Università Ca' Foscari di Venezia

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis: TD-2007-1

Secure Implementations of Typed Channel Abstractions

Marco Giunti

Supervisor
Prof. Michele Bugliesi

PhD Coordinator
Prof. Simonetta Balsamo

January, 2007

Author's Web Page: http://www.dsi.unive.it/~giunti

Author's e-mail: giunti@dsi.unive.it

Author's address:

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino, 155
30172 Venezia Mestre – Italia
tel. +39 041 2348411
fax. +39 041 2348419
web: http://www.dsi.unive.it

Abstract

The use of abstractions in the analysis of distributed computer systems permits to focus on the main features of the system while ignoring low-level details represented abstractly such as those inherent in the mechanism by which the components of the system communicate. In process calculi, the communication medium of distributed systems is abstracted by the notion of *channel* that ideally represents the address of the endpoints of communication. Further features of such systems as a client-server communication paradigm where clients and servers are allowed to read from and write on a resource, respectively, may be modeled by using typed abstractions and by relying on a type-checker to regulate the behavior of processes. However, such abstractions are reasonable only insofar as the low-level details do not hide particulars that compromise the feasibility of implementing the model.

In this dissertation, we show that the type-based discretional access control to communication channels of process calculi may be enforced in untyped, open, distributed environments by using cryptography without any assumption on the behavior of the processes running in such environments. We develop a typed pi calculus with matching in which capability types are employed to realize the policies for the access to communication channels, and we present implementations that compile the high-level processes of the pi calculus into low-level principals of a cryptographic process calculus based on the applied pi calculus. In these translations, the type capabilities of the high-level calculus are implemented as term capabilities protected by encryption keys only known to the intended receivers. As such, the implementation is effective even when the low-level compiled principals are deployed in open contexts, for which no assumption on trust and behavior may be made.

Our technique and results draw on, and extend, previous work on secure implementation of channel abstractions in a dialect of the join calculus [8]. In particular, our translation preserves the forward secrecy of communications in calculi which support the dynamic exchange of write *and/or* read access rights among processes. We establish the adequacy and full abstraction of the implementation by contrasting the untyped equivalences of the low-level cryptographic calculus, with the typed equivalences of the high-level source calculus.

Acknowledgments

I would like to thank Michele Bugliesi for his great involvement in the supervision of my studies. Michele has been a friendly and patient supervisor always available for discussions; he gave me good advises and taught me the rigour of research. Thanks to Sabina Rossi for encouraging me to understate the PhD program. I am grateful to my parents for having been very understanding and supporting during all my studies. My wife Angela has been so close and careful, giving me the strength to go on with my doctorate; thank you, love.

Contents

List of Tables

1

Introduction

In this thesis we study secure implementations of concurrent systems communicating through channel abstractions. Ideally, a concurrent system is formed by a set of computation units, named processes, that interact among each other by using communication links. We view such units as representation of distributed agents located on different machines connected by a network. The communication mechanism may be abstracted by interpreting the endpoints of communication as addresses that take the form of *channel names* from a countable set, and by representing communication as an atomic synchronization among a sender process and a receiver process, writing on and reading from the same channel, respectively. This permits to focus on the high-level properties of concurrent systems ignoring the low-level details of communication, e.g. routing, physical addressing.

The π-calculus [74] is one of the most studied models for concurrent systems based on channel abstractions. The expressions of the language are processes generated by a concise grammar that includes input and output name-prefixes representing the sender and the receiver process, respectively, a restriction operator to make channels private, and parallel composition representing concurrent computation. The dynamics of processes are described by a formal semantics that includes the rule for atomic synchronization mentioned above; an algebraic theory to manipulate processes and formally compare their behaviour is provided. Particularly, we study the asynchronous π-calculus [26, 65] that mainly differs from the π-calculus because of the output prefix has no continuation. Indeed, we believe that abstract models for concurrent distributed systems should consider an asynchronous communication paradigm in order to adequately represent communications arising in such systems.

