In this case $E\{P_1, \dots, P_k\} \succ (\nu \tilde{q})\langle n[Q_1] \rangle Q_2$, $Q_1 \xrightarrow{\tau} Q'_1$, and $R = (\nu \tilde{q})(n[Q'_1] \mid Q_2)$. From $E\{P_1, \dots, P_k\} \succ (\nu \tilde{q})\langle n[Q_1] \rangle Q_2$ it follows $\{\tilde{q}\} \cap fn(P_i) = \emptyset$, since $fn(P_i) \subseteq fn(E\{P_1, \dots, P_k\})$. By Lemma 4.6.6, $E\{P_1, \dots, P_k\} \succ (\nu \tilde{q})\langle n[Q_1] \rangle Q_2$ implies that there are four cases to consider:

1. $E \succ (\nu \tilde{q})\langle n[E'] \rangle Q_2$ and $Q_1 = E'\{P_1, \dots, P_k\}$. By induction we have that $Q_1 = E'\{P_1, \dots, P_k\} \xrightarrow{\tau} Q'_1$ implies one of the following:

- (a) $\exists i$ s.t. $P_i \rightarrow P'_i$ with $Q'_1 \equiv E'\{P_1, \dots, P'_i, \dots, P_k\}$. We have $R \equiv (\nu \tilde{q})(n[E'\{P_1, \dots, P'_i, \dots, P_k\}] \mid Q_2)$, and $E \equiv (\nu \tilde{q})(n[E'] \mid Q_2)$, so case 1. pertains.
- (b) $E' \rightarrow E''$ with $Q'_1 \equiv E''\{P_1, \dots, P_k\}$. From $E \succ (\nu \tilde{q})(n[E'] \mid Q_2)$ and $E' \rightarrow E''$ we can derive $E \rightarrow (\nu \tilde{q})(n[E''] \mid Q_2)$. We have $R \equiv (\nu \tilde{q})(n[E''\{P_1, \dots, P_k\}] \mid Q_2)$, so case 2. pertains.
- (c) $\exists i, j$ s.t. $E'\{P_1, \dots, P_k\} = E''\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}_{k-1}$, $P_i \mid P_j \rightarrow \bar{P}$ and $Q'_1 \equiv E''\{P_1, \dots, \bar{P}, \dots, P_k\}_{k-1}$. Let $\bar{E}_{k-1} = (\nu \tilde{q})(n[E''\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}_{k-1}] \mid Q_2)$, then $E\{P_1, \dots, P_k\} = \bar{E}\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}_{k-1}$ and $R \equiv \bar{E}\{P_1, \dots, \bar{P}, \dots, P_k\}_{k-1}$, hence case 3. pertains.
- (d) $\exists i$ such that $E'\{P_1, \dots, -, \dots, P_k\} \bullet P_i \rightsquigarrow Q'_1$. From $E \succ (\nu \tilde{q})(n[E'] \mid Q_2)$ we get that $E \equiv (\nu \tilde{q})(n[E'] \mid Q_2)$. Also, $R \equiv (\nu \tilde{q})(n[Q'_1] \mid Q_2)$. By Lemma 4.6.11(2) $E'\{P_1, \dots, -, \dots, P_k\} \bullet P_i \rightsquigarrow Q'_1$ implies that $n[E'\{P_1, \dots, -, \dots, P_k\}] \mid Q_2 \bullet P_i \rightsquigarrow n[Q'_1] \mid Q_2$. By Lemma 4.6.11(3), from $\{\tilde{q}\} \cap fn(P_i) = \emptyset$ we have that $(\nu \tilde{q})(n[E'\{P_1, \dots, -, \dots, P_k\}] \mid Q_2) \bullet P_i \rightsquigarrow (\nu \tilde{q})(n[Q'_1] \mid Q_2)$. By Lemma 4.6.11(1), $E\{P_1, \dots, -, \dots, P_k\} \bullet P_i \rightsquigarrow R$. Hence case 4. pertains.
- (e) Similarly to case 1.(e) above, we prove that case 5. pertains.
2. $E \succ (\nu \tilde{q})(n[Q_1] \mid E_1)$ and $Q_2 = E_1\{P_1, \dots, P_k\}$. Let $E' = (\nu \tilde{q})(n[Q'_1] \mid E_1)$. Since $E \equiv (\nu \tilde{q})(n[Q'_1] \mid E_1)$ and $Q_1 \xrightarrow{\tau} Q'_1$, we get that $E \rightarrow E'$. Moreover $R \equiv (\nu \tilde{q})(n[Q'_1] \mid E_1\{P_1, \dots, P_k\}) \equiv E'\{P_1, \dots, P_k\}$. Hence case 2. pertains.
3. $E \succ (\nu \tilde{q})(n[E_1] \mid E_2)$ with $Q_1 = E_1\{P_1, \dots, P_l\}_l$ and $Q_2 = E_2\{P_{l+1}, \dots, P_k\}_{k-l}$. The proof follows as in case 1. above, using the induct. hyp. on $Q_1 = E_1\{P_1, \dots, P_l\}_l \xrightarrow{\tau} Q'_1$.
4. $\exists i$ such that $P_i \succ (\nu \tilde{q})(n[Q_1] \mid P')$, $E \equiv - \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}$, $Q_2 \equiv P' \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}$, and $\{\tilde{q}\} \cap fn(E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}) = \emptyset$. Let $P'' = (\nu \tilde{q})(n[Q'_1] \mid P')$. From $Q_1 \xrightarrow{\tau} Q'_1$ and $P_i \equiv (\nu \tilde{q})(n[Q_1] \mid P')$, we get that $P_i \rightarrow P''$. Moreover, $R \equiv (\nu \tilde{q})(n[Q'_1] \mid P' \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}_{k-1}) \equiv E\{P_1, \dots, P_{i-1}, P'', P_{i+1}, \dots, P_k\}$. Hence case 1. pertains.
- (TRANS IN) In this case $E\{P_1, \dots, P_k\} \succ (\nu \tilde{q})(n[Q_1] \mid Q_2)$, $Q_1 \xrightarrow{\text{in } m} Q'_1$, $Q_2 \succ (\nu \tilde{r})(m[Q'_2] \mid Q''_2)$, $fn(n[Q_1]) \cap \{\tilde{r}\} = \emptyset$ and $R = (\nu \tilde{q}, \tilde{r})(m[n[Q'_1] \mid Q'_2] \mid Q''_2)$. We may assume that $fn(P_i) \cap \{\tilde{q}\} = \emptyset, i = 1, \dots, k$. By Lemma 4.6.6 there are four cases to consider:

1. $E \equiv (\nu \tilde{q})(n[E'] \mid Q_2)$ and $Q_1 = E'\{P_1, \dots, P_k\}$. From $Q_1 \xrightarrow{\text{in } m} Q'_1$ we have that $E'\{P_1, \dots, P_k\} \xrightarrow{\text{in } m} Q'_1$ and By Lemma 4.6.7 there are two subcases:
 - (a) $E' \equiv (\nu \tilde{s})(\text{in } m.R_1 \mid E'')$, $Q'_1 \equiv (\nu \tilde{s})(R_1 \mid E''\{P_1, \dots, P_k\})$, $\{\tilde{s}\} \cap (\{m\} \cup \text{fn}(P_i)) = \emptyset$. Let $\overline{E} = (\nu \tilde{q}, \tilde{r})(m[n[(\nu \tilde{s})(R_1 \mid E'')]] \mid Q'_2 \mid Q''_2)$. Then $E \rightarrow \overline{E}$ and $R \equiv \overline{E}\{P_1, \dots, P_k\}$, hence case 2. pertains.
 - (b) $\exists i$ and a $(k-1)$ -hole evaluation context E'' such that $P_i \xrightarrow{\text{in } m} P'$, $E' \equiv - \mid E''_{k-1}$, $Q'_1 \equiv P' \mid E''\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$. Then from $Q_2 \succ (\nu \tilde{r})\langle m[Q'_2] \rangle Q''_2$ and $\text{fn}(n[Q_1]) \cap \{\tilde{r}\} = \emptyset$ we have $E \equiv (\nu \tilde{q}, \tilde{r})(n[- \mid E''_{k-1}] \mid m[Q'_2] \mid Q''_2)$, $P_i \xrightarrow{\text{in } m} P'$ and $R \equiv (\nu \tilde{q}, \tilde{r})(m[n[P' \mid E''\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}]] \mid Q'_2 \mid Q''_2)$, hence case 4. (Inter In) pertains.
2. $E \equiv (\nu \tilde{q})(n[Q_1] \mid E')$ and $Q_2 = E'\{P_1, \dots, P_k\}$. From $Q_2 \succ (\nu \tilde{r})\langle m[Q'_2] \rangle Q''_2$, by Lemma 4.6.6 there are four subcases:
 - (a) $E' \equiv (\nu \tilde{r})(m[E''] \mid Q''_2)$ and $Q'_2 = E''\{P_1, \dots, P_k\}$. Then let $\overline{E} = (\nu \tilde{q}, \tilde{r})(m[n[Q'_1] \mid E''] \mid Q''_2)$, then $E \rightarrow \overline{E}$ and $R \equiv \overline{E}\{P_1, \dots, P_k\}$, hence case 2. pertains.
 - (b) $E' \equiv (\nu \tilde{r})(m[Q'_2] \mid E'')$ and $Q''_2 = E''\{P_1, \dots, P_k\}$. Then let $\overline{E} = (\nu \tilde{q}, \tilde{r})(m[n[Q'_1] \mid Q'_2] \mid E'')$, then $E \rightarrow \overline{E}$ and $R \equiv \overline{E}\{P_1, \dots, P_k\}$, hence case 2. pertains.
 - (c) $E' \equiv (\nu \tilde{r})(m[E_1] \mid E_2)$ and $Q'_2 = E_1\{P_1, \dots, P_l\}_l$, $Q''_2 = E_2\{P_{l+1}, \dots, P_k\}_{k-l}$. Let $\overline{E} = (\nu \tilde{q}, \tilde{r})(m[n[Q'_1] \mid E_1] \mid E_2)$, then $E \rightarrow \overline{E}$ and $R \equiv \overline{E}\{P_1, \dots, P_k\}$, hence case 2. pertains.
 - (d) $\exists i$ and a $(k-1)$ multihole evaluation context E'' such that $P_i \succ (\nu \tilde{r})\langle m[Q'_2] \rangle P'$, $E' \equiv - \mid E''_{k-1}$, $Q'_2 \equiv P' \mid E''\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$ and $\{\tilde{r}\} \cap \text{fn}(E''\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}) = \emptyset$. Then $E \equiv (\nu \tilde{q})(n[Q_1] \mid - \mid E'')$, $Q_1 \xrightarrow{\text{in } m} Q'_1$ and $\{\tilde{r}\} \cap \text{fn}(n[Q_1]) = \emptyset$, thus case 4. (Inter Amb)(3) pertains.
3. $E \equiv (\nu \tilde{q})(n[E_1] \mid E_2)$ and $Q_1 = E_1\{P_1, \dots, P_l\}$, $Q_2 = E_2\{P_{l+1}, \dots, P_k\}$. From $Q_1 \xrightarrow{\text{in } m} Q'_1$, by Lemma 4.6.7 we have two subcases:
 - (a) $E_1 \equiv (\nu \tilde{s})(\text{in } m.R_1 \mid E'_1)$, $Q'_1 \equiv (\nu \tilde{s})(R_1 \mid E'_1\{P_1, \dots, P_l\})$, $\tilde{s} \cap \text{fn}(P_i) = \emptyset$, $i = 1, \dots, l$. Now from $Q_2 \succ (\nu \tilde{r})\langle m[Q'_2] \rangle Q''_2$ and $Q_2 = E_2\{P_{l+1}, \dots, P_k\}$ by Lemma 4.6.6 there are four subcases:

i. $E_2 \equiv (\nu \tilde{r})(m[E'_2] \mid Q''_2)$ and $Q'_2 = E'_2\{P_{l+1}, \dots, P_k\}$. Then

$$E \equiv (\nu \tilde{q}, \tilde{r})(n[(\nu \tilde{s})(\text{in } m.R_1 \mid E'_1)] \mid m[E'_2] \mid Q''_2), \text{ and}$$

$$E \rightarrow (\nu \tilde{q}, \tilde{r})(m[n[(\nu \tilde{s})(R_1 \mid E'_1)] \mid E'_2] \mid Q''_2) = \overline{E}\{P_1, \dots, P_k\}.$$

Now $R \equiv \overline{E}\{P_1, \dots, P_k\}$, hence case 2. pertains.

ii. $E_2 \equiv (\nu \tilde{r})(m[Q'_2] \mid E'_2)$ and $Q''_2 = E'_2\{P_{l+1}, \dots, P_k\}$. This subcase is similar to the previous one, hence case 2. pertains.

iii. $E_2 \equiv (\nu \tilde{r})(m[E'_2] \mid E''_2)$ and $Q'_2 = E'_2\{P_{l+1}, \dots, P_q\}$, $Q''_2 = E'_2\{P_{q+1}, \dots, P_k\}$. This subcase is similar to the previous one, hence case 2. pertains.

iv. $\exists i, l+1 \leq i \leq k$ and a (k-1-2)-hole evaluation context E'_2 s.t. $P_i \succ (\nu \tilde{r})(m[Q'_2])P'$, $E_2 \equiv - \mid E'_2$ and $Q''_2 \equiv P' \mid E'_2\{P_{l+1}, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$. Then $E \equiv (\nu \tilde{q})(n[(\nu \tilde{s})(\text{in } m.R_1 \mid E'_1)] \mid - \mid E'_2)$, $\tilde{r} \cap n[Q_1] = \emptyset$, $R \equiv (\nu \tilde{q}, \tilde{r})(m[n[(\nu \tilde{s})(R_1 \mid E'_1)] \mid Q'_2] \mid P' \mid E'_2)$, hence case 4. (Inter Amb)(3) pertains.

(b) $\exists i, 1 \leq i \leq l$ and a (ℓ -1)-hole evaluation context E'_1 s.t. $P_i \xrightarrow{\text{in } m} P'_i$, $E_1 \equiv - \mid E'_1$ and $Q'_1 \equiv P'_i \mid E'_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_l\}$. Now from $Q_2 \succ (\nu \tilde{r})(m[Q'_2])Q''_2$ and $Q_2 = E_2\{P_{l+1}, \dots, P_k\}$ by Lemma 4.6.6 there are four subcases:

i. $E_2 \equiv (\nu \tilde{r})(m[E'_2] \mid Q''_2)$ and $Q'_2 = E'_2\{P_{l+1}, \dots, P_k\}$. Then

$$E \equiv (\nu \tilde{q}, \tilde{r})(n[- \mid E'_1] \mid m[E'_2] \mid Q''_2), \text{ and}$$

$$R \equiv (\nu \tilde{q}, \tilde{r})(m[n[P'_i \mid E'_1] \mid E'_2] \mid Q''_2), \text{ hence case 4. (Inter In) pertains.}$$

ii. $E_2 \equiv (\nu \tilde{r})(m[Q'_2] \mid E'_2)$ and $Q''_2 = E'_2\{P_{l+1}, \dots, P_k\}$. This subcase is similar to the previous one, hence case 4. (Inter In) pertains.

iii. $E_2 \equiv (\nu \tilde{r})(m[E'_2] \mid E''_2)$ and $Q'_2 = E'_2\{P_{l+1}, \dots, P_q\}$, $Q''_2 = E'_2\{P_{q+1}, \dots, P_k\}$. This subcase is similar to the previous one, hence case 4. (Inter In) pertains.

iv. $\exists j, l+1 \leq j \leq k$ and a (k-1-2)-hole evaluation context E'_2 s.t. $P_j \succ (\nu \tilde{r})(m[Q'_2])P'_j$, $E_2 \equiv - \mid E'_2$ and $Q''_2 \equiv P' \mid E'_2\{P_{l+1}, \dots, P_{j-1}, P_{j+1}, \dots, P_k\}$. Then $E\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_{j-1}, P_{j+1}, \dots, P_k\} \equiv (\nu \tilde{q}, \tilde{r})(n[-1 \mid E'_1] \mid -2 \mid E'_2)$, $\tilde{r} \cap n[Q_1] = \emptyset$, $R \equiv (\nu \tilde{q}, \tilde{r})(m[n[P'_i \mid E'_1] \mid Q'_2] \mid P'_j \mid E'_2)$, hence case 5. (Inter IN) pertains.

4. $\exists i, 1 \leq i \leq k$ and a (k-1)-hole evaluation context E' such that $P_i \succ (\nu \tilde{q})(n[Q_1])P'_i$, $E \equiv - \mid E'$, $Q_2 \equiv P'_i \mid E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$ and $\{\tilde{q}\} \cap fn(E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}) = \emptyset$. Let $R_1 = E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$. From $Q_2 \equiv P'_i \mid R_1$ and from $Q_2 \succ$

$(\nu\tilde{r})(m[Q'_2])Q''_2$, that implies $\{\tilde{r}\} \cap fn(m[Q'_2]) \neq \emptyset$, we have two subcases:

(a) there exist Q_3, Q_4 such that $Q''_2 = Q_3 \mid Q_4$ and $P'_i \equiv (\nu\tilde{r})(m[Q'_2] \mid Q_3)$, $E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\} = Q_4$, $fn(Q_4) \cap \{\tilde{r}\} = \emptyset$. Then we have $P_i \equiv (\nu\tilde{q})(n[Q_1] \mid (\nu\tilde{r})(m[Q'_2] \mid Q_3))$, $Q_1 \xrightarrow{\text{in } m} Q'_1$. Thus $P_i \rightarrow P'_i$ where $P'_i = (\nu\tilde{q}, \tilde{r})(m[n[Q'_1] \mid Q'_2] \mid Q_3)$, $E\{P_1, \dots, P_k\} \equiv P_i \mid Q_4$ and $R \equiv P'_i \mid Q_4 = E\{P_1, \dots, P'_i, \dots, P_k\}$, hence case 1. pertains.

(b) $\exists Q_3, Q_4$ such that $Q''_2 = Q_3 \mid Q_4$ and $P'_i = Q_4$, $E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\} = (\nu\tilde{r})(m[Q'_2] \mid Q_3)$, $fn(Q_4) \cap \{\tilde{r}\} = \emptyset$. From $\tilde{q} \cap fn(E'\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}) = \emptyset$, we have $\{\tilde{q}\} \cap fn(m[Q'_2]) = \emptyset$ and by Lemma 4.6.6 we have four subcases:

- i. $E' \equiv (\nu\tilde{r})(m[E_1] \mid Q_3)$ and $Q'_2 = E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$. Then $E \equiv - \mid (\nu\tilde{r})(m[E_1] \mid Q_3)$ and $R \equiv (\nu\tilde{q}, vr)(P'_i \mid m[n[Q'_1] \mid E_1] \mid Q_3)$, hence case 4. (Inter Amb) (1) pertains.
- ii. $E' \equiv (\nu\tilde{r})(m[Q'_2] \mid E_1)$ and $Q_3 = E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$. The proof follows similarly to the previous subcase, hence case 4. (Inter Amb) (1) pertains.
- iii. $E' \equiv (\nu\tilde{r})(m[E_1] \mid E_2)$. The proof follows similarly to the previous subcase, hence case 4. (Inter Amb) (1) pertains.
- iv. $\exists j 1 \leq j \leq k, j \neq i$ and a (k-2)-hole evaluation context E_1 such that $P_j \succ (\nu\tilde{r})(m[Q'_2])P'_j$, $E' \equiv - \mid E_1$ and $Q_3 \equiv P'_j \mid E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_{j-1}, P_{j+1}, \dots, P_k\}$. Let $\overline{E} = - \mid E_1$ a (k-1) evaluation context. Then $E \equiv \overline{E}\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}$ and $P_i \mid P_j \equiv (\nu\tilde{q})(n[Q_1] \mid P'_i \mid (\nu\tilde{r})(m[Q'_2] \mid P'_j))$, thus $P_i \mid P_j \rightarrow \overline{P} = (\nu\tilde{q}, \tilde{r})(P'_i \mid m[Q'_2] \mid n[Q'_1] \mid P'_j)$. Now, we have that $P'_i \mid P'_j \mid E_1\{\dots\} = Q_4 \mid Q_3 = Q''_2$, then $R \equiv \overline{P} \mid E_1\{\dots\} = \overline{E}\{P_1, \dots, \overline{P}, \dots, P_k\}$, hence case 3. pertains.

(τ I/O) In this case $E\{P_1, \dots, P_k\} \succ (\nu\tilde{q})(\langle M \rangle Q_1)Q_2$, $Q_2 \xrightarrow{(M)} Q'_2$, and $R = (\nu\tilde{q})(Q_1 \mid Q'_2)$. We may assume that $fn(P_i) \cap \{\tilde{q}\} = \emptyset, i = 1, \dots, k$. By Lemma 4.6.5 there are four cases to consider:

1. $E \succ (\nu\tilde{q})(n[E'])Q_2$ and $\langle M \rangle Q_1 = n[E'\{P_1, \dots, P_k\}]$, that is obviously a vacuous case.

2. $E \succ (\nu \tilde{q})(\langle M \rangle Q_1)E'$ and $Q_2 = E'\{P_1, \dots, P_k\}$. From $Q_2 \xrightarrow{(M)} Q'_2$, by Lemma 4.6.8 there are two subcases:

- (a) $E' \equiv (\nu \tilde{r})(\langle x \rangle R_1 \mid E'')$, $Q'_2 \equiv (\nu \tilde{r})(R_1\{x := M\} \mid E''\{P_1, \dots, P_k\})$ and $(fn(P_i) \cup fn(M)) \cap \{\tilde{r}\} = \emptyset$. Then $E \equiv (\nu \tilde{q})(\langle M \rangle Q_1 \mid (\nu \tilde{r})(\langle x \rangle R_1 \mid E''))$ and since \tilde{r} are bound we may assume $(fn(Q_1) \cap \{\tilde{r}\} = \emptyset$. Thus $E\{P_1, \dots, P_k\} \rightarrow (\nu \tilde{q})(Q_1 \mid (\nu \tilde{r})(R_1\{x := M\} \mid E''\{P_1, \dots, P_k\}))$, hence case 2. pertains.
- (b) $\exists i$ and a $(k-1)$ -hole evaluation context E_1 such that $P_i \xrightarrow{(M)} P'_i, E' \equiv - \mid E_1$ and $Q'_2 \equiv P' \mid E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$. Then $E \equiv (\nu \tilde{q})(\langle M \rangle Q_1 \mid - \mid E_1)$, and $R \equiv (\nu \tilde{q})(Q_1 \mid P'_1 \mid E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\})$, hence case 4. (Inter Input) pertains.

3. $E \succ (\nu \tilde{q})(\langle n[E_1] \rangle E_2, \langle M \rangle Q_1 = n[E_1\{P_1, \dots, P_l\}])$ and $Q_2 = E'\{P_{l+1}, \dots, P_k\}$, that is obviously a vacuous case.

4. $\exists i$ and a $(k-1)$ -hole evaluation context E_1 s.t. $P_i \succ (\nu \tilde{q})(\langle M \rangle Q_1)P'_i$, $E \equiv - \mid E_1$, $Q_2 \equiv P'_i \mid E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$ and $fn(E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}) \cap \{\tilde{q}\} = \emptyset$. From $Q_2 \equiv P'_i \mid E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$ and $Q_2 \xrightarrow{(M)} Q'_2$, we can easily see that there are two subcases. Let be $R_1 = E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}$

- (a) $P'_i \xrightarrow{(M)} P''_i$ and $Q'_2 \equiv P''_i \mid R_1$. Then $P_i \rightarrow P'''_i = (\nu \tilde{q})(Q_1 \mid P''_i)$ and $R \equiv P'''_i \mid R_1 = E\{P_1, \dots, P'''_i, \dots, P_k\}$, hence case 1. pertains.
- (b) $R_1 \xrightarrow{(M)} R'_1$ and $Q'_2 \equiv P'_i \mid R'_1$. Then by Lemma 4.6.8 we have two subcases:
 - i. $E_1 \equiv (\nu \tilde{s})(\langle x \rangle R' \mid E'_1)$, $R'_1 \equiv (\nu \tilde{s})(R'\{x := M\} \mid E'_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\})$ and $\tilde{s} \cap (fn(M) \cup fn(P_j)) = \emptyset$. Then $E \equiv (\nu \tilde{q}, \tilde{s})(- \mid \langle x \rangle R' \mid E'_1)$ and from $Q'_2 \equiv P'_i \mid R'_1$, $R = (\nu \tilde{q})(Q_1 \mid Q'_2) \equiv (\nu \tilde{q}, \tilde{s})(P'_i \mid Q_1 \mid R'\{x := M\} \mid E'_1)$, hence case 4. (Inter Output) pertains.
 - ii. $\exists j, j \neq i$ and a $(k-2)$ -hole evaluation context E_2 such that $P_j \xrightarrow{(M)} P'_j$, $E_1 \equiv - \mid E_2$ and $R'_1 \equiv P'_j \mid E_2\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_{j-1}, P_{j+1}, \dots, P_k\}$. Then $E\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_{j-1}, P_{j+1}, \dots, P_k\} \equiv -_1 \mid -_2 \mid E_2\{\dots\}$. Now $P_i \mid P_j \rightarrow \overline{P} = (\nu \tilde{q})(Q_1 \mid P'_i \mid P'_j)$ and from $fn(E_1\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_k\}) \cap \{\tilde{q}\} = \emptyset$, and $Q'_2 \equiv P'_i \mid R'_1$, we have that $R \equiv \overline{P} \mid E_2\{P_1, \dots, P_{i-1}, P_{i+1}, \dots, P_{j-1}, P_{j+1}, \dots, P_k\}$, hence case 3. pertains.

The remaining cases follow similarly. \square

Proof of Theorem 4.4.9 [Activity] $E\{P_1, \dots, P_k\} \rightarrow R$ if and only if:

(Act Proc) $\exists i$ such that $P_i \rightarrow P'_i$ with $R \equiv E\{P_1, \dots, P'_i, \dots, P_k\}$, or

(Act Ctx) $E \rightarrow E'$ with $R \equiv E'\{P_1, \dots, P_k\}$, or

(Act Holes) $\exists i, j$ and a $(k-1)$ holes evaluation context E'_{k-1} s.t. $E\{P_1, \dots, P_k\}_k \equiv E'\{P_1, \dots, (P_i \mid P_j), \dots, P_k\}_{k-1}$, and there is a reduction $P_i \mid P_j \rightarrow \bar{P}$ with $R \equiv E'\{P_1, \dots, \bar{P}, \dots, P_k\}_{k-1}$.

(Act Inter one hole) $\exists i$ such that $E\{P_1, \dots, -, \dots, P_k\}_1 \bullet P_i \rightsquigarrow R$.

(Act Inter two holes) $\exists i, j$ such that $E\{P_1, \dots, P_{i-1}, -, P_{i+1}, \dots, P_{j-1}, -, P_{j+1}, \dots, P_k\} \bullet (P_i, P_j) \rightsquigarrow R$.

Right-to-left direction: it is a routine calculation.

Left-to-right direction: assume $E\{P_1, \dots, P_k\} \rightarrow R$, then by Theorem 4.4.3, there is Q with $E\{P_1, \dots, P_k\} \xrightarrow{\tau} Q$ and $Q \equiv R$. Proof follows by Proposition 4.6.12.

4.6.3 Proof of Context Theorem 4.4.4

In order to prove the Context Theorem, we introduce the following definition:

Definition 4.6.13 ($P \sim Q$). Assume $\Delta \vdash P : T, \Delta \vdash Q : T$. Let be $\Delta \triangleright P \sim Q$ if and only if for all Δ -compatible substitutions ϑ s.t. $\text{Dom}(\vartheta) = \text{fv}(P) \cup \text{fv}(Q)$, for all closed $(\Gamma/\Delta|_{\vartheta}, T)$ evaluation contexts E , and for all n such that $\Lambda(\Gamma(n)) \in \mathbf{L}$ one has $E\{P\vartheta\} \Downarrow_n \iff E\{Q\vartheta\} \Downarrow_n$. \square

The proof of Theorem 4.4.4 is then reduced to proving $\Delta \triangleright P \cong_L Q$ if and only if $\Delta \triangleright P \sim Q$. Before proving that, we give a number of useful lemmas.

Lemma 4.6.14. Assume $\Delta \vdash P : T$ and $\Delta \vdash P' : T$. If $\Delta \triangleright P \sim P'$, then $\Delta \triangleright \mathbf{C}(P) \sim \mathbf{C}(P')$, that is, we prove that \sim is a congruence.

Proof. If comes as in [CG99a]; we give here only a subcase, distinctive of our calculus. We prove that if $\Delta \triangleright P \sim P'$ then $\Delta \triangleright (x)^\eta P \sim (x)^\eta P'$. Consider any Δ -compatible substitution ϑ with $\text{Dom}(\vartheta) = (\text{fv}(P) \cup \text{fv}(P')) - \{x\}$. From $\Delta \triangleright P \sim P'$ it follows that $\Delta \triangleright P\vartheta \sim P'\vartheta$

and that $fv(P\vartheta) \cup fv(P'\vartheta) \subseteq \{x\}$. Consider any closed $(\Gamma/\Delta|_{\vartheta}, T)$ evaluation context E and any name n such that $\Lambda(\Gamma(n)) \in L$, we just need to prove $E\{(x)^\eta P\vartheta\} \Downarrow_n$ if and only if $E\{(x)^\eta P'\vartheta\} \Downarrow_n$. More specifically, we prove that, given any P, P' such that $\Delta \triangleright P \sim P'$ and $fv(P) \cup fv(P') \subseteq \{x\}$, if $E\{(x)^\eta P\} \Downarrow_n$, then $E\{(x)^\eta P'\} \Downarrow_n$. We show it by induction on the length of the derivation $E\{(x)^\eta P\} \Downarrow_n$.

▷ $E\{(x)^\eta P\} \Downarrow_n$, then by Proposition 4.4.6 there are two cases: (i) $(x)^\eta P \Downarrow_n$, that is impossible, thus it is a vacuous case, or (ii) $E\{Q\} \Downarrow_n$ for all Q , thus also $E\{(x)^\eta P'\} \Downarrow_n$ as desired.

▷ $E\{(x)^\eta P\} \rightarrow R$ and $R \Downarrow_n$. By Activity Theorem (instantiated for one-hole evaluation contexts), one of the following cases must hold:

(Act Proc) that is $(x)^\eta P \rightarrow P_1$ and $R \equiv E\{P_1\}$, that is impossible, thus it is a vacuous case.

(Act Ctx) That is $E \rightarrow E'$ and $R \equiv E'\{(x)^\eta P\}$. Then by induction, we have $E'\{(x)^\eta P'\} \Downarrow_n$, thus, from $E \rightarrow E'$, we have $E\{(x)^\eta P'\} \Downarrow_n$ as desired.

(Act Inter one hole) That is $E \bullet (x)^\eta P \rightsquigarrow R$. By analyzing the interaction rules, $E \bullet (x)^\eta P \rightsquigarrow R$ can only be derived using one of the following cases: (Inter Input), (Inter Input n), (Inter Input \uparrow from n), (Inter Input from n), or (Inter Input upwd). We only give the first case, the other follow similarly. In case (Inter Input) we have $E \bullet (x)^\eta P \rightsquigarrow R$ with $E \equiv (\nu \tilde{r})E'\{- \mid \langle M \rangle R'\}$, $(x)^\eta P \xrightarrow{(M)} P\{x := M\}$ and $R \equiv (\nu \tilde{r})E'\{P\{x := M\} \mid R'\}$, $\{\tilde{r}\} \cap fn(P) = \emptyset$. From $(\nu \tilde{r})E'\{P\{x := M\} \mid R'\} \Downarrow_n$ and $\Delta \triangleright P \sim P'$ we have $R_1 = (\nu \tilde{r})E'\{P'\{x := M\} \mid R'\} \Downarrow_n$. Now, $E\{(x)^\eta P'\} \equiv (\nu \tilde{r})E'\{(x)^\eta P' \mid R'\} \rightarrow R_1$, thus $E\{(x)^\eta P'\} \Downarrow_n$ as desired

□

Lemma 4.6.15. *If $\Delta \triangleright P \sim Q$ then $\Delta \triangleright P \cong_L Q$.*

Proof. We must show that, for all $(\Gamma/\Delta, T)$ -context $\mathbf{C}()$ with $\mathbf{C}(P)$ and $\mathbf{C}(Q)$ closed, for all name n such that $\Lambda(\Gamma(n)) \in L$, we have $\mathbf{C}(P) \Downarrow_n \iff \mathbf{C}(Q) \Downarrow_n$. By Lemma 4.6.14, $\Delta \triangleright P \sim Q$ implies $\Delta \triangleright \mathbf{C}(P) \sim \mathbf{C}(Q)$. Therefore $\mathbf{C}(P) \Downarrow_n \iff \mathbf{C}(Q) \Downarrow_n$ follows from the definition of $\Delta \triangleright \mathbf{C}(P) \sim \mathbf{C}(Q)$, given that $\mathbf{C}(P)$ and $\mathbf{C}(Q)$ are closed. □

Lemma 4.6.16. *If $\Delta \triangleright P \cong_L Q$, then $\Delta \triangleright P\{x := M\} \cong_L Q\{x := M\}$.*

Proof. We start showing that $\Delta \triangleright (\nu m)(m[\langle M \rangle] \mid (x)^m P) \sim P\{x := M\}$, for any P such that $m \notin \text{fn}(P)$.

Consider any Δ -compatible substitution ϑ with $\text{Dom}(\vartheta) = (\text{fv}(P) \cup \text{fv}(M)) - \{x\}$. Consider any closed $(\Gamma/\Delta|_{\vartheta}, T)$ evaluation context E and any name n such that $\Lambda(\Gamma(n)) \in L$, we prove $E\{(\nu m)(m[\langle M \rangle] \mid (x)^m P)\} \Downarrow_n \iff E\{P\{x := M\}\} \Downarrow_n$. The right-to-left direction follows by an easy calculation. We prove the opposite direction by induction on the length of the derivation $E\{(\nu m)(m[\langle M \rangle] \mid (x)^m P)\} \Downarrow_n$.

- $\triangleright E\{(\nu m)(m[\langle M \rangle] \mid (x)^m P)\} \Downarrow_n$ is impossible, then it is a vacuous case.
- $\triangleright E\{(\nu m)(m[\langle M \rangle] \mid (x)^m P)\} \rightarrow R_1, R_1 \Downarrow_n$. By Activity Theorem (instantiated for one-hole evaluation contexts), one of the following cases must hold:
 - (Act Proc)** that is $R_1 \equiv E\{(\nu m)(m[\mathbf{0}]) \mid P\{x := M\}\}$. Now, since the process $(\nu m)(m[\mathbf{0}])$ cannot participate in any transition, we have that for any name ℓ , $E\{(\nu m)(m[\mathbf{0}]) \mid P\{x := M\}\} \Downarrow_\ell \iff E\{P\{x := M\}\} \Downarrow_\ell$. Thus from $R_1 \Downarrow_n$ we have $E\{P\{x := M\}\} \Downarrow_n$ as desired.
 - (Act Ctx)** That is $E \rightarrow E'$ and $R_1 \equiv E'\{(\nu m)(m[\langle M \rangle] \mid (x)^m P)\}$. By induction we have $E'\{P\{x := M\}\} \Downarrow_n$, then from $E \rightarrow E'$ we conclude $E\{P\{x := M\}\} \Downarrow_n$.
 - (Act Inter)** That is $E \bullet (\nu m)(m[\langle M \rangle] \mid (x)^m P) \rightsquigarrow R$. By analyzing the interaction rules we see that the process $(\nu m)(m[\langle M \rangle] \mid (x)^m P)$ cannot participate in any interaction with the context, thus this is a vacuous case.