The general question we address in this dissertation is whether the level of abstraction provided by process calculi is adequate for describing concurrent, distributed systems. The details ignored by the abstraction should not compromise the implementability of the model; a feasible implementation of the channel abstraction in distributed scenarios is therefore a fundamental issue to be settled. In this thesis, following a common approach (e.g., [1, 8]), we refer to implementation as a translation from an high-level distributed calculus using private channels to a low-level cryptographic calculus using non-private channels. Indeed, from an abstract point

of view, the low-level language models the open and distributed communications that characterize actual network protocols. To study implementations of the channel abstraction, it is therefore important to understand which is the communication paradigm modeled by the abstraction. Consider the communication rule of the pi calculus:

$$a\langle b\rangle \,|\, a(x).P \longrightarrow P\{b/x\} \qquad (*)$$

Since we interpret the output and input processes as a sender and a receiver, respectively, that are physically located in different machines, we see this rule as a model of point-to-point connection among two machines using a dedicated line.

Evidently, implementing distributed systems that communicate by using dedicated lines is hardly feasible. Rather, our implementations should use a public network as communication medium. Relying on the representation of the public network with a channel *net* known to the environment, we can implement the sender and the receiver in a subset of the pi calculus where the only free channel is *net* with little effort; to avoid confusions, we call this subset the low-level language and the full pi calculus the high-level language. For instance, we may use the channel name as an address inside the message, and add a filtering mechanism to the receiver. The following reduction on low-level processes[1] corresponds to the high-level reduction $(*)$:

$$net\langle b, a\rangle \,|\, net(x, y).(\mathsf{if}\ y = a\ \mathsf{then}\ P) \longrightarrow \mathsf{if}\ a = a\ \mathsf{then}\ P\{b/x\}$$

In the same vein, we can interpret the reduction $(*)$ as a communication in multicast by providing an implementation with replicated emissions of the form $!\, net\langle b, a\rangle$.

A more subtle question concerns the implementability of the restriction operator. Typically, this construct is used in process calculi to protect channels and the communications occurring on them. For instance the operator can prevent interferences in the execution of a protocol: we can make non-interruptible the communication of b in $(*)$ by restricting the communication channel.

$$(\mathsf{new}\ a)(a\langle b\rangle \,|\, a(x).P) \longrightarrow (\mathsf{new}\ a)(P\{b/x\}) \qquad (**)$$

In our view, the reduction $(**)$ models a communication occurring on a dedicated line such that the communication is invisible to the environment, i.e. the environment has not physical access to the line. Since we want to rely on public communications in our low-level model, the problem is to protect the low-level protocol from interferences without using private channels. A possible solution is to use cryptography to protect the content of the emission on the high-level channel; therefore our

[1] Clearly, the input process should also be instrumented with a recovery mechanism to handle all those cases in which it picks up "wrong packets" from the channel *net* representing the public network: at this stage, however, we omit this and other details for ease of readability.

low-level language should provide primitives to encrypt and decrypt messages. The high-level channel a is mapped into a couple of low-level cryptographic keys a^+, a^- such that the sender packages messages with the encryption key a^+ while the receiver uses the decryption key a^- to recover the content of the packet. The following synchronizations of low-level principals correspond to the high-level reduction $(**)$:

$$(\mathsf{new}\ a^+, a^-)net\langle\{b\}_{a^+}\rangle \mid net(y).(\mathsf{decrypt}\ y\ \mathsf{as}\ \{x\}_{a^-}\ \mathsf{in}\ P)$$
$$\longrightarrow \mathsf{decrypt}\ \{b\}_{a^+}\ \mathsf{as}\ \{x\}_{a^-}\ \mathsf{in}\ P$$

However, this does not prevent the low-level environment to notice that the communication in $(**)$ occurred, while $(*)$ is invisible to the high-level environment. For instance, the low-level environment may notice that the emission $net\langle\{b\}_{a^+}\rangle$ is not more available after the reduction $(**)$, and infer that $(**)$ occurred. While some traffic is practically always present on the network interfaces that link actual distributed systems, this rarely prevents information leaks through traffic analysis (e.g., due to message sizes, addresses). On the other hand, due to the model's limitations, we do not handle such attacks; therefore assuming the presence of a noise process creating endless fresh non-intelligible messages on the network interface represents a possible solution to this problem.

Protecting access to resources Next we investigate whether the typed abstractions used in process calculi to regulate the access to channels can be implemented in open, distributed systems. The need of controlling the access to channels it is often relevant both for design and security issues. From a security perspective, having knowledge of a channel name means possess a communication capability; in this vision, channels indeed represent the abstract resources of our concurrent systems. In the original pi calculus [74], the only way to protect a channel is by using the restriction operator, as we have seen in $(**)$: the restriction prevents the environment both to read from and write on the channel a.