Now we proceed with lemma's proof. Assume $\Delta \triangleright P \cong_L Q$, then $\Delta \triangleright (\nu m)(m[\langle M \rangle] \mid (x)^m P) \cong_L (\nu m)(m[\langle M \rangle] \mid (x)^m Q)$. Then from $\Delta \triangleright (\nu m)(m[\langle M \rangle] \mid (x)^m R) \sim R\{x := M\}$ and Lemma 4.6.15 we have $\Delta \triangleright P\{x := M\} \cong_L Q\{x := M\}$ as desired. \square

Proof of Theorem 4.4.4 [Context]

By definition of $\Delta \triangleright P \sim Q$, the just have to show that $\Delta \triangleright P \cong_L Q$ if and only if $\Delta \triangleright P \sim Q$. Right-to-left direction follows from Lemma 4.6.15. For the opposite direction, assume $\Delta \triangleright P \cong_L Q$ and consider any Δ -compatible substitution ϑ with $\text{Dom}(\vartheta) = \text{fv}(P) \cup \text{fv}(Q)$, any closed $(\Gamma/\Delta|_{\vartheta}, T)$ evaluation contexts E , and any name n such that $\Lambda(\Gamma(n)) \in L$. By Lemma 4.6.16 $\Delta \triangleright P \cong_L Q$ implies $\Delta \triangleright P\vartheta \cong_L Q\vartheta$. Since \cong_L is a congruence, we have that

$\Delta \triangleright E\{P\vartheta\} \cong_L E\{Q\vartheta\}$. By definition of $\Delta \triangleright E\{P\vartheta\} \cong_L E\{Q\vartheta\}$ and the fact that $E\{P\vartheta\}$ and $E\{Q\vartheta\}$ are closed, we have that $E\{P\vartheta\} \Downarrow_n \iff E\{Q\vartheta\} \Downarrow_n$, hence $\Delta \triangleright P \sim Q$. \square

Chapter 5

Controlling Interferences in Boxed Ambients

Boxed Ambients expressiveness is achieved at the price of communication interferences on message reception whose resolution requires synchronization of activities at multiple, distributed locations. In this chapter we study a variant of BA aimed at controlling communication interferences as well as mobility ones. The new calculus draws inspiration from Safe Ambients (SA [LS00]) (with passwords [MH02]) and modifies the communication mechanism of BA. Expressiveness is maintained through a new form of co-capability that at the same time registers incoming agents with the receiver ambient and performs access control. We prove that new calculus has a rich semantics theory, including a sound labelled transition system characterization, and an expressive, yet simple type system. Finally, we characterize its expressiveness with respect to both BA and SA through a set of examples.

5.1 Introduction

In the calculus of Mobile Ambients, ambient's migration, triggered by the processes they enclose, models mobility of entire domains and active computational environments. This model captures several aspects of current real-world mobility of agents within distributed systems, but poses some new hard problems. Paper [LS00] pointed out a set of so-called grave interferences, i.e. situations where the inherent nondeterminism of movement becomes critical. For instance, in

$$k[n[\text{in } m.P \mid \text{out } k.R] \mid m[Q]]$$

the ambient n has two possibilities: either it enters m , or it exits k ; moreover, performing an action results in disallowing the other. Then, although n 's next movement is beyond

n 's control, the difference that such a choice brings about is so big that it is difficult to see how such a situation could have been purposely programmed. Levi and Sangiorgi's proposal of Safe Ambients (SA) [LS00] counters it by using 'co-actions' to grant ambients a form of control over other ambients' access. A process willing to be entered will manifest that explicitly, as e.g. in

$$n[\text{in } m.P \mid Q] \mid m[\overline{\text{in}} m.R \mid S] \rightarrow m[n[P \mid Q] \mid R \mid S],$$

and similarly for **out** and **open**. Building on such infrastructure, a type-system enforced the notion of *single-threadness* ensuring that at any time ambients are willing to engage in at most one activity across boundaries.

Recently, Merro and Hennessy [MH02] studied a richer version of Safe Ambients, called SAP, where incoming ambients must be able to present a suitable password in order to cross ambients' boundaries. Paper [MH02] develops a tractable semantic theory for SAP in the form of a labelled transition system (LTS) based characterization of its (reduction) barbed congruence. We will use some of these ideas in the present chapter to develop the algebraic theory of (a richer version of) Boxed Ambients.

In BA, besides mobility interferences, we have to deal also with communication interferences. Consider BA's communications across ambient boundaries:

$$\begin{aligned} (x)^n P \mid n[\langle M \rangle Q \mid R] &\rightarrow P\{x := M\} \mid n[Q \mid R] \\ \langle M \rangle P \mid n[(x)^\dagger Q \mid R] &\rightarrow P \mid n[Q\{x := M\} \mid R] \end{aligned}$$

Such reductions correspond to precise design choices we discussed in Chapter 2, but have the drawback of introducing a great amount of non-local nondeterminism and communication interference. This is exemplified perfectly by the term below, where a single message issued in n unleashes a nondeterministic race among three potential receivers located in three different ambients, possibly belonging to three different administrative domains.

$$m[(x)^n P \mid n[\langle M \rangle \mid (x)Q \mid k[(x)^\dagger R]]]$$

This raises difficulties for a distributed implementation of BA, as there is a hidden, non trivial distributed consensus problem to address at each communication. Another instructive example is $(\nu a)(a[\langle a \rangle^\dagger P] \mid Q)$, where the process inside a subverts the clear intention to have a private to Q . These forms of interference are as grave as those that led to the definition of SA, as they too should be regarded as programming errors.

In this chapter we propose a variant of BA aimed at addressing such issue and at providing a fresh foundation for the ideas behind BA. The resulting calculus, NBA, takes inspiration from SA(P), and is based on the idea that each ambient comes equipped with *two* mutually non-interfering channels, respectively for local and upward communications. New rules for hierarchical communication are the following, where no communication interference is possible:

$$\begin{aligned} (x)^n P \mid n[\langle M \rangle^\uparrow Q \mid R] &\rightarrow P\{x := M\} \mid n[Q \mid R] \\ \langle M \rangle^n P \mid n[(x)^\uparrow Q \mid R] &\rightarrow P \mid n[Q\{x := M\} \mid R] \end{aligned}$$

Observe that the upward channel can be thought of as a gateway between parent and child, located at the child's and traveling with it, and poses no particular implementation challenges.

From the theoretical viewpoint, a first consequence of the elimination of unwanted interferences is a set of good, expected algebraic laws for NBA, as illustrated in §5.3. Also, the type system for BA results considerably simplified. In particular, the types of ambients and capabilities need only record upward exchanges, while processes are characterized by their local and hierarchical exchanges. The details will be discussed in §5.4.

Observe however that limiting ourselves to banning communication interferences as above would result in a poorly expressive calculus (although some of its good properties have been underlined in [CBC02]). For instance, in $n[P]$ there would be no way for P to communicate with its subambients, unless their names were statically known. In order to regain expressive power we need to reinstate a mechanism for an ambient to learn dynamically the names of incoming ambients. Essentially, our idea is to introduce co-actions of the form $\overline{\text{in}}(x)$ that have the effect of binding such names to the variable x .

Observe that a purely binding mechanism such as this would not in itself be able to control access, but only to register it. Similarly to SA, it expresses a general willingness to accept incoming ambients; in addition to that, the receiving ambient learns the incoming ambient's name. It can thus be thought as (an abstraction of) an access protocol as it would actually take place in real computational domains, where newly arrived agents would have to register themselves in order to be granted access to local resources. In order to provide ambients with a finer mechanism of access control, we add a second component to our (co-)capabilities and write rules as the one below.

$$a[\text{in}\langle b, k \rangle.P_1 \mid P_2] \mid b[\overline{\text{in}}(x, k).Q_1 \mid Q_2] \rightarrow b[a[P_1 \mid P_2] \mid Q_1\{x := a\} \mid Q_2]$$

In practical terms, this enhances our access protocol with a form of controlling the credentials of incoming processes (k in the rule above), as a preliminary step to the registration protocol. An example of the practical relevance and naturality of this mechanism, is the negotiation of credential that takes place when connecting to a wireless LAN or to a LAN using DHCP or to a ISP using PPP.

Remarkably, our admission mechanism resembles quite closely the notion of passwords as developed in [MH02]. As a consequence, we benefit from results similar to those in [MH02]. In particular, we present a labelled transition semantics for NBA that yields a bisimulation congruence sound with respect to (reduction) barbed congruence, and we use it to prove a number of laws. Passwords also have a relevant role in the type system, where their types keep track of the type of (upward exchanges of) incoming ambients, so contributing effectively to a clean and easy type system.

We will see that, besides having practical, implementation-oriented features and enjoying good theoretical properties, such as a rich and tractable algebraic theory and a simple type system, at the same time NBA remains expressive enough. In particular, by means of examples and encodings in §5.5 we show that the expressive power we loose with respect to BA is essentially that directly related to communication interferences.

5.2 The NBA Calculus

The syntax of processes is given by the following productions; it is similar to that of BA (cf. Section 2.2), except that we have replicated prefixing rather than full replication, and, as in [MH02], each of the original capabilities has a co-capability with an extra argument which may be looked upon as a *password*.

<i>Expressions</i>	M, N	$::=$	m, n, \dots	names
			$ $	
			x, y, \dots	variables
			$ $	
			$\text{in}\langle M, N \rangle$	enter M with password N
			$ $	
			$\text{out}\langle M, N \rangle$	exit M with password N
			$ $	
			$M.M$	path

<i>Locations</i>	η	$::=$	M	names and variables
			$ $	\uparrow enclosing ambient
			$ $	\star local
<i>Prefixes</i>	π	$::=$	M	path
			$ $	$(x_1, \dots, x_k)^\eta$ input
			$ $	$\langle M_1, \dots, M_k \rangle^\eta$ output
			$ $	$\overline{\text{in}}(x, M)$ allow enter with password M
			$ $	$\overline{\text{out}}(x, M)$ allow exit with pwd M
<i>Processes</i>	P	$::=$	$\mathbf{0}$	nil process
			$ $	$P \mid P$ composition
			$ $	$(\nu n)P$ restriction
			$ $	$M[P]$ ambient
			$ $	$\pi.P$ prefixing
			$ $	$!\pi.P$ replication

Meaningless terms such as $\text{in}\langle n, p \rangle[P]$ will be ruled out by the type system in §5.4. We use a number of notational conventions. We reserve m, n, p, q to range over ambient names, x, y, z over variables, and a, b, c over both. We write $\langle \tilde{M} \rangle^\eta, (\tilde{x})^\eta, (\nu \tilde{n})$ for $\langle M_1, \dots, M_k \rangle^\eta, (x_1, \dots, x_k)^\eta, (\nu n_1, \dots, n_k)$. We omit trailing dead processes, writing M for $M.\mathbf{0}$, $\langle \tilde{M} \rangle$ for $\langle \tilde{M} \rangle \mathbf{0}$, and $n[]$ for $n[\mathbf{0}]$. The operators $(\nu n)P$, $\overline{\text{in}}(x, M).P$, $\overline{\text{out}}(x, M).P$, and $(\tilde{x})^\eta P$ act as binders for names n , x , and \tilde{x} , respectively. The sets of free names and free variables of P , $fn(P)$ and $fv(P)$, are defined accordingly. Finally, we assume that processes are closed terms, that is any P such that $fv(P) = \emptyset$.

5.2.1 Reduction and Behavioural Semantics

The dynamics of the calculus is given in the form of a reduction relation and a structural congruence. Structural congruence is used to bring the participants of a potential interaction to contiguous positions. Its formal definition is similar to that in §2.2.

$$(\text{Red Struct}) \quad P \equiv Q, Q \rightarrow R, R \equiv S \Rightarrow P \rightarrow S$$

The reduction relation \rightarrow is defined as the least relation on processes closed under parallel composition, restriction, and ambient operator which satisfies the following rules.

Mobility

$$\begin{aligned}
 (\text{enter}) \quad & n[\text{in}\langle m, k \rangle.P_1 \mid P_2] \mid m[\overline{\text{in}}(x, k).Q_1 \mid Q_2] \rightarrow m[n[P_1 \mid P_2] \mid Q_1\{x := n\} \mid Q_2] \\
 (\text{exit}) \quad & n[m[\text{out}\langle n, k \rangle.P_1 \mid P_2] \mid Q] \mid \overline{\text{out}}(x, k).R \rightarrow m[P_1 \mid P_2] \mid n[Q] \mid R\{x := m\}
 \end{aligned}$$

Communication

$$\begin{aligned}
 (\text{local}) \quad & (\tilde{x})P \mid \langle \tilde{M} \rangle Q \rightarrow P\{\tilde{x} := \tilde{M}\} \mid Q \\
 (\text{input } n) \quad & (\tilde{x})^n P \mid n[\langle \tilde{M} \rangle^\dagger Q \mid R] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R] \\
 (\text{output } n) \quad & \langle \tilde{M} \rangle^n P \mid n[(\tilde{x})^\dagger Q \mid R] \rightarrow P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R]
 \end{aligned}$$

Mobility rules, inspired by [MH02], requires the two ambients involved in the move to agree on some password k : in addition the target of the move gets to know the name of the entering ambient as a result of synchronisation. Similar ideas apply to the *(exit)* rule: in particular, the co-exit capability is placed in the target ambient, as in [MH02]. Unlike [MH02], however, the co-exit does not mention the exited ambient (so, in a way, our co-exit provides for lesser control over ambient movement).

The communication rules are a simplification of those of BA. As in BA communication is *synchronous*, *polyadic*, and, most importantly, *located* and *across* boundaries. In both rules *(input n)* and *(output n)* the underlying anonymous channel is supposed to be allocated at the sub-ambient n . In all communication rules we assume that tuples have the same arity, a condition that later will be enforced by the type system.

As to behavioral equivalences, in this Chapter we focus on *reduction barbed congruence* [HY95], a standard equality based on the notions of reduction relation and observability. Even if we could have adapted to NBA the typed observational equivalence of Chapter 4, we think that reduction barbed congruence is more suitable for the new calculus. In NBA, indeed, the use of powerful co-capabilities gives a precise control of ambient movements; then using a finer equivalence that compares process behaviors step-by-step seems to be a more reasonable choice.

In ambients the observation predicate $P \downarrow_n$ is used to denote the possibility of process P of interacting with the environment via the ambient n . In MA, [CG99a], this is true

whenever $P \equiv (\nu \tilde{m})(n[P_1] | P_2)$ where $n \notin \{\tilde{m}\}$. This is because in MA no authorization is required to cross a boundary, and the presence of an ambient n at top level denotes a potential interaction between the process and the environment via n . However in the presence of co-capabilities the process $(\nu \tilde{m})(n[P_1] | P_2)$ only represents a potential interaction if P_1 can exercise an appropriate co-capability.

Definition 5.2.1 (Barbs). We write $P \downarrow_n$ iff $P \equiv (\nu \tilde{m})(n[\overline{\text{in}}(x, k).Q | R] | S)$ for $\{n, k\} \cap \{\tilde{m}\} = \emptyset$. We write $P \Downarrow_n$ if $P \Longrightarrow P'$ and $P' \downarrow_n$. \square

We can now define our behavioral equivalence.

Definition 5.2.2 (Reduction Barbed Congruence). Reduction barbed congruence, written \cong , is the largest congruence relation over arbitrary terms such that, when restricted to processes, it is reduction closed and barb preserving; that is $P \cong Q$ implies:

1. whenever $P \rightarrow P'$, there exists Q' such that $Q \Longrightarrow Q'$ and $P' \cong Q'$ (*reduction closed*);
2. for each name n , if $P \downarrow_n$ then $Q \Downarrow_n$ (*barb preserving*). \square

Notice that, similarly to Section 4.3, in our setting there is at least another significant choice of barb, reflecting the other form of interaction a boxed ambient may engage with the context, that is, via communication. In particular, we could reasonably define that P has a barb at n if and only if $P \equiv (\nu \tilde{m})(n[\langle \tilde{M} \rangle^\dagger P' | Q'] | Q'')$ for $n \notin \{\tilde{m}\}$. However, we can show that either choices of barb result in the same equivalence (cf. Theorem 5.3.7).

5.3 LTS Semantics and Algebraic Theory

In this section we show that NBA comes equipped with a robust and rich algebraic theory. More precisely, we provide a set of laws which formalize the absence of interferences at both communication and mobility level. Our behavioral equality, reduction barbed congruence, has all of the extensional properties we require of but it is very difficult to reason about; see for example the proof of the equational laws in [LS00]. However, *bisimulation* relations, because of their co-inductive nature, provide powerful proof techniques for establishing equivalences, [San92, SM92, San96]; these are based on descriptions of processes in terms

of a *labelled transition system* (LTS). We now prepare the ground to define a labelled bisimilarity which implies reduction barbed congruence.

We start defining an alternative operational semantics for NBA, based on a labelled transition system. As a matter of simplicity, we only consider monadic communication. However, dealing with polyadic NBA just requires additional details to check the arity of communicated tuples. Transitions are of the form $P \xrightarrow{\alpha} O$ where intuitively α codifies some small contexts with which P may interact. Labels α and outcomes O are defined by the following grammar:

$$\begin{aligned}
\text{Prefixes} \quad \mu &::= \text{in}\langle m, k \rangle \mid \text{out}\langle m, k \rangle \mid \overline{\text{in}}(m, k) \mid \overline{\text{out}}(m, k) \mid (M)^n \mid \langle - \rangle^n \\
\text{Labels} \quad \alpha &::= \mu \mid \tau \mid \text{enter}\langle m, k \rangle \mid m \overline{\text{enter}}(n, k) \mid \text{exit}\langle m, k \rangle \mid \\
&\quad \text{pop}\langle k \rangle \mid n \text{ get } M \mid n \text{ put } - \\
\text{Concretions} \quad K &::= (\nu \tilde{p})\langle P \rangle Q \mid (\nu \tilde{p})\langle M \rangle P \\
\text{Outcomes} \quad O &::= P \mid K
\end{aligned}$$

The outcome O may be a simple process Q , if for example α is a prefix from the language, or a concretion, of the form $(\nu \tilde{p})\langle P \rangle Q$ and $(\nu \tilde{p})\langle M \rangle Q$. Intuitively, in $(\nu \tilde{p})\langle P \rangle Q$, the process P represents a moving ambient, while in $(\nu \tilde{p})\langle M \rangle Q$, M represents some information transmitted to the environment. In both cases the process Q represents the remaining system which is not affected by the move (resp. the output), and \tilde{p} is the set of private names shared by P (resp. M) and Q .

Although our bisimilarity will consider only actions from process to process, the actions having concretions as derivatives will help us to formally define the τ -transitions of the system. More precisely, concretions represent partial derivatives which need a contribution from the environment to be complete (such contribution is modeled via corresponding higher-order transitions we will show later).

Rules for the labelled transitions are shown in the following, where we use the convention that if O is the concretion $(\nu \tilde{p})\langle P \rangle Q$, then

- ▷ $(\nu r)O$ is a shorthand for $(\nu \tilde{p})\langle P \rangle (\nu r)Q$, if $r \notin \text{fn}(P)$, and for $(\nu r, \tilde{p})\langle P \rangle Q$ otherwise.
- ▷ $O|R$ is defined to be the concretion $(\nu \tilde{p})\langle P \rangle (Q|R)$, where \tilde{p} are chosen, using α -conversion if necessary, so that $\text{fn}(R) \cap \{\tilde{p}\} = \emptyset$.

Similar conventions holds for concretions $(\nu \tilde{p})\langle M \rangle Q$.

Visible Transitions

$$\begin{array}{c}
\text{(CAP)} \quad \frac{\text{cap} \in \{\text{in}, \text{out}\}}{\text{cap}\langle n, k \rangle.P \xrightarrow{\text{cap}\langle n, k \rangle} P} \quad \text{(CO-CAP)} \quad \frac{\text{cap} \in \{\text{in}, \text{out}\}}{\overline{\text{cap}}(x, k).P \xrightarrow{\overline{\text{cap}}(n, k)} P\{x := n\}} \quad \text{(PATH)} \quad \frac{M_1.(M_2.P) \xrightarrow{\alpha} P'}{(M_1.M_2).P \xrightarrow{\alpha} P'} \\
\\
\text{(INPUT)} \quad \frac{}{(x)^\eta P \xrightarrow{(M)^\eta} P\{x := M\}} \quad \text{(OUTPUT)} \quad \frac{}{\langle M \rangle^\eta P \xrightarrow{\langle - \rangle^\eta} (\nu)\langle M \rangle P} \\
\\
\text{(GET)} \quad \frac{P \xrightarrow{(M)^\dagger} P_1}{m[P] \xrightarrow{m \text{ get } M} m[P_1]} \quad \text{(PUT)} \quad \frac{P \xrightarrow{\langle - \rangle^\dagger} (\nu\tilde{p})\langle M \rangle P_1 \quad (m \notin \{\tilde{p}\})}{m[P] \xrightarrow{m \text{ put } -} (\nu\tilde{p})\langle M \rangle m[P_1]} \\
\\
\text{(AMB ENTER)} \quad \frac{P \xrightarrow{\text{in}\langle n, k \rangle} P'}{m[P] \xrightarrow{\text{enter}\langle n, k \rangle} (\nu)\langle m[P'] \rangle \mathbf{0}} \quad \text{(AMB CO-ENTER)} \quad \frac{P \xrightarrow{\overline{\text{in}}(n, k)} P'}{m[P] \xrightarrow{m \overline{\text{enter}}(n, k)} (\nu)\langle P' \rangle \mathbf{0}} \\
\\
\text{(AMB EXIT)} \quad \frac{P \xrightarrow{\text{out}\langle n, k \rangle} P'}{m[P] \xrightarrow{\text{exit}\langle n, k \rangle} (\nu)\langle m[P'] \rangle \mathbf{0}} \quad \text{(EXIT)} \quad \frac{P \xrightarrow{\text{exit}\langle n, k \rangle} (\nu\tilde{p})\langle m[P_1] \rangle P_2}{n[P] \xrightarrow{\text{pop}\langle k \rangle} (\nu\tilde{p})\langle m \rangle (m[P_1] \mid n[P_2])}
\end{array}$$

 τ transitions

$$\begin{array}{c}
\text{(\tau-ENTER)} \quad \frac{fn(P) \cap \{\tilde{q}\} = fn(Q) \cap \{\tilde{p}\} = \emptyset \quad \tilde{p}, \tilde{q} \text{ distincts} \quad P \xrightarrow{\text{enter}\langle m, k \rangle} (\nu\tilde{p})\langle n[P_1] \rangle P_2 \quad Q \xrightarrow{m \overline{\text{enter}}(n, k)} (\nu\tilde{q})\langle Q_1 \rangle Q_2}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p}, \tilde{q})\langle m[P_1] \mid n[P_2] \rangle} \quad \text{(\tau-AMB)} \quad \frac{P \xrightarrow{\tau} P'}{n[P] \xrightarrow{\tau} n[P']} \\
\\
\text{(\tau-EXIT)} \quad \frac{fn(Q) \cap \{\tilde{p}\} = \emptyset \quad P \xrightarrow{\text{pop}\langle k \rangle} (\nu\tilde{p})\langle m \rangle P' \quad Q \xrightarrow{\overline{\text{out}}(m, k)} Q'}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p})\langle P' \mid Q' \rangle} \quad \text{(\tau-PUT)} \quad \frac{fn(Q) \cap \{\tilde{p}\} = \emptyset \quad P \xrightarrow{\langle - \rangle^n} (\nu\tilde{p})\langle M \rangle P_1 \quad Q \xrightarrow{n \text{ get } M} Q_1}{P \mid Q \xrightarrow{\tau} (\nu\tilde{p})\langle P_1 \mid Q_1 \rangle}
\end{array}$$

$$\begin{array}{c}
(\tau\text{-EXCHANGE}) \quad fn(P) \cap \{\tilde{q}\} = \emptyset \\
\frac{P \xrightarrow{(M)} P_1 \quad Q \xrightarrow{\langle - \rangle} (\nu \tilde{q}) \langle M \rangle Q_1}{P \mid Q \xrightarrow{\tau} (\nu \tilde{q})(P_1 \mid Q_1)}
\end{array}
\quad
\begin{array}{c}
(\tau\text{-GET}) \quad fn(P) \cap \{\tilde{q}\} = \emptyset \\
\frac{P \xrightarrow{(M)^n} P_1 \quad Q \xrightarrow{n \text{ put } -} (\nu \tilde{q}) \langle M \rangle Q_1}{P \mid Q \xrightarrow{\tau} (\nu \tilde{q})(P_1 \mid Q_1)}
\end{array}$$

Structural Rules

$$\begin{array}{ccc}
(\text{PAR}) & (\text{RES}) & (\text{REPL}) \\
\frac{P \xrightarrow{\alpha} O}{P \mid Q \xrightarrow{\alpha} O \mid Q} & \frac{P \xrightarrow{\alpha} O \quad n \notin fn(\alpha)}{(\nu n)P \xrightarrow{\alpha} (\nu n)O} & \frac{\pi.P \xrightarrow{\alpha} O}{!\pi.P \xrightarrow{\alpha} !\pi.P \mid O}
\end{array}$$

Most of the rules above are similar to those for SAP [MH02] (and SA [LS00]). For instance, rule (AMB CO-ENTER) says that ambient $m[P]$ is willing to accept an incoming ambient n exhibiting the password k . Rule (EXIT) says how to deal with an ambient $m[P]$ which wants to move out of another ambient n . In this case we have to record the name m in a buffer. This name will be useful in the rule (τ -EXIT). The main novelties are the rules for communications across ambients, and the rules dealing with the new feature of getting the name of the incoming ambient; they all should be self-explanatory.

As expected the LTS based semantics coincides with the reduction semantics.

Lemma 5.3.1.

1. If $P \xrightarrow{\text{in}\langle m, k \rangle} P'$ then there exist \tilde{p}, P_1, P_2 with $m, k \notin \tilde{p}$, such that $P \equiv (\nu \tilde{p})(\text{in}\langle m, k \rangle.P_1 \mid P_2)$ and $P' = (\nu \tilde{p})(P_1 \mid P_2)$.
2. If $P \xrightarrow{\overline{\text{in}}\langle m, k \rangle} P'$ then there exist \tilde{p}, P_1, P_2 with $m, k \notin \tilde{p}$, such that $P \equiv (\nu \tilde{p})(\overline{\text{in}}\langle x, k \rangle.P_1 \mid P_2)$ and $P' = (\nu \tilde{p})(P_1\{x := m\} \mid P_2)$.
3. If $P \xrightarrow{\text{enter}\langle m, k \rangle} O$ then there exist \tilde{p}, n, P_1, P_2 with $m, k \notin \tilde{p}$, such that $O = (\nu \tilde{p})\langle n[P_1] \rangle P_2$, $P \equiv (\nu \tilde{p})(n[R] \mid P_2)$ and $R \xrightarrow{\text{in}\langle m, k \rangle} P_1$.
4. If $P \xrightarrow{m \overline{\text{enter}}\langle n, k \rangle} O$ then there exist \tilde{p}, P_1, P_2 with $m, n, k \notin \tilde{p}$, such that $O = (\nu \tilde{p})\langle P_1 \rangle P_2$, $P \equiv (\nu \tilde{p})(m[R] \mid P_2)$ and $R \xrightarrow{\overline{\text{in}}\langle n, k \rangle} P_1$.
5. If $P \xrightarrow{(\tilde{M})^\eta} P'$ then there exist \tilde{p}, P_1, P_2 with $(fn(M) \cup fn(\eta)) \cap \tilde{p} = \emptyset$, such that $P \equiv (\nu \tilde{p})(\langle \tilde{x} \rangle^\eta.P_1 \mid P_2)$ and $P' = (\nu \tilde{p})(P_1\{\tilde{x} := \tilde{M}\} \mid P_2)$.
6. If $P \xrightarrow{\langle - \rangle^\eta} P'$ then there exist \tilde{p}, P_1, P_2 such that $fn(\eta) \notin \tilde{p}$ $P' = (\nu \tilde{p})\langle \tilde{M} \rangle (P_1 \mid P_2)$ and $P \equiv (\nu \tilde{p})(\langle \tilde{M} \rangle^\eta.P_1 \mid P_2)$.

7. If $P \xrightarrow{m \text{ get } \tilde{M}} P'$ then there exist \tilde{p}, P_1, P_2 such that $m \notin \tilde{p}$, $fn(\tilde{M}) \cap \tilde{p} = \emptyset$
 $P' = (\nu \tilde{p})(m[P_1] \mid P_2)$, $P \equiv (\nu \tilde{p})(m[R] \mid P_2)$ and $R \xrightarrow{(\tilde{M})^\dagger} P_1$.

□

Lemma 5.3.2. If $P \equiv Q$ and $Q \xrightarrow{\alpha} Q'$, then $\exists P'$ such that $P \xrightarrow{\alpha} P'$ and $P' \equiv Q'$. □

Theorem 5.3.3 (Harmony).

1. If $P \xrightarrow{\tau} P'$ then $P \rightarrow P'$.
2. If $P \rightarrow P'$ then $P \xrightarrow{\tau} \equiv P'$.

Proof. of 1. The proof is by induction on the derivation of $P \xrightarrow{\tau} P'$ and a case analysis on the last rule applied. We give only four cases, the others follow similarly.

(τ -ENTER) In this case we have $P = P_1 \mid P_2$, where $P_1 \xrightarrow{\text{enter}\langle m, k \rangle} (\nu \tilde{p})\langle n[P'_1] \rangle P''_1$,
 $P_2 \xrightarrow{m \text{ enter}\langle n, k \rangle} (\nu \tilde{q})\langle P'_2 \rangle P''_2$, with $fn(P_1) \cap \tilde{q} = \emptyset$, $fn(P_2) \cap \tilde{p} = \emptyset$ and
 $P' = (\nu \tilde{p}, \tilde{q})(m[P'_2 \mid n[P'_1]] \mid P''_1 \mid P''_2)$. By Lemma 5.3.1 there exist $\tilde{r}, \tilde{s}, R_1, R_2, S_1, S_2$
such that $P \equiv (\nu \tilde{p})(n[(\nu \tilde{r})\text{in}\langle m, k \rangle.R_1 \mid R_2] \mid R'_1) \mid (\nu \tilde{q})(m[(\nu \tilde{s})\text{in}\langle x, k \rangle.S_1 \mid S_2] \mid P''_2)$.
From the hypothesis and the fact that \tilde{r}, \tilde{s} are bound names, we may extrude the scope
as follows: $P \equiv (\nu \tilde{p}, \tilde{q})(\nu \tilde{r}, \tilde{s})(n[\text{in}\langle m, k \rangle \dots] \mid m[\text{in}\langle x, k \rangle \dots] \mid P''_1 \mid P''_2)$.

Then $P \rightarrow (\nu \tilde{p}, \tilde{q})(m[(\nu \tilde{s})(S_1\{x := n\} \mid S_2) \mid n[(\nu \tilde{r})(R_1 \mid R_2)]] \mid P''_1 \mid P''_2)$, and we con-
clude observing that by Lemma 5.3.1 $(\nu \tilde{s})(S_1\{x := n\} \mid S_2) = P'_2$ and $(\nu \tilde{r})(R_1 \mid R_2) = P'_1$.

(τ -EXCHANGE) In this case $P = P_1 \mid P_2$ with $P_1 \xrightarrow{(\tilde{M})} P'_1$, $P_2 \xrightarrow{\langle - \rangle} (\nu \tilde{p})\langle \tilde{M} \rangle P'_2$,
 $fn(P_1) \cap \tilde{p} = \emptyset$ and $P' = (\nu \tilde{p})(P'_1 \mid P'_2)$. Now, by Lemma 5.3.1 there exist $\tilde{q}, R_1, R_2, S_1, S_2$
such that $P \equiv (\nu \tilde{q})(\langle \tilde{x} \rangle.R_1 \mid R_2) \mid (\nu \tilde{p})(\langle \tilde{M} \rangle.S_1 \mid S_2)$ From $fn(P_1) \cap \tilde{p} = \emptyset$ and the fact that \tilde{q}
are bound names, we may extrude the scope as follows: $P \equiv (\nu \tilde{p}, \tilde{q})(\langle \tilde{x} \rangle.R_1 \mid R_2 \mid \langle \tilde{M} \rangle.S_1 \mid S_2)$.
Then $P \rightarrow (\nu \tilde{p})(\langle \tilde{q} \rangle(R_1\{\tilde{x} := \tilde{M}\} \mid R_2) \mid S_1 \mid S_2)$, and we conclude observing that by
Lemma 5.3.1 $(\nu \tilde{q})(R_1\{\tilde{x} := \tilde{M}\} \mid R_2) = P'_1$ and $S_1 \mid S_2 = P'_2$.

(τ -PUT) In this case $P = P_1 \mid P_2$ with $P_1 \xrightarrow{\langle - \rangle^n} (\nu \tilde{p})\langle \tilde{M} \rangle P'_1$, $P_2 \xrightarrow{n \text{ get } \tilde{M}} P'_2$,
 $fn(P_2) \cap \tilde{p} = \emptyset$ and $P' = (\nu \tilde{p})(P'_1 \mid P'_2)$. Now, by Lemma 5.3.1 there exist $\tilde{q}, \tilde{s}, R_1, R_2, S_1, S_2, S_3$
such that $P \equiv (\nu \tilde{p})(\langle \tilde{M} \rangle^n.R_1 \mid R_2) \mid (\nu \tilde{q})(n[(\nu \tilde{s})(\tilde{x})^\dagger.S_1 \mid S_2] \mid S_3)$ From $fn(P_2) \cap \tilde{p} = \emptyset$
and the fact that \tilde{q}, \tilde{s} are bound names, we may extrude the scope as follows: $P \equiv$
 $(\nu \tilde{p}, \tilde{q}, \tilde{s})(\langle \tilde{M} \rangle^n.R_1 \mid R_2 \mid n[(\tilde{x})^\dagger.S_1 \mid S_2] \mid S_3)$. Then we have that

$P \rightarrow (\nu \tilde{p})(R_1 \mid R_2 \mid (\nu \tilde{q})(n[(\nu \tilde{s})S_1\{\tilde{x} := \tilde{M}\} \mid S_2] \mid S_3))$, and we conclude observing that by Lemma 5.3.1 $R_1 \mid R_2 = P'_1$ and $(\nu \tilde{q})(n[(\nu \tilde{s})S_1\{\tilde{x} := \tilde{M}\} \mid S_2] \mid S_3) = P'_2$.

(τ -AMB) In this case $P = n[P_1]$, $P_1 \xrightarrow{\tau} P'_1$ and $P' = n[P'_1]$. By induction we have that $P_1 \rightarrow P'_1$, thus by rule (RED AMB), it follows $n[P_1] \rightarrow n[P'_1]$ as desired.

Proof of 2. The proof is by induction on the derivation of $P \rightarrow P'$ and a case analysis on the last rule applied. We give only three cases, the others follow similarly.

(ENTER) In this case we have that $P = n[\text{in}\langle m, k \rangle.P_1 \mid P_2] \mid m[\overline{\text{in}}(x, k).P_3 \mid P_4]$ and $P' = m[P_3\{x := n\} \mid P_4 \mid n[P_1 \mid P_2]]$. Then we can easily calculate $n[\text{in}\langle m, k \rangle.P_1 \mid P_2] \xrightarrow{\text{enter}\langle m, k \rangle} (\nu)\langle n[P_1 \mid P_2] \rangle \mathbf{0}$ and $m[\overline{\text{in}}(x, k).P_3 \mid P_4] \xrightarrow{m \overline{\text{enter}}(n, k)} (\nu)\langle P_3\{x := n\} \mid P_4 \rangle \mathbf{0}$. Then by (τ -ENTER) we have $P \xrightarrow{\tau} m[P_3\{x := n\} \mid P_4 \mid n[P_1 \mid P_2]] \mid \mathbf{0} \mid \mathbf{0}$, that is $P \xrightarrow{\tau} \equiv P'$ as desired.

(RED NEW) In this case $P = (\nu n)P_1$, $P_1 \rightarrow P'_1$ and $P' = (\nu n)P'_1$. By induction we have that $P_1 \xrightarrow{\tau} \equiv P'_1$, then by (Res) $(\nu n)P_1 \xrightarrow{\tau} \equiv (\nu n)P'_1$, that is $P \xrightarrow{\tau} \equiv P'$ as desired.

(RED STRUCT) In this case $P \equiv Q$, $Q \rightarrow Q'$ and $Q' \equiv P'$. By induction we have that $Q \xrightarrow{\tau} \equiv Q'$. From $P \equiv Q$ and $Q \xrightarrow{\tau} \equiv Q'$, by Lemma 5.3.2(1) there exists a process P_1 such that $P \xrightarrow{\tau} P'_1$ and $P'_1 \equiv Q'$. Then from $P'_1 \equiv Q'$ and $Q' \equiv P'$ we obtain $P \xrightarrow{\tau} \equiv P'$ as desired. \square

We now re-examine our definition of barbed congruence \cong , in the light of the new semantics based on labelled transitions. As already mentioned, the predicate $P \downarrow_n$ detects the ability of the process P to interact with its environment via the ambient n . In traditional process calculi, like the pi-calculus, barbs are defined using the visible actions of the labelled transition system. We next study how the definition of barbed congruence is affected by inheriting the definition of barb from the LTS. In fact, we can show that our definition of barb coincides with the choice of one particular action.

Lemma 5.3.4. $P \downarrow_n$ if and only if $P \xrightarrow{n \overline{\text{enter}}(m, k)}$ for some m, k .

Proof. Directly by the definition of $P \downarrow_n$ and an inspection of the transition rules. \square

Next, we show that for all possible labels generated by the lts the resulting definitions of barbed congruence collapse, and coincide with \cong . We write $P \xrightarrow{\alpha}$ to say that $P \xrightarrow{\alpha} P'$ for some P' .

Definition 5.3.5. For $\alpha \in \text{Labels}$ we write $P \downarrow_{\alpha}$ if $P \xrightarrow{\alpha}$, and $P \Downarrow_{\alpha}$ if $P \Longrightarrow \xrightarrow{\alpha}$. Let $\gamma \in \text{Labels} \setminus \{\tau\}$, and define \cong_{γ} to be the largest congruence that *i*) is reduction closed and *ii*) preserves γ -barbs, i.e. $P \cong_{\gamma} Q$ and $P \downarrow_{\gamma}$ implies $Q \downarrow_{\gamma}$. \square

Proposition 5.3.6. Assume $P \cong_{\gamma} Q$. Then

1. $P \Longrightarrow P'$ implies $Q \Longrightarrow Q'$ for some Q' such that $P' \cong_{\gamma} Q'$
2. $P \Downarrow_{\gamma}$ if and only if $Q \Downarrow_{\gamma}$

Proof. Part (1) is proved by induction on the number of steps in $P \Longrightarrow P'$. If $P' = P$, then choose $Q' = Q$. Otherwise, assume $P \rightarrow P^* \xRightarrow{n} P'$. Since $P \cong_{\gamma} Q$, there exists Q^* such that $Q \Longrightarrow Q^*$ and $P^* \cong_{\gamma} Q^*$. Now the proof follows by the induction hypothesis.

For part (2), assume $P \cong_{\gamma} Q$, and $P \Downarrow_{\gamma}$. By definition, there exists P' such that $P \Longrightarrow P' \downarrow_{\gamma}$. Since $P \cong_{\gamma} Q$, by part (1) there exists Q' such that $Q \Longrightarrow Q'$ and $P' \cong_{\gamma} Q'$. Thus $Q \Longrightarrow Q' \downarrow_{\gamma}$. \square

Theorem 5.3.7. For all $\gamma \in \text{Labels} \setminus \{\tau\}$, $P \cong Q$ if and only if $P \cong_{\gamma} Q$.

Proof. Since the definitions of \cong and \cong_{γ} differ only in the notion of barb, it is enough to show that the two barbs imply each other.

- $\triangleright \gamma = n \text{ put } -$. Consider the implication from left to right first. Let $P \cong Q$ and $P \downarrow_{n \text{ put } -}$: we want to show that $Q \Downarrow_{n \text{ put } -}$. Consider the following context, where ℓ is fresh in P and Q :

$$\mathbf{C}() \triangleq () \mid (x)^n \ell [\overline{\text{in}}(x, k).0]$$

Given any R with ℓ fresh in R , it is easy to show that $R \downarrow_{n \text{ put } -}$ if and only if $\mathbf{C}(R) \downarrow_{\ell}$. This is enough to complete the proof, for $P \downarrow_{n \text{ put } -}$ implies $\mathbf{C}(P) \downarrow_{\ell}$, and since $P \cong Q$, one has $\mathbf{C}(Q) \downarrow_{\ell}$ which implies $Q \Downarrow_{n \text{ put } -}$.

For the reverse implication, let $P \cong_{n \text{ put } -} Q$, and $P \downarrow_n$. Consider the context defined as follows:

$$\mathbf{C}^k() \triangleq () \mid \ell [\text{in}\langle n, k \rangle.\text{out}\langle n, \ell \rangle.\langle \cdot \rangle^{\uparrow}] \mid \overline{\text{out}}(x, \ell).0$$

Given any R with ℓ fresh in R , it is easily shown that

- ▷ if $R \Downarrow_n$ then there exists k such that $\mathbf{C}^k(R) \Downarrow_{\ell \text{ put } -}$
- ▷ $\mathbf{C}^k(R) \Downarrow_{\ell \text{ put } -}$ implies $R \Downarrow_n$

Now, $P \Downarrow_n$ implies that there exists k such that $\mathbf{C}^k(P) \Downarrow_{\ell \text{ put } -}$. Thus $\mathbf{C}^k(Q) \Downarrow_{\ell \text{ put } -}$ since $P \cong_n \text{ put } Q$, and then $Q \Downarrow_n$ as desired.

- ▷ $\gamma = \text{pop}\langle k \rangle$. For the implication from left to right, choose the following context, with ℓ fresh in P and Q :

$$\mathbf{C}() \triangleq () \mid \overline{\text{out}}(x, k). \ell[\overline{\text{in}}(y, h)]$$

The proof proceeds as in the previous case as for all R with $\ell \notin \text{fn}(R)$, we have $R \Downarrow_{\text{pop}\langle k \rangle}$ if and only if $\mathbf{C}(R) \Downarrow_{\ell}$. For the reverse implication, choose the context:

$$\mathbf{C}^k() \triangleq () \mid \ell[\text{in}\langle n, k \rangle. \text{out}\langle n, h \rangle]$$

with h fresh. For each R with $h \notin \text{fn}(R)$, we have $i)$ $R \Downarrow_n$ implies that $\mathbf{C}^k(R) \Downarrow_{\text{pop}\langle h \rangle}$ for a suitable k , and $ii)$ $\mathbf{C}^k(R) \Downarrow_{\text{pop}\langle h \rangle}$ implies $R \Downarrow_n$. From this, we conclude as in the previous cases.

- ▷ $\gamma = \text{out}\langle n, k \rangle$. For the implication from left to right, choose the context

$$\mathbf{C}() \triangleq n[()] \mid \overline{\text{out}}(x, k). \ell[\overline{\text{in}}(y, h)]$$

Again, if $\ell \notin \text{fn}(R)$, one has $R \Downarrow_{\text{out}\langle n, k \rangle}$ if and only if $\mathbf{C}(R) \Downarrow_{\ell}$. For the reverse implication, choose the context:

$$\mathbf{C}^k() \triangleq () \mid \ell[\text{in}\langle n, k \rangle. \text{out}\langle n, h \rangle. \text{out}\langle \ell, h \rangle] \mid \overline{\text{out}}(x, h)$$

with h fresh, and verify that $R \Downarrow_n$ if and only if $\mathbf{C}^k(R) \Downarrow_{\text{out}\langle \ell, h \rangle}$.

- ▷ $\gamma = \text{in}\langle n, k \rangle$. For the implication from left to right, choose the context

$$\mathbf{C}() \triangleq a[()] \mid n[\overline{\text{in}}(x, k). b[\text{out}\langle n, h \rangle. \overline{\text{in}}(x, k)]] \mid \overline{\text{out}}(y, h)$$

with a, b, h fresh, and verify that $R \Downarrow_{\text{in}\langle n, k \rangle}$ if and only if $\mathbf{C}(R) \Downarrow_b$. For the reverse implication, choose

$$\mathbf{C}^k() \triangleq () \mid a[\text{in}\langle n, k \rangle. \text{out}\langle n, h \rangle] \mid \overline{\text{out}}(x, h). \text{in}\langle a, h \rangle.$$

with a, h fresh, and verify that $P \Downarrow_n$ if and only if $\mathbf{C}^k(P) \Downarrow_{\text{in}\langle a, h \rangle}$.

- ▷ The other cases are handled similarly. □

5.3.1 A characterization of barbed congruence

In order to define the labelled bisimilarity that is contained into barbed congruence, we add higher order rules to the LTS above, so that the resulting LTS only contains transitions of the form $P \xrightarrow{\lambda} P'$, that is from processes to processes. We assume that λ ranges over the new set of labels, that are those of the previous section (still represented by α), plus higher order labels of the form αP or $\alpha P Q$ where P and Q are processes. In particular, in the higher order transitions $P \xrightarrow{\alpha R} P'$, the process R represents the contribution given by the context so that the action performed by P is complete. As an example, rule (AMB CO-ENTER HO) models the situation where the environment provide an ambient $n[Q]$ moving into m .

$$\begin{array}{c}
\text{(OUTPUT HO)} \quad fv(Q) \subseteq \{x\}, \eta \neq \uparrow \\
\frac{P \xrightarrow{\langle - \rangle^\eta} (\nu \tilde{p}) \langle M \rangle P' \quad \tilde{p} \cap fn(Q) = \emptyset}{P \xrightarrow{\langle - \rangle^\eta Q} (\nu \tilde{p}) (P' \mid Q\{x := M\})}
\end{array}
\quad
\begin{array}{c}
\text{(OUTPUT}^\uparrow\text{ HO)} \quad fv(R) \subseteq \{x\} \\
\frac{P \xrightarrow{\langle - \rangle^\uparrow} (\nu \tilde{p}) \langle M \rangle P' \quad \tilde{p} \cap fn(n[Q], R) = \emptyset}{P \xrightarrow{\langle - \rangle^\uparrow n[Q] R} (\nu \tilde{p}) (n[P' \mid Q] \mid R\{x := M\})}
\end{array}$$

$$\begin{array}{c}
\text{(PUT HO)} \quad fv(Q) \subseteq \{x\} \\
\frac{P \xrightarrow{m \text{ put } -} (\nu \tilde{p}) \langle M \rangle P' \quad \tilde{p} \cap fn(Q) = \emptyset}{P \xrightarrow{m \text{ put } - Q} (\nu \tilde{p}) (P' \mid Q\{x := M\})}
\end{array}
\quad
\begin{array}{c}
\text{(EXIT HO)} \quad fv(Q) \subseteq \{x\} \\
\frac{P \xrightarrow{\text{pop} \langle k \rangle} (\nu \tilde{p}) \langle M \rangle P' \quad \tilde{p} \cap fn(Q) = \emptyset}{P \xrightarrow{\text{pop} \langle k \rangle Q} (\nu \tilde{p}) (P' \mid Q\{x := M\})}
\end{array}$$

$$\begin{array}{c}
\text{(AMB EXIT HO)} \quad fv(R) \subseteq \{x\} \\
\frac{P \xrightarrow{\text{exit} \langle n, k \rangle} (\nu \tilde{p}) \langle m[P_1] \rangle P_2 \quad \tilde{p} \cap fn(Q, R) = \emptyset}{P \xrightarrow{\text{exit} \langle n, k \rangle QR} (\nu \tilde{p}) (m[P_1] \mid n[P_2 \mid Q] \mid R\{x := m\})}
\end{array}$$

$$\begin{array}{c}
\text{(AMB ENTER HO)} \quad fv(Q) \subseteq \{x\} \\
\frac{P \xrightarrow{\text{enter} \langle n, k \rangle} (\nu \tilde{p}) \langle m[P_1] \rangle P_2 \quad \tilde{p} \cap fn(Q) = \emptyset}{P \xrightarrow{\text{enter} \langle n, k \rangle Q} (\nu \tilde{p}) (n[m[P_1] \mid Q\{x := m\}] \mid P_2)}
\end{array}$$

$$\begin{array}{c}
\text{(AMB CO-ENTER HO)} \\
\frac{P \xrightarrow{m \text{ enter} \langle n, k \rangle} (\nu \tilde{p}) \langle P_1 \rangle P_2 \quad \tilde{p} \cap fn(Q) = \emptyset}{P \xrightarrow{m \text{ enter} \langle n, k \rangle Q} (\nu \tilde{p}) (m[n[Q] \mid P_1] \mid P_2)}
\end{array}$$

In the rule (AMB EXIT HO) we can imagine the environment wrapping the process P with an ambient $n[Q]$, and receiving the name m of the exiting ambient at R . In output rules (OUTPUT HO), (OUTPUT[↑] HO), (PUT HO) the contribution provided by the environment is an input process that reads the message contained in P . In particular, in rule (OUTPUT[↑] HO) the environment wraps the upward output with an ambient $n[Q]$ and reads the message M with a process R , in parallel with $n[P' \mid Q]$. Finally, rule (AMB ENTER HO) models the situation where the environment provides an ambient $n[Q]$ that receives the incoming ambient $m[P_1]$.

We are now ready to give the labelled bisimilarity. As usual we focus on weak bisimilarities based on weak transitions. We use the following notation:

- i) $\xRightarrow{\lambda}$ denotes $\xrightarrow{\tau^*} \xrightarrow{\lambda} \xrightarrow{\tau^*}$;
- ii) $\xRightarrow{\hat{\lambda}}$ denotes $\xrightarrow{\tau^*}$ if $\lambda = \tau$ and $\xRightarrow{\lambda}$ otherwise.

Definition 5.3.8 (Bisimilarity \approx). A symmetric relation \mathcal{R} over processes is a *bisimulation* if $P \mathcal{R} Q$ and $P \xrightarrow{\lambda} P'$ implies that there exists Q' such that $Q \xRightarrow{\hat{\lambda}} Q'$ and $P' \mathcal{R} Q'$. P and Q are bisimilar, written $P \approx Q$, if $P \mathcal{R} Q$ for some bisimulation \mathcal{R} . \square

Notice that the bisimilarity above considers only actions from processes to processes, which typically involve higher-order actions. To this regard, it is worth pointing out that the structural rules of the LTS of the previous section only apply when $\lambda \in \text{Labels}$: consequently, there are no structural rules associated with higher-order transitions.

Finally, we note that bisimilarity is *not* preserved by arbitrary substitutions: if $P = l[m[\text{out}\langle z, l \rangle.\overline{\text{in}}(_, m)]]$, then $P \approx \mathbf{0}$, but $P\{z := l\} \not\approx \mathbf{0}$. Therefore we consider the full closure of \approx under substitutions.

Definition 5.3.9 (Full bisimilarity). Let P, Q be arbitrary processes, possibly containing free variables. P and Q are *full bisimilar*, $P \approx_c Q$, if $P\sigma \approx Q\sigma$ for every substitution σ . \square

Applying the proof techniques of [MH02] we can prove, first that full bisimilarity is a congruence, then that full bisimilarity is contained into barbed congruence. We start with a few simple properties of the labelled transitions.

Lemma 5.3.10.

1. If $P \xrightarrow{\text{out}\langle n,k \rangle 0R} P'$ then $n[P] \xrightarrow{\text{pop}\langle k \rangle R} P'$.
2. If $P \xrightarrow{\langle - \rangle^\dagger n[0]R} P'$ then $n[P] \xrightarrow{n \text{ put } -R} P'$.

Proof. By rule induction. □

Theorem 5.3.11. *Full bisimilarity is a congruence*

Proof. It is easy to show that \approx_c is preserved by input prefixes (these include proper input prefixes and co-capability prefixes). For instance, assuming $P \approx_c Q$, we need to show that $(x)^\eta P\sigma \approx (x)^\eta Q\sigma$ for all closing substitutions σ . By definition, one has that $((x)^\eta P)\sigma = (x)^{\eta\sigma}(P\sigma)$ (with σ capture free). The only moves from $(x)^{\eta\sigma}(P\sigma)$ are of the form $(x)^{\eta\sigma}(P\sigma) \xrightarrow{(M)} P\sigma\{x := M\}$ for an arbitrary expression (message) M . Since also $(x)^{\eta\sigma}(Q\sigma) \xrightarrow{(M)} Q\sigma\{x := M\}$, it remains to show that $P\sigma\{x := M\} \approx Q\sigma\{x := M\}$. But this follows directly from the assumption $P \approx_c Q$.

For the remaining constructs we can safely restrict to closed terms (i.e. processes) in the language, and prove that \approx is a congruence. We treat all the constructs simultaneously, as follows. Let \mathcal{S} be the least equivalence relation for processes that contains \approx and is closed by prefix, parallel composition, restriction and ambient, i.e.:

- ▷ $\approx \subseteq \mathcal{S}$
- ▷ PSQ implies $\pi.P \mathcal{S} \pi.Q$ and $!PS!Q$
- ▷ PSQ implies $(P \mid R) \mathcal{S} (Q \mid R)$ for all processes R
- ▷ PSQ implies $n[P] \mathcal{S} n[Q]$ and $(\nu n)P \mathcal{S} (\nu n)Q$.

We show that \mathcal{S} is a bisimulation up to \equiv . The theorem follows directly from this fact (for, then, \mathcal{S} is itself a bisimulation, hence $\mathcal{S} \subseteq \approx$, which implies $\mathcal{S} = \approx$). The proof is by induction on the formation of \mathcal{S} .

- ▷ PSQ because $P \approx Q$. This case follows by definition.
- ▷ $\pi.P \mathcal{S} \pi.Q$ because PSQ . There are five sub-cases to consider since the case when π is an input prefix has already been worked out above. If π is a capability, say M , the only move from $M.P$ is of the form $M.P \xrightarrow{M} P$. Then $M.Q \xrightarrow{M} Q$, and this concludes the proof because PSQ by hypothesis. There are two more subcases for output prefixes.

- ▷ $\pi.P \xrightarrow{\lambda} P'$ because $\pi = \langle M \rangle^\eta$ with $\eta \neq \uparrow$, $\lambda = \langle - \rangle^\eta R$ and P' is structurally equivalent to $P \mid R\{x := M\}$. The same move is also available to $\langle M \rangle^\eta Q$, hence one has $\langle M \rangle^\eta Q \xrightarrow{\lambda} Q \mid R\{x := M\}$. Since PSQ by hypothesis, and since \mathcal{S} is closed by parallel composition, we conclude $(P \mid R\{x := M\})\mathcal{S}(Q \mid R\{x := M\})$, as desired.
- ▷ The case when $\pi = \langle M \rangle^\uparrow$ is similar: it also requires the closure of \mathcal{S} by the ambient constructor.
- ▷ $!PS!Q$ because PSQ . We proceed by a case analysis of why $!P \xrightarrow{\lambda} O$, with O a process (not a concretion). We start with the structural case, below.
 - ▷ $!P \xrightarrow{\lambda} !P \mid P'$, derived from $P \xrightarrow{\lambda} P'$ for some process P' . Since PSQ , we use induction to find a move $Q \xRightarrow{\lambda} Q'$ with $P'SQ'$. Thus $!Q \xRightarrow{\lambda} !Q \mid Q'$ by an application of (REPL). Then we have $!PS!Q$ and $P'SQ'$. Since \mathcal{S} is closed by parallel composition, this implies $(!P \mid P')\mathcal{S}(!Q \mid Q')$, as desired.

The remaining cases arises when λ is an higher order action, we show only a couple of representative cases, the others follows similarly.

- ▷ $!P \xrightarrow{\langle - \rangle^R} P' = (\nu \tilde{p})(P_1 \mid R\{x := M\})$, because $!P \xrightarrow{\langle - \rangle} (\nu \tilde{p})\langle M \rangle P_1$ with $\tilde{p} \cap fn(R) = \emptyset$ and $fv(R) \subseteq \{x\}$. The last transition must have been derived by structural rule (REPL) from $P \xrightarrow{\langle - \rangle} (\nu \tilde{p})\langle M \rangle P_2$ with $P_1 = !P \mid P_2$ and $\tilde{p} \cap fn(!P) = \emptyset$. Then we have that $P \xrightarrow{\langle - \rangle^R} (\nu \tilde{p})(P_2 \mid R\{x := M\})$. Now, from the hypothesis PSQ , there exists a process Q' such that $Q \xRightarrow{\langle - \rangle^R} Q'$ and $Q'\mathcal{S}(\nu \tilde{p})(P_2 \mid R\{x := M\})$. This reduction must have the following form:

$$Q \Longrightarrow V \xrightarrow{\langle - \rangle^R} Z \Longrightarrow Q'$$

An inspection of the transition rules shows that $Z = (\nu \tilde{q})(Q_2 \mid R\{x := M_1\})$ and the transition $V \xrightarrow{\langle - \rangle^R} Z$ must have been derived from $V \xrightarrow{\langle - \rangle} (\nu \tilde{q})\langle M_1 \rangle Q_2$ with $\tilde{q} \cap fn(R) = \emptyset$ and $fv(R) \subseteq \{x\}$. Then, by repeated applications of structural rules, we have $!Q \Longrightarrow !Q \mid V \xrightarrow{\langle - \rangle} (\nu \tilde{q})\langle M_1 \rangle (Q_2 \mid !Q)$ for $fn(!Q) \cap \tilde{q} = \emptyset$. Thus, $!Q \Longrightarrow !Q \mid V \xrightarrow{\langle - \rangle^R} (\nu \tilde{q})(Q_2 \mid !Q \mid R\{x := M_1\})$, where $(\nu \tilde{q})(Q_2 \mid !Q \mid R\{x := M_1\}) = Z \mid !Q$ and $Z \mid !Q \Longrightarrow Q' \mid !Q$. That is $!Q \xRightarrow{\langle - \rangle^R} Q' \mid !Q$. It remains to show $(\nu \tilde{p})(P_2 \mid !P \mid R\{x := M\}) \mathcal{S} (Q' \mid !Q)$,

assuming $!PS!Q$ and $Q'S(\nu\tilde{p})(P_2 \mid R\{x := M\})$. This follows easily observing that $(\nu\tilde{p})(P_2 \mid !P \mid R\{x := M\}) \equiv !P \mid (\nu\tilde{p})(P_2 \mid R\{x := M\})$ since $\tilde{p} \cap \text{fn}(!P) = \emptyset$.

▷ $!P \xrightarrow{\langle - \rangle^\dagger n[R_1] R_2} P' = (\nu\tilde{p})(n[P_1 \mid R_1] \mid R_2\{x := M\})$, that derives from $!P \xrightarrow{\langle - \rangle^\dagger} (\nu\tilde{p})\langle M \rangle P_1$ with $\tilde{p} \cap (\text{fn}(R_2) \cup \text{fn}(n[R_1])) = \emptyset$ and $\text{fv}(R_2) \subseteq \{x\}$. The last transition must have been derived by structural rule (REPL) from $P \xrightarrow{\langle - \rangle^\dagger} (\nu\tilde{p})\langle M \rangle P_2$ with $P_1 = !P \mid P_2$ and $\tilde{p} \cap \text{fn}(!P) = \emptyset$. Then we have that $P \xrightarrow{\langle - \rangle^\dagger n[R_1 \mid !Q] R_2} (\nu\tilde{p})(n[P_2 \mid R_1 \mid !Q] \mid R_2\{x := M\})$ for $\tilde{p} \cap \text{fn}(!Q) = \emptyset$. Now, from the hypothesis PSQ , $\exists Q'$ such that $Q \xrightarrow{\langle - \rangle^\dagger n[R_1 \mid !Q] R_2} Q'$ and $Q'S(\nu\tilde{p})(n[P_2 \mid R_1 \mid !Q] \mid R_2\{x := M\})$. This reduction must have the following form:

$$Q \Longrightarrow V \xrightarrow{\langle - \rangle^\dagger n[R_1 \mid !Q] R_2} Z \Longrightarrow Q'$$

An inspection of transition rules shows that $Z = (\nu\tilde{q})(n[Q_2 \mid R_1 \mid !Q] \mid R_2\{x := M_1\})$ and the transition $V \xrightarrow{\langle - \rangle^\dagger n[R_1 \mid !Q] R_2} Z$ must have been derived from $V \xrightarrow{\langle - \rangle^\dagger} (\nu\tilde{q})\langle M_1 \rangle Q_2$ with $\tilde{q} \cap (\text{fn}(R_2) \cup \text{fn}(n[R_1])) = \emptyset$ and $\text{fv}(R_2) \subseteq \{x\}$. Then, by repeated applications of structural rules, for $\text{fn}(!Q) \cap \tilde{q} = \emptyset$, we have

$$!Q \Longrightarrow !Q \mid V \xrightarrow{\langle - \rangle^\dagger} (\nu\tilde{q})\langle M_1 \rangle Q_2 \mid !Q$$

Thus $!Q \xrightarrow{\langle - \rangle^\dagger n[R_1] R_2} (\nu\tilde{q})(n[Q_2 \mid !Q \mid R_1] \mid R_2\{x := M_1\}) = Z$, then $!Q \xrightarrow{\langle - \rangle^\dagger n[R_1] R_2} Q'$. It remains to show $Q'S(\nu\tilde{p})(n[!P \mid P_2 \mid R_1] \mid R_2\{x := M\})$, assuming $Q'S(\nu\tilde{p})(n[P_2 \mid R_1 \mid !Q] \mid R_2\{x := M\})$; this follows from $!PS!Q$ and the fact that \mathcal{S} is closed by parallel composition, ambient construct and restriction.

▷ $(P \mid R) \mathcal{S} (Q \mid R)$ because PSQ . We proceed by a case analysis of why $P \mid R \xrightarrow{\lambda} O$, with O a process (not a concretion). There thirteen cases in all to consider, plus their duals. We start with the structural case, below.

▷ $P \mid R \xrightarrow{\lambda} P' \mid R$ because $P \xrightarrow{\lambda} P'$. Since PSQ , by induction hypothesis we find a weak transition $Q \xRightarrow{\lambda} Q'$ with $P'SQ'$. Thus, we also have a weak transition $Q \mid R \xRightarrow{\lambda} Q' \mid R$, and since \mathcal{S} is closed by parallel composition, $P' \mid RSQ' \mid R$ as desired.

Then there are six cases of τ -transitions, plus their duals.

▷ $P \mid R \xrightarrow{\tau} O$ because $P \xrightarrow{\text{enter}\langle m,k \rangle} (\nu \tilde{p})(n[P_1])P_2$ and $R \xrightarrow{m \overline{\text{enter}}(n,k)} (\nu \tilde{r})(R_1)R_2$, with $O \equiv (\nu \tilde{r})(\nu \tilde{p})(m[R_1 \mid n[P_1]] \mid P_2 \mid R_2)$, and $R_1 \equiv R_x\{x := n\}$ for a suitable R_x . We must find a matching move $Q \mid R \Longrightarrow O'$ with OSO' .
By rule (ENTER HO) $P \xrightarrow{\text{enter}\langle m,k \rangle R_x} (\nu \tilde{p})(m[n[P_1] \mid R_x\{x := n\}] \mid P_2)$.
Since PSQ , by induction we find a weak transition $Q \xrightarrow{\text{enter}\langle m,k \rangle R_x} Q'$, with $Q'SP'$. This reduction must have the form $Q \Longrightarrow V \xrightarrow{\text{enter}\langle m,k \rangle R_x} Z \Longrightarrow Q'$.
An inspection of the LTS shows that Z must be of the form $(\nu \tilde{q})(m[l[Q_1] \mid R_x\{x := l\}] \mid Q_2)$ for suitable names l, \tilde{q} and processes Q_1 and Q_2 , and that the transition $V \xrightarrow{\text{enter}\langle m,k \rangle R_x} Z$ has been derived from $V \xrightarrow{\text{enter}\langle m,k \rangle} (\nu \tilde{q})(l[Q_1])Q_2$. From $R \xrightarrow{m \overline{\text{enter}}(n,k)} (\nu \tilde{r})(R_1)R_2$, it follows easily that $R \xrightarrow{m \overline{\text{enter}}(l,k)} (\nu \tilde{r})(R_x\{x := l\})R_2$. Hence by an application of the rule (τ ENTER)

$$Q \mid R \Longrightarrow V \mid R \xrightarrow{\tau} (\nu \tilde{r})(Z \mid R_2) \Longrightarrow (\nu \tilde{r})(Q' \mid R_2)$$

From $P'SQ'$, since \mathcal{S} is closed by restriction and parallel composition, it follows that $O = (\nu \tilde{r})(P' \mid R_2)\mathcal{S}(\nu \tilde{r})(Q' \mid R_2) = O'$, as desired.

▷ $P \mid R \xrightarrow{\tau} O$ because $P \xrightarrow{m \overline{\text{enter}}(n,k)} (\nu \tilde{p})(P_1)P_2$ and $R \xrightarrow{\text{enter}\langle m,k \rangle} (\nu \tilde{r})(n[R_1])R_2$, with $O \equiv (\nu \tilde{r})(\nu \tilde{p})(m[P_1 \mid n[R_1]] \mid R_2 \mid P_2)$. We must find a matching move $Q \mid R \Longrightarrow O'$ with OSO' .

By (CO-ENTER HO), $P \xrightarrow{m \overline{\text{enter}}(n,k)R_1} P' = (\nu \tilde{p})(m[n[R_1] \mid P_1] \mid P_2)$, with $\tilde{p} \cap \text{fn}(R_1) = \emptyset$. Since PSQ , we find a weak transition

$$Q \Longrightarrow V \xrightarrow{m \overline{\text{enter}}(n,k)R_1} Z \Longrightarrow Q'$$

with $Q'SP'$. An inspection of the transition rules shows that Z must be of the form $(\nu \tilde{q})(m[n[R_1] \mid Q_1] \mid Q_2)$ for suitable names \tilde{q} , and processes Q_1 and Q_2 . In particular, the transition $V \xrightarrow{m \overline{\text{enter}}(n,k)R_1} Z$ must have been derived

from $V \xrightarrow{m \text{ enter}(n,k)} (\nu \tilde{q})\langle Q_1 \rangle Q_2$. Thus, by an application of $(\tau \text{ ENTER})$

$$\begin{aligned} Q \mid R &\Longrightarrow V \mid R \\ &\xrightarrow{\tau} (\nu \tilde{r})(\nu \tilde{q})(m[n[R_1] \mid Q_1] \mid R_2 \mid Q_2) \\ &= (\nu \tilde{r})(Z \mid R_2) \\ &\Longrightarrow (\nu \tilde{r})(Q' \mid R_2) \end{aligned}$$

From $P'SQ'$, since \mathcal{S} is closed by restriction and parallel composition, it follows that $O = (\nu \tilde{r})(P' \mid R_2)\mathcal{S}(\nu \tilde{r})(Q' \mid R_2) = O'$, as desired.

▷ $P \mid R \xrightarrow{\tau} O$ because $P \xrightarrow{\text{pop}\langle k \rangle} (\nu \tilde{p})\langle m \rangle P'$ and $R \xrightarrow{\text{out}(m,k)} R'$, where O structurally equivalent to $(\nu \tilde{p})(P' \mid R')$ and R' is of the form $R_x\{x := m\}$ for a suitable R_x .

By the rule (EXIT HO) , we derive $P \xrightarrow{\text{pop}\langle k \rangle R_x} O$. Since PSQ , we may use the induction hypothesis to find a transition $Q \Longrightarrow V \xrightarrow{\text{pop}\langle k \rangle R_x} Z \Longrightarrow O'$ with OSO' .

An inspection of the transition rules shows that $V \xrightarrow{\text{pop}\langle k \rangle R_x} Z$ must have been derived from $V \xrightarrow{\text{pop}\langle k \rangle} (\nu \tilde{r})\langle l \rangle V'$ for suitable V' and l , with $Z \equiv (\nu \tilde{r})(V' \mid R_x\{x := l\})$. Also, from $R \xrightarrow{\text{out}(m,k)} R'$, it is easy to see that $R \xrightarrow{\text{out}(l,k)} R_x\{x := l\}$. Thus $V \mid R \xrightarrow{\tau} Z$, and we are done, since $Q \mid R \Longrightarrow V \mid R \xrightarrow{\tau} Z \Longrightarrow O'$

▷ $P \mid R \xrightarrow{\tau} O$ because $P \xrightarrow{\text{out}(m,k)} P'$ and $R \xrightarrow{\text{pop}\langle k \rangle} (\nu \tilde{r})\langle m \rangle R'$ with O structurally equivalent to $(\nu \tilde{r})(R' \mid P')$. Since PSQ , by induction hypothesis, we know that $Q \Longrightarrow U \xrightarrow{\text{out}(m,k)} Z \Longrightarrow Q'$. Thus $Q \mid R \Longrightarrow U \mid R \xrightarrow{\text{out}(m,k)} (\nu \tilde{r})(R' \mid Z) \Longrightarrow (\nu \tilde{r})(R' \mid Q') = O'$. Now, OSO' derives from $P'SQ'$ because \mathcal{S} is closed by parallel composition and restriction.

▷ $P \mid R \xrightarrow{\tau} O$ because $P \xrightarrow{\langle - \rangle} (\nu \tilde{p})\langle M \rangle P'$, $R \xrightarrow{\langle M \rangle} R'$ and O is structurally equivalent to $(\nu \tilde{p})(P' \mid R')$ and R' is of the form $R_x\{x := M\}$.

From $P \xrightarrow{\langle - \rangle} (\nu \tilde{p})\langle M \rangle P'$, by (Output HO) we derive $P \xrightarrow{\langle - \rangle R_x} O$. By induction hypothesis, since PSQ , we have $Q \Longrightarrow U \xrightarrow{\langle - \rangle R_x} Z \Longrightarrow O'$ with OSO' . The previous higher-order transition must have been derived from $U \xrightarrow{\langle - \rangle} (\nu \tilde{q})\langle N \rangle V$ with Z of the form $(\nu \tilde{q})(V \mid R_x\{x := N\})$. Thus, since

$R \xrightarrow{(N)} R_x\{x := N\}$, we have $U \mid R \xrightarrow{\tau} Z$ and then $Q \mid R \Longrightarrow U \mid R \xrightarrow{\tau} Z \Longrightarrow O'$ as desired.

- ▷ The dual case of the previous transition, (τ -EXCHANGE), and the two cases of (τ -GET) and (τ -PUT) follow the same pattern outline in the previous cases.