However, the specification of modern computer systems usually require finer methods to protect access to resources (see [67, 18]). For instance, in systems based on a client-server communication paradigm it is often requested that the server eventually writes on a channel while a client eventually read from a channel. A model language suitable for the design of distributed systems should therefore provide the possibility to specify systems satisfying such constraints. In [88] this problem is tackled by using typed abstractions (see [90, 36]) that regulate the correct use of resources. The authors introduce a sorting discipline on channel names that separate the capability to *read* from channels from the capability to *write* on channels. Processes wishing to read from or writing on a channel need to possess the read or the write capability on that channel, respectively; a process may distribute capabilities on names only if it possesses them. The discipline is enforced by a type system that typechecks only processes obeying at the rules above. Assuming that all processes in the concurrent system are statically typed by the same type system,

one can deploy systems enjoying discretional access to resources by programming an adequate distribution of capabilities.

This leads to a crucial implementation problem, since this assumption is too strong for open, distributed systems. Such systems indeed are typically composed by several independent machines that are not administrated by an unique global authority. At the high-level, the access control is based on the use of a static type system that predicates that processes receiving a resource will not use it in a forbidden modality, i.e. typed contexts behave well. But what happens when we deploy such systems in contexts that do not respect the high level type assumptions? Stated more explicitly: can we compile our typed processes as low-level principals to run in distributed, open environments while preserving the high-level reasonings on the protection of resources we have established?

This appears to be an important question, as it constitutes a fundamental prerequisite to the use of typed process calculi as an abstract specification tool for concurrent computations in distributed, open systems.

Implementing typed processes in untyped contexts The objective of this dissertation is to provide a secure implementation of an abstract machine where the distribution of capabilities on channel abstractions is type-based, to low-level agents operating on a model of network where no assumptions on behaviour or trust can be made, and the correct distribution of capabilities is enforced by using cryptography. More in detail, we translate pi calculus processes using private channels and typed abstractions, into lower-level processes using open communications and cryptographic functions, and we prove that the implementation is secure.

The formal tool we use to establish the security of our implementations is the concept of *full abstraction* [1]; roughly, an encoding fully abstract equates encoded expressions if and only if the source expressions are equivalent. We say that two expressions are equivalent in a given language, noted $E_1 \cong E_2$, if yield the same results in all contexts of the language (cf. [45, 76]). Formally, a translation $[\![\cdot]\!]$ of expressions of a language H into expressions of a language L is said to be fully abstract if for all expressions E_1, E_2 of the language H we have (i) $[\![E_1]\!] \cong^L [\![E_2]\!]$ implies $E_1 \cong^H E_2$ and (ii) $E_1 \cong^H E_2$ implies $[\![E_1]\!] \cong^L [\![E_2]\!]$. As Abadi noted in [1], the direction (ii) is particularly relevant for security concerns, since it ensures that the translation does not introduce information leaks.

More in detail we want to preserve *typed equivalences* of the form $\mathrm{I} \models P \cong Q$ stating the indistinguishability (hence the equivalence) of P, Q in all enclosing contexts that satisfy the typed assumptions in I, i.e. all processes typechecked by I (cf. [23, 60, 91]). As we will argue, we believe that no fully abstract implementation may rely on static typing alone, as distributed and open networks do not validate any useful assumption on the trustworthiness, hence the well-typeness, of the contexts where (the low-level agents representing) our typed processes operate. Indeed requiring all low-level processes running in the system to be statically typed by

the same type system is a too strong assumption for distributed and open systems. Therefore, the low-level environment cannot be trusted to follow the type discipline. Rather than *assuming* that a context satisfies the constraints imposed by a typing assumption, our implementations should *enforce* them.

Enforcing access control with dynamic typing To make a fully abstract implementation feasible, the solution we propose here is to adopt a new typing discipline for the source calculus, based on a combination of static and dynamic typing to control the interaction with the context. Dynamic typing is an appealing solution when dealing with computing in wide-area distributed systems; static verification of such systems is impractical since the typing information may be partial or missing and the components of the system may be unknown or partially known to each other. To avoid to stop the execution and re-check, components must dynamically carry sufficient behavioral information that can be checked at runtime by the system's components (e.g., [82]).

We formalize our approach by developing a typed version of the (asynchronous) pi calculus where the values emitted in the output construct are coerced with the intended type to be released, noted $p\langle a@T\rangle$, regardless of the type of p. A static typing system guarantees that a may indeed be assigned the coercion type T, while a mechanism of dynamically typed synchronization guarantees that a is received only at types less informative than T, so as to guarantee type soundness of each exchange.