Finally we have seven cases for the higher-order transitions. We give the case of ($\text{OUTPUT}^\dagger \text{HO}$), which is the most complex. The other cases are similar and simpler.

- ▷ $P \mid R \xrightarrow{\langle - \rangle^\dagger n[R_1] R_2} O$, because $P \mid R \xrightarrow{\langle - \rangle^\dagger} K_S = (\nu \tilde{s})\langle M \rangle S$, and O is structurally equivalent to $(\nu \tilde{s})(n[S \mid R_1] \mid R_2\{x := M\})$. We have two possible sub-cases, depending on whether P or R move. We consider the latter case first.

If $R \xrightarrow{\langle - \rangle^\dagger} K_R$, then $K_S = K_R \mid P$, which implies $K_R = (\nu \tilde{s})\langle M \rangle R'$ and $S = R' \mid P$. Thus $O \equiv \mathbf{C}(P) = (\nu \tilde{s})(n[R' \mid P \mid R_1] \mid R_2\{x := M\})$. Clearly, $Q \mid R \xrightarrow{\langle - \rangle^\dagger n[R_1] R_2} \mathbf{C}(Q)$. By induction hypothesis PSQ , and since \mathcal{S} is closed by all the operators in the context $\mathbf{C}()$, we have $\mathbf{C}(P)\mathcal{S}\mathbf{C}(Q)$ as desired.

If instead P moves, i.e. $P \xrightarrow{\langle - \rangle^\dagger} K_P$, then $K_S = K_P \mid R$, which implies $K_P = (\nu \tilde{s})\langle M \rangle P'$ and $S = P' \mid R$. Thus O is structurally equivalent to $(\nu \tilde{s})(n[P' \mid R \mid R_1] \mid R_2\{x := M\})$. Now from $P \xrightarrow{\langle - \rangle^\dagger} K_P$, by ($\text{OUTPUT}^\dagger \text{HO}$), we derive $P \xrightarrow{\langle - \rangle^\dagger n[R \mid R_1] R_2} O$. Since PSQ , by the induction hypothesis there exists a weak transition

$$Q \Longrightarrow U \xrightarrow{\langle - \rangle^\dagger n[R \mid R_1] R_2} Z \Longrightarrow O'$$

with OSO' . An inspection of the transition rules shows that the last transition comes from $U \xrightarrow{\langle - \rangle^\dagger} (\nu \tilde{m})\langle N \rangle Q'$ with $Z = (\nu \tilde{m})(n[Q' \mid R \mid R_1] \mid R_2\{x := N\})$. Then, an application of (PAR) gives $U \mid R \xrightarrow{\langle - \rangle^\dagger} (\nu \tilde{m})\langle N \rangle Q' \mid R$, from which $U \mid R \xrightarrow{\langle - \rangle^\dagger n[R_1] R_2} Z$. We are done, since $Q \mid R \Longrightarrow U \mid R$ and $Z \Longrightarrow O'$.

- ▷ $n[P] \mathcal{S} n[Q]$ because PSQ . There are again several subcases to consider, one for each possible transition.

- ▷ $n[P] \xrightarrow{n \text{ get } M} O$ because $P \xrightarrow{(M)^\dagger} P'$ and $O \equiv n[P']$. Since PSQ , by the induction hypothesis, we know that $Q \xrightarrow{(M)^\dagger} Q'$ with $P'SQ'$. Thus $n[Q] \xrightarrow{n \text{ get } M} n[Q'] = O'$, and OSO' because \mathcal{S} is closed by the ambient constructor.
- ▷ $n[P] \xrightarrow{\text{exit}\langle m, k \rangle RS} O$ because $n[P] \xrightarrow{\text{exit}\langle m, k \rangle} (\nu)\langle n[P'] \rangle 0$, where O is structurally equivalent to $n[P'] \mid m[R] \mid S\{x := n\}$. The latter transition must have been derived from $P \xrightarrow{\text{out}\langle m, k \rangle} P'$. Since PSQ , by the induction hypothesis there exists Q' such that it follows by induction that $Q \Rightarrow \xrightarrow{\text{out}\langle m, k \rangle} \Rightarrow Q'$ and $P'SQ'$. Then

$$n[Q] \xrightarrow{\text{exit}\langle m, k \rangle RS} n[Z] \mid m[R] \mid S\{x := n\}$$

That OSO' follows again from $P'SQ'$ and from \mathcal{S} being closed under parallel composition and ambient construction.

- ▷ $n[P] \xrightarrow{\text{pop}\langle k \rangle R} O$ because $n[P] \xrightarrow{\text{pop}\langle k \rangle} (\nu\tilde{p})\langle m \rangle (m[P_1] \mid n[P_2])$, where O is structurally equivalent to $(\nu\tilde{p})(m[P_1] \mid n[P_2] \mid R\{x := m\})$. The latter transition must have been derived from $P \xrightarrow{\text{exit}\langle n, k \rangle} (\nu\tilde{p})\langle m[P_1] \rangle P_2$, from which $P \xrightarrow{\text{exit}\langle n, k \rangle 0R} O$. Since PSQ , by induction, we find a weak transition $Q \Rightarrow \xrightarrow{\text{exit}\langle n, k \rangle 0R} Z \Rightarrow O'$ where $Z \equiv (\nu\tilde{q})(l[Q_1] \mid n[Q_2] \mid R\{x := l\})$ and OSO' . By Lemma 5.3.10(1), we then have the desired weak reduction $n[Q] \xrightarrow{\text{pop}\langle k \rangle R} (\nu\tilde{q})(l[Q_1] \mid n[Q_2] \mid R\{x := l\}) \Rightarrow O'$.
- ▷ $n[P] \xrightarrow{n \text{ put } \langle - \rangle R} O$ because $n[P] \xrightarrow{n \text{ put } \langle - \rangle} (\nu\tilde{p})\langle M \rangle n[P']$, where O is structurally equivalent to $(\nu\tilde{p})(n[P'] \mid R\{x := M\})$. The latter transition must have been derived from $P \xrightarrow{\langle - \rangle^\dagger} (\nu\tilde{p})\langle M \rangle P'$, from which we derive $P \xrightarrow{\langle - \rangle^\dagger n[0]R} (\nu\tilde{p})(n[P'] \mid R\{x := M\})$ by an application of (OUTPUT[†] HO). Since PSQ , by induction hypothesis it follows that there exists O' such that $Q \xrightarrow{\langle - \rangle^\dagger n[0]R} O'$ and OSO' . An inspection of the transition rules shows that O' is of the form $(\nu\tilde{q})(n[Q'] \mid R\{x := N\})$ for suitable Q', R and N . By Lemma 5.3.10(2), we then have $n[Q] \xrightarrow{n \text{ put } \langle - \rangle R} O'$ as desired.
- ▷ The remaining cases, namely (AMB ENTER HO) and (AMB CO-ENTER HO) similar to and simpler than the previous ones.

▷ $(\nu n)P \mathcal{S} (\nu n)Q$ because PSQ . We proceed by a case analysis of why $(\nu n)P \xrightarrow{\lambda} O$, with O a process (not a concretion). We start with the structural case, below.

▷ $(\nu n)P \xrightarrow{\lambda} (\nu n)P'$, derived from $P \xrightarrow{\lambda} P'$ for some process P' and $n \notin fn(\lambda)$. Since PSQ , we use induction to find a move $Q \xRightarrow{\lambda} Q'$ with $P'\mathcal{S}Q'$. Thus $(\nu n)Q \xRightarrow{\lambda} (\nu n)Q'$ by an application of (RES). Now, from $P'\mathcal{S}Q'$, since \mathcal{S} is closed by restriction, we also have $(\nu n)P'\mathcal{S}(\nu n)Q'$, as desired.

The remaining cases arises when λ is an higher order action, we show only one representative case, the others follows similarly.

▷ $(\nu n)P \xrightarrow{\langle - \rangle R} P' \equiv (\nu n, \tilde{p})(P_1 \mid R\{x := M\})$, from $(\nu n)P \xrightarrow{\langle - \rangle} (\nu n, \tilde{p})\langle M \rangle P_1$ with $\{n, \tilde{p}\} \cap fn(R) = \emptyset$ and $fv(R) \subseteq \{x\}$. The last transition must have been derived by structural rule (RES) from $P \xrightarrow{\langle - \rangle} (\nu \tilde{p})\langle M \rangle P_1$. Then we have that $P \xrightarrow{\langle - \rangle R} (\nu \tilde{p})(P_1 \mid R\{x := M\})$. Now, from the hypothesis PSQ , there exists a process Q' such that $Q \xRightarrow{\langle - \rangle R} Q'$ and $Q'\mathcal{S}(\nu \tilde{p})(P_1 \mid R\{x := M\})$. This reduction must have the following form:

$$Q \Longrightarrow V \xrightarrow{\langle - \rangle R} Z \Longrightarrow Q'$$

An inspection of the transition rules shows that $Z = (\nu \tilde{q})(Q_1 \mid R\{x := M_1\})$ and the transition $V \xrightarrow{\langle - \rangle R} Z$ must have been derived from $V \xrightarrow{\langle - \rangle} (\nu \tilde{q})\langle M_1 \rangle Q_1$ with $\tilde{q} \cap fn(R) = \emptyset$ and $fv(R) \subseteq \{x\}$. Then, by repeated applications of (RES), we have $(\nu n)Q \Longrightarrow (\nu n)V \xrightarrow{\langle - \rangle} (\nu n, \tilde{q})\langle M_1 \rangle Q_1$. Thus we have that $(\nu n)Q \xRightarrow{\langle - \rangle R} (\nu n, \tilde{q})(Q_1 \mid R\{x := M_1\})$, where $(\nu n, \tilde{q})(Q_1 \mid R\{x := M_1\}) = (\nu n)Z$ and $(\nu n)Z \Longrightarrow (\nu n)Q'$. That is $(\nu n)Q \xRightarrow{\langle - \rangle R} (\nu n)Q'$. It remains to show $(\nu n, \tilde{p})(P_1 \mid R\{x := M\}) \mathcal{S} (\nu n)Q'$, assuming $Q'\mathcal{S}(\nu \tilde{p})(P_1 \mid R\{x := M\})$. This follows easily since \mathcal{S} is closed under restriction.

□

Theorem 5.3.12 (Soundness of full bisimilarity). *If $P \approx_c Q$ then $P \cong Q$.*

Proof. It is enough to show that $\mathcal{S} = \{(P, Q) \mid P \approx_c Q\}$ is a barbed bisimulation, up to \equiv . Assume PSQ . If $P \downarrow_n$ then, by Lemma 5.3.4 $P \xrightarrow{n \text{ enter}(m,k)} \cdot$. Since $P \approx_c Q$ we know that $Q \xRightarrow{n \text{ enter}(m,k)}$, from which $Q \downarrow_n$. Now assume that $P \rightarrow P'$. By Theorem

5.3.3 $P \xrightarrow{\tau} \equiv P'$. Since $P \approx_c Q$, there exists Q' such that $Q \Longrightarrow Q'$ and $P' \equiv \approx_c \equiv Q'$. Thus also $P' \equiv S \equiv Q'$, as desired. \square

Theorem 5.3.12 shows that the bisimilarity we defined is sufficient to use it as a proof technique to prove that two processes are reduction barbed congruent. It would be interesting to further compare the two equivalences, finding out if \approx_c is strictly finer than \cong or if they are actually the same. However we could not prove the converse of Theorem 5.3.12, and we could neither find out a counterexample. The main problem is the fact that processes like $(x)^\dagger P$ or $\overline{\text{in}}(x, k).P$ are visible at *top level* for the bisimilarity, while they need to be inserted in a context of the form $\mathbf{C}() = n[Q_1 \mid ()] \mid Q_2$ to be observed by the barbed congruence. Then when testing, for example $(x)^\dagger P$, we have $(x)^\dagger P \xrightarrow{(M)^\dagger} P' \xrightarrow{\lambda} P''$ with the bisimilarity, while to test the same process with barbed congruence, we have to insert it within a context like $n[(x)^\dagger P] \mid \langle M \rangle^n R$, where R must now be able to observe $n[P]$ instead of the simple P . This suggests that in order to find a bisimulation that precisely corresponds to reduction barbed congruence we have to enrich the given LTS with additional higher order rules that handles also the cases of $(x)^\dagger, \text{in}\langle a, k \rangle, \text{out}\langle a, k \rangle, \overline{\text{in}}(x, k)$. We leave this problem to future work, as well as the proof or the confutation of the converse of Theorem 5.3.12.

5.3.2 Algebraic Laws

In the remainder of the section we give a collections of laws which hold in NBA. All these laws can be easily proved using our notion of bisimilarity by exhibiting the appropriate bisimulation relation.

In the following we write $\Pi_{i \in I} P_i$ for the parallel composition of the processes P_i for $i \in I$. The first set of laws is about garbage collection.

Theorem 5.3.13 (Garbage Collection Laws).

1. $l[\Pi_{i \in I} (\tilde{x}_i)^{n_i} P_i \mid \Pi_{j \in J} (\tilde{x}_j) P_j \mid \Pi_{h \in H} \langle \tilde{M}_h \rangle^{m_h} P_h] \cong \mathbf{0}$
2. $l[\Pi_{i \in I} (\tilde{x}_i)^{n_i} P_i \mid \Pi_{j \in J} \langle \tilde{M}_j \rangle P_j \mid \Pi_{h \in H} \langle \tilde{M}_h \rangle^{m_h} P_h] \cong \mathbf{0}$

Proof. In both cases, the set containing the sole pair of the two processes (in the left and in the right hand side of \cong) is a full bisimulation: this follows by observing that none of the processes in the two laws has any transition. \square

Taking $I = J = H = \emptyset$ in the previous laws, one also derives $l[] \cong \mathbf{0}$. This law holds in SA(P) and not in MA , nor in the calculus we gave in Chapter 2. The second set of laws is about communication.

Theorem 5.3.14 (Buffer Laws). *For any finite J :*

1. $l[\prod_{j \in J} \langle \tilde{M}_j \rangle] \cong \prod_{j \in J} l[\langle \tilde{M}_j \rangle]$.
2. $l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \cong \prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow]$.

Proof. The first law follows directly by Theorem 5.3.13(2), as both sides of the law are equivalent to the nil process $\mathbf{0}$. When $J = \emptyset$, the second law also follows by Theorem 5.3.13. For $J \neq \emptyset$, define:

$$\mathcal{S} = \{ (l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \mid R, \prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow] \mid R) \mid J \text{ finite, } R \text{ process} \} \cup \approx_c$$

We show that \mathcal{S} is a bisimulation. The left-hand side may perform three moves:

- ▷ $l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \mid R \xrightarrow{\lambda} P'$ since $R \xrightarrow{\lambda} R'$ and $P' = l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \mid R'$.
Then $\prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow] \mid R \xrightarrow{\lambda} \prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow] \mid R' = Q'$ and $P' \mathcal{S} Q'$.
- ▷ $l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \mid R \xrightarrow{\lambda} P'$ since $l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \xrightarrow{\lambda} P_1$ and $P' = P_1 \mid R$.
Then $\lambda = l \text{ put } -S$ and $P_1 = l[\prod_{j \in J - \{k\}} \langle \tilde{M}_j \rangle^\uparrow] \mid S\{\tilde{x} := \tilde{M}_k\}$, for $k \in J$. For the right-hand side, first observe that $l[\langle M_k \rangle^\uparrow] \xrightarrow{l \text{ put } -S} l[] \mid S\{x := M_k\}$. Then, by repeated application of the (PAR) rule, one has that $\prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow] \xrightarrow{l \text{ put } -S} \prod_{j \in J - \{k\}} l[\langle \tilde{M}_j \rangle^\uparrow] \mid l[] \mid S\{\tilde{x} := \tilde{M}_k\} \mid R = Q'$. Now, $l[\prod_{j \in J - \{k\}} \langle \tilde{M}_j \rangle^\uparrow] \mid S\{\tilde{x} := \tilde{M}_k\} \mathcal{S} \prod_{j \in J - \{k\}} l[\langle \tilde{M}_j \rangle^\uparrow] \mid S\{\tilde{x} := \tilde{M}_k\}$, and from $l[] \approx_c \mathbf{0}$, it follows that $P' \mathcal{S} Q'$ as desired.
- ▷ $l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \mid R \xrightarrow{\tau} P'$. Then $l[\prod_{j \in J} \langle \tilde{M}_j \rangle^\uparrow] \xrightarrow{l \text{ put } -R} P'$. Now, reasoning as in the previous case, we have that $\prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow] \xrightarrow{l \text{ put } -R} Q'$ with $P' \mathcal{S} Q'$.
But we also have that $\prod_{j \in J} l[\langle \tilde{M}_j \rangle^\uparrow] \mid R \xrightarrow{\tau} Q'$, thus we are done since $P' \mathcal{S} Q'$.

The dual case, where the right hand side moves is exactly the same. \square

The two laws show how parallel composition of local and upward *asynchronous* messages distributes on the ambient construct. The latter law holds for the calculus BA: in neither

case (BA, or NBA) it extends to the case of (upward) output prefixes with non-nil continuation. The former law, instead, does not hold in BA. For example, consider the two processes $P = l[\langle M_1 \rangle \mid \langle M_2 \rangle]$ and $Q = l[\langle M_1 \rangle] \mid l[\langle M_2 \rangle]$. Then the context

$$\mathbf{C}() = () \mid n[\text{in } l.(x)^\dagger(x)^\dagger \text{out } l.\langle \rangle^\dagger]$$

distinguishes them. In fact, $\mathbf{C}(P) \Downarrow_n$ whereas $\mathbf{C}(Q) \not\Downarrow_n$. This implies (in fact, confirms) that local communication are not visible in NBA whereas they are in BA.

Theorem 5.3.15 (Communication Laws). *If $|\tilde{x}| = |\tilde{M}|$ then:*

1. $l[(\tilde{x})P \mid \langle \tilde{M} \rangle Q] \cong l[P\{\tilde{x} := \tilde{M}\} \mid Q]$
2. $(\nu l)((\tilde{x})^l P \mid l[\langle \tilde{M} \rangle^\dagger Q]) \cong (\nu l)(P\{\tilde{x} := \tilde{M}\} \mid l[Q])$
3. $m[(\tilde{x})^l P \mid l[\langle \tilde{M} \rangle^\dagger Q]] \cong m[P\{\tilde{x} := \tilde{M}\} \mid l[Q]]$

The dual laws of 2 and 3 (resulting from exchanging input with output prefixes) hold as well.

Proof. By exhibiting the appropriate bisimulation. In all cases, the bisimulation has the form $\{(LHS, RHS)\} \cup \mathcal{I}$, where LHS and RHS denote the left-hand side and the right-hand side of the identity, respectively. \square

Law 1 shows that NBA does not suffer from interferences on local communications: this law holds in Safe Ambients but not in Mobile and Boxed Ambients. The remaining laws are peculiar of NBA and show that, unlike BA, no interferences on upward communications are possible.

The following theorem presents some algebraic laws of mobility inherited from SAP where co-actions and passwords allow to avoid interferences on movements.

Theorem 5.3.16 (Mobility Laws).

1. $(\nu p)(m[\text{in}\langle n, p \rangle.P] \mid n[\overline{\text{in}}(x, p).Q]) \cong (\nu p)(n[Q\{x := m\} \mid m[P]])$
2. $l[m[\text{in}\langle n, p \rangle.P] \mid n[\overline{\text{in}}(x, p).Q]] \cong l[n[Q\{x := m\} \mid m[P]]]$
3. $(\nu p)(n[m[\text{out}\langle n, p \rangle.P]] \mid \overline{\text{out}}(x, p).Q) \cong (\nu p)(m[P] \mid Q\{x := m\})$
4. $l[n[m[\text{out}\langle n, p \rangle.P]] \mid \overline{\text{out}}(x, p).Q] \cong l[m[P] \mid Q\{x := m\}].$

Proof. Again, by exhibiting the appropriate bisimulation. In all cases, the bisimulation has the form $\{(LHS, RHS)\} \cup \mathcal{I}$, where LHS and RHS denote the left-hand side and the right-hand side of the identity, respectively. \square

5.4 The Type System

The absence of interferences in communication has other interesting payoffs besides those we have discussed in the previous section. Specifically, as we discuss below, it greatly simplifies the structure of the type system over the initial definition of Boxed Ambients. The structure of the types is similar to (but simpler than) that of Section 2.4.

$$\begin{array}{ll} \textit{Expression Types } W & ::= \text{ Name}[E] \quad \text{ambient/password} \\ & \mid \text{ Cap}[E] \quad \text{capability} \end{array}$$

$$\begin{array}{ll} \textit{Exchange Types } E, F & ::= \text{ shh} \quad \text{no exchange} \\ & \mid W_1 \times \cdots \times W_k \quad \text{tuples } (k \geq 0) \end{array}$$

$$\textit{Process Types } T ::= [E, F] \quad \text{composite exchange}$$

The main difference with respect to the type systems of Section 2.4 is in the structure of ambient types, that are defined here as one-place constructors of the form $\text{Name}[E]$, tracing the upward exchanges of ambients with this type. This simplification over previous systems (in which ambient types are two-place constructors) is a direct consequence of the semantics of communication, which guarantees the absence of interferences between the local exchanges of an ambient and the upward exchanges of its nested sub-ambients.

In addition, in the present system the types of the form $\text{Name}[E]$ also serve as the types of passwords: hence, $\text{Name}[E]$ is indeed the class of *name* types. When used as a password type, $\text{Name}[E]$ enables upward exchanges of type E for any process that relies on a password with this type to cross an ambient boundary. There is no type confusion in this double role of name types, as different uses of a name have different imports in the typing rules, with no risk of confusion. An alternative, perhaps more easily understood solution would be to use two different constructors for ambient and password names: however, this would also have the undesired effect of disallowing the same name to be used in the two roles, a feature that is harmless, and rather convenient in many examples.

The remaining types have the same interpretation as in previous type systems. In particular, $\text{Cap}[E]$ is the type of capabilities exercised within ambients with upward exchange of type E , and $[E, F]$ is the type of processes with local exchange of type E and upward exchanges of type F . The role of the type **shh** also is similar to that played by the corresponding type in the type system of Section 2.4. As in that case, it indicates absence of exchanges, and can be assigned to any process that does not have any local or upward exchange. In addition, in the type systems for NBA, **shh** provides for a *silent* mode for mobility in a way similar to, but substantially simpler than, the moded types of Section 2.6. Specifically, the typing rules guarantee that an ambient, say n , crossing a boundary with a password of type $\text{Name}[\text{shh}]$ does not disclose its name to the target ambient.

We proceed with the presentation of the typing rules, starting with rules for valid type environments, which are standard.

$$\begin{array}{c} \text{(ENV EMPTY)} \\ \hline \emptyset \vdash \diamond \end{array} \qquad \begin{array}{c} \text{(ENV NAME)} \\ \Gamma \vdash \diamond \quad a \notin \text{Dom}(\Gamma) \\ \hline \Gamma, a : W \vdash \diamond \end{array}$$

Typing of Expressions

$$\begin{array}{c} \text{(PROJECTION)} \\ \Gamma, a : W, \Gamma' \vdash \diamond \\ \hline \Gamma, a : W, \Gamma' \vdash a : W \end{array} \qquad \begin{array}{c} \text{(PATH)} \\ \Gamma \vdash M_1 : \text{Cap}[E_1] \quad \Gamma \vdash M_2 : \text{Cap}[E_2] \\ \hline \Gamma \vdash M_1.M_2 : \text{Cap}[E_1 \sqcup E_2] \end{array}$$

$$\begin{array}{c} \text{(ENTER)} \\ \Gamma \vdash M : \text{Name}[E] \quad \Gamma \vdash N : \text{Name}[F] \quad (F \leq G) \\ \hline \Gamma \vdash \text{in}\langle M, N \rangle : \text{Cap}[G] \end{array}$$

$$\begin{array}{c} \text{(EXIT)} \\ \Gamma \vdash M : \text{Name}[E] \quad \Gamma \vdash N : \text{Name}[F] \quad (F \leq G) \\ \hline \Gamma \vdash \text{out}\langle M, N \rangle : \text{Cap}[G] \end{array}$$

The notation $F \leq G$, with F and G exchange types, is defined as in Definition 2.4.1, namely: $F \leq G$ if and only if $F \in \{\text{shh}, G\}$. The rule (PROJECTION) is standard. The rules (ENTER) and (EXIT) define the types of capabilities in terms of the type of the component passwords: together with the typing rules for the process constructs for ambients and

mobility, they construe the types of passwords as *interfaces* for mobility. In particular, if the type F associated with the password N is a message type W (equivalently, a tuple), then N constrains any ambient relying upon N for mobility to have upward exchanges of type W (cf. the rules (PREFIX) and (AMB). If, instead, $F = \text{shh}$, then N enforces no constraint, as the type G in the conclusion of the rule can be any exchange type. The rule (PATH) follows the same intuition: it is applicable only when $E_1 \sqcup E_2$ is defined (where \sqcup is the (partial) lub operator associated with \leq).

Typing of Processes The following rules are standard.

$$\begin{array}{ccc}
\text{(REPL)} & \text{(DEAD)} & \text{(NEW)} \\
\frac{\Gamma \vdash P : [E, F]}{\Gamma \vdash !P : [E, F]} & \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0} : [E, F]} & \frac{\Gamma, n : \text{Name}[G] \vdash P : [E, F]}{\Gamma \vdash (\nu n : \text{Name}[G])P : [E, F]}
\end{array}$$

The rule (AMB) is standard, and construes the type $\text{Name}[E]$ as the interface of the ambient M for any process that knows the name M : any such process may have sound E exchanges with M , as the process enclosed within M has upward exchanges of this type.

$$\begin{array}{cc}
\text{(AMB)} & \text{(PAR)} \\
\frac{\Gamma \vdash M : \text{Name}[E] \quad \Gamma \vdash P : [F, E]}{\Gamma \vdash M[P] : [G, H]} & \frac{\Gamma \vdash P : [E, F] \quad \Gamma \vdash Q : [E, F]}{\Gamma \vdash P \mid Q : [E, F]}
\end{array}$$

The rules for the mobility co-actions provide similar guarantees for the exchanges a process may have with ambients whose name the process gets to know by exercising the co-capability. In this case, it is the type of the password M that acts as interface: if M has a type $\text{Name}[\tilde{W}]$, as in the rules (CO-ENTER) and (CO-EXIT), we are guaranteed that \tilde{W} is indeed the type of the exchanges of the incoming ambient. If instead the password type is $\text{Name}[\text{shh}]$, then no such guarantee can be provided, as it is easily verified by an inspection of the (PREFIX) rule (and of the rules for communication below). To recover soundness, the rules (CO-ENTER-SILENT) and (CO-EXIT-SILENT) require that no use be made by the continuation process P of the variable x (hence of the name of the incoming ambient, unless the name was known already to P).

$$\begin{array}{c}
\text{(PREFIX)} \\
\frac{\Gamma \vdash M : \text{Cap}[F] \quad \Gamma \vdash P : [E, G] \quad (F \leq G)}{\Gamma \vdash M.P : [E, G]} \\
\\
\text{(CO-ENTER)} \\
\frac{\Gamma \vdash M : \text{Name}[\tilde{W}] \quad \Gamma, x : \text{Name}[\tilde{W}] \vdash P : [E, F]}{\Gamma \vdash \overline{\text{in}}(x, M).P : [E, F]} \\
\\
\text{(CO-EXIT)} \\
\frac{\Gamma \vdash M : \text{Name}[\tilde{W}] \quad \Gamma, x : \text{Name}[\tilde{W}] \vdash P : [E, F]}{\Gamma \vdash \overline{\text{out}}(x, M).P : [E, F]} \\
\\
\text{(CO-ENTER-SILENT)} \\
\frac{\Gamma \vdash M : \text{Name}[\text{shh}] \quad \Gamma \vdash P : [E, F] \quad (x \notin \text{fv}(P))}{\Gamma \vdash \overline{\text{in}}(x, M).P : [E, F]} \\
\\
\text{(CO-EXIT-SILENT)} \\
\frac{\Gamma \vdash M : \text{Name}[\text{shh}] \quad \Gamma \vdash P : [E, F] \quad (x \notin \text{fv}(P))}{\Gamma \vdash \overline{\text{out}}(x, M).P : [E, F]}
\end{array}$$

An alternative, and still sound solution, would be to generalize the (CO-ENTER) and (CO-EXIT) rules by (systematically) replacing the type \tilde{W} with a generic exchange type G . With this solution, the two additional rules (CO-ENTER-SILENT) and (CO-EXIT-SILENT) could be dispensed with. On the other hand, the resulting system would be less general than the present one, in that any ambient using a silent password (i.e. one of type $\text{Name}[\text{shh}]$) for mobility would be required to be upward silent. With our solution, instead, there is no such constraint: the typing rules only prevent upward exchanges with the processes enclosed into ambients reached by the use of a silent password. The last set of rules, are those for input output, that are again completely standard.

$$\begin{array}{cc}
\text{(INPUT)} & \text{(OUTPUT)} \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [\tilde{W}, E]}{\Gamma \vdash (\tilde{x} : \tilde{W})P : [\tilde{W}, E]} & \frac{\Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : [\tilde{W}, E]}{\Gamma \vdash \langle \tilde{M} \rangle P : [\tilde{W}, E]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } \uparrow \text{)} \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [E, \tilde{W}]}{\Gamma \vdash (\tilde{x} : \tilde{W})^\uparrow P : [E, \tilde{W}]}
\end{array}
\quad
\begin{array}{c}
\text{(OUTPUT } \uparrow \text{)} \\
\frac{\Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : [E, \tilde{W}]}{\Gamma \vdash \langle \tilde{M} \rangle^\uparrow P : [E, \tilde{W}]}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } M \text{)} \\
\frac{\Gamma \vdash M : \mathbf{Name}[\tilde{W}] \quad \Gamma, \tilde{x} : \tilde{W} \vdash P : [G, H]}{\Gamma \vdash (\tilde{x} : \tilde{W})^M P : [G, H]}
\end{array}$$

$$\begin{array}{c}
\text{(OUTPUT } N \text{)} \\
\frac{\Gamma \vdash N : \mathbf{Name}[\tilde{W}] \quad \Gamma \vdash \tilde{M} : \tilde{W} \quad \Gamma \vdash P : [G, H]}{\Gamma \vdash \langle \tilde{M} \rangle^N P : [G, H]}
\end{array}$$

With the standard technique, we prove that the type system is sound.

Proposition 5.4.1 (Subject Reduction). *If $\Gamma \vdash P : T$ is a derivable judgment in NBA, and $P \rightarrow Q$, then $\Gamma \vdash Q : T$ is also a derivable judgment.* \square

5.5 Examples

We illustrate the expressive power of NBA, and of its algebraic theory with several examples. The first is a rather standard expressiveness test, namely the encoding of the following, choice-free fragment of the pi-calculus [MPW92].

$$P \in \pi ::= \overline{a} \langle \tilde{b} \rangle . P \mid a(\tilde{x}) . P \mid P \mid P \mid (\nu a) P$$

5.5.1 Encoding of Channels

Among the possible solutions, we present the most compact one, which further illustrates the power of our co-actions. The encoding is defined compositionally:

$$\begin{aligned}
\llbracket \overline{a} \langle \tilde{b} \rangle P \rrbracket &\triangleq (\nu p) (a[\langle \tilde{b} \rangle^\uparrow a[\mathbf{out} \langle a, p \rangle]] \mid \overline{\mathbf{out}}(x, p). \llbracket P \rrbracket) \\
&\quad x \notin \mathit{fv}(P) \quad p \notin (\mathit{fn}(P) \cup \{\tilde{b}\}) \\
\llbracket a(\tilde{x}) P \rrbracket &\triangleq (\tilde{x})^a \llbracket P \rrbracket \\
\llbracket (\nu a) P \rrbracket &\triangleq (\nu a) \llbracket P \rrbracket & \llbracket !\pi . P \rrbracket &\triangleq !\llbracket \pi . P \rrbracket \\
\llbracket P \mid Q \rrbracket &\triangleq \llbracket P \rrbracket \mid \llbracket Q \rrbracket & \llbracket \mathbf{0} \rrbracket &\triangleq \mathbf{0}
\end{aligned}$$

Given the direct nature of the encoding, its operational correctness is simple to prove. We refer to the following formulation of the commitment semantics of the fragment of the pi-calculus. The definition is adapted from [Mil99]. It uses concretions of the form $(\nu \tilde{p})\langle \tilde{q} \rangle P$ with $\{\tilde{p}\} \subseteq \{\tilde{q}\}$, and relies on the same conventions for the notation $(\nu n)O$ and $O \mid Q$ defined earlier in the chapter.

$$\begin{array}{c}
\text{(INPUT)} \\
\hline
a(\tilde{x}).P \xrightarrow{a(\tilde{b})} P\{\tilde{x} := \tilde{b}\}
\end{array}
\qquad
\begin{array}{c}
\text{(OUTPUT)} \\
\hline
\bar{a}\langle \tilde{b} \rangle.P \xrightarrow{\bar{a}} (\nu)(\tilde{b})P
\end{array}$$

$$\begin{array}{c}
\text{(COMM)} \\
\hline
\frac{P \xrightarrow{\bar{a}} (\nu \tilde{c})\langle \tilde{b} \rangle P' \quad Q \xrightarrow{a(\tilde{b})} Q' \quad fn(Q) \cap \{\tilde{c}\} = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu \tilde{c})(P' \mid Q')}
\end{array}$$

$$\begin{array}{c}
\text{(RES)} \\
\hline
\frac{P \xrightarrow{\alpha} O \quad n \notin fn(\alpha)}{(\nu n)P \xrightarrow{\alpha} (\nu n)O}
\end{array}
\qquad
\begin{array}{c}
\text{(PAR)} \\
\hline
\frac{P \xrightarrow{\alpha} O}{P \mid Q \xrightarrow{\alpha} O \mid Q}
\end{array}
\qquad
\begin{array}{c}
\text{(REPL)} \\
\hline
\frac{\pi.P \xrightarrow{\alpha} O}{!\pi.P \xrightarrow{\alpha} O \mid !\pi.P}
\end{array}$$

Definition 5.5.1 (Barbs).

1. $P \downarrow_a$ if $P \xrightarrow{a(\tilde{b})} P'$ for some name \tilde{b} .
2. $P \downarrow_{\bar{a}}$ if $P \xrightarrow{\bar{a}} O$ for some concretion O . □

Definition 5.5.2 (Reduction Barbed Congruence for π -calculus). Reduction barbed congruence, written \cong , is the largest congruence relation over processes which is reduction closed and barb preserving. □

Let \succsim denote the *expansion* relation [AKH92], an asymmetric variant of \cong such that $P \succsim Q$ holds if $P \cong Q$, and P has at least as many τ -moves as Q . The formal definition is as follows. First we need the following notation: we write $P \xrightarrow{\hat{\tau}} P'$ if $P \xrightarrow{\tau} P'$ or $P = P'$.

Definition 5.5.3 (Expansion [SW01]). A relation \mathcal{R} is an *expansion* if whenever $P \mathcal{R} Q$,

- i) For each name n , $P \downarrow_n$ implies $Q \downarrow_n$.

- ii) For each name n , $Q \downarrow_n$ implies $P \downarrow_n$.
- iii) $P \xrightarrow{\tau} P'$ implies $Q \xrightarrow{\tau} Q'$ with $P' \mathcal{R} Q'$
- iv) $Q \xrightarrow{\tau} Q'$ implies $P \xrightarrow{\hat{\tau}} P'$ with $P' \mathcal{R} Q'$

Q expands P if PRQ for some expansion \mathcal{R} . We note \gtrsim the largest congruence relation contained within the expansion, and with \gtrsim_c its closure under substitution defined by $P \gtrsim_c Q$ if $P\sigma \gtrsim Q\sigma$ for all substitution σ . \square

Lemma 5.5.4 (Operational Correspondence). *Let P be a π -calculus process.*

1. Assume $P \xrightarrow{\alpha} O$. Then:

- (a) $\alpha = a(\tilde{b})$ implies that O is a process and $\llbracket P \rrbracket \xrightarrow{(\tilde{b})^a} \gtrsim_c \llbracket O \rrbracket$
- (b) $\alpha = \bar{a}$ implies $O \equiv (\nu \tilde{c})\langle \tilde{b} \rangle P'$ and $\llbracket P \rrbracket \xrightarrow{a \text{ put } \langle - \rangle} (\nu \tilde{c})\langle \tilde{b} \rangle P^*$ with $P^* \gtrsim_c \llbracket P' \rrbracket$
- (c) $\alpha = \tau$ implies that O is a process and $\llbracket P \rrbracket \xrightarrow{\tau} \gtrsim_c \llbracket O \rrbracket$

2. Assume $\llbracket P \rrbracket \xrightarrow{\alpha} O$. Then:

- (a) $\alpha = (\tilde{b})^a$ implies that O is a process and $\exists P' \in \pi$ such that $P \xrightarrow{a(\tilde{b})} P'$ with $O \gtrsim_c \llbracket P' \rrbracket$.
- (b) $\alpha = a \text{ put } \langle - \rangle$ implies $O \equiv (\nu \tilde{c})\langle \tilde{b} \rangle P_1$ and $\exists P' \in \pi$ s.t. $P \xrightarrow{\bar{a}} (\nu \tilde{c})\langle \tilde{b} \rangle P'$ and $P_1 \gtrsim_c \llbracket P' \rrbracket$
- (c) $\alpha = \tau$ implies that O is a process, and $\exists P' \in \pi$ such that $P \xrightarrow{\tau} P'$ and $O \gtrsim_c \llbracket P' \rrbracket$

Proof. of 1. by transition induction:

- $\triangleright P \xrightarrow{\alpha} O$ is $a(\tilde{x}).P_1 \xrightarrow{a(\tilde{b})} P_1\{\tilde{x} := \tilde{b}\}$. Then $\llbracket P \rrbracket = (x)^a \llbracket P_1 \rrbracket$ and $\llbracket P \rrbracket \xrightarrow{(\tilde{b})^a} \llbracket P_1 \rrbracket\{\tilde{x} := \tilde{b}\}$. Then we conclude observing that $\llbracket P_1 \rrbracket\{\tilde{x} := \tilde{b}\} = \llbracket P_1\{\tilde{x} := \tilde{b}\} \rrbracket$.
- $\triangleright P \xrightarrow{\alpha} O$ is $\bar{a}\langle \tilde{b} \rangle.P_1 \xrightarrow{\bar{a}} (\nu)\langle \tilde{b} \rangle P_1$.

Then $\llbracket P \rrbracket = (\nu p)(a[\langle \tilde{b} \rangle^\dagger a[\text{out}\langle a, p \rangle]] \mid \overline{\text{out}}(x, p).\llbracket P_1 \rrbracket)$ where $p \notin (fn(P_1) \cup \{\tilde{b}\})$, $fv(P_1) \cap x = \emptyset$ and $\llbracket P \rrbracket \xrightarrow{a \text{ put } -} (\nu)\langle \tilde{b} \rangle(\nu p)(a[a[\text{out}\langle a, p \rangle]] \mid \overline{\text{out}}(x, p).\llbracket P_1 \rrbracket)$. Let P^* be the process $(\nu p)(a[a[\text{out}\langle a, p \rangle]] \mid \overline{\text{out}}(x, p).\llbracket P_1 \rrbracket)$, we have to prove $P^* \gtrsim_c \llbracket P'_1 \rrbracket$. This comes from $a[\] \gtrsim_c \mathbf{0}$ and the fact that P^* can only perform the following move: $P^* \xrightarrow{\tau} \llbracket P_1 \rrbracket \mid (\nu p)a[\] \mid a[\]$.

- ▷ $P \xrightarrow{\alpha} 0$ is $P_1 \mid P_2 \xrightarrow{\tau} O$ from $P_1 \xrightarrow{\bar{a}} (\nu\tilde{c})(\tilde{b})P'_1$, $P_2 \xrightarrow{a(\tilde{b})} P'_2$, $O = (\nu\tilde{c})(P'_1 \mid P'_2)$ and $fn(P_2) \cap \{\tilde{c}\} = \emptyset$. By induction we have that there exist P_1^*, P_2^* such that $\llbracket P_1 \rrbracket \xrightarrow{a \text{ put } -} (\nu\tilde{c})(\tilde{b})P_1^*$, $P_1^* \gtrsim_c \llbracket P'_1 \rrbracket$ and $\llbracket P_2 \rrbracket \xrightarrow{(\tilde{b})^a} P_2^*$, $P_2^* \gtrsim_c \llbracket P'_2 \rrbracket$. Then from $fn(P_2) \cap \{\tilde{c}\} = \emptyset$ and the fact that for all process R , $fn(\llbracket R \rrbracket) = fn(R)$, we have $fn(\llbracket P_2 \rrbracket) \cap \{\tilde{c}\} = \emptyset$, then $\llbracket P_1 \mid P_2 \rrbracket \xrightarrow{\tau} (\nu\tilde{c})(P_1^* \mid P_2^*)$. Moreover, from $P_1^* \gtrsim_c \llbracket P'_1 \rrbracket$ and $P_2^* \gtrsim_c \llbracket P'_2 \rrbracket$ and the fact that \gtrsim_c is a congruence, we have $(\nu\tilde{c})(P_1^* \mid P_2^*) \gtrsim_c (\nu\tilde{c})(\llbracket P'_1 \rrbracket \mid \llbracket P'_2 \rrbracket)$ and we conclude observing that $(\nu\tilde{c})(\llbracket P'_1 \rrbracket \mid \llbracket P'_2 \rrbracket) = \llbracket (\nu\tilde{c})(P'_1 \mid P'_2) \rrbracket$.
- ▷ $P \xrightarrow{\alpha} O$ is $(\nu n)P \xrightarrow{\alpha} (\nu n)O$ since $P \xrightarrow{\alpha} O$ and $n \notin fn(\alpha)$. We have to distinguish three cases, depending on α ; we only show the case $\alpha = a(\tilde{b})$, the others follow similarly. From $P \xrightarrow{a(\tilde{b})} O$, by induction there exists a process P' such that $\llbracket P \rrbracket \xrightarrow{(\tilde{b})^a} P'$ and $P' \gtrsim_c \llbracket O \rrbracket$, then $(\nu n)\llbracket P \rrbracket \xrightarrow{(\tilde{b})^a} (\nu n)P'$ and $(\nu n)P' \gtrsim_c (\nu n)\llbracket O \rrbracket$ since \gtrsim_c is a congruence. Now we conclude observing that $(\nu n)\llbracket O \rrbracket = \llbracket (\nu n)O \rrbracket$ and $(\nu n)\llbracket P \rrbracket = \llbracket (\nu n)P \rrbracket$.

▷ The remaining cases follow by induction similarly to the previous one.

Proof of 2. by induction on the structure of P . The case $P = \mathbf{0}$ is vacuous.

- ▷ $P = a(\tilde{x}).P_1$. In this case we have $\llbracket P \rrbracket = (\tilde{x})^a \llbracket P_1 \rrbracket$, thus $\alpha = (\tilde{b})^a$ and $O = \llbracket P_1 \rrbracket \{\tilde{x} := \tilde{b}\}$. Let $P' = P_1 \{\tilde{x} := \tilde{b}\}$, then $P \xrightarrow{a(\tilde{b})} P'$ and we conclude observing that $O = \llbracket P_1 \{\tilde{x} := \tilde{b}\} \rrbracket = \llbracket P' \rrbracket$.
- ▷ $P = \bar{a}(\tilde{b}).P_1$. In this case we have $\llbracket P \rrbracket = (\nu p)(a[\langle \tilde{b} \rangle^\dagger a[\text{out}\langle a, p \rangle]] \mid \overline{\text{out}}(x, p). \llbracket P_1 \rrbracket)$, with $p \notin (fn P_1 \cup \{\tilde{b}\})$ and $fv(P_1) \cap x = \emptyset$. Thus $\alpha = a \text{ put } -$ and we have that $O = (\nu)\langle \tilde{b} \rangle (\nu p)(a[a[\text{out}\langle a, p \rangle]] \mid \overline{\text{out}}(x, p). \llbracket P_1 \rrbracket)$. Let $P' = P_1$, then $P \xrightarrow{\bar{a}} (\nu)\langle \tilde{b} \rangle P_1$. Finally, $(\nu p)(a[a[\text{out}\langle a, p \rangle]] \mid \overline{\text{out}}(x, p). \llbracket P_1 \rrbracket) \gtrsim_c \llbracket P_1 \rrbracket$ comes from the fact that P^* can only perform the following move: $P^* \xrightarrow{\tau} \llbracket P_1 \rrbracket \mid (\nu p)a[\] \mid a[\]$, together with $a[\] \gtrsim_c \mathbf{0}$.
- ▷ $P = P_1 \mid P_2$. If the hypothesis comes by rule (PAR), then we conclude by induction, using the fact that \gtrsim_c is a congruence. If $\alpha = \tau$ then the hypothesis comes from $\llbracket P_1 \rrbracket \xrightarrow{(\tilde{b})^a} P_1^*$, $\llbracket P_2 \rrbracket \xrightarrow{a \text{ put } -} (\nu\tilde{c})(\tilde{b})P_2^*$, $fn(\llbracket P_1 \rrbracket) \cap \{\tilde{c}\} = \emptyset$ and $O =$

$(\nu\tilde{c})(P_1^* \mid P_2^*)$. By induction we have that there exist P'_1, P'_2 such that $P_1 \xrightarrow{a(\tilde{b})} P'_1$, $P_1^* \gtrsim_c \llbracket P'_1 \rrbracket$ and $P_2 \xrightarrow{\tilde{a}} (\nu\tilde{c})\langle\tilde{b}\rangle P'_2$, $P_2^* \gtrsim_c \llbracket P'_2 \rrbracket$. Let $P' = (\nu\tilde{c})(P'_1 \mid P'_2)$. Then from $fn(\llbracket P_1 \rrbracket) \cap \{\tilde{c}\} = \emptyset$ we have $fn(P_1) \cap \{\tilde{c}\} = \emptyset$, thus $P \xrightarrow{\tau} P'$. Moreover, from $P_1^* \gtrsim_c \llbracket P'_1 \rrbracket$ and $P_2^* \gtrsim_c \llbracket P'_2 \rrbracket$, since \gtrsim_c is a congruence, we have that $O = (\nu\tilde{c})(P_1^* \mid P_2^*) \gtrsim_c (\nu\tilde{c})(\llbracket P'_1 \rrbracket \mid \llbracket P'_2 \rrbracket)$, and we conclude observing that $(\nu\tilde{c})(\llbracket P'_1 \rrbracket \mid \llbracket P'_2 \rrbracket) = \llbracket P' \rrbracket$

▷ The remaining cases follow easily by induction using the fact that \gtrsim_c is a congruence. \square

The result extends to weak reductions.

Proposition 5.5.5 ([Bor98]). *Let P a pi-calculus process.*

1. *if $P \Longrightarrow P'$ then $\llbracket P \rrbracket \Longrightarrow \gtrsim_c \llbracket P' \rrbracket$*
2. *if $\llbracket P \rrbracket \Longrightarrow Q$, then there exists P' such that $P \Longrightarrow P'$ and $Q \gtrsim_c \llbracket P' \rrbracket$*
3. *$P \Downarrow_n$ if and only if $\llbracket P \rrbracket \Downarrow_n$.*

Proof. Items 1 and 2 are both items are proved by induction on the number of reduction steps. Item 3 follows easily from 1 and 2.

1. The base case is trivial. For the inductive case, assume $P \Longrightarrow^{n-1} P^* \rightarrow P'$. By induction hypothesis $\llbracket P \rrbracket \Longrightarrow R \gtrsim_c \llbracket P^* \rrbracket$. From $P^* \rightarrow P'$, by Lemma 5.5.4(1) we know that $\llbracket P^* \rrbracket \rightarrow \gtrsim_c \llbracket P' \rrbracket$. From $R \gtrsim_c \llbracket P^* \rrbracket$ and $\llbracket P^* \rrbracket \rightarrow \gtrsim_c \llbracket P' \rrbracket$, we know that $R \Longrightarrow \gtrsim_c \llbracket P' \rrbracket$. Thus $\llbracket P \rrbracket \Longrightarrow R \Longrightarrow \gtrsim_c \llbracket P' \rrbracket$ as desired.
2. The base case is again trivial. For the inductive step, assume $\llbracket P \rrbracket \Longrightarrow Q' \rightarrow Q$. By induction hypothesis there exists P' such that $P \Longrightarrow P'$ with $Q' \gtrsim_c \llbracket P' \rrbracket$. From this, and from $Q' \rightarrow Q$ we have two possible cases: either $Q \gtrsim_c \llbracket P' \rrbracket$, or $\llbracket P' \rrbracket \rightarrow P'' \lesssim_c Q$. In the first case we are done. In the second, by Lemma 5.5.4(2) there is P^* such that $P' \rightarrow P^*$ with $P'' \gtrsim_c \llbracket P^* \rrbracket$. Thus, there is P^* such that $P \Longrightarrow P' \rightarrow P^*$ with $Q \gtrsim_c P'' \gtrsim_c \llbracket P^* \rrbracket$, as desired.
3. From the definition of the encoding and theorem 5.3.7, it is verified that $P \Downarrow_n$ if and only if $\llbracket P \rrbracket \Downarrow_n$.

Then, for the (only if) part of the claim, assume $P \Longrightarrow P' \downarrow_n$. By (1) we have that $\llbracket P \rrbracket \Longrightarrow R \gtrsim_c \llbracket P' \rrbracket$. Thus $R \downarrow_n$ and hence also $P \downarrow_n$.

For the (if) part, assume $\llbracket P \rrbracket \Longrightarrow Q \downarrow_n$. By (2) there exists P' such that $P \Longrightarrow P'$ and $Q \gtrsim_c \llbracket P' \rrbracket$. Thus $\llbracket P' \rrbracket \downarrow_n$ which implies $P' \downarrow_n$ and then $P \downarrow_n$. \square

Based on that, and on the compositionality of $\llbracket \cdot \rrbracket$, we can show that the encoding is sound, in the following sense.

Theorem 5.5.6 (Equational Soundness). *If $\llbracket P \rrbracket \cong \llbracket Q \rrbracket$ then $P \cong Q$.*

Proof. We show that $\mathcal{S} = \{(P, Q) \mid \llbracket P \rrbracket \cong \llbracket Q \rrbracket\}$ is a barbed congruence.

- ▷ Let $\mathbf{C}(P) \downarrow_n$, then by Proposition 5.5.5(3) we also have $\llbracket \mathbf{C}(P) \rrbracket \downarrow_n$, by compositionality of the encoding we have $\mathbf{C}_1(\llbracket P \rrbracket) \downarrow_n$ where $\mathbf{C}_1() = \llbracket \mathbf{C}() \rrbracket$. Now, from $\llbracket P \rrbracket \cong \llbracket Q \rrbracket$ we have that $\mathbf{C}_1(\llbracket Q \rrbracket) \downarrow_n$, that is $\llbracket \mathbf{C}(Q) \rrbracket \downarrow_n$, and we conclude $\mathbf{C}(Q) \downarrow_n$ by Proposition 5.5.5(3).
- ▷ Let be $\mathbf{C}(P) \rightarrow P'$, then by 5.5.5(1) we also have $\llbracket \mathbf{C}(P) \rrbracket \Longrightarrow R \gtrsim_c \llbracket P' \rrbracket$. By the compositionality of the encoding we have that $\llbracket \mathbf{C}(P) \rrbracket = \mathbf{C}_1(\llbracket P \rrbracket)$ where $\mathbf{C}_1() = \llbracket \mathbf{C}() \rrbracket$, thus, from the hypothesis $\llbracket P \rrbracket \cong \llbracket Q \rrbracket$ there exists a process S such that $\llbracket \mathbf{C}(Q) \rrbracket = \mathbf{C}_1(\llbracket Q \rrbracket) \Longrightarrow S$ and $S \cong R$. Now, by 5.5.5(2) there exists a process Q' such that $\mathbf{C}(Q) \Longrightarrow Q'$ and $S \gtrsim_c \llbracket Q' \rrbracket$. Then we have $\llbracket P' \rrbracket \lesssim_c R \cong S \gtrsim_c \llbracket Q' \rrbracket$, thus $\llbracket P' \rrbracket \cong \llbracket Q' \rrbracket$, that is $(P', Q') \in \mathcal{S}$ as desired.
- ▷ The symmetric cases are similar.

\square

5.5.2 NBA versus BA and SA

The next suite of examples relate NBA and its ‘parent’ calculi BA and SA, reinforcing our claim that NBA removes non-determinism from BA, and providing more a formal statement about their relative expressive power.

A point-to-point communication server

Our first example is a system that represents a server for point-to-point communication.

$$w(k) = k[\overline{\text{in}}(x, k). \overline{\text{in}}(y, k). (! (z)^x \langle z \rangle^y \mid ! (z)^y \langle z \rangle^x)]$$

$w(k)$ is a bidirectional forwarder for any pair of incoming ambients. An agent willing to participate in a point-to-point communication must know the password k and should be implemented as the process $A(a, k, P, Q) = a[\text{in} \langle k, k \rangle . P \mid \text{out} \langle k, k \rangle . Q]$ (with P performing the expected (upward) exchanges). A complete implementation of the point-to-point server can be then defined as shown below:

$$\text{p2p}(k) = (\nu r) (r[\langle \rangle^\dagger] \mid ! ()^r (w(k) \mid \overline{\text{out}}(-, k). \overline{\text{out}}(-, k). r[\langle \rangle^\dagger]))$$

The process $\text{p2p}(k)$ accepts a pair of ambients within the forwarder, provides them with the necessary support of the point-to-point exchange and then lets them out before preparing a new instance of $w(k)$ for a new protocol session. Given the configuration $\text{p2p}(k) \mid A(k, a_1, P_1, Q_1) \mid \cdots \mid A(k, a_n, P_n, Q_n)$, we are guaranteed that at most one pair of agents can be active within k at any given time (k is locked until the two ambients are inside k). In particular, one has:

$$\begin{aligned} & (\nu k) (\text{p2p}(k) \mid A(k, a_1, \langle M \rangle^\dagger P_1, Q_1) \mid A(k, a_2, \langle x \rangle^\dagger P_2 \{x\}, Q_2) \mid \Pi_{i \in I} A(K, a_i, R_i, S_i)) \\ & \implies \cong (\nu k) (\text{p2p}(k) \mid a_1[P_1 \mid Q_1] \mid a_2[P_1 \{x := M\} \mid Q_2] \mid \Pi_{i \in I} A(K, a_i, R_i, S_i)) \end{aligned}$$

This says that once (and if) the two agents have reached the forwarder, no other agent knowing the key k can interfere and prevent them from completing their exchange. The equivalence above follows by the mobility laws of Theorem 5.3.16 and the laws of Theorem 5.3.13. In particular, once the two ambients are back at top level, the currently active instance of the forwarder k has the form $k[! (z)^{a_1} \langle z \rangle^{a_2} \mid ! (z)^{a_2} \langle z \rangle^{a_1}] \cong \mathbf{0}$.

The use of the forwarder to implement a point-to-point communication protocol may at first appear artificial, for it would seem that two ambients wishing to communicate are likely to know their partner's name, and could then interact without intervention of a third party. Indeed, in NBA the example can be simplified with these assumption. In BA, instead, the knowledge of names still leaves a number of problems due to possible communication interferences.

To illustrate the point, let us consider the following BA coding of the protocol.

$$a[\text{in } b. \text{in } k. P] \mid b[k[! (x). \langle x \rangle^a \mid ! (x)^a. \langle x \rangle] \mid Q]$$

Here Q can read from and write to k to exchange values with P inside a , but it is not obvious what P should do. Clearly, with k as above, it must use local communication to talk with k , and hence with Q . However, this would not avoid interference with its own local exchanges. Not is it clear how to ban such unwanted effect, as there seem to be similar problems with several different implementations of k . For instance, using $k[!(x).\langle x \rangle]$ a and b may end up re-reading their own messages; while with $k[!(x).\langle x \rangle \mid (x)^\dagger.\langle x \rangle^a]$ the upward read by k may mistakenly synchronize with local output in b .

A print server

Our next example implements a print server to print jobs arriving off the network in the order of arrival. We give the implementation in steps. First consider the following process that assigns a progressive number to each incoming job.

$$\text{enqueue}(k) = (\nu c) (c[\langle 1 \rangle^\dagger] \mid !(n)^{c\overline{\text{in}}}(x, k).\langle n \rangle^x c[\langle n+1 \rangle^\dagger])$$

The (private) ambient c holds the current value of the counter. The process accepts a job and delivers it the current number. Then, it updates the counter and prepares for the next round. Now, we can show how this can be turned into a print server ($\text{enqueue}(k)$ is defined as above):

$$\begin{aligned} \text{prtsrv}(k) &= k[\text{enqueue}(k) \mid \text{print}] \\ \text{print} &= (\nu c) (c[\langle 1 \rangle^\dagger] \mid !(n)^{c\overline{\text{out}}}(x, n).(data)^x(P\{data\} \mid c[\langle n+1 \rangle^\dagger])) \\ \text{job}(M, s) &= (\nu p)p[\text{in}\langle s, s \rangle.(n)^\dagger(\nu q)q[\text{out}\langle p, n \rangle.\langle M \rangle^\dagger]] \end{aligned}$$

The process $\text{job}(M, s)$ enters the server $\text{prtsrv}(s)$, it is assigned a number to be used as a password for carrying the job M to the printer process P (note that the use of passwords is critical here).

This situation appears hard to implement with SA(P) or BA. In SA(P) because one would need to know the names of the incoming jobs to be able to assign them their numbers. In BA because dequeuing the jobs (according to the intended FIFO policy) requires a test of the number a job has been assigned, and an atomic implementation of such test appears to be problematic.

BA \lesssim NBA + Guarded Choice

In order to relate BA and NBA more formally, and support our claim that the only loss of expressiveness in the passage is very directly related the removal of grave communication interferences, we present an encoding of BA into an extended version of NBA. Precisely, we enrich NBA with a limited, focused form of nondeterminism, viz. sum operator, that we use in the encoding to circumscribe the nondeterminism in communication typical of BA. Besides showing the relationship between the two calculi, this has the further advantage of localizing their differences in a single construct.

The encoding is defined parametrically over four names n, mv, pr, pw : n is the name of the ambient (if any) that encloses the process that we are encoding, while the remaining three names are used as passwords. To ease the notation, we use the following shorthands: $\overline{\text{cross}} = \overline{\text{in}}(x, mv) \mid \overline{\text{out}}(x, mv)$, $\text{in } n = \text{in}\langle n, mv \rangle$, and $\text{out } n = \text{out}\langle n, mv \rangle$. The encoding is defined in terms of two mutually recursive translations, $\llbracket \cdot \rrbracket_n$ and $\langle \cdot \rangle_n$, with $\llbracket \cdot \rrbracket_n$ leading. The interesting cases are given below:

$$\begin{aligned}
\llbracket P \rrbracket_n &= \overline{\text{cross}} \mid \langle P \rangle_n \\
\langle \text{in } a.P \rangle_n &= \text{in}\langle a, mv \rangle. \langle P \rangle \\
\langle \text{out } a.P \rangle_n &= \text{out}\langle a, mv \rangle. \langle P \rangle \\
\langle m[P] \rangle_n &= m[\llbracket P \rrbracket_m] \\
\langle (x)^a P \rangle_n &= (x)^a \langle P \rangle_n \\
\langle (x)P \rangle_n &= (x) \langle P \rangle_n + (x)^\dagger \langle P \rangle_n + \overline{\text{out}}(y, pw)(x)^y \langle P \rangle_n & y \notin fv(P) \\
\langle (x)^\dagger P \rangle_n &= (\nu p)p[\text{out}\langle n, pr \rangle.(x)^\dagger \text{in}\langle n, p \rangle.\langle x \rangle^\dagger] \mid \overline{\text{in}}(y, p)(x)^y \langle P \rangle_n & p \notin fn(P), y \notin fv(P) \\
\langle \langle M \rangle^a P \rangle_n &= \langle M \rangle^a \langle P \rangle_n \\
\langle \langle M \rangle P \rangle_n &= \langle M \rangle \langle P \rangle_n + \langle M \rangle^\dagger \langle P \rangle_n + \overline{\text{out}}(y, pr)\langle M \rangle^y \langle P \rangle_n & y \notin fv(P) \\
\langle \langle M \rangle^\dagger P \rangle_n &= (\nu p)p[\text{out}\langle n, pw \rangle.\langle M \rangle^\dagger \text{in}\langle n, p \rangle.\langle \rangle^\dagger] \mid \overline{\text{in}}(y, p)()^y \langle P \rangle_n & p \notin fn(P), y \notin fv(P)
\end{aligned}$$

The remaining cases are defined homomorphically. The encoding $\llbracket \cdot \rrbracket_n$ provides unboundedly many co-capabilities, at all nesting levels, so that ambient mobility in BA is rendered faithfully. As for the translation of the communication primitives, the intuition is the following. The upward exchanges of a BA term are dealt with by the taxi ambients that exit the enclosing ambient n to deliver output (or collect input) and then return to n to unlock the continuation P . The use of restricted names as passwords is essential here for the continuation P to be able to identify its helper taxi ambient without risk of

confusion. As for the translation of a local input/output, the three branches of the choice reflect the three possible synchronizations: local, from upward, from a nested ambient. Note, in particular, that the right-most branch of these choices may only match upward requests that are encoding of upward requests in a BA term: this is guaranteed by the use of the two passwords \mathbf{pr} and \mathbf{pw} that regulate the moves of the read/write taxi ambients. The use of two different passwords ensure that they do not interfere with each other, nor they interfere with other BA ambients' moves (the latter use \mathbf{mv}).

Using the algebraic laws in §5.3.2 we can show that the encoding is operationally correct (and equationally sound) for *single-threaded* terms. Our notion of single-threadedness here, although morally identical to SA's, is adapted to BA to record that engaging in inter-ambient communications is an activity across ambient boundaries that may as well create grave interferences. For instance, $a[\langle x \rangle^\dagger \mid \text{out}\langle n, k \rangle.P]$ cannot be considered single-threaded, as illustrated by, say, the context $\overline{\text{out}}(x, k).R \mid n[(x)^a Q \mid -]$. We give a syntactical definition of single-threadedness. P is a single threaded process if it does not contain any subprocess of the form $H \mid H$, where H is built according to the following production:

$$H ::= (\nu \tilde{p}) \pi_1 \dots \pi_k M.P \mid (\nu \tilde{p}) \pi_1 \dots \pi_k \langle M \rangle^\dagger P \mid (\nu \tilde{p}) \pi_1 \dots \pi_k (x)^\dagger P \quad k \geq 0$$

Theorem 5.5.7.

If P and Q are single-threaded, then $\llbracket P \rrbracket_n \cong \llbracket Q \rrbracket_n$ implies $P \cong Q$. □

The proof follows the same pattern as the one outlined for the encoding of the pi-calculus. The additional hypothesis on the two source terms P and Q is needed to guarantee the atomicity of the protocol that implements an upward exchange (once the taxi ambient leaves n , we need to make sure that no process inside n causes n to move).

Typed Encoding. The encoding extends smoothly to the typed case. The most interesting aspect of the typed version is the encoding of types, given below:

$$\begin{aligned} \llbracket \text{amb}[E, F] \rrbracket &= \text{Name}[\llbracket E \rrbracket] & \llbracket \text{shh} \rrbracket &= \text{shh} \\ \llbracket \text{cap}[E] \rrbracket &= \text{Cap}[\text{shh}] & \llbracket [E, F] \rrbracket &= [\llbracket E \rrbracket, \llbracket F \rrbracket] \end{aligned}$$

Perhaps surprisingly, the type traced in $\llbracket \text{amb}[E, F] \rrbracket$ is (the encoding of) the type of the local exchanges: this is because the upward exchanges of in BA are implemented by

the helper taxi ambients, whose type will trace the (encoding of) the type F . The local exchanges (again of the source term) are used for typing the upward and local exchanges generated by the translation. The translation of the capability and process types follows the same intuitions, and are direct consequence of the fact that the upward exchanges in the source ambient types are disregarded in the translation (for the reasons we just explained)

The typed encoding is given in the following. The main difference from the untyped case is in the use of a family of passwords \mathbf{pr}_W and \mathbf{pw}_W , indexed on types, with the implicit assumption that $\mathbf{pr}_W, \mathbf{pw}_W : \mathbf{Name}[W]$ for all (NBA) types W . This indexing is required in the typed case, for each of these passwords enables exchanges of the corresponding type. The same would seem needed for the \mathbf{mv} password. However, since the co-capabilities that enable mobility à la BA do not have any continuation, we can safely keep with the sole \mathbf{mv} , provided that we stipulate $\mathbf{mv} : \mathbf{Name}[\mathbf{shh}]$.

$$\begin{aligned}
\llbracket \Gamma \triangleright P \rrbracket_n &= \overline{\mathbf{cross}} \mid \langle \Gamma \triangleright P \rangle_n \\
\langle \Gamma \triangleright P \mid Q \rangle_n &= \langle \Gamma \triangleright P \rangle_n \mid \langle \Gamma \triangleright Q \rangle_n & \langle \Gamma \triangleright M.P \rangle_n &= M.\langle \Gamma \triangleright P \rangle_n \\
\langle \Gamma \triangleright (\nu a:W)P \rangle_n &= (\nu a:\langle W \rangle)\langle \Gamma, a:W \triangleright P \rangle_n & \langle \Gamma \triangleright \mathbf{0} \rangle_n &= \mathbf{0} \\
\langle \Gamma \triangleright !P \rangle_n &= !\langle \Gamma \triangleright P \rangle_n & \langle \Gamma \triangleright m[P] \rangle_n &= m[\llbracket \Gamma \triangleright P \rrbracket_m] \\
\langle \Gamma \triangleright (x:W)^\uparrow P \rangle_n &= (\nu p : \mathbf{Amb}[\langle W \rangle]) \ p[\mathbf{out}\langle n, \mathbf{pr}_{\langle W \rangle} \rangle.(x:\langle W \rangle)^\uparrow \mathbf{in}\langle n, p \rangle.\langle x \rangle^\uparrow] \mid \\
&\quad \overline{\mathbf{in}}(y, p)(x:\langle W \rangle)^p \langle \Gamma, x:W \triangleright P \rangle_n \\
&\quad (\Gamma(n) = \mathbf{Amb}[E, W]) \quad y \notin \mathbf{fv}(P), p \notin \mathbf{fn}(P) \\
\langle \Gamma \triangleright (x:W)^a P \rangle_n &= (x:\langle W \rangle)^a \langle \Gamma, x:W \triangleright P \rangle_n \\
&\quad (\Gamma(a) = \mathbf{Amb}[W, E]) \\
\langle \Gamma \triangleright \langle M \rangle^\uparrow P \rangle_n &= (\nu p : \mathbf{Amb}[\langle W \rangle]) \ p[\mathbf{out}\langle n, \mathbf{pw}_{\langle W \rangle} \rangle.\langle M \rangle^\uparrow \mathbf{in}\langle n, p \rangle.\langle M \rangle^\uparrow] \mid \\
&\quad \overline{\mathbf{in}}(y, p)(x:\langle W \rangle)^p \langle \Gamma, x:W \triangleright P \rangle_n \\
&\quad x, y \notin \mathbf{fv}(P), p \notin \mathbf{fn}(P) \quad \Gamma(n) = \mathbf{Amb}[E, W]) \\
\langle \Gamma \triangleright \langle M \rangle^a P \rangle_n &= \langle M \rangle^a \langle \Gamma \triangleright P \rangle_n \\
\langle \Gamma \triangleright (x:W)P \rangle_n &= (x:\langle W \rangle)\langle \Gamma, x:W \triangleright P \rangle_n + (x:\langle W \rangle)^\uparrow \langle \Gamma, x:W \triangleright P \rangle_n + \\
&\quad + \overline{\mathbf{out}}(y, \mathbf{pw}_{\langle W \rangle})(x:\langle W \rangle)^y \langle \Gamma, x:W \triangleright P \rangle_n \\
&\quad (\Gamma(n) = \mathbf{Amb}[W, E]) \quad y \notin \mathbf{fv}(P) \\
\langle \Gamma \triangleright \langle M \rangle P \rangle_n &= \langle M \rangle \langle \Gamma \triangleright P \rangle_n + \langle M \rangle^\uparrow \langle \Gamma \triangleright P \rangle_n + \overline{\mathbf{out}}(y, \mathbf{pr}_{\langle W \rangle}) \langle M \rangle^y \langle \Gamma \triangleright P \rangle_n \\
&\quad (\Gamma(n) = \mathbf{Amb}[W, E]) \quad y \notin \mathbf{fv}(P)
\end{aligned}$$

In order to type the processes that come from the translation, we have to add a typing rule for guarded choice:

$$\text{(SUM)} \quad \frac{\Gamma \vdash P : T \quad \Gamma \vdash Q : T}{\Gamma \vdash P + Q : T}$$

The following theorem shows the correspondence between BA's type system of Section 2.4 and the type system for NBA (enriched with rule (SUM) above) presented in Section 5.4.

Theorem 5.5.8 (Soundness of Typing). *If $\Gamma \vdash P : [E, F]$ is derivable in BA, and $\Gamma(n) = \text{Amb}[E, F]$, then $\llbracket \Gamma \rrbracket \vdash \llbracket \Gamma \triangleright P \rrbracket_n : \llbracket [E, F] \rrbracket$ is derivable in NBA.* \square

Chapter 6

Secrecy in Open Networks

We study the problem of protecting migrating agents against the untrusted sites they traverse. We extend the calculus of Chapter 5 with specific primitives; the resulting calculus (CBA), is meant to be a formal framework for reasoning about protection policies and security protocols over distributed, mobile infrastructures, and aims to stand to ambients as the spi calculus stands to π . We further present a sound type system that separates trusted and untrusted data and code, while allowing safe interactions with untrusted sites. We then study a basic *secrecy* property, proving that well typed processes do not disclose their secrets in the presence of opponents. Finally, a few examples and an encoding of the spi calculus show the expressive power of the calculus.

6.1 Introduction

In previous chapters we developed a number of type systems that verify the *internal* security of software systems: such systems assume that all processes, as well as their context, are well typed. However, when working with open networks, we cannot expect that all processes respect a given typing discipline, and we must instead consider that some processes can visit untrusted sites, they can interact with untrusted components and they are subject to attacks. To cope with this situation, we propose a solution in which types are used to reason about programs and ensure their correctness (w.r.t. a security policy), while a weak form of cryptography is employed to protect agents in their interactions with external world.