By breaking the dependency between the types of the transmission channels and the types of the names transmitted, distinctive of the approaches to typing in the pi calculus tradition [88, 60], we can safely reduce the capability types to the simplest, flat structure that only exhibits the read/write access rights on channels, regardless of the types of the values transmitted, namely $T \in \{\mathsf{rw}, \mathsf{r}, \mathsf{w}\}$. Furthermore, and more interestingly for our present concerns, the combination of type coercion and dynamically typed synchronization allows us to gain further control on the interactions among processes as well as between processes and their enclosing context.

To illustrate, we program in this language a process that releases the read capability of a resource a that we assume unknown to the environment, and provides a receiver on the same channel a for internal use: $p\langle a@\mathsf{r}\rangle \mid a(x).\mathbf{0}$. The intention of the specification is clear, and best formalized by the following typed equation stating the invisibility of the read resource a:

$$p : T \models p\langle a@\mathsf{r}\rangle \mid a(x).\mathbf{0} \;\cong\; p\langle a@\mathsf{r}\rangle$$

Dynamic typing will prevent processes of the form $P = p(x@\mathsf{w}).P'$ to receive the resource a and use it for write access, revealing the presence of $a(x).\mathbf{0}$. Indeed, in this example the process P asks for a name to be used in write modality and, in contrast, the channel a is released in read modality; therefore dynamic typing does not permit the synchronization among $p\langle a@\mathsf{r}\rangle$ and P.

As the example shows, the knowledge of the environment on p is irrelevant to establishing the equation above; this is the main difference from our approach and that of [88, 60]. We can therefore rely on a adequate programming of the distribution of capabilities in order to guarantee that the system does not release more capabilities than the intended ones.

1.1 Main contributions

In this thesis we provide a fully abstract implementation of a typed pi calculus with matching into a lower-level calculus using open communications and cryptography. Since our high-level process calculus is a conservative extension of the untyped pi calculus, we also have a direct secure implementation of the pi calculus with matching.

This result shows that the dynamic management of capabilities enforced at the high level by using a global type system, can be effectively enforced in untyped, open, distributed environments by using cryptography without any assumption on the behavior or trust of the low-level contexts. To the best of our knowledge, this is the first result of this kind for typed process calculi. Further, we resolved the problem of forward secrecy of communications (see [1]) that is the main obstacle to the implementation of process calculi which supports the dynamic exchange of write and/or read access rights among processes. Particularly, we devised the first secure implementation of the untyped pi calculus with matching; an implementation of the pi calculus without matching already exists (cf. [51, 8]).

We proposed a typed version of the pi calculus, namely the API@ calculus, that, in our opinion, can be useful to model the distribution of capabilities on abstract resources; we believe that the declarative style characterizing our calculus and type theory can make easier and less error-prone the hard task of specifying distributed systems where the access to resources is discretional. We provided an observational theory for the calculus, and we gave a coinductive proof technique for observational equivalence.

Publications

The API@ calculus, it's observational theory and a proof technique for observational equivalence have been published in [29]. The secure implementations of the API@ calculus have been published in [30].

1.2 Structure of the thesis

In Chapter 2 we introduce syntax and operational semantics of API@ processes and we prove type soundness. Then we introduce a notion of contextual equivalence among processes, namely typed behavioral equivalence, and we provide it with a

coinductive proof technique. A few equations characterizing the calculus are draw. Finally we relate the API@ calculus with the (asynchronous version of the) pi calculus of [60] by providing a sound translation of the former in to the latter. Comparison with related work is available at the end of the chapter. This chapter is based on [29].

In Chapter 3 we review the implementation language that is an asynchronous version of the applied pi calculus [7]. We introduce semantics for the destructors of interest that include functions implementing symmetric/asymmetric encryption and digital signatures. Next we review the syntax and the operational semantics of applied pi processes. We assume that destructors are only used in let-expressions and may not occur in arbitrary terms; the semantics is given in terms of labelled transitions and does not rely on active substitutions as in the original formulation.We define a notion of term-indexed behavioral equivalence among processes and we prove some useful closure properties. Finally we review some related work.

Chapter 4 contains implementations of API@ calculus in the applied pi calculus. We first review the main obstacles arising in implementing fully abstract the pi calculus and in particular the problem of preserving the forward secrecy of communications. Then we describe the implementation framework, and we present a first compositional compilation of high-level processes into low-level ones. We show that the implementation is sound by establishing the operational correspondence of the translation. Next, we enhance the design by giving a fully abstract implementation that relies on a proxy server to prevent compiled processes to have undesired interactions with low-level, possibly malicious contexts. Comparison with related work is analyzed at the end of the chapter. This chapter is based on [30].

Chapter 5 contain the proofs of the results of Chapter 4; an up-to technique to prove the operational correspondence of "non-prompt" [84] translations is given. Chapter 6 extends the fully abstract centralized implementation provided in Chapter 4 by introducing a distributed implementation based on the partitioning of the network in domains each of which is administrated by a proxy server. To model this partitionining of the network, we extend the calculus presented in Chapter 2 by labelling processes with domain labels; we let domains have not impact on the dynamics and/or the typing of the API@ calculus. We define a notion of contextual equivalence that is closed under composition of known domains. Finally we provide a fully abstract translation of the high-level processes labelled with domains into low-level ones. The distributed implementation has been published in [30].

2

A pi calculus with dynamic typing for resource access control

In this chapter we introduce the high level language that we use to specify concurrent systems supporting the dynamic exchange of write and/or read access rights among processes. The language is a typed variant of the pi calculus [74, 72, 104, 73] with a matching construct and polyadic communications, and is a conservative extension of the untyped pi calculus. We provide the calculus with a typed equational theory, and we give a co-inductive proof technique useful to prove equivalences among processes. The technique relies on labelled transitions on configurations of the form $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ where α range over typed input/output actions and silent actions, and I is a type environment less informative than the environment that typechecks P. Basically input and output transitions arise only if they are allowed by I; silent transitions are invisible to the environment. Finally we compare our process calculus with the pi calculus of Hennessy and Rathke [60] which embeds input/output types generalizing those of [88], and we provide a sound encoding of the former into the latter. The main results of this chapter has been published in [29].

Background

The use of types to control the behavior of processes in the pi calculus is a long known and well established technique. The idea was first introduced by Pierce and Sangiorgi in their seminal work on the subject [88], and is best illustrated by their motivating example:

$$S = (\mathsf{new}\ s)!d\langle s\rangle \mid !s(x).print\langle x\rangle \qquad C = d(x).x\langle j\rangle$$

S is a print spooler serving requests from a private channel s that it communicates to its clients via the public channel d. C is one such client, that receives s and uses it to print the job j.

While the intention of the specification is clear, reasoning on its properties is subtler. For instance, given the initial configuration $S \mid C$, can we prove that the jobs sent by C are eventually received and printed? Stated in more formal terms:

is there a proof of the following equation?

$$S \mid C \quad \stackrel{?}{\cong} \quad S \mid print\langle j \rangle \tag{2.1}$$

Here we take $P \cong Q$ to mean that P and Q are behaviorally indistinguishable, i.e. they have the same observable behavior when executed in any arbitrary context. Back to our example, (2.1) is easily disproved by exhibiting a context that interferes with the intended protocol between S and C. A first example is the context $\mathscr{C}_1[-] = - \mid d(x).!x(y).\mathbf{0}$, that initially behaves as a client, to receive s, but then steals the jobs intended for S. A second example is the context $\mathscr{C}_2[-] = - \mid (\mathsf{new}\ s')d\langle s' \rangle$, which may succeed in transmitting to C a dead-ended channel that will never serve the purpose C expected of it.

As shown in [88], hostile contexts such as those above can be ruled out by resorting to a system of capability types to control the transmission and/or reception of values over channels based on the possession of corresponding type capabilities. In our example, that system allows us to protect against contexts like $\mathscr{C}_1$ by requiring that clients be only granted write capabilities on the channel s, and by reserving read capabilities on s to the spooler. Similarly, we may build safeguards against attackers like $\mathscr{C}_2$ by demanding that clients only have read access on d.