In the π -calculus secure communication relies on the use of *private channels*, whose

names are restricted. For instance, the process

$$(\nu n)(\bar{n}\langle m \rangle \mid n(x).P)$$

creates a private channel n and uses it for transmitting the name m . Intuitively, this communication is secure since no third process may use and interfere with the channel n . However, in a distributed system, the two subprocesses $\bar{n}\langle m \rangle$ and $n(x).P$ may execute on different machines and the network between these machines may not be physically secure. A secure channel like n should then be implemented by exchanging encrypted messages along a public channel. The π -process above can be expressed in lower-level terms by translating it into the spi-calculus [Aba98], where the use of cryptography is explicit:

$$(\nu n)(\bar{p}\langle \{m\}_n \rangle \mid p(y).case\ y\ of\ \{x\}_n\ in\ P) \quad (1)$$

Here process n still appears under a restriction, but now it is used as an encryption key rather than as a channel. The message m is encrypted using n and the encrypted packet is communicated along a public channel p ; a receiver that knows the key n may then decrypt that packet to obtain m . Even if a third process may steal the message $\{m\}_n$ by reading from the public channel, it has no access to the secret name m since it does not know the key n ¹.

Moving from π calculus to spi calculus, it is important to study the relation between the abstractions that a calculus provides for security (e.g. the restriction operator) and their implementations. A similar analysis can be conducted in the model of Mobile Ambients, where the security of an ambient is based on the privacy of its name: names are unforgeable and their scope must be controlled carefully. When ambients move along the network, the question arises as to whether the privacy of restricted names can indeed be guaranteed by the implementation. The following process implements the idea of cryptography as k -envelope [Car99], that is, encryption under a key k is obtained by putting a message into an ambient whose name k is restricted, and decryption is obtained by opening the ambient k and reading the content.

$$(\nu n)(a[n[out\ a.\ in\ b.\langle m \rangle]] \mid b[open\ n.\langle x \rangle.P]) \quad (2)$$

The two ambients a and b communicate by means of a subambient n that carries the message m from a to b . This protocol corresponds to a π -calculus's private channel,

¹See [Aba98] for a discussion on several weaknesses of the translation, as given.

in that communication relies on a “private agent” n whose name is restricted. As in the π calculus, communication is secure in the sense that no context can discover m by interacting with the process, and no context can cause a different message to be sent to b . On the other hand, if the two ambients a and b are located at remote sites it is difficult to ensure that the name of the ambient n is not revealed as it moves to carry $\langle m \rangle$ from a to b . In particular, current implementations of ambient calculi suffer of this problem. In [FLA00] each ambient contains an Ambient Manager that in turn contains pointers to its children to collect requests of operations; in [SV01] each ambient has a pointer to its parent and it reveals its presence (and its name) when it exposes a (co-)action. In both cases the implementation of a moving ambient n reveals the name n to the ambients it traverses, and the knowledge of the name n is sufficient for an opponent to have full access to n . In other words, as in the π calculus, it is difficult to render the semantics of the restriction operator ν . Even if we do not exclude the possibility of finding a robust implementation that keeps ambient names secret, we argue that reducing the secrecy of an ambient to the secrecy of its name is a risk. We therefore aim at finding more robust solutions.

A closer look to processes (1) and (2) shows a basic difference between the two packets that carry the secret message m : in (2) the wrapper ambient n is an active agent that is much more powerful than the passive encryption $\{m\}_n$ of (1). In fact, one may exploit the power of agents, committing their security to their internal threads; as an example, recasting the process (2) in the model of Safe Ambients we obtain

$$(\nu n)(a[\ n[\text{out } a.\text{in } b.\overline{\text{open}} \ n.\langle m \rangle] \mid \overline{\text{out}} \ a \] \mid b[\overline{\text{in}} \ b.\text{open } n.(x).P]) \quad (3)$$

By simply exhibiting the intended cocapabilities, we may ensure that, even though the name n is revealed, no Trojan horse can enter n nor any malicious process can disclose n before it has reached its final destination. This example suggests that any ambient can subjectively regulate its interactions with the external environment, and protect its content by simply disallowing communications with the untrusted sites it traverses. However, this is not enough since once the name n is revealed, an attacker can use it to masquerade as a malicious process within an ambient that has the trusted name n . In the Safe Ambient model, a solution is to further protect n by putting it in a box that navigates the ambient

n :

$$(\nu n)(a[p[\text{out } a.\text{in } b.\overline{\text{open}} \ p \mid n[\overline{\text{open}} \ n.\langle m \rangle] \mid \overline{\text{out}} \ a.P] \mid b[\overline{\text{in}} \ b.\text{open } p.\text{open } n.(x).Q]]) \\ \rightarrow \dots \rightarrow (\nu n)(a[P] \mid b[\text{open } n.(x).Q \mid n[\overline{\text{open}} \ n.\langle m \rangle]])$$

With this protocol the name n and the message m are protected by the wrapper ambient p that can be opened only at the target site b . A minor problem remains since, due to the communication model of (S)MA, the opening of n and the reading of its content are two distinct steps. This is problematic: consider a more general ambient b that contains more than one thread: $b[\text{open } n.(x).Q \mid Q' \mid n[\overline{\text{open}} \ n.\langle m \rangle]]$, if Q' is also a reader process, nothing ensures that the message m will be read by the leftmost thread that actually knows the secret name n . As we said this is a problem of the communication model of MA, that is solved in Boxed Ambients using the different communication primitives available in our calculus. The process above can be translated in the NBA calculus of Chapter 5 as follows:

$$(\nu n)(a[p[\text{out}\langle a, a \rangle.\text{in}\langle b, b \rangle \mid n[\text{out}\langle p, n \rangle.\langle m \rangle^\dagger] \mid \overline{\text{out}}(x, a) \mid \\ b[\overline{\text{in}}(x, b).\overline{\text{out}}(y, n).(z)^y.Q \mid Q']])$$

This protocol solves the previous problem: m cannot be read by the process Q' , unless Q' knows the secret name n . Furthermore, note that while the carrier ambient p moves, its subambient n is confined within p since only the target ambient b can offer a $\overline{\text{out}}(y, n)$ capability with the right password n . There is a final problem here. In the most general case, the ambient n is not simply a packet driving passive data, but it is a mobile agent containing an active process P . In this case we have to be even more careful, to make sure that P does not drag n out of p before this reaches a trusted site.

To sum up, we observed that an agent, in order to protect its content, should not (only) rely on the privacy of its name, but on powerful mechanisms that filter accesses from external entities. The scientific literature on mobile agent security focuses mainly on the dual problems of protecting a host from incoming agents, and protecting a mobile agent from malicious hosts. Cryptography is used effectively in the latter case, by setting up a network of trusted sites and a mechanism of authentication between such sites and encrypted agents on the move. In general, an agent can encrypt itself (or its critical parts), and the ciphertext is put in a clear-code wrapper that navigates the encrypted payload to the desired destination ([Tsc99, WBS98]). However, such a model disrupts the power of

mobile agents, in that what is sent along the network is just a passive encrypted packet. Other approaches use cryptography to enhance the security of mobile agents by relying on digital signatures to provide for authentication and non-repudiation, or on the use of encrypted functions [ST98].

In this chapter we propose a different, flexible solution for the problems we discussed so far; the idea is similar to that proposed for process (3), that is to let an ambient block its interactions with the external environment. More precisely, we introduce a specific primitive to let an ambient *dynamically* block its external interactions: consider the following reduction:

$$n[\text{lock } k.P \mid Q] \rightarrow n\{P \mid Q\}_k$$

by exercising the capability `lock k` in one of its internal threads, the ambient n blocks all its interactions with the outside, included those contained in the thread Q . The flexibility of this model comes from the fact that a “locked” ambient $n\{P \mid Q\}_k$ is still active: it may then freely move in the network (revealing its name) and perform local computation. Only its external interactions are blocked, and so remain until the ambient reaches a computational environment that knows the key k . The mechanism to unlock interaction is associated to movements and co-capabilities containing keys as, e.g., in

$$n\{\text{in } m.P \mid Q\}_k \mid m[\overline{\text{in}} \{x\}_k.R \mid S] \rightarrow m[n\{P \mid Q\} \mid R\{x := n\} \mid S]$$

The resulting model can be roughly viewed as a model of symmetric cryptography, where keys are managed using cryptographic primitives (`lock k` corresponds to encryption, while co-capabilities correspond to decryption) and an encrypted ambient is an ambient whose access upward channel is blocked. Nevertheless this idea remains at the abstraction level where an ambient is just a model for both a mobile device and a piece of mobile code. More concretely, ‘encrypting’ a mobile device is nothing else than disconnecting it from the network, while ‘decrypting’ means to re-connect it, possibly elsewhere, in order to let the device communicate again with other agents or hosts. On the other hand an ‘encrypted’ code that is still able to move is much more tricky; encrypting an agent can be thought of as disallowing external accesses to its communication buffer and encrypting the messages already contained in that buffer. In this way, if the ‘encrypted’ agent enters a malicious host, no communications may take place, and even if the host tries to disassemble the agent, it can not read those values that have been encrypted with a secret key (as long

as we assume perfect cryptography). This model of symmetric cryptography fits well the capability-based model of security inherited from MA, as cryptographic operations are linked to ambient mobility. However, talking about cryptography may be misleading since no code gets really encrypted: the process inside an 'encrypted' ambient is still active, it is just refusing external accesses. On the other hand, we show that our mechanism of locking/unlocking ambient's external interactions based on the knowledge of a secret key brings the usual benefits (in terms of privacy and integrity) provided in distributed system by cryptography.

Secure communication as in (1) can now be achieved by putting the clear message into an ambient that blocks (encrypts) itself, moves over the network towards its destination, gets unblocked (decrypted) in the act of entering it, and becomes then ready to deliver its message. All this is illustrated by the term below (cf. §6.2 for the details).

$$(\nu k)a[\ n[\text{lock } k.\text{out } a.\text{in } b.\langle M \rangle^\dagger] \] \mid \overline{\text{out}} \mid b[\overline{\text{in}} \{x\}_k.(y)^x.P \mid Q]$$

Another key-point of this chapter is the development of a sound type system (see Section 6.3) that takes into account untrusted processes (*opponents*) interacting with trusted components that move along open networks. The type system implements the idea that world is divided in two parts: a trusted *system* and an untrusted *context*. Relying on this intuition, an “unknown” type **Public** is assigned to anything that comes from the external environment; public values are then handled very carefully since there is no precise knowledge of what a particular piece of data they are. This type system is then the basis for the study of *secrecy* property of CBA: in Section 6.4 we prove that a well typed process P do not disclose its secrets to any adversary context that knows public names of P and that may traverse the hierarchy of subambients contained in P . Even if this type system deals with untrusted networks, a similar technique can be used to generalize the type systems presented in Chapter 3 and in Chapter 4 to study access control and information flow security in presence of untrusted components.

The chapter ends with an example and an encoding of the spi calculus into CBA that represent a starting point for a deeper comparison between our approach and a more standard use of cryptography.

6.2 The CBA calculus

CBA are a variant of the calculus we presented in Chapter 5. Basically, the passwords that in Chapter 5 were used to authorize ambient movements, become here locking keys used to lock and unlock ambients carrying confidential data. Such a difference is reflected both in the semantics and in the syntax of movement cocapabilities: instead of $\overline{\text{in}}(x, k), \overline{\text{out}}(x, k)$ we write $\overline{\text{in}}\{x\}_k, \overline{\text{out}}\{x\}_k$ in analogy with the spi-calculus syntax. Moreover, since we do not want to enforce the authentication/unlocking mechanism on *every* movement, CBA recovers standard $\text{in } a, \text{out } a, \overline{\text{in}}, \overline{\text{out}}$ capabilities. Then two forms of cocapabilities, $\overline{\text{in}}, \overline{\text{out}}$ and $\overline{\text{in}}\{x\}_k, \overline{\text{out}}\{x\}_k$, allow two kinds of movements: usual $\text{in} - \overline{\text{in}}$ ($\text{out} - \overline{\text{out}}$) synchronizations provide for standard mobility, while $\overline{\text{in}}\{x\}_k, \overline{\text{out}}\{x\}_k$ are used to model a more complicate access protocol where the incoming agent is unlocked and its name is registered in the new computation environment. Note that two distinct forms of capabilities are necessary since standard capabilities can be only partially simulated with NBA's capabilities, as we will see in Section 6.2.2.

6.2.1 Syntax

The syntax of the calculus is defined by the following productions, where we assume that names range over m, n, \dots , variables range over x, y, \dots and $a, b \dots$ are used for both names and variables.

<i>Expressions</i>	M, N	$::=$	$k, m - q,$	names
			$ $	$x - z$ variables
			$ $	$\text{in } M$ enter M
			$ $	$\text{out } M$ exit M
			$ $	$\overline{\text{in}}$ let enter
			$ $	$\overline{\text{out}}$ let exit
			$ $	$M.M$ path

<i>Locations</i>	η	$::=$	a	names and variables
			$ $	\uparrow enclosing ambient
			$ $	\star local

<i>Prefixes</i>	π	$::=$	M	path
			$(x_1, \dots, x_k)^\eta$	input
			$\langle M_1, \dots, M_k \rangle^\eta$	output
			$\overline{\text{in}} \{x\}_M$	unlock entering ambient
			$\overline{\text{out}} \{x\}_M$	unlock exiting ambient
			$\text{lock } M$	locking
<i>Processes</i>	P	$::=$	$\mathbf{0}$	stop
			$\pi.P$	prefixing
			$(\nu n)P$	restriction
			$P \mid P$	composition
			$M[P]$	plain ambient
			$M\{P\}_M$	locked ambient
			$!\pi.P$	replication

An ambient may have two states: plain ambient, $a[P]$, or locked ambient, $a\{P\}_k$, that is an ambient whose access channel is locked (encrypted) with the key k .

In CBA there are three new processes that deal with the locking mechanism. Two movement cocapabilities $\overline{\text{in}} \{x\}_k.P$, $\overline{\text{out}} \{x\}_k.P$ authenticate, unlock and learn the name of the incoming ambient locked with key k . An additional process $\text{lock } k.P$ is used to lock the enclosing ambient under the key k . Note that we do not communicate the capability $\text{lock } k$: communicating only the locking ability on a key k is meaningless if we think locking keys as shared cryptographic keys. The rest of the syntax is standard. In the following we will write $a\{P\}$ to indicate the ambient a when it is not relevant if that ambient is locked or not.

6.2.2 Semantics

The operational semantics of the calculus is given as usual in the form of a reduction relation and a structural congruence. The definition of structural congruence is similar to that in §2.2: it is the least congruence relation that is a commutative monoid for $\mathbf{0}$ and \mid and that is closed under rules of §2.2 plus the following rule:

$$(\text{Struct Res Lock}) \quad (\nu n)a\{P\}_k \equiv a\{(\nu n)P\}_k \quad n \notin \{a, k\}$$

Structural congruence is connected to the reduction relation via the standard (Red Struct) rule:

$$(Red\ Struct) \quad P \equiv Q, Q \rightarrow R, R \equiv S \Rightarrow P \rightarrow S$$

The reduction relation \rightarrow is defined as the least relation on processes closed under parallel composition, restriction, and ambient operator (both locked or not, i.e. $a\{P\} \rightarrow a\{Q\}$ if $P \rightarrow Q$) which satisfies the following rules:

Mobility (I)

$$\begin{aligned} (enter) \quad & n\{\text{in } m.P \mid Q\} \mid m\{\overline{\text{in}} .R \mid S\} \rightarrow m\{n\{P \mid Q\} \mid R \mid S\} \\ (exit) \quad & n\{m\{\text{out } n.P \mid Q\} \mid R\} \mid \overline{\text{out}} .S \rightarrow m\{P \mid Q\} \mid n\{R\} \mid S \end{aligned}$$

Mobility (II) and Locking

$$\begin{aligned} (K-enter) \quad & n\{\text{in } m.P \mid Q\} \mid m\{\overline{\text{in}} \{x\}_k .R \mid S\} \rightarrow m\{n[P \mid Q] \mid R\{x := n\} \mid S\} \\ (K-exit) \quad & n\{P \mid m\{\text{out } n.Q \mid R\}\} \mid \overline{\text{out}} \{x\}_k .S \rightarrow n\{P\} \mid m[Q \mid R] \mid S\{x := m\} \\ (lock) \quad & n[\text{lock } k.P \mid Q] \rightarrow n\{P \mid Q\}_k \end{aligned}$$

Communication

$$\begin{aligned} (local) \quad & (\tilde{x})P \mid \langle \tilde{M} \rangle Q \rightarrow P\{\tilde{x} := \tilde{M}\} \mid Q \\ (input\ n) \quad & (\tilde{x})^n P \mid n[\langle \tilde{M} \rangle^\dagger Q \mid R] \rightarrow P\{\tilde{x} := \tilde{M}\} \mid n[Q \mid R] \\ (output\ n) \quad & \langle \tilde{M} \rangle^n P \mid n[(\tilde{x})^\dagger Q \mid R] \rightarrow P \mid n[Q\{\tilde{x} := \tilde{M}\} \mid R] \end{aligned}$$

The two rules *(enter)* and *(exit)* allow any ambient, locked or not, to traverse any other ambient, locked or not. The only constraint for such a movement is that the target ambient must explicitly cooperate by offering a cocapability.

An alternative mechanisms for mobility, analogous to that of NBA (cf. §5.2), is represented by the rules *(K-enter)*, *(K-exit)*. As in Chapter 5, with those two rules the target ambients learn the names of incoming ambients. However, the use of keys is different from that of passwords of Section 5.2. In Chapter 5 the passwords contained in (co-)capabilities

provide ambients with an access control mechanism where the credentials of the incoming ambients are checked as a preliminary step to the registration protocol. Here the access protocol is even more complicate: besides checking the credential of an incoming ambient and learning its name, rules $(K\text{-}enter), (K\text{-}exit)$ allow the unlocking of an ambient when it arrives in a computational environment that knows its locking key.

To sum up, in CBA agents may freely move using standard rules $(enter)$ and $(exit)$; in addition, a more complex access protocol is used to deal with locked agents. With such a protocol we have that in a single reduction step (i) the target site recognizes the locking key of the locked ambient, (ii) the moving ambient is first unlocked and then unleashed at the target site, (iii) the new computational environment gets to know the name of the incoming ambient and can use it for communications. Note that the use of $\overline{\text{out}}$ placed in parallel with the exited ambient is fundamental, since it regulates the entrance in the surrounding computational environment.

Finally, rule $(lock)$ shows the dual unlocking operation: the capability lock k instructs a process to lock its enclosing ambient under a key k . Rule $(lock)$, together with the syntax of processes, state that ambients cannot be locked more than once. This requirement matches the idea that locking an ambient means to block its access channel.

An alternative presentation of the calculus could be given by separating the locking primitives from those for mobility. Specifically, one could introduce an explicit unlocking prefix such as $\text{unlock}\{x\}_k.P$, and define its semantics by the reduction

$$\text{unlock}\{x\}_k.P \mid n \llbracket Q \rrbracket_k \rightarrow P\{x := n\} \mid n[Q]$$

This rule, together with rules $(enter)$ and $(exit)$ would implement an equivalent (albeit not atomic) unlocking mechanism. However, our present solution appears to be more faithful to the current practice in distributed and mobile systems, in which the protocols for agent authentication and certification take place at domain (i.e. ambient) boundaries rather than after such boundaries have been crossed.

As for communication, CBA inherits the communication model of NBA. Furthermore, locked ambients do not communicate: this reflects the idea that a locked ambient is an ambient whose access channel is blocked, and exchanges are allowed only in a computational environment that recognizes (authenticates) and unlocks that ambient.

6.3 Type System for partial typing

In this section we develop the type system of CBA. The type system provides for partial typing, that is it takes into account untrusted, ill-formed processes, letting them interact with trusted components. In particular, a distinct type, Un , is used to type any public process that is potentially ill-formed and that does not respect any security discipline. The type system then implements the idea that world is divided in two parts: a disciplined, trusted, *system* and an untrusted *opponent*. Relying on this intuition, we assign an “unknown” type Public to any data that comes from the untrusted world and we are very careful in handling such values since we have no knowledge of what a particular piece of data they are.

In addition to this security-related purpose, our type system has also the descriptive nature of the previous type systems for Boxed Ambients, in that types reflect the intended usage of some piece of data. Nevertheless, we cannot assume that (terms containing other) terms of type Public are well formed, essentially because we cannot expect that the opponent obeys our term formation rules. As a consequence, terms like $\text{in out } a$ or $\overline{\text{in}} [P]$ will not be ruled out by the type system.

Finally, differently from the previous chapters, we do not use a typed syntax for CBA. This is a key point when modeling distributed systems, where we cannot assume that external processes, especially the untrusted ones, come equipped with type annotations belonging to a single, locally trusted type system.

6.3.1 Type System

The structure of types for CBA is the following:

<i>Expression Types</i>	W	$::=$	$\text{Amb}[E]$	ambient
			$ $ $\text{Key}[E]$	key
			$ $ Public	unknown
<i>Exchange Types</i>	E, F	$::=$	shh	no exchange
			$ $ $W_1 \times \dots \times W_k$	tuple exchange
<i>Process Types</i>	T	$::=$	$[E, F]$	process
			$ $ Un	untrusted process

As we said above, the new type **Public** is used to type public expressions. In particular, since here capabilities do not contain passwords as in Chapter 5, they can be considered public values (as in Chapter 4). Then expressions like in a or in $a.out\ b.\overline{in}$ are of type **Public**. Ambient types instead denote “secret”, i.e. trusted, ambients. A new type $\text{Key}[E]$ is the type of a locking key that can be used to lock ambients of type $\text{Amb}[E]$. Notice that, according to our model, only ambients (and no generic expressions) can be locked. Furthermore, types $\text{Key}[E]$ are secret values.

As for process types, $[E, F]$ is the type of processes enclosed in a trusted ambient, while **Un** is the type of any untrusted process. As we said above, the type **Un** is used to represent potential attackers that may interact with trusted processes. We proceed with the presentation of the typing rules, starting with rules for valid type environments:

$$\begin{array}{c} \text{(EMPTY)} \\ \hline \emptyset \vdash \diamond \end{array} \quad \begin{array}{c} \text{(ENV } x) \\ \Gamma \vdash \diamond \quad x \notin \text{Dom}(\Gamma) \\ \hline \Gamma, x : W \vdash \diamond \end{array}$$

There is no subtyping: trusted and untrusted components must be kept distinct. Nevertheless the typing rules for processes allow non trivial forms of interactions between “public” and “secret” processes.

Typing of Expressions

$$\begin{array}{c} \text{(PROJECTION)} \\ \Gamma \vdash \diamond \quad \Gamma(M) = W \\ \hline \Gamma \vdash M : W \end{array} \quad \begin{array}{c} \text{(PATH)} \\ \Gamma \vdash M_1 : \text{Public} \quad \Gamma \vdash M_2 : \text{Public} \\ \hline \Gamma \vdash M_1.M_2 : \text{Public} \end{array}$$

$$\begin{array}{c} \text{(CO-IN)} \\ \Gamma \vdash \diamond \\ \hline \Gamma \vdash \overline{in} : \text{Public} \end{array} \quad \begin{array}{c} \text{(CO-OUT)} \\ \Gamma \vdash \diamond \\ \hline \Gamma \vdash \overline{out} : \text{Public} \end{array}$$

$$\begin{array}{c} \text{(IN } M) \\ \Gamma \vdash M : W \quad W \in \{\text{Amb}[E], \text{Public}\} \\ \hline \Gamma \vdash in\ M : \text{Public} \end{array} \quad \begin{array}{c} \text{(OUT } M) \\ \Gamma \vdash M : W \quad W \in \{\text{Amb}[E], \text{Public}\} \\ \hline \Gamma \vdash out\ M : \text{Public} \end{array}$$

In the typing rules above every (co-)capability is assigned type **Public**; this fact, together with rule (PREFIX), allows trusted ambients to traverse untrusted ones and vice versa, without breaking the soundness of the type system. Note that rule (PATH) can be used to type also ill-formed path as $a.b$, provided that a, b are **Public** values. Similarly, rule (IN M) allows a ill-formed capability in $(a.b)$ to be exercised within a trusted ambient. As we said above, this type system allows the formation of meaningless terms; this is not a problem as long as malformed terms contained within trusted processes do not violate the security of trusted systems. In the following Section we prove that this is exactly what happens.

Typing of Processes

The basic idea of typing rules for processes is that for each process constructor there are two possibilities: either (i) it is a trusted process that respects the security discipline, or (ii) it is an untrusted process: it could be an attacker or a trusted process tainted by an interaction with an untrusted component via its public names. For prefixes the two cases can be summarized in a single rule, where T stands for either $[E, F]$ or **Un**:

$$\frac{\text{(PREFIX)} \quad \Gamma \vdash M : \mathbf{Public} \quad \Gamma \vdash P : T}{\Gamma \vdash M.P : T}$$

There are four rules to type ambients. Rule (AMB LOCK) is used to type a correctly locked ambient: the key N is used to lock an ambient of the expected ambient type, and the process P has a corresponding “trusted” process type. Rule (AMB) is standard, while the rules (UNTRUSTED AMB/AMB LOCK) are used to type untrusted, possibly ill-formed, ambients.

$$\begin{array}{c} \text{(AMB)} \\ \frac{\Gamma \vdash M : \mathbf{Amb}[E] \quad \Gamma \vdash P : [F, E]}{\Gamma \vdash M[P] : T} \end{array} \qquad \begin{array}{c} \text{(UNTRUSTED AMB)} \\ \frac{\Gamma \vdash M : \mathbf{Public} \quad \Gamma \vdash P : \mathbf{Un}}{\Gamma \vdash M[P] : T} \end{array}$$

$$\begin{array}{c} \text{(AMB LOCK)} \\ \frac{\Gamma \vdash N : \mathbf{Key}[E] \quad \Gamma \vdash M : \mathbf{Amb}[E] \quad \Gamma \vdash P : [F, E]}{\Gamma \vdash M \{ P \}_N : T} \end{array}$$

$$\begin{array}{c}
\text{(UNTRUSTED AMB LOCK)} \\
\frac{\Gamma \vdash N : \text{Public} \quad \Gamma \vdash M : \text{Public} \quad \Gamma \vdash P : \text{Un}}{\Gamma \vdash M \{ P \}_N : T}
\end{array}$$

There are two ways to type the locking operation $\text{lock } M$, depending on the trust level of the key. If M is public, then the process $\text{lock } M.P$ is subject to attacker's interactions, thus it must be typed as an unsafe process with type Un (rule (UNTRUSTED LOCK)). On the other hand, rule (LOCK) handles the case where M is a secret key of type $\text{Key}[E]$, where E corresponds to the type of upward exchanges of the ambient that will be locked with the key M .

$$\begin{array}{c}
\text{(LOCK)} \\
\frac{\Gamma \vdash M : \text{Key}[E] \quad \Gamma \vdash P : [F, E]}{\Gamma \vdash \text{lock } M.P : [F, E]}
\end{array}
\qquad
\begin{array}{c}
\text{(UNTRUSTED LOCK)} \\
\frac{\Gamma \vdash M : \text{Public} \quad \Gamma \vdash P : \text{Un}}{\Gamma \vdash \text{lock } M.P : \text{Un}}
\end{array}$$

The following rules (CO-IN KEY), (CO-OUT KEY) are used to unlock an ambient that enters a computational environment that knows its secret locking key M . When M is a secret key, we assume that the content of the locked text is sensitive, thus the continuation process P , as well as the unlocked ambient, must be enclosed into a trusted ambient. On the other hand if M is a public value, no assumptions can be made on the process that uses such a key, or on the ambient that gets unlocked; thus rules (UNTRUSTED CO-IN), (UNTRUSTED CO-OUT) infer the unstructured type Un .

$$\begin{array}{c}
\text{(CO-IN KEY)} \\
\frac{\Gamma \vdash M : \text{Key}[E] \quad \Gamma, x:\text{Amb}[E] \vdash P : [G, H]}{\Gamma \vdash \overline{\text{in}} \{x\}_M.P : [G, H]}
\end{array}
\qquad
\begin{array}{c}
\text{(UNTRUSTED CO-IN)} \\
\frac{\Gamma \vdash M : \text{Public} \quad \Gamma, x:\text{Public} \vdash P : \text{Un}}{\Gamma \vdash \overline{\text{in}} \{x\}_M.P : \text{Un}}
\end{array}$$

$$\begin{array}{c}
\text{(CO-OUT KEY)} \\
\frac{\Gamma \vdash M : \text{Key}[E] \quad \Gamma, x:\text{Amb}[E] \vdash P : [G, H]}{\Gamma \vdash \overline{\text{out}} \{x\}_M.P : [G, H]}
\end{array}
\qquad
\begin{array}{c}
\text{(UNTRUSTED CO-OUT)} \\
\frac{\Gamma \vdash M : \text{Public} \quad \Gamma, x:\text{Public} \vdash P : \text{Un}}{\Gamma \vdash \overline{\text{out}} \{x\}_M.P : \text{Un}}
\end{array}$$

The following rules are standard, just note that since we use an untyped syntax, in the hypothesis of rule (NEW) we choose an appropriate expression type for the name n .

$$\begin{array}{c}
\text{(DEAD)} \\
\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0} : T}
\end{array}
\quad
\begin{array}{c}
\text{(PAR)} \\
\frac{\Gamma \vdash P : T \quad \Gamma \vdash Q : T}{\Gamma \vdash P \mid Q : T}
\end{array}
\quad
\begin{array}{c}
\text{(NEW)} \\
\frac{\Gamma, n : W \vdash P : T}{\Gamma \vdash (\nu n)P : T}
\end{array}
\quad
\begin{array}{c}
\text{(REPL)} \\
\frac{\Gamma \vdash P : T}{\Gamma \vdash !P : T}
\end{array}$$

As for communication, local and upward exchanges are handled similarly. If the exchanges occur in an untrusted environment, the (UNTRUSTED ...) rules below ensure that only public, unstructured, values are exchanged (this is a strong requirement but it is necessary for soundness). On the other hand, standard communication rules apply to the case where the exchanges happen within trusted ambients.

$$\begin{array}{c}
\text{(LOCAL INPUT)} \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [W_1 \times \dots \times W_k, E]}{\Gamma \vdash (x_1, \dots, x_k)P : [W_1 \times \dots \times W_k, E]}
\end{array}
\quad
\begin{array}{c}
\text{(UNTRUSTED LOCAL INPUT)} \\
\frac{\Gamma, \tilde{x} : \mathbf{Public} \vdash P : \mathbf{Un}}{\Gamma \vdash (x_1, \dots, x_k)P : \mathbf{Un}}
\end{array}$$

$$\begin{array}{c}
\text{(LOCAL OUTPUT)} \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [W_1 \times \dots \times W_k, E]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle P : [W_1 \times \dots \times W_k, E]}
\end{array}
\quad
\begin{array}{c}
\text{(UNTRUSTED LOCAL OUTPUT)} \\
\frac{\Gamma \vdash M_i : \mathbf{Public} \quad i = 1, \dots, k \quad \Gamma \vdash P : \mathbf{Un}}{\Gamma \vdash \langle M_1, \dots, M_k \rangle P : \mathbf{Un}}
\end{array}$$

$$\begin{array}{c}
\text{(INPUT } \uparrow) \\
\frac{\Gamma, \tilde{x} : \tilde{W} \vdash P : [E, W_1 \times \dots \times W_k]}{\Gamma \vdash (x_1, \dots, x_k)^\uparrow P : [E, W_1 \times \dots \times W_k]}
\end{array}
\quad
\begin{array}{c}
\text{(UNTRUSTED INPUT } \uparrow) \\
\frac{\Gamma, \tilde{x} : \mathbf{Public} \vdash P : \mathbf{Un}}{\Gamma \vdash (x_1, \dots, x_k)^\uparrow P : \mathbf{Un}}
\end{array}$$

$$\begin{array}{c}
\text{(OUTPUT } \uparrow) \quad i = 1, \dots, k \\
\frac{\Gamma \vdash M_i : W_i \quad \Gamma \vdash P : [E, W_1 \times \dots \times W_k]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^\uparrow P : [E, W_1 \times \dots \times W_k]}
\end{array}
\quad
\begin{array}{c}
\text{(UNTRUSTED OUTPUT } \uparrow) \\
\frac{\Gamma \vdash M_i : \mathbf{Public} \quad i = 1, \dots, k \quad \Gamma \vdash P : \mathbf{Un}}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^\uparrow P : \mathbf{Un}}
\end{array}$$

Three typing rules handle input (output) from a subambient M , depending on the type of M . Rule (INPUT M AMB) is used for communications between two trusted ambients. Rule (INPUT M UNTRUSTED) says that both trusted and untrusted ambients may read *public* values from an untrusted subambient. Finally, rule (UNTRUSTED INPUT M) states that even untrusted processes may access trusted ambients, but only to read values of type *Public*. Similar rules hold for the output case. In addition, in these rules we do not require that the arity of the downward communication matches that of the ambient M ; this again allows a flexible typing of opponent processes.

$$\begin{array}{c}
(\text{INPUT } M \text{ AMB}) \\
\frac{\Gamma \vdash M : \text{Amb}[W_1 \times \dots \times W_k] \quad \Gamma, \tilde{x} : \tilde{W} \vdash P : [E, F]}{\Gamma \vdash (x_1, \dots, x_k)^M P : [E, F]}
\end{array}$$

$$\begin{array}{c}
(\text{INPUT } M \text{ UNTRUSTED}) \\
\frac{\Gamma \vdash M : \text{Public} \quad \Gamma, \tilde{x} : \tilde{\text{Public}} \vdash P : T}{\Gamma \vdash (x_1, \dots, x_k)^M P : T}
\end{array}$$

$$\begin{array}{c}
(\text{UNTRUSTED INPUT } M) \\
\frac{\Gamma \vdash M : \text{Amb}[\text{Public}_1 \times \dots \times \text{Public}_n] \quad \Gamma, \tilde{x} : \tilde{\text{Public}} \vdash P : \text{Un}}{\Gamma \vdash (x_1, \dots, x_k)^M P : \text{Un}}
\end{array}$$

$$\begin{array}{c}
(\text{OUTPUT } M \text{ AMB}) \\
\frac{\Gamma \vdash M : \text{Amb}[W_1 \times \dots \times W_k] \quad \Gamma \vdash M_i : W_i \quad i = 1, \dots, k \quad \Gamma \vdash P : [E, F]}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^M P : [E, F]}
\end{array}$$

$$\begin{array}{c}
(\text{OUTPUT } M \text{ UNTRUSTED}) \\
\frac{\Gamma \vdash M : \text{Public} \quad \Gamma \vdash M_i : \text{Public} \quad i = 1, \dots, k \quad \Gamma \vdash P : T}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^M P : T}
\end{array}$$

$$\begin{array}{c}
(\text{UNTRUSTED OUTPUT } M) \\
\frac{\Gamma \vdash M : \text{Amb}[\text{Public}_1 \times \dots \times \text{Public}_n] \quad \Gamma \vdash M_i : \text{Public} \quad i = 1, \dots, k \quad \Gamma \vdash P : \text{Un}}{\Gamma \vdash \langle M_1, \dots, M_k \rangle^M P : \text{Un}}
\end{array}$$

Trusted versus Untrusted Processes

Allowed interactions between trusted and untrusted components can be summarized as follows:

- ▷ *Mobility* is totally free: using the reduction rules (*enter*), (*exit*) trusted ambients may traverse untrusted ones and vice versa.
- ▷ *Unlocking* is obviously disallowed: no trusted ambient can be unlocked within an untrusted one and vice versa.

- ▷ *Communication* is restricted:
 - ▷ Local exchanges between trusted and untrusted processes are disallowed.
 - ▷ Trusted ambients may communicate **Public** values to untrusted ones.
 - ▷ Untrusted ambients may communicate **Public** values to trusted ones.

Properties of the Type System

As we said above, the type system we presented can be used for reasoning about processes containing both trusted and untrusted components. The following proposition corresponds to the usual “typability lemma”, stating that any process P can be typed with type **Un** assuming that its names and variables are of type **Public**.

Proposition 6.3.1 (Typability).

1. For all M , if $fn(M) = \{a_1, \dots, a_n\}$ and $fv(M) = \{x_1, \dots, x_m\}$, then
 $a_1 : \text{Public}, \dots, a_n : \text{Public}, x_1 : \text{Public}, \dots, x_m : \text{Public} \vdash M : \text{Public}.$
2. For all P , if $fn(P) = \{a_1, \dots, a_n\}$ and $fv(P) = \{x_1, \dots, x_m\}$, then
 $a_1 : \text{Public}, \dots, a_n : \text{Public}, x_1 : \text{Public}, \dots, x_m : \text{Public} \vdash P : \text{Un}.$

Proof. By induction on the structure of M or P . □

Proposition 6.3.1 will be essential in the following section, where we use types to study the secrecy properties of systems that mix trusted processes and untrusted opponents.

We end this section with the proof that the type system is sound.

Lemma 6.3.2.

1. Subject Congruence

- (a) If $\Gamma \vdash P : T$ and $P \equiv Q$ then $\Gamma \vdash Q : T$.
- (b) If $\Gamma \vdash P : T$ and $Q \equiv P$ then $\Gamma \vdash Q : T$.

2. Substitution

Assume $\Gamma, x : W, \Gamma' \vdash P : T$. For any N , if $\Gamma, \Gamma' \vdash N : W$,
 then $\Gamma, \Gamma' \vdash P\{x := N\} : T$.

Proof. Standard □

Proposition 6.3.3 (Subject Reduction). *If $\Gamma \vdash P : T$ and $P \rightarrow Q$, then $\Gamma \vdash Q : T$.*

Proof. By induction on the derivation of $P \rightarrow Q$ and a case analysis on the last rule applied.

(enter) In this case we have that $P = a\{\text{in } b.P' \mid Q'\} \mid b\{\overline{\text{in}}.R \mid S\}$ and $Q = b\{R \mid S \mid a[P' \mid Q']\}$. Then $\Gamma \vdash P : T$ comes from (i) $\Gamma \vdash a\{\text{in } b.P' \mid Q'\} : T$ and (ii) $\Gamma \vdash b\{\overline{\text{in}}.R \mid S\} : T$. We have to distinguish a number of subcases, depending on the typing rules used to type ambients a and b .

- ▷ Consider the case where (i) and (ii) come by (AMB LOCK). Then (i) must have been derived from $\Gamma \vdash k : \text{Key}[E_1]$, $\Gamma \vdash a : \text{Amb}[E_1]$ and $\Gamma \vdash \text{in } b.P' \mid Q' : [F_1, E_1]$. From the last judgment we have $\Gamma \vdash Q' : [F_1, E_1]$ and $\Gamma \vdash \text{in } b.P' : [F_1, E_1]$. The last judgment must have been derived by (PREFIX) from $\Gamma \vdash \text{in } b : \text{Public}$ and $\Gamma \vdash P' : [F_1, E_1]$. Then from $\Gamma \vdash P' : [F_1, E_1]$, $\Gamma \vdash Q' : [F_1, E_1]$, $\Gamma \vdash k : \text{Key}[E_1]$ and $\Gamma \vdash a : \text{Amb}[E_1]$, by (PAR) and (AMB LOCK) we have $\Gamma \vdash a\{P' \mid Q'\}_{\mathbb{I}_k} : [F_2, E_2]$. With a similar argument we have that (ii) comes from $\Gamma \vdash k_1 : \text{Key}[E_2]$, $\Gamma \vdash b : \text{Amb}[E_2]$, $\Gamma \vdash R : [F_2, E_2]$ and $\Gamma \vdash S : [F_2, E_2]$. Then by (PAR) $\Gamma \vdash R \mid S \mid a\{P' \mid Q'\}_{\mathbb{I}_k} : [F_2, E_2]$ and we conclude by (AMB LOCK).
- ▷ Consider the case where (i) comes by (AMB LOCK) and (ii) comes by (UNTRUSTED AMB). Now, as in the previous case, (i) must have been derived from $\Gamma \vdash k : \text{Key}[E_1]$, $\Gamma \vdash a : \text{Amb}[E_1]$, $\Gamma \vdash P' : [F_1, E_1]$ and $\Gamma \vdash Q' : [F_1, E_1]$. Then by (PAR) and (AMB LOCK) we have $\Gamma \vdash a\{P' \mid Q'\}_{\mathbb{I}_k} : \text{Un}$. On the other hand, (ii) comes from $\Gamma \vdash b : \text{Public}$ and $\Gamma \vdash \overline{\text{in}}.R \mid S : \text{Un}$. The last judgment comes from $\Gamma \vdash S : \text{Un}$ and $\Gamma \vdash \overline{\text{in}}.R : \text{Un}$, that must have been derived by (PREFIX) from $\Gamma \vdash \overline{\text{in}} : \text{Public}$ and $\Gamma \vdash R : \text{Un}$. Now, by (PARA) we have $\Gamma \vdash R \mid S \mid a\{P' \mid Q'\}_{\mathbb{I}_k} : \text{Un}$ and we conclude by (UNTRUSTED AMB).
- ▷ The other cases follow similarly.

(exit) In this case we have that $P = a\{b\{\text{out } a.P' \mid Q'\} \mid R\} \mid \overline{\text{out}}.S$ and $Q = b\{P' \mid Q'\} \mid a\{R\} \mid S$. We have to distinguish a number of cases, depending on the typing rules used to type ambients a and b . We show just a case, where both ambients are typed with (UNTRUSTED AMB), the other cases follow similarly. The hypothesis

$\Gamma \vdash P : T$ comes from $\Gamma \vdash a[b[\text{out } a.P' \mid Q'] \mid R] : T$ and $\Gamma \vdash \overline{\text{out}}.S : T$. If the first judgment has been derived by (UNTRUSTED AMB), we also have $\Gamma \vdash a : \text{Public}$ and $\Gamma \vdash b[\text{out } a.P' \mid Q'] \mid R : \text{Un}$, that implies $\Gamma \vdash b[\text{out } a.P' \mid Q'] : \text{Un}$ and $\Gamma \vdash R : \text{Un}$. Then by (UNTRUSTED AMB) we have $\Gamma \vdash a[R] : T$. Now, if also $b[\dots]$ comes by (UNTRUSTED AMB), we have $\Gamma \vdash b : \text{Public}$, $\Gamma \vdash \text{out } a.P' \mid Q' : \text{Un}$. From the last judgment we have $\Gamma \vdash P' : \text{Un}$ and $\Gamma \vdash Q' : \text{Un}$, thus also $\Gamma \vdash P' \mid Q' : \text{Un}$ by (PAR). By (UNTRUSTED AMB) we have then $\Gamma \vdash b[P' \mid Q'] : T$. Finally, from $\Gamma \vdash \overline{\text{out}}.S : T$, by (PREFIX), we have $\Gamma \vdash S : T$, that, together with $\Gamma \vdash b[P' \mid Q'] : T$ and $\Gamma \vdash a[R] : T$, by (PAR), gives $\Gamma \vdash Q : T$ as desired.

(K-enter) In this case we have that $P = a\{\text{in } b.P' \mid Q'\}_{k'} \mid b\{\overline{\text{in}} \{x\}_k.R \mid S\}$ and $Q = b\{R\{x := a\} \mid S \mid a[P' \mid Q']\}$. Then the hypothesis $\Gamma \vdash P : T$ comes from (i) $\Gamma \vdash a\{\text{in } b.P' \mid Q'\}_{k'} : T$ and (ii) $\Gamma \vdash b\{\overline{\text{in}} \{x\}_k.R \mid S\} : T$. We have to distinguish a number of cases. We show just a case, as representative. We assume that $\Gamma \vdash a\{\dots\}_{k'} : T$ comes by rule (AMB LOCK) and $\Gamma \vdash b[\dots] : T$ by rule (AMB). Then (i) must have been derived from $\Gamma \vdash k : \text{Key}[E_1]$, $\Gamma \vdash a : \text{Amb}[E_1]$ and $\Gamma \vdash \text{in } b.P' \mid Q' : [F_1, E_1]$. From the last one we have $\Gamma \vdash Q' : [F_1, E_1]$ and $\Gamma \vdash \text{in } b.P' : [F_1, E_1]$. The last judgment must have been derived by (PREFIX) from $\Gamma \vdash \text{in } b : \text{Public}$ and $\Gamma \vdash P' : [F_1, E_1]$. Then from $\Gamma \vdash P' : [F_1, E_1]$, $\Gamma \vdash Q' : [F_1, E_1]$, $\Gamma \vdash k : \text{Key}[E_1]$ and $\Gamma \vdash a : \text{Amb}[E_1]$, by (PAR) and (AMB LOCK) we have $\Gamma \vdash a[P' \mid Q'] : [F_2, E_2]$. On the other hand, (ii) comes from $\Gamma \vdash b : \text{Amb}[E_2]$, $\Gamma \vdash S : [F_2, E_2]$ and $\Gamma \vdash \overline{\text{in}} \{x\}_k.R : [F_2, E_2]$. The last judgment must have been derived by (CO-IN KEY) from $\Gamma \vdash k : \text{Key}[E_1]$ and $\Gamma, x : \text{Amb}[E_1] \vdash R : [F_2, E_2]$, that, together with $\Gamma \vdash a : \text{Amb}[E_1]$, by Substitution Lemma 6.3.2(2), implies $\Gamma \vdash R\{x := a\} : [F_2, E_2]$. Now, by (PAR), we have $\Gamma \vdash R\{x := a\} \mid S \mid a[P' \mid Q'] : [F_2, E_2]$, and we conclude by (AMB).

The remaining cases are similar, except for a number of cases that are vacuous, namely, when one of the two ambients $b\{\dots\}$ and $a\{\dots\}_{k'}$ is typed with (UNTRUSTED AMB/AMB LOCK) and the other with a different rule. As an example, if $a\{\dots\}_{k'}$ is typed with rule (AMB LOCK), it follows $\Gamma \vdash k : \text{Key}[E]$; now, if $b\{\dots\}_{k'}$ comes instead by (UNTRUSTED AMB LOCK), we have that $\Gamma \vdash \overline{\text{in}} \{x\}_k.R : \text{Un}$ is a derivable judgment, that must be derived by (UNTRUSTED CO-IN) from $\Gamma \vdash k : \text{Public}$, hence a contradiction arises. The other vacuous cases follow similarly.

(K-exit) Similar to case (K-enter).

(lock) In this case we have that $P = a[\text{lock } k.R \mid S]$ and $Q = a\{\!| R \mid S |\!\}_k$. We have to distinguish two subcases:

- ▷ The hypothesis $\Gamma \vdash P : T$ have been derived by rule (AMB) from $\Gamma \vdash a : \text{Amb}[E]$ and $\Gamma \vdash \text{lock } k.R \mid S : [F, E]$. The last judgment must have been derived by (PARA) from $\Gamma \vdash S : [F, E]$ and $\Gamma \vdash \text{lock } k.R : [F, E]$. The last one, in turn, must have been derived by (LOCK) from $\Gamma \vdash k : \text{Key}[E]$ and $\Gamma \vdash R : [F, E]$. Now, we conclude by (PARA) and (AMB LOCK) from $\Gamma \vdash k : \text{Key}[E]$, $\Gamma \vdash a : \text{Amb}[E]$, $\Gamma \vdash R : [F, E]$ and $\Gamma \vdash S : [F, E]$.
- ▷ The hypothesis $\Gamma \vdash P : T$ have been derived by rule (UNTRUSTED AMB) from $\Gamma \vdash a : \text{Public}$ and $\Gamma \vdash \text{lock } k.R \mid S : \text{Un}$. The last judgment must have been derived by (PARA) from $\Gamma \vdash S : \text{Un}$ and $\Gamma \vdash \text{lock } k.R : \text{Un}$. The last one, in turn, must have been derived by (UNTRUSTED LOCK) from $\Gamma \vdash k : \text{Public}$ and $\Gamma \vdash R : \text{Un}$. Now, from $\Gamma \vdash k : \text{Public}$ and $\Gamma \vdash a : \text{Public}$, together with $\Gamma \vdash R : \text{Un}$ and $\Gamma \vdash S : \text{Un}$, we conclude by (PARA) and (UNTRUSTED AMB LOCK).

(local) In this case we have that $P = (x_1, \dots, x_k)R \mid \langle M_1, \dots, M_k \rangle S$ and $Q = R\{\tilde{x} := \tilde{M}\} \mid S$. We have to distinguish two cases, depending on the type T .

- ▷ If $T = [E, F]$, then the hypothesis must have been derived from (i) $\Gamma \vdash (\tilde{x})R : [E, F]$ and (ii) $\langle \tilde{M} \rangle S : [E, F]$. (i) must have been derived by (LOCAL INPUT) from $E = W_1 \times \dots \times W_k$ and $\Gamma, \tilde{x} : \tilde{W} \vdash R : [E, F]$. (ii) in turn, must have been derived by (LOCAL OUTPUT) from $\Gamma \vdash M_i : W_i, i = 1, \dots, k$ and $\Gamma \vdash S : [E, F]$. Now, by Lemma 6.3.2(2) (Substitution) we have $\Gamma \vdash R\{x_i := M_i\} : [E, F]$, that, together with $\Gamma \vdash S : [E, F]$, by (PARA), gives $\Gamma \vdash Q : T$ as desired.
- ▷ If $T = \text{Un}$, then the hypothesis must have been derived from (i) $\Gamma \vdash (\tilde{x})R : \text{Un}$ and (ii) $\langle \tilde{M} \rangle S : \text{Un}$. (i) must have been derived by (UNTRUSTED LOCAL INPUT) from $\Gamma, x_i : \text{Public} \vdash R : \text{Un}, i = 1, \dots, k$. (ii) in turn, must have been derived by (UNTRUSTED LOCAL OUTPUT) from $\Gamma \vdash M_i : \text{Public}, i = 1, \dots, k$ and $\Gamma \vdash S : \text{Un}$. Now, by Lemma 6.3.2(2)(Substitution) we have $\Gamma \vdash R\{x_i := M_i\} : \text{Un}$, that, together with $\Gamma \vdash S : \text{Un}$, by (PARA), gives $\Gamma \vdash Q : T$ as desired.

(input M) In this case we have that $P = (x_1 \dots, x_k)^a R \mid a[\langle N_1, \dots, N_k \rangle S \mid U]$ and $Q = R\{\tilde{x} := \tilde{N}\} \mid a[S \mid U]$. We have to distinguish two subcases, depending on the type T .

▷ If $T = [E, F]$, then the hypothesis must have been derived from (i) $\Gamma \vdash (x_1 \dots, x_k)^a R : [E, F]$ and (ii) $\Gamma \vdash a[\langle N_1, \dots, N_k \rangle S \mid U] : [E, F]$. Now, (i) can come from one of the following two cases:

▷ $\Gamma \vdash a : \text{Amb}[W_1 \times \dots \times W_k]$ and $\Gamma, \tilde{x} : \tilde{W} \vdash R : [E, F]$. In this case (ii) must have been derived by (AMB) from $\Gamma \vdash a : \text{Amb}[W_1 \times \dots \times W_k]$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S \mid U : [G, W_1 \times \dots \times W_k]$. From the last judgment we have $\Gamma \vdash U : [G, W_1 \times \dots \times W_k]$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S : [G, W_1 \times \dots \times W_k]$, that must have been derived by (OUTPUT \uparrow) from $\Gamma \vdash N_i : W_i$ and $\Gamma \vdash S : [G, W_1 \times \dots \times W_k]$. Now, from $\Gamma, \tilde{x} : \tilde{W} \vdash R : [E, F]$ and $\Gamma \vdash N_i : W_i$, by Lemma 6.3.2(2)(Substitution) we have $\Gamma \vdash R\{\tilde{x} := \tilde{N}\} : [E, F]$. Moreover, from $\Gamma \vdash S : [G, W_1 \times \dots \times W_k]$, $\Gamma \vdash U : [G, W_1 \times \dots \times W_k]$ and $\Gamma \vdash a : \text{Amb}[W_1 \times \dots \times W_k]$, by (AMB) we have $\Gamma \vdash a[S \mid U] : [E, F]$ and we conclude by (PARA).

▷ $\Gamma \vdash a : \text{Public}$ and $\Gamma, x_i : \text{Public} \vdash R : [E, F]$. In this case (ii) must have been derived by (UNTRUSTED AMB) from $\Gamma \vdash a : \text{Public}$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S \mid U : \text{Un}$. From the last judgment we have $\Gamma \vdash U : \text{Un}$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S : \text{Un}$, that must have been derived by (UNTRUSTED OUTPUT \uparrow) from $\Gamma \vdash N_i : \text{Public}$ and $\Gamma \vdash S : \text{Un}$. Now, from $\Gamma, x_i : \text{Public} \vdash R : [E, F]$ and $\Gamma \vdash N_i : \text{Public}$, by Lemma 6.3.2(2) (Substitution) we have $\Gamma \vdash R\{\tilde{x} := \tilde{N}\} : [E, F]$. Moreover, from $\Gamma \vdash S : \text{Un}$, $\Gamma \vdash U : \text{Un}$ and $\Gamma \vdash a : \text{Public}$, by (UNTRUSTED AMB) we have $\Gamma \vdash a[S \mid U] : [E, F]$ and we conclude by (PARA).

▷ If $T = \text{Un}$, then the hypothesis must have been derived from (i) $\Gamma \vdash (x_1, \dots, x_k)^a R : \text{Un}$ and (ii) $\Gamma \vdash a[\langle N_1, \dots, N_k \rangle S \mid U] : \text{Un}$. Now, (i) can come from one of the following two cases:

▷ $\Gamma \vdash a : \text{Amb}[\text{Public}_1 \times \dots \times \text{Public}_k]$ and $\Gamma, x_i : \text{Public} \vdash R : \text{Un}$. In this case (ii) must have been derived by (AMB) from $\Gamma \vdash a : \text{Amb}[\text{Public}_1 \times \dots \times \text{Public}_k]$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S \mid U : [G, \text{Public}_1 \times \dots \times \text{Public}_k]$. From the last judgment we have $\Gamma \vdash U : [G, \text{Public}_1 \times \dots \times \text{Public}_k]$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S : [G, \text{Public}_1 \times \dots \times \text{Public}_k]$

$\dots \times \text{Public}_k]$, that must have been derived by (OUTPUT \uparrow) from $\Gamma \vdash N_i : \text{Public}$ and $\Gamma \vdash S : [G, \text{Public}_1 \times \dots \times \text{Public}_k]$. Now, from $\Gamma, x_i : \text{Public} \vdash R : \text{Un}$ and $\Gamma \vdash N_i : \text{Public}$, by Lemma 6.3.2(2)(Substitution) we have $\Gamma \vdash R\{\tilde{x} := \tilde{N}\} : \text{Un}$. Moreover, from $\Gamma \vdash S : [G, \text{Public}_1 \times \dots \times \text{Public}_k]$, $\Gamma \vdash U : [G, \text{Public}_1 \times \dots \times \text{Public}_k]$ and $\Gamma \vdash a : \text{Amb}[\text{Public}_1 \times \dots \times \text{Public}_k]$, by (AMB) we have $\Gamma \vdash a[S \mid U] : \text{Un}$ and we conclude by (PARA).

▷ $\Gamma \vdash a : \text{Public}$ and $\Gamma, x_i : \text{Public} \vdash R : \text{Un}$. In this case (ii) must have been derived by (UNTRUSTED AMB) from $\Gamma \vdash a : \text{Public}$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S \mid U : \text{Un}$. From the last judgment we have $\Gamma \vdash U : \text{Un}$ and $\Gamma \vdash \langle N_1, \dots, N_k \rangle S : \text{Un}$, that must have been derived by (UNTRUSTED OUTPUT \uparrow) from $\Gamma \vdash N_i : \text{Public}$ and $\Gamma \vdash S : \text{Un}$. Now, from $\Gamma, x_i : \text{Public} \vdash R : \text{Un}$ and $\Gamma \vdash N_i : \text{Public}$, by Lemma 6.3.2(2)(Substitution) we have $\Gamma \vdash R\{\tilde{x} := \tilde{N}\} : \text{Un}$. Moreover, from $\Gamma \vdash S : \text{Un}$, $\Gamma \vdash U : \text{Un}$ and $\Gamma \vdash a : \text{Public}$, by (UNTRUSTED AMB) we have $\Gamma \vdash a[S \mid U] : \text{Un}$ and we conclude by (PARA).

(output M) This case follows similarly to the previous one.

(Red Par) The reduction is $P' \mid R \rightarrow Q' \mid R$ derived from $P' \rightarrow Q'$. The judgment in the hypothesis, $\Gamma \vdash P' \mid R : T$ must depend on two judgments $\Gamma \vdash P' : T$ and $\Gamma \vdash R : T$. By the induction hypothesis, $\Gamma \vdash Q' : T$ and $\Gamma \vdash Q' \mid R : T$ derives by (PAR).

(Red Struct) follows by the induction hypothesis and Lemma 6.3.2(1), while (Red New) follows directly by the induction hypothesis.

(Red Amb) The reduction is $a\{P'\} \rightarrow a\{Q'\}$, derived from $P' \rightarrow Q'$, and the judgment $\Gamma \vdash a\{P'\} : T$, in the hypothesis, must depend on $\Gamma \vdash P' : T'$ for a suitable T' . By the inductive hypothesis, one has $\Gamma \vdash Q' : T'$. From this $\Gamma \vdash a\{Q'\} : T$ derives from $\Gamma \vdash Q' : T'$ by the same steps that derived $\Gamma \vdash a\{P'\} : T$ from $\Gamma \vdash P' : T'$.

□

6.4 Secrecy

In the context of name-passing calculi such as π or spi, secrecy clearly concerns names privacy. In a system of mobile agents instead, protecting names would be only part of the secrecy issue: for instance, one can think of keeping secret the very presence of an agent at a given site. In Section 6.1 we pointed out that it is very difficult to keep secret

an agent that is ready to interact with its context. We also mentioned the possibility of encrypting agents, reducing their active code to a passive packet that is forwarded along the untrusted network towards a trusted site. This is a simple way of totally protecting an agent, and this mechanism can be recast to the spi calculus model since an encrypted agent is nothing else than a private data ready to be communicated. However, this is a drastic method that achieves agent secrecy at the price of strongly constraining the power of the agent model. On the contrary, the CBA model we proposed allows to reveal the presence of an ambient, protecting instead its interactions with the context. By disallowing external communications via a locking mechanism, what we mean to protect is the content of ambients, that is stored values and subambients. Protecting subambients is achieved again by means of locking/unlocking them, while in order to protect data we resort to name secrecy, adapting to the new framework the ideas developed for the spi-calculus.

We refer to a rather standard notion of secrecy in the literature of security protocols, namely: *a process preserves the secrecy of a piece of data M if the process never sends M in clear on a public channel, or anything that would permit the computation of M , even in interaction with the attacker.*

The formal definition of secrecy is inspired by [AB02] and adapted to our framework. In particular, we represent an attacker as an arbitrary (but closed) context $\mathbf{A}()$. As an example, a potentially malicious environment surrounding a trusted agent is modeled with contexts of the form $a[Q \mid ()]$, while the context $a[\text{in } p.\text{in } q.Q \mid Q'] \mid ()$ is an attacker agent that visits a trusted site. In addition, we characterize the initial knowledge of the attacker in terms of the names, the keys and the capabilities initially known to it. Perhaps interestingly, the knowledge of capabilities is important here, since by exercising (a number of) capabilities an adversary may approach an agent and interact with it, even without knowing its name. As an example, if we take the process $b[\overline{\text{in}} \{x\}_k.\langle a \rangle^x]$, it is not necessary for an opponent to know the name b in order to read the value a , but it is sufficient for him to know the capability $\text{'in } b'$ and the key $'k'$.

In order to formally define the adversary, we use two sets $fen(P)$ and $fc(P)$ that collect, respectively, the free names used by P to exchange values, and the capabilities on free names appearing in P . The inductive definitions of $fen(P)$ and $fc(P)$ are straightforward, and can be found in Appendix A.

Definition 6.4.1 (S-adversary). Let S be a finite set of names and capabilities. The

closed context $\mathbf{A}()$ is an S -adversary if $fen(\mathbf{A}()) \cup fc(\mathbf{A}()) \subseteq S$. \square

Next, we define what it means to preserve a secret: since capabilities are public, the definition of secrecy only applies to names.

Definition 6.4.2 (Revealing Names, Preserving Secrecy of Names).

Let P be a process, M a name and S a finite set of names and capabilities.

- \triangleright P reveals M to S if and only if there exists an S -adversary $\mathbf{A}()$ and a name $c \in S$ such that $\mathbf{A}(P)$ is closed and $\mathbf{A}(P) \Longrightarrow \mathbf{C}(c[\langle M \rangle^\dagger \mid Q])$, for some context $\mathbf{C}()$ where c is not bound in $\mathbf{C}()$, and for some process Q .
- $(\nu n)P$ reveals n to S if and only if there is a name m with $m \notin S \cup fn(P)$ such that $P\{n := m\}$ reveals m to S .
- \triangleright P preserves the secrecy of M from S if and only if it does not reveal M to S .
- $(\nu n)P$ preserves the secrecy of n from S if and only if it does not reveal n to S . \square

Notice that, even if we only specify the adversary's initial knowledge, an attacker may dynamically acquire new names and new capabilities by creating fresh names, by receiving messages and by unlocking ambients locked with a key it knows (remind that CBA's unlocking primitives learn the name of the unlocked ambient). Nevertheless, Definition 6.4.2 is enough to capture the intended notion of secrecy since the publishing of M on c may be the result of the interaction between P and the context. As an example, take $S = \{c\}$, and consider the process $P = c[\langle a \rangle^\dagger] \mid a[\langle k \rangle^\dagger]$. P does not preserve the secrecy of k from S , even though S does not include a . In fact, one can take the S -adversary $\mathbf{A}() = (x)^c(y)^x c[\langle y \rangle^\dagger] \mid ()$, and note that $\mathbf{A}(P) \Longrightarrow c[\langle k \rangle^\dagger] \mid c[] \mid a[]$.

The following theorem states that a well typed process P preserves the secrecy of its private names from any adversary that initially knows all the public names in P and has the capabilities to move in and out any ambient of P , included its secret ambients.

Note that, by definition, an adversary has access to the anonymous top level channel of any process. Thus a (trusted) agent that actually allows interactions with adversaries must be such that its top level processes have type Un . Let indeed $\mathbf{A}()$ be an adversary context built with public names and capabilities contained in a process P ; by Typability Lemma 6.3.1, in order for P to interact with the opponent, i.e. to fill the hole in the context $\mathbf{A}()$, it must have type Un (remind that trusted processes, in particular trusted

ambients, may be safely typed with type Un). This explains the hypothesis of the following secrecy theorem. There are obviously other safe processes, in particular those that can be typed with type $[E, F]$, but we do not consider them since their typing implies that they can be put only within trusted contexts.

Theorem 6.4.3 (Secrecy). *Let P be a process such that $\Gamma \vdash P : \text{Un}$ and $\Gamma \vdash s : W$ with $W \neq \text{Public}$. Let $S = \{a \mid \Gamma \vdash a : \text{Public}\} \cup \{\text{in } a, \text{out } a \mid a \in \text{Dom}(\Gamma)\}$. Then P preserves the secrecy of s from S .*

Proof. By contradiction. Assume that P reveals s to S ; by Definition 6.4.2 there exists an S -adversary $\mathbf{A}()$ such that $\mathbf{A}(P)$ is closed and $\mathbf{A}(P) \Longrightarrow \mathbf{C}(c[\langle s \rangle^\uparrow \mid Q])$ with $c \in S$. Since $\mathbf{A}()$ is an S -adversary and any name and capability contained in S is of type Public , by Proposition 6.3.1 (Typability), we have that $\Gamma|_S \vdash \mathbf{A}() : \text{Un}$ is a derivable judgment assuming $\Gamma|_S \vdash () : \text{Un}$. Then, by weakening, $\Gamma \vdash () : \text{Un}$ and $\Gamma \vdash \mathbf{A}() : \text{Un}$ are also derivable, and from $\Gamma \vdash P : \text{Un}$ we have that $\Gamma \vdash \mathbf{A}(P) : \text{Un}$ is a derivable judgment. By Proposition 6.3.3 (Subject Reduction) we have $\Gamma \vdash \mathbf{C}(c[\langle s \rangle^\uparrow \mid Q]) : \text{Un}$, thus there exists a type environment Δ and a process type T such that $\Gamma, \Delta \vdash c[\langle s \rangle^\uparrow \mid Q] : T$. Now, from $c \in S$ we have $\Gamma \vdash c : \text{Public}$, that, together with the previous judgment, gives $\Gamma \vdash \langle s \rangle^\uparrow \mid Q' : \text{Un}$, that implies $\Gamma \vdash \langle s \rangle^\uparrow : \text{Un}$, hence $\Gamma \vdash s : \text{Public}$, that contradicts the hypothesis $\Gamma \vdash s : W$ with $W \neq \text{Public}$. \square

6.5 Examples

In this section we illustrate the CBA calculus with two examples: the first one shows the use of the locking/unlocking mechanism to protect a mobile agent that moves in a network, while the second one is drawn from the literature on cryptographic protocols and is intended as a first test of the 'cryptographic' interpretation of CBA.

We start introducing a matching construct that allows one to test equality of names (or variables); it will be useful to write the following examples. Let the process $[a = b]P$ be a simpler version of the spi-calculus process $[M \text{ is } N]P$ in which we compare names rather than full expressions: $[a = b]P$ behaves like P if a and b are the same name (or variable), otherwise it is stuck. This behavior can be simulated in CBA as follows (with

$c \notin fn(P)$:

$$[a = b]P \triangleq (\nu c) (c[\langle \rangle^a \mid b[(\rangle^\dagger \langle \rangle^\dagger] \mid (\rangle^b \langle \rangle^\dagger] \mid (\rangle^c P)$$

With the definition above, we have that $[a = b]P \implies (\nu c)(c[b[0]]) \mid P$ as expected.

The first example is quite general, it can be thought of as an agent that performs a query on a distributed database: e.g. in the e-commerce framework, we can think about an agent that visits different stocks looking for which one holds a particular item to be sold.

We assume that there is a user U that spawns an agent that navigates the network to visit databases A_i . The agent asks each database for the presence of a given item it, and collects the names of databases that actually contain that item. To protect the agent while moving along the network we use the CBA mechanism, blocking *only* that part of the agent that is intended to interact with the different databases. Let k be a locking key shared between the user U and the databases A_i . The user can be represented as $U[(\nu c)c[P \mid R] \mid Q]$, where c is a private agent that contains two threads: R is a router process that controls agent movements and P is the main thread that interacts with the visited databases. We use two ambients l and r to let the two threads correctly synchronize within c :

$$\begin{aligned} c[(\nu l, r)(l[\langle \rangle^\dagger] \mid & \mid (\rangle^l.\text{lock } k.(r[\langle \rangle^\dagger] \mid \langle \text{it} \rangle^\dagger(x, y)^\dagger([x = \text{yes}]\langle y \rangle \mid l[\langle \rangle^\dagger]))) \\ & \mid (\rangle^r.\text{out } U \dots \text{in } A_1.(\rangle^r.\text{out } A_1 \dots \text{in } A_2.(\rangle^r.\text{out } A_2 \dots \text{in } U))] \end{aligned}$$

The router process navigates c along the network to visit the databases, first A_1 then A_2 , but before moving outside U, A_1, A_2 , it waits the agreement of the other thread. This is done in order to enforce the fact that the ambient c moves in the unreliable network only after having been locked with the secret key k . On the other hand, the main thread within c first lets c to be locked, then it releases the router process and waits to interact with the external context. It is worth noticing that while the ambient c is moving in the untrusted environment, its main thread is still offering the upward output $\langle \text{it} \rangle^\dagger$, but this is safe since the CBA model ensures that upward communications are disallowed as long as c is locked. It remains to explain how c actually interacts with the distributed databases A_i , that we represent as follows:

$$A_i[\overline{\text{in}} \{z\}_k.(x)^z P_i^x \langle M_i^x, A_i \rangle^z]$$

where the process P_i^x represents a calculation performed by the database to find out if it contains the item x . The result of such a calculation is the expression M_i^x (that again depends on x) that we assume to be one of the two special names *yes* or *no*, depending on the actual presence of x within A_i .

Now, when the agent c enters A_i it gets unlocked, thus its upward interactions can synchronize with those contained in the database. Then c asks A_i for the presence of the item it and if the result is *yes*, it stores the name of the database in its local buffer. After such an interaction, the ambient c is ready to continue its search at a different site, thus it releases a new instance of its main thread, then it gets locked and reactivates its router thread.

The main point with this example is that the information collected by c are kept secret in its local buffer. No external, untrusted opponent may steal it; in addition, no opponent may fill c with unreliable information (*integrity* property), since the locking mechanism ensures that c only interacts with trusted databases.

We now present a more complex example: we encode in CBA (a version of) the Wide Mouthed Frog protocol, whose abstract definition is the following.

Msg. 1	$A \longrightarrow S : A$	Msg. 5	$B \longrightarrow S : N_b$
Msg. 2	$S \longrightarrow A : N_s$	Msg. 6	$S \longrightarrow B : \{S, A, B, K_{AB}, N_b\}_{K_{SB}}$
Msg. 3	$A \longrightarrow S : \{A, B, K_{AB}, N_s\}_{K_{AS}}$	Msg. 7	$A \longrightarrow B : \{M\}_{K_{AB}}$
Msg. 4	$S \longrightarrow B : *$		

In this protocol, first A contacts a trusted server S to get a nonce N_s , then it sends to S a message containing its identity, the name of the principle it wants to communicate with, the fresh encryption key it wants to use for such communication and the nonce it got from S , everything encrypted using a key shared between A and S . At this stage, the server S asks B for a nonce N_b , and then S sends to B a message containing the private key received from A and the nonce N_b , everything encrypted with the key K_{SB} shared between S and B . Finally, A transmits to B a secret message M encrypted with the key K_{AB} that B has already received.