Both the requirements are expressed formally by the typing assumption $d : ((T)^{\mathsf{w}})^{\mathsf{r}}$: this typing grants read-only access on d and write-only access to any name received on d, as desired. We may now refine the equation in (2.1) into its typed version below

$$d : ((T)^{\mathsf{w}})^{\mathsf{r}} \ \models \ S \mid C \ \cong \ S \mid print\langle j \rangle. \tag{2.2}$$

Typed equations of the form $\mathrm{I} \models P \cong Q$ express behavioral equivalences between processes in any context that typechecks in the type environment I. Here I represents the context's view of the processes under observation, given in terms of a set of typing assumptions on the names shared between the processes and the context itself. Incidentally, but importantly, the typing assumptions on the shared names may in general be different –in fact, more accurate– for the processes than they are for the context. To illustrate, in (2.2), a context is only assumed to have read-capabilities on d, while for the system to typecheck the name d must be known at the lower (hence more accurate) type $((T)^{\mathsf{w}})^{\mathsf{rw}}$, so that to allow S to write and C to read. Similarly, for the system $S \mid C$ to typecheck, the name s must be known at the type $(T)^{\mathsf{rw}}$ including both a write-capability, granted to S, and read-capability, granted to any process that receives s: the context, instead, will only acquire s at the super-type $(T)^{\mathsf{w}}$ determined by the type of the transmission channel d.

Typed processes in untyped contexts

Given the type for d available to the context, it is not difficult to be convinced that (2.2) above (under appropriate hypotheses on the context's view of the name *print*,

see Section 2.3) represents a valid equivalence as no context that typechecks under $d : ((T)^{\mathsf{w}})^{\mathsf{r}}$ may tell the two processes apart.

Typed equivalences like these are very useful, and effective in all situations in which we have control on the contexts observing our processes, i.e. in all situations in which we may assume that such contexts are well-typed, hence behave according to the invariants enforced by the typing system.

The question we address in this thesis is whether the same kind of reasoning can still be relied upon when our processes are to be deployed in distributed, open environments. Stated more precisely: can we implement our typed processes as low-level agents to be executed in arbitrary, open networks, while at the same time preserving the typed behavioral congruences available for the source processes?

We argue that no implementation with the desired properties may rely on static typing alone, as distributed and open networks do not validate any useful assumption on the trustworthiness, hence the well-typeness, of the contexts where (the low-level agents representing) our typed processes operate. Rather than *assuming* that a context satisfies the constraints imposed by a typing assumption, our implementations should *enforce* them.

The implementation schema we envision here is one in which the statically checked possession and distribution of type capabilities in the source-level processes is realized in terms of the possession and the dynamic distribution of corresponding term-level capabilities in the implementation agents. For instance, each channel could be implemented by means of a pair of cryptographic keys representing the write and read capabilities. If designed carefully, and instrumented with adequate measures to protect against hostile contexts (see [1, 8], Chapter 4) this represents a viable idea to pursue.

The problem remains, however, to make sure that the implementation preserves the desired typed equations of the source calculus: for that to be the case, one must guarantee that for each name, the distribution of the term capabilities in the low-level agents match the corresponding type capabilities in the source level process. While this is possible for the names that are statically shared with the context, ensuring such correspondence is much harder, if at all possible, for the names that are dynamically acquired by the context. There are two fundamental difficulties:

- first, as we have observed, the type at which the context acquires a name depends on the type of the channel over which the name is communicated;
- secondly, the type of the transmission channel may vary dynamically in ways that cannot be predicted statically.

The dynamic evolution is particularly problematic in a calculus with matching because, as noticed by Hennessy and Rathke in [60], matching makes it possible to progressively refine the type at which a name is known during the computation. This is best illustrated by the following typed labelled transition, borrowed from

[60], that formalizes the effect of emitting a name on a public channel.

$$\frac{\mathrm{I}^{\mathsf{r}}(a)\downarrow}{\mathrm{I}\triangleright a\langle n\rangle.P \xrightarrow{a\langle n\rangle} \mathrm{I}\sqcap n:\mathrm{I}^{\mathsf{r}}(a)\triangleright P}$$

The configuration $\mathrm{I}\triangleright P$ represents a process P operating in a context that typechecks under I, and $\mathrm{I}^{\mathsf{r}}(a)\downarrow$ indicates that I (hence the context) has a read capability on the name a. The meet $\mathrm{I}\sqcap n:\mathrm{I}^{\mathsf{r}}(a)$ in the resulting configuration represents the ability of the context to "merge" its current type for n with the type determined by receiving n on a: as explained in [60], taking the meet mimics the ability of the context to match n with the names already known to it (possibly, at different types), and obtain a more informative type based on that. For instance if I contains the typing assumptions $n:(S)^{\mathsf{r}}, a:((S)^{\mathsf{w}})^{\mathsf{r}}, q:S$, then the context $\mathscr{C}[-]=a(x).[x=n]\,n\langle q\rangle\,|\,n(y).\mathbf{0};\mathbf{0}$ is typechecked by the typing system of [60]; here the ability to check if the channel received through a at type $(S)^{\mathsf{w}}$ is equal to the channel n known at type $(S)^{\mathsf{r}}$ let the context use both the read and the write capability of n. The problem is that the effect of this type refinement may propagate dynamically in ways that cannot be determined statically. To illustrate, let I and P be the typing environment and the process defined as follows:

$$\begin{aligned}\mathrm{I} &= n:((T)^{\mathsf{w}})^{\mathsf{r}}, a:(((T)^{\mathsf{r}})^{\mathsf{r}})^{\mathsf{r}}\\ P &= a\langle n\rangle\,|\,(\mathsf{new}\ p:(T)^{\mathsf{rw}})n\langle p\rangle\end{aligned}$$

We have $\mathrm{I}^{\mathsf{r}}(n)=(T)^{\mathsf{w}}$ and $\mathrm{I}^{\mathsf{r}}(a)=((T)^{\mathsf{r}})^{\mathsf{r}}$, and from this we compute $\mathrm{I}'\triangleq \mathrm{I}\sqcap n:\mathrm{I}^{\mathsf{r}}(a)=a:(((T)^{\mathsf{r}})^{\mathsf{r}})^{\mathsf{r}}, n:((T)^{\mathsf{rw}})^{\mathsf{r}}$. Now we see that a context that typechecks under I will acquire p at the type $(T)^{\mathsf{w}}$ or at the type $(T)^{\mathsf{rw}}$ depending on which one of the two names p and n it receives first in its interactions with process P. This is reflected by the following two transition sequences available from $\mathrm{I}\triangleright P$.

$$\begin{array}{lllll}\mathrm{I}\triangleright P & \xrightarrow{(\mathsf{new}\ p)n\langle p\rangle} & \mathrm{I}, p:(T)^{\mathsf{w}}\triangleright a\langle n\rangle & \xrightarrow{a\langle n\rangle} & \mathrm{I}\sqcap n:((T)^{\mathsf{r}})^{\mathsf{r}}, p:(T)^{\mathsf{w}}\triangleright \mathbf{0}\\ \mathrm{I}\triangleright P & \xrightarrow{a\langle n\rangle} & \mathrm{I}'\triangleright(\mathsf{new}\ p:(T)^{\mathsf{rw}})n\langle p\rangle & \xrightarrow{(\mathsf{new}\ p)n\langle p\rangle} & \mathrm{I}', p:(T)^{\mathsf{rw}}\triangleright\mathbf{0}\end{array}$$

Clearly, this dynamic evolution of the typing knowledge of the context is problematic, as a fully abstract implementation would need to tune the distribution of the term-capabilities associated with n on the types available dynamically to the context: as the example shows, this is in general impossible to achieve at compile time, by simply inspecting the structure of the source level processes.

Enforcing access control with dynamic typing

While the problem with the previous example is a direct consequence of the presence of matching, a more fundamental obstacle against full abstraction is in the very

structure of the capability types adopted in the source calculus, and in the way that structure determines the acquisition of new capabilities on a name. As we have shown, acquiring a name, say n, at a type not only informs on how n will be used, but also determines how other names transmitted over n will be circulated and used in the system. These invariants are all encoded in the static type of the channel at which n is received, and clearly they will not be guaranteed if that channel is shared with an untyped context.

To make a fully abstract implementation feasible, the solution we propose here is to adopt a new typing discipline for the source calculus, based on a combination of static and dynamic typing to control the interaction with the context.

We formalize our approach by introducing a typed variant of the (asynchronous) pi calculus. In this calculus, named API@, the types at which the emitted values are to be received by the context are decided by the output sites. This is accomplished by introducing a new output construct, noted $a\langle v@T\rangle$, that relies on type coercion to enforce the delivery of v at the type T, regardless of the type of the communication channel a. A static typing system will ensure that v has indeed the type T to which it is coerced, while a mechanism of dynamically typed synchronization guarantees that v is received only at supertypes of T, so as to guarantee the type soundness of each exchange of values.

By breaking the dependency between the types of the transmission channels and the types of the names transmitted, in API@ we may safely dispense with the nested types of [88, 60], and rely instead on channel types with a flat structure that only exhibits the read/write access rights associated with the channels, regardless of the types of the values they transmit. Needless to say, the resulting discipline of static typing is much looser: to compensate for that one then needs a dynamically typed operational semantics to ensure type soundness.

What is more interesting and relevant for our present concerns is that the new typing discipline makes it possible to recover fully abstract implementations, i.e. implementations for which the typed congruences of the source calculus are preserved even in the presence of untyped, and potentially hostile contexts. We show how this can be accomplished in Chapter 4. Before that, we introduce the source calculus formally, and look at the consequences of the new typing discipline on the ability to reason about process behavior.

2.1 Syntax of API@ calculus

Given the intended use of calculus as a specification language for distributed systems, we opt for an asynchronous version of the calculus. However, the same results and technique would apply, mutatis mutandis, to the synchronous case.

We presuppose countable sets of names and variables, ranged over by $a - n$ and $x, y, \ldots$ respectively. We use u, v to range collectively over names and variables whenever the distinction does not matter. The structure of processes is defined by

the following productions:

$$
\begin{array}{lcll}
P, Q, \ldots & ::= & \mathbf{0} \mid P \,|\, Q \mid (\mathsf{new}\ n : A)P \mid \,!\, P & \textit{pi calculus} \\
& \mid & [u = v]\, P; Q & \textit{matching} \\
& \mid & u\langle \tilde{v} @ \tilde{A} \rangle & \textit{type-coerced output} \\
& \mid & u(\tilde{x} @ \tilde{A}).P & \textit{typed input}
\end{array}
$$

We use $\tilde{u}$ and $\tilde{A}$ to note (possibly empty) tuples of values and types, respectively, and the notation $\tilde{v}@\tilde{A}$ as a shorthand for $v_1@A_1, \ldots, v_n@A_n$.

The reading of the process forms is standard, with the exception of the constructs for input/output. The process $u\langle \tilde{v}@\tilde{A} \rangle$ represents the polyadic output of values $v_1, \ldots, v_n$ respectively at the types $A_1, \ldots, A_n$: the rules of the operational semantics will ensure that outputs at this types only synchronizes with input prefixes expecting values at types respectively higher than (or equal to) $A_1, \ldots, A_n$. The input process $u(\tilde{x} : \tilde{A}).P$ acts as a binder for the variables $x_1, \ldots, x_n$. The inert process is represented by $\mathbf{0}$; concurrent computation is expressed by using parallel composition $P \,|\, Q$. The matching process $[u = v]\, P; Q$ behaves as a conditional that proceeds as P if $u = v$, and as Q otherwise. We often omit trailing inert processes and abbreviate $[u = v]\, P; \mathbf{0}$ with $[u = v]\, P$. Process $(\mathsf{new}\ n : A)P$ acts as a binder of the name n in P; the type information is used to ensure that n is used consistently in P. Finally $!\, P$ represents the replication of P. We let the free names of P, noted $\mathit{fn}(P)$ be the set of names occurring in P that are not bound; similarly the free variables of P, noted $\mathit{fv}(P)$ are the variables not bound in P. We assume a well-defined capture-free substitution operation $P\{v/x\}$ which allow the value v to be substituted to x in the process P. We use the notation $P\{\tilde{v}/\tilde{x}\}$ to indicate nested substitutions applications $((P\{v_1/x_1\})\ldots)\{v_n/x_n\}$. A process with no free variables is said to be closed. We let two processes P, Q be α-equivalent, noted $P = Q$, when Q may from obtained by P by a finite number of changes of bound names and/or bound variables.

The types of API@ include a top type and capability types for names. As introduced in Chapter 1, the channel types we use have the flat structure defined in Table 2.1 that only exhibits the read/write access rights on channels. The types $\mathsf{ch}(\cdot)$ are the types of channels, built around the capabilities r, w and rw which provide *read*, *write* and *full fledged* access to the channel, respectively. To ease the notation, we henceforth denote channel types by only mentioning the associated capabilities. $\top$ is the type of all values and may be used to export a name without providing any capability for synchronization or exchange. The ability of match identity of names still give a distinguishing power to contexts knowing a name at type top [104].

The subtyping relation $<:$, also in Table 2.1, is the preorder that satisfies the expected relationship over capability types, and admits a meet operator; we will often write $T :> S$ to indicate that $S <: T$. A binary operator over the types introduced above is said to be a meet if it is commutative, associative and satisfies (i) $A <: B$ and $A <: C$ implies $A <: B \sqcap C$ and (ii) $A \sqcap B <: A$. We define the

Table 2.1 The typing system

Types

$$A, B, \ldots \quad ::= \quad \mathsf{ch}(\mathsf{rw}) \mid \mathsf{ch}(\mathsf{r}) \mid