This protocol can be implemented in CBA in a number of ways, we present one of these possibilities based on the following definitions.

$$A = A[(\nu K_{AB}) \text{ pub}[\text{out } A.\text{in } S.(x_n)^\dagger a_1 \text{ out pub}.\langle A, B, K_{AB}, x_n \rangle^\dagger \text{ } \text{ }_{K_{AS}}] \\ | a_2 \text{ out } A.\text{in } B.\langle M \rangle^\dagger \text{ } \text{ }_{K_{AB}}]]$$

The principal A spawns an ambient pub that goes to the server, takes a fresh nonce, then it let exit an ambient encrypted with the key K_{AS} that carries the composite message containing the private key K_{AB} . Finally, an ambient a_2 encrypted with the key K_{AB} is used to bring the secret message M from A to B . The definition of the server S is the following, where \mathcal{P} is the set of all the principals that may use the server S .

$$S = S[(\nu N_s)(\langle N_s \rangle^{\text{pub}} | \prod_{A \in \mathcal{P}} \overline{\text{out}} \{y\}_{K_{AS}}.(x_a, x_b, x_k, x_n)^y.[x_a = A][x_n = N_s] \\ \prod_{B \in \mathcal{P}} [x_b = B] \text{ pub}[\text{out } S.\text{in } x_b.(z_n)^\dagger b_1 \text{ out pub}.\langle S, x_a, x_b, x_k, z_n \rangle^\dagger \text{ } \text{ }_{K_{SB}}]])]$$

The server S waits for someone that starts the protocol sending an ambient whose name is pub ; when such an ambient arrives, S communicates a fresh nonce N_s to it. Then S waits for an encrypted message containing the nonce N_s , that arrives from one of the principals it knows. When such a message arrives, it also contains the name of a second principal, bound to x_b , and a fresh key, bound to x_k . At this stage S spawns an ambient pub that asks x_b for a fresh nonce and sends to x_b a message containing the private key x_k , enclosed within an ambient b_1 encrypted with the key shared between S and x_b .

$$B = B[(\nu N_b)(\langle N_b \rangle^{\text{pub}} | \overline{\text{out}} \{y\}_{K_{SB}}.(x_s, x_a, x_b, x_k, z_n)^y[x_b = B][z_n = N_b] \\ \overline{\text{in}} \{y\}_{x_k(x)}^y F(y))]$$

The principal B waits for an ambient that starts the protocol asking for a fresh nonce N_b . In the second step B waits from the server an ambient encrypted with the key K_{SB} carrying the nonce N_b and a private key, bound to x_k . Finally, B is ready to decrypt with key x_k an ambient that carries a private message coming from the principal whose name is bound to x_a .

Then the protocol above can be represented in CBA by the process $\mathcal{P}(M) = (A \mid B \mid S \mid \overline{\text{out}})$. Moreover, let

$$\begin{aligned} \Gamma = & K_{AS}:\text{Key}[\text{Amb}[E_A] \times \text{Amb}[E_B] \times \text{Key}[F] \times \text{Public}], \\ & K_{SB}:\text{Key}[\text{Amb}[E_S] \times \text{Amb}[E_A] \times \text{Amb}[E_B] \times \text{Key}[F] \times \text{Public}], \\ & \text{pub} : \text{Amb}[\text{Public}], \quad a_2 : \text{Amb}[F], \\ & a_1 : \text{Amb}[\text{Amb}[E_A] \times \text{Amb}[E_B] \times \text{Key}[F] \times \text{Public}], \\ & b_1 : \text{Amb}[\text{Amb}[E_S] \times \text{Amb}[E_A] \times \text{Amb}[E_B] \times \text{Key}[F] \times \text{Public}] \end{aligned}$$

then choosing for the restricted names the following types:

$$N_s : \text{Public}, \quad N_b : \text{Public}, \quad K_{AB} : \text{Key}[F]$$

a routine check shows that the protocol is well typed. Thus, assuming $\Gamma \vdash M : F \not\leq \text{Public}$, Theorem 6.4.3 guarantees that the secret message M can not be revealed to an untrusted external process.

6.6 Encoding of Spi calculus

As a further example of the expressive power of CBA, we conclude this chapter with an encoding of the spi-calculus in CBA. In particular, we encode the following fragment of the asynchronous spi-calculus, that disregards constructs like pairs, natural numbers and matching.

<i>Expressions</i>	$M, N ::=$	$n \mid x \mid \{M_1, \dots, M_n\}_N$	
<i>Processes</i>	$P, Q ::=$	$\overline{M}\langle N_1, \dots, N_n \rangle$	asynch. output
		$\mid M(x_1, \dots, x_n)P$	input
		$\mid P \mid Q$	composition
		$\mid (\nu n)P$	restriction
		$\mid \mathbf{0}$	stop
		$\mid \text{case } M \text{ of } \{x_1, \dots, x_n\}_N \text{ in } P$	decryption

The operational semantics of this fragment can be given in terms of structural congruence and reduction relations, following [AG99]. Below we only give the definition of reduction, and refer to [AG99] for full details.

<i>(I/O)</i>	$\overline{n}\langle M_1, \dots, M_n \rangle.P \mid n(x_1, \dots, x_n)Q \rightarrow P \mid Q\{x_i := M_i\}$
<i>(Decrypt)</i>	$\text{case } \{M_1, \dots, M_n\}_k \text{ of } \{x_1, \dots, x_n\}_k \text{ in } P \rightarrow P\{x_i := M_i\}$
<i>(Red Para)</i>	$P \rightarrow Q \Rightarrow P \mid R \rightarrow Q \mid R$
<i>(Red Res)</i>	$P \rightarrow Q \Rightarrow (\nu n)P \rightarrow (\nu n)Q$
<i>(Red Struct)</i>	$P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \rightarrow Q$

The basic idea of the encoding is to represent an encrypted data with an 'encrypted' ambient that contains that data. Furthermore, since in CBA we cannot exchange encrypted packets, what gets communicated is a pointer to that packet, that is the name of the ambient that contains the data.

To give the intuition, let a, b stand for a name or a variable. An encryption packet $\{a\}_k$ is encoded as an ambient -named p - that stores the name a and that is obtained by an encoding function $\llbracket \cdot \rrbracket_p$. Transmitting that packet over a channel b , as in $\bar{b}\langle\{a\}_k\rangle$, is represented by the process $(\nu p)(b[\langle p \rangle^\dagger] \mid \llbracket \{a\}_k \rrbracket_p)$.

Decryption on the other hand, is obtained by letting the ambient p , that represents the packet, enter a temporary ambient c where p is 'decrypted' with the key k . Specifically, $\llbracket \{a\}_k \rrbracket_p$ stands for the process $p[(x)^\dagger.\text{lock } k.\text{in } x.\langle a \rangle^\dagger]$ that reads the name of the ambient where it is expected to enter to be 'decrypted'. Dually, the decryption of the packet $\{a\}_k$, that is $\llbracket \text{case } \{a\}_k \text{ of } \{x\}_k \text{ in } P \rrbracket$, is represented by the process $(\nu p)(\llbracket \{a\}_k \rrbracket_p \mid (\nu c)(\langle c \rangle^p \mid c[\bar{\text{in}} \{y\}_k.(x)^y\langle x \rangle^\dagger] \mid (x)^c P))$.

We give the formal definition of the encoding in terms of three translation maps, where p is a name:

$$\begin{aligned} \langle \cdot \rangle_p & : \text{Expressions} \mapsto \text{Expressions} \\ \llbracket \cdot \rrbracket_p & : \text{Expressions} \mapsto \text{Processes} \\ \llbracket \cdot \rrbracket & : \text{Processes} \mapsto \text{Processes} \end{aligned}$$

Informally, $\langle M \rangle_p$ returns the data M if M is a name or a variable, instead, if M is an encryption packet, it returns the pointer to (i.e. the name of the ambient that represents) that packet. Furthermore, if M is an encryption, $\llbracket M \rrbracket_p$ returns an ambient p that represents that encryption. Note that multiple encryption is then represented as a chain of pointers: $\llbracket \{\{a\}_{k_1}\}_{k_2} \rrbracket_p = (\nu q)q[(x)^\dagger.\text{lock } k_1.\text{in } x.\langle a \rangle^\dagger] \mid p[(x)^\dagger.\text{lock } k_2.\text{in } x.\langle q \rangle^\dagger]$.

The three maps are defined inductively as follows, by generalizing the previous intuitions:

$$\begin{aligned} \langle a \rangle_p & = a \\ \langle \{M_1, \dots, M_n\}_k \rangle_p & = p \\ \llbracket a \rrbracket_p & = \mathbf{0} \\ \llbracket \{M_1, \dots, M_n\}_k \rrbracket_p & = (\nu q_1, \dots, q_n)(\Pi_{i=1}^n \llbracket M_i \rrbracket_{q_i} \mid \\ & \quad !p[(x)^\dagger.\text{lock } k.\text{in } x.\langle \langle M_1 \rangle_{q_1}, \dots, \langle M_n \rangle_{q_n} \rangle^\dagger]) \end{aligned}$$

$$\begin{aligned}
\llbracket \mathbf{0} \rrbracket &= \mathbf{0} \\
\llbracket (\nu n)P \rrbracket &= (\nu n)\llbracket P \rrbracket \\
\llbracket P \mid Q \rrbracket &= \llbracket P \rrbracket \mid \llbracket Q \rrbracket \\
\llbracket \bar{b}\langle M_1, \dots, M_n \rangle \rrbracket &= (\nu q_1, \dots, q_n)(\Pi_{i=1}^n \llbracket M_i \rrbracket_{q_i} \mid b[\langle \langle M_1 \rangle_{q_1}, \dots, \langle M_n \rangle_{q_n} \rangle^\dagger]) \\
\llbracket b(x_1, \dots, x_n)P \rrbracket &= (x_1, \dots, x_n)^b \llbracket P \rrbracket \\
\llbracket \text{case } M \text{ of } \{x_1, \dots, x_n\}_k \text{ in } P \rrbracket &= (\nu p)(\llbracket M \rrbracket_p \mid (\nu c)(\langle c \rangle^{\langle M \rangle_p} \mid \\
&\quad c[\bar{\text{in}} \{y\}_k.(x_1, \dots, x_n)^y \langle x_1, \dots, x_n \rangle^\dagger] \mid (x_1, \dots, x_n)^c P))
\end{aligned}$$

We end the section with a simple example of the encoding: consider the spi-calculus process $P = (\nu a)(\bar{a}\langle \{M\}_k \rangle \mid a(x).\text{case } x \text{ of } \{y\}_k \text{ in } P')$ that reduces in two steps to $P'\{y := M\}$. The encoding of P is the following CBA process:

$$\begin{aligned}
\llbracket P \rrbracket &= (\nu a)(\nu q)(\llbracket \{M\}_k \rrbracket_q \mid a[\langle \langle \{M\}_k \rangle_q \rangle^\dagger]) \mid \\
&\quad (x)^a(\nu p)(\llbracket x \rrbracket_p \mid (\nu c)(\langle c \rangle^{\langle x \rangle_p} \mid c[\bar{\text{in}} \{y\}_k.(x)^y \langle x \rangle^\dagger] \mid (y)^c \llbracket P' \rrbracket))
\end{aligned}$$

where $\llbracket \{M\}_k \rrbracket_q = (\nu q_1)(\llbracket M \rrbracket_{q_1} \mid !q[(x)^\dagger.\text{lock } k.\text{in } x.\langle \langle M \rangle_{q_1} \rangle^\dagger])$, $\langle \{M\}_k \rangle_q = q$, $\llbracket x \rrbracket_p = \mathbf{0}$ and $\langle x \rangle_p = x$. Then

$$\begin{aligned}
\llbracket P \rrbracket &\equiv (\nu a, q, q_1)(\llbracket M \rrbracket_{q_1} \mid q[(x)^\dagger.\text{lock } k.\text{in } x.\langle \langle M \rangle_{q_1} \rangle^\dagger] \mid a[\langle q \rangle^\dagger]) \mid \\
&\quad (x)^a(\nu c)(\langle c \rangle^x \mid c[\bar{\text{in}} \{y\}_k.(x)^y \langle x \rangle^\dagger] \mid (y)^c \llbracket P' \rrbracket) \\
&\rightarrow (\nu a, q, q_1)(\llbracket M \rrbracket_{q_1} \mid q[(x)^\dagger.\text{lock } k.\text{in } x.\langle \langle M \rangle_{q_1} \rangle^\dagger] \mid a[\] \mid \\
&\quad (\nu c)(\langle c \rangle^q \mid c[\bar{\text{in}} \{y\}_k.(x)^y \langle x \rangle^\dagger] \mid (y)^c \llbracket P' \rrbracket) \\
&\Rightarrow (\nu a, q, q_1, c)(\llbracket M \rrbracket_{q_1} \mid a[\] \mid q\{\text{in } c.\langle \langle M \rangle_{q_1} \rangle^\dagger\}_k \mid \\
&\quad c[\bar{\text{in}} \{y\}_k.(x)^y \langle x \rangle^\dagger] \mid (y)^c \llbracket P' \rrbracket) \\
&\Rightarrow (\nu a, q, q_1, c)(\llbracket M \rrbracket_{q_1} \mid a[\] \mid c[q[\langle \langle M \rangle_{q_1} \rangle^\dagger] \mid (x)^q \langle x \rangle^\dagger] \mid (y)^c \llbracket P' \rrbracket) \\
&\Rightarrow (\nu a, q, q_1, c)(\llbracket M \rrbracket_{q_1} \mid a[\] \mid c[q[\] \] \mid \llbracket P' \rrbracket\{y := \langle M \rangle_{q_1}\})
\end{aligned}$$

Now, if we assume $(\nu n)n[\mathbf{0}] \equiv \mathbf{0}$, we have that $\llbracket P \rrbracket \Rightarrow (\nu q_1)(\llbracket M \rrbracket_{q_1} \mid \llbracket P' \rrbracket\{y := \langle M \rangle_{q_1}\})$ and we conclude observing that, if we also assume $!P \mid !P \equiv !P$, we have that the following property holds for substitutions:

$$\llbracket P\{x := M\} \rrbracket \equiv (\nu p)(\llbracket M \rrbracket_p \mid \llbracket P \rrbracket\{x := \langle M \rangle_p\})$$

6.7 Conclusion and Related Work

In this chapter we have investigated the protection of migrating agents against the untrusted networks they traverse. Our primitives are best understood as low-level primitives

to be employed for a secure implementation of the abstract mechanisms for secrecy found in mainstream ambient calculi such as the Ambient calculus.

The idea of using capabilities that carry passwords comes back to SAP calculus [MH02]. In fact, NBA can be encoded CBA simply by defining the capability $\text{in}\langle n, k \rangle$ as $\text{lock } k.\text{in } n$, and similarly for $\text{out}\langle n, k \rangle$. On the other hand, the locking model of CBA appears to provide strictly more flexibility, and expressive power, than the use of passwords in NBA: a CBA agent can be locked by any of its local threads requesting a locking. Hence, an agent can be locked and protected from undesired interactions by firing an action in one of local threads, and it not clear that a corresponding mechanism can be recovered in NBA (or SAP).

In this chapter we then studied the secrecy property of CBA; a corresponding analysis has already been conducted in the spi-calculus. In [Aba99] secrecy property of security protocols is studied by means of a type system that distinguishes data of type *Public*, *Secret* and *Any*. Besides the underlying calculus, the main difference with our work is the definition of secrecy: in [Aba99] a process $P\{M\}$ does not leak its secret M if $P\{M\}$ is equivalent (with testing equivalence) to $P\{N\}$, where the secret data M has been substituted with N . On the contrary, our definition of secrecy is borrowed from [AB02] (that, in turn, comes from the work of Dolev and Yao [DY83]) and adapted to CBA; we say that a process respects the privacy of a secret data if it turns out that any opponent context will never know that secret data. This security property has been studied in the pi-calculus (without cryptography) framework in [CGG00b] and in (a generalization of) the spi-calculus in [AB02]. In [HR99] Hennessy and Riely addressed the security issue in the context of open network. They study an extension of the $D\pi$ -calculus with a type system that labels some location as untrusted and relies on run-time type checking to enforce security restrictions for processes coming from untrusted locations.

Finally we presented an encoding of spi calculus into CBA and we showed an example based on a simple security protocol that gives a first idea of how the CBA model can be related to cryptography. However, a deeper comparison between our approach and that of the spi-calculus deserves to be made: in particular it would be interesting to study what benefits (if any) come from using mobile agents in the design of complex security protocols such as multi party computation or secret sharing. This is the subject of future work, starting with a deeper study of the spi encoding given in section 6.4.

Chapter 7

Conclusions

The design of foundational calculi is a subtle activity, in which the choice of the core set of primitives is the result of a tension between expressive power –the ability of such primitives to provide effective encodings for the intended protocols and systems– and formal parsimony.

Mobile Ambients certainly represent one of the most successful examples of foundational calculi for mobile computing and wide-area networks. On the other hand, their mobility and reconfiguration capabilities have also been the source of many difficulties with reasoning about ambient programs.

Motivated by these consideration, the calculus of Boxed Ambients restricts the Ambient Calculus by removing the ability to open ambients and extends it with explicit communication primitives across ambient boundaries. Removing the *open* capability ensures that the code of untrusted agents will not mingle with more trusted code. Providing explicit communication avoid the need for some rather cumbersome encoding needed by the original ambient calculus.

We have explored diverse aspects of the new calculus, and analyzed the impact of the new primitives in providing support for different facets of security in distributed systems. In particular:

- ▷ We have developed a number of type systems that enforce a given security policy on trusted systems, showing how they can be generalized to the case where systems are composed of both trusted and untrusted components.
- ▷ The use of co-capabilities and the communication model of NBA provide the calculus with a rich (and tractable) algebraic theory.

- ▷ The access protocol of NBA where incoming agents must be “authenticated” and registered, reflects what happens in real computational domains.

As for future work, we plan to investigate on an alternative approach to the problem of detecting malicious flows of information, in order to allow the design of secure processes that are more flexible than those allowed by the type system of Chapter 4. The study of the algebraic theory of NBA can be also improved: we plan to investigate on the completeness of the coinductive characterization of barbed congruence we gave in Chapter 5. Finally, the work presented in Chapter 6 can be extended in many directions: besides secrecy, we think that the CBA calculus is a suitable framework for the study of other security properties of agents moving in untrusted networks, as integrity or authentication. Furthermore, it would be interesting to deeply study the relations between our model and a more classical use of cryptography to protect agents, testing our framework on complex security protocols in order to better exploit the power of agents. In addition, it would be interesting to translate Boxed Ambients into the lower-level CBA calculus and to study such a translation in order to fill the gap between the abstractions provided by BA for the security of agents, and their implementation.

One important point that we did not address in this dissertation is the issue of implementing a distributed, mobile system. We argued that the design choices underlying Boxed Ambients have implementation oriented features. We expect that the absence of *open* and the new communication model of NBA, where each ambient contains two mutually non-interfering channels respectively for local and upward communication, enable the development of effective implementations of the calculus. Nevertheless, a precise analysis of the implementation of Boxed Ambients over a real distributed system deserves to be made.

Appendix A

In the following we inductively define two sets $fen(P)$ and $fc(P)$ that collect, respectively, the free names used in a CBA process P to exchange values, and the capabilities on free names appearing in P . For the definition of $fc(P)$ we use the subscript S to collect bound names. These definitions easily generalize to the case of considering an arbitray context instead of a process P .

$$\begin{array}{ll}
fen(n) &= \{n\} \\
fen(x) &= \emptyset \\
fen(\text{in } n) &= \emptyset \\
fen(\text{out } n) &= \emptyset \\
fen(M_1.M_2) &= fen(M_1) \cup fen(M_2) \\
fen(\overline{\text{in}}) &= \emptyset \\
fen(\mathbf{0}) &= \emptyset \\
fen(M.P) &= fen(M) \cup fen(P) \\
fen((\tilde{x})^a P) &= \{a\} \cup fen(P) \\
fen((\tilde{x})^\dagger P) &= fen(P) \\
fen((\tilde{x})P) &= fen(P) \\
fen(\overline{\text{out}}) &= \emptyset \\
fen(\langle M_1, \dots, M_k \rangle^a P) &= \{a\} \cup fen(P) \cup fen(M_1) \cup \dots \cup fen(M_k) \\
fen(\langle M_1, \dots, M_k \rangle^\dagger P) &= fen(P) \cup fen(M_1) \cup \dots \cup fen(M_k) \\
fen(\langle M_1, \dots, M_k \rangle P) &= fen(P) \cup fen(M_1) \cup \dots \cup fen(M_k) \\
fen(\overline{\text{in}} \{x\}_M.P) &= fen(M) \cup fen(P) \\
fen(\overline{\text{out}} \{x\}_M.P) &= fen(M) \cup fen(P) \\
fen(\text{lock } M.P) &= fen(M) \cup fen(P) \\
fen((\nu n)P) &= fen(P) \setminus \{n\} \\
fen(P \mid Q) &= fen(P) \cup fen(Q) \\
fen(!\pi.P) &= fen(\pi.P) \\
fen(M[P]) &= fen(M) \cup fen(P) \\
fen(M \parallel P \parallel N) &= fen(M) \cup fen(P) \cup fen(N)
\end{array}$$

$$\begin{aligned}
fc_S(n) &= \emptyset & fc_S(\mathbf{0}) &= \emptyset \\
fc_S(x) &= \emptyset & fc_S(M.P) &= fc_S(M) \cup fc_S(P) \\
fc_S(\text{in } n) &= \text{if } n \notin S \text{ then } \{\text{in } n\} \text{ else } \emptyset & fc_S((\tilde{x})^a P) &= fc_S(P) \\
fc_S(\text{out } n) &= \text{if } n \notin S \text{ then } \{\text{out } n\} \text{ else } \emptyset & fc_S((\tilde{x})^\dagger P) &= fc_S(P) \\
fc_S(M_1.M_2) &= fc_S(M_1) \cup fc_S(M_2) & fc_S((\tilde{x})P) &= fc_S(P) \\
fc_S(\overline{\text{in}}) &= \emptyset & fc_S(\overline{\text{out}}) &= \emptyset \\
fc_S(\langle M_1, \dots, M_k \rangle^a P) &= fc_S(P) \cup fc_S(M_1) \cup \dots \cup fc_S(M_k) \\
fc_S(\langle M_1, \dots, M_k \rangle^\dagger P) &= fc_S(P) \cup fc_S(M_1) \cup \dots \cup fc_S(M_k) \\
fc_S(\langle M_1, \dots, M_k \rangle P) &= fc_S(P) \cup fc_S(M_1) \cup \dots \cup fc_S(M_k) \\
fc_S(\overline{\text{in}} \{x\}_M.P) &= fc_S(M) \cup fc_S(P) \\
fc_S(\overline{\text{out}} \{x\}_M.P) &= fc_S(M) \cup fc_S(P) \\
fc_S(\text{lock } M.P) &= fc_S(M) \cup fc_S(P) \\
fc_S(M \{ P \}_N) &= fc_S(M) \cup fc_S(P) \cup fc_S(N) \\
fc_S(!\pi.P) &= fc_S(P) \\
fc_S((\nu n)P) &= fc_{S \cup \{n\}}(P) \\
fc_S(P \mid Q) &= fc_S(P) \cup fc_S(Q) \\
fc_S(M[P]) &= fc_S(M) \cup fc_S(P)
\end{aligned}$$

References

- [AB02] M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. In *Proceedings of POPL'02*, pages 33–44. ACM Press, 2002.
- [Aba98] M. Abadi. Protection in programming-language translations. In *Automata, Languages and Programming: 25th International Colloquium, ICALP'98*, pages 868–883. Springer-Verlag, 1998.
- [Aba99] M. Abadi. Secrecy by Typing in Security Protocols. *Journal of the ACM*, 46(5):749–786, 1999.
- [AG99] M. Abadi and A. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation*, 148(1):1–70, 1999.
- [AKH92] S. Arun-Kumar and M. Hennessy. An Efficiency Preorder for Processes. *Acta Informatica*, 29:737–760, 1992.
- [BBC02] V. Bono, M. Bugliesi, and S. Crafa. Typed interpretations of extensible objects. *ACM Transactions on Computational Logic*, 3(4):562–603, August 2002.
- [BC01a] G. Boudol and I. Castellani. Non-interference for concurrent programs. In *Proceedings of ICALP'2001*, number 2076 in Lecture Notes in Computer Science, pages 328–395. Springer, 2001.
- [BC01b] M. Bugliesi and G. Castagna. Secure safe ambients. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages*, pages 222–235, London, 2001. ACM Press.

- [BCC00] M. Bugliesi, G. Castagna, and S. Crafa. Typed mobile objects. In *Proceedings of CONCUR 2000 (11th. International Conference on Concurrency Theory)*, number 1877 in Lecture Notes in Computer Science, pages 504–520. Springer, 2000.
- [BCC01a] M. Bugliesi, G. Castagna, and S. Crafa. Boxed ambients. In *TACS 2001 (4th. International Symposium on Theoretical Aspects of Computer Science)*, number 2215 in Lecture Notes in Computer Science, pages 38–63, Sendai, Japan, 2001. Springer.
- [BCC01b] M. Bugliesi, G. Castagna, and S. Crafa. Reasoning about security in mobile ambients. In *CONCUR 2001 (12th. International Conference on Concurrency Theory)*, number 2154 in Lecture Notes in Computer Science, pages 102–120, Aalborg, Denmark, 2001. Springer.
- [BCC01c] M. Bugliesi, G. Castagna, and S. Crafa. Subtyping and matching for Mobile Objects. In *ICTCS 2001 (7th. Italian Conference on Theoretical Computer Science)*, number 2202 in Lecture Notes in Computer Science, pages 235–255. Springer-Verlag, 2001.
- [BCF02] C. Braghin, A. Cortesi, and R. Focardi. Control Flow Analysis of Mobile Ambients with Security Boundaries. In B. Jacobs and A. Rensink, editors, *Proc. of Fifth IFIP International Conference on Formal Methods for Open Object-Based Distributed Systems (FMOODS’02)*, pages 197–212. Kluwer Academic Publisher, 2002.
- [BCMS02] M. Bugliesi, S. Crafa, M. Merro, and V. Sassone. Communication interference in mobile boxed ambients. In *In FSTTCS02 Conf. on the Foundations of Software Technology and Theoretical Computer Science 2002*, number 2556 in Lecture Notes in Computer Science, pages 71–84, 2002.
- [BCPS03] M. Bugliesi, S. Crafa, A. Prelic, and V. Sassone. Secrecy in untrusted networks. In *Thirtieth International Colloquium on Automata, Languages and Programming, ICALP2003*, number 2719 in Lecture Notes in Computer Science, pages 969–983, 2003.

- [Bor98] M. Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theoretical Computer Science*, 195, 1998.
- [Bou92] G. Boudol. Asynchrony and the π -calculus. Research Report 1702, INRIA, <http://www-sop.inria.fr/mimosa/personnel/Gerard.Boudol.html>, 1992.
- [BP76] D.E. Bell and L. La Padula. Secure computer system: Unified exposition and multics interpretation,. Technical Report MTR-2997, MITRE Corporation, Bedford, MA 01730, March 1976.
- [BV02] C. Bryce and J. Vitek. The JavaSeal mobile agent kernel. *Autonomous Agents and Multi-Agent Systems*, 2002. To appear.
- [Car99] L. Cardelli. Abstractions for mobile computations. In *Secure Internet Programming*, number 1603 in Lecture Notes in Computer Science, pages 51–94. Springer, 1999.
- [Car00] L. Cardelli. Global computing. In *IST FET Global Computing Consultation Workshop*. 2000. Slides.
- [CBC02] S. Crafa, M. Bugliesi, and G. Castagna. Information Flow Security for Boxed Ambients. *ENTCS*, 66(3), 2002.
- [CG98] L. Cardelli and A. Gordon. Mobile ambients. In *FoSSaCS'98*, number 1378 in Lecture Notes in Computer Science, pages 140–155. Springer, 1998.
- [CG99a] L. Cardelli and A. Gordon. Equational properties for mobile ambients. In *Proceedings FoSSaCS'99*. Springer LNCS. Full version available as Microsoft Research Technical Report MSR-TR-99-11, 1999.
- [CG99b] L. Cardelli and A. Gordon. Types for mobile ambients. In *Proceedings of POPL'99*, pages 79–92. ACM Press, 1999.
- [CGG99] L. Cardelli, G. Ghelli, and A. Gordon. Mobility types for mobile ambients. In *Proceedings of ICALP'99*, number 1644 in Lecture Notes in Computer Science, pages 230–239. Springer, 1999.

- [CGG00a] L. Cardelli, G. Ghelli, and A. D. Gordon. Ambient groups and mobility types. In *International Conference IFIP TCS*, number 1872 in Lecture Notes in Computer Science, pages 333–347. Springer, August 2000.
- [CGG00b] L. Cardelli, G. Ghelli, and A. D. Gordon. Secrecy and group creation. In *CONCUR2000:Concurrency Theory*, number 1877 in Lecture Notes in Computer Science, pages 365–379. Springer-Verlag, August 2000.
- [DCS00] M. Dezani-Ciancaglini and I. Salvo. Security types for safe mobile ambients. In *Proceedings of ASIAN’00*, pages 215–236. Springer, 2000.
- [DLB00] P. Degano, F. Levi, and C. Bodei. Safe ambients: Control flow analysis and security. In *Proceedings of ASIAN’00*, volume 1961 of LNCS, pages 199–214. Springer, 2000.
- [DNFP98] R. De Nicola, G. Ferrari, and R. Pugliese. Klaim: a kernel language for agents interaction and mobility. Number 24 in IEEE Transactions on Software Engineering, pages 315–330. 1998.
- [DNFP99] R. De Nicola, G. Ferrari, and R. Pugliese. Types as specifications of access policies. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, number 1603 in LNCS. Springer, 1999.
- [DNFPV00] R. De Nicola, G. Ferrari, R. Pugliese, and B Venneri. Types for access control. *Theoretical Computer Science*, 240(1):215–254, 2000.
- [DoD85] US Department of Defense. DoD trusted computer system evaluation criteria, (the orange book). DOD 5200.28-STD, 1985.
- [DY83] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, 1983.
- [FG95] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, 1995.

- [FG97] R. Focardi and R. Gorrieri. Non interference: Past, present and future. In *Proceedings of DARPA Workshop on Foundations for Secure Mobile Code*, pages 26–28, march 1997.
- [FGL⁺96] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *7th International Conference on Concurrency Theory (CONCUR'96)*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer, 1996.
- [FLA00] C. Fournet, J.-J. Levy, and Shmitt. A. An asynchronous, distributed implementation of mobile ambients. In *International Conference IFIP TCS*, number 1872 in *Lecture Notes in Computer Science*. Springer, 2000.
- [GM82] J.A. Goguen and J. Meseguer. Security policy and security models. In *Proceedings of Symposium on Secrecy and Privacy*, pages 11–20. IEEE Computer Society, april 1982.
- [Gol99] D. Gollmann. *Computer Security*. John Wiley & Sons Ltd., 1999.
- [Hen00] M. Hennessy. The security picalculus and non-interference. Technical Report CS-05-2000, University of Sussex, School of Cognitive and Computing Sciences, Brighton BN1 9QH, UK, Nov. 2000.
- [HR98] Ms Hennessy and J. Riely. Resource access control in systems of mobile agents (extended abstract). In *Proc. of 3rd International Workshop on High-Level Concurrent Languages (HLCL'98)*. 1998.
- [HR99] M. Hennesy and J. Riely. Type-safe execution of mobile agents in anonymous networks. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, number 1603 in LNCS. Springer, 1999.
- [HR00] M. Hennessy and J. Riely. Information flow vs. resource access in the asynchronous π -calculus (extended abstract). In *Automata, Languages and Programming, 27th International Colloquium*, volume 1853 of *Lecture Notes in Computer Science*, pages 415–427. Springer, 2000.

- [HVV00] K. Honda, V.T. Vasconcelos, and N. Yoshida. Secure information flow as typed process behaviour. In *ESOP '00*, volume 1782 of *Lecture Notes in Computer Science*, pages 180–199. Springer, 2000.
- [HY95] K. Honda and N. Yoshida. On Reduction-based Process Semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.
- [LR] X. Leroy and F. Rouaix. Security properties of typed applets. In *Proc. of POPL'98*, pages 391–400. ACM Press.
- [LS00] F. Levi and D. Sangiorgi. Controlling interference in ambients. In *POPL2000*, pages 352–364. ACM Press, 2000.
- [MH02] M. Merro and M. Hennessy. Bisimulation Congruences in Safe Ambients. In *POPL'02 Proc. 29th ACM Symposium on Principles of Programming Languages*, pages 71–80. ACM Press, 2002.
- [Mil99] R. Milner. *Communicating and Mobile Systems: the π -Calculus*. Cambridge University Press, 1999.
- [MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I and II. *Information and Computation*, 100:1–77, September 1992.
- [MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In *ICALP'92*, number 623 in *Lecture Notes in Computer Science*, pages 685–695. Springer, 1992.
- [MS02] M. Merro and V. Sassone. Typing and Subtyping Mobility in Boxed Ambients. In *Proceedings of Concur'02*, number 2421 in *Lecture Notes in Computer Science*. Springer, 2002.
- [NN00] F. Nielson and H.R. Nielson. Shape analysis for mobile ambients. In *POPL'00*, pages 135–148. ACM Press, 2000.
- [NNHJ99] F. Nielson, H.R. Nielson, R.R. Hansen, and J.G. Jensen. Validating firewalls in mobile ambients. In *CONCUR'99*, number 1664 in *Lecture Notes in Computer Science*, pages 463–477. Springer, 1999.
- [RH99] J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. In *Proceedings of POPL'99*, pages 93–104. ACM Press, 1999.

- [San92] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis CST-99-93, Department of Computer Science, University of Edinburgh, 1992.
- [San96] D. Sangiorgi. Bisimulation for Higher-Order Process Calculi. *Information and Computation*, 131(2):141–178, 1996.
- [Sew98] Peter Sewell. Global/local subtyping and capability inference for a distributed π -calculus. *Lecture Notes in Computer Science*, 1443:695–??, 1998.
- [SM92] D. Sangiorgi and R. Milner. The problem of “Weak Bisimulation up to”. In *Proc. CONCUR '92*, volume 630 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 1992.
- [ST98] T. Sander and C. Tschudin. Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1998. IEEE Computer Society Press.
- [SV00] P. Sewell and J. Vitek. Secure composition of untrusted code: Wrappers and causality types. In *13th IEEE Computer Security Foundations Workshop*, 2000. To Appear in *Journal of Computer Security*.
- [SV01] D. Sangiorgi and A. Valente. A distributed abstract machine for safe ambients. In *Proc. of ICALP 2001*, pages 408–420, 2001.
- [SW01] D. Sangiorgi and D. Walker. *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
- [Tsc99] C. Tschudin. Mobile agent security. In M. Klusch, editor, *Intelligent information agents: agent based information discovery and management in the Internet*, pages 431–446. Springer-Verlag, 1999.
- [TZH02] D. Teller, P. Zimmer, and D. Hirschhoff. Using ambients to control resources. In *CONCUR 2002*, number 2421 in *Lecture Notes in Computer Science*. Springer, 2002.

- [VC99] J. Vitek and G. Castagna. Seal: A framework for secure mobile computations. In *Internet Programming Languages*, number 1686 in Lecture Notes in Computer Science. Springer, 1999.
- [VS97] D. Volpano and G. Smith. A type-based approach to program security. In *Proc. 7th Int'l Joint Conference on the Theory and Practice of Software Development*, number 1214 in Lecture Notes in Computer Science, pages 607–621. Springer, 1997.
- [VS98] D. Volpano and G. Smith. Secure information flow in a multi-threaded imperative language. In *Proc. of POPL'98*, pages 355–364. ACM Press, 1998.
- [VSI96] D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4(3):167–187, 1996.
- [WBS98] U. G. Wilhelm, L. Buttyà, and S. Staamann. On the problem of trust in mobile agent systems. In *Symposium on Network and Distributed System Security*. Internet Society, 1998.
- [Zim00] P. Zimmer. Subtyping and typing algorithms for mobile ambients. In *Proceedings of FoSSaCS'99*, volume 1784 of *LNCS*, pages 375–390. Springer, 2000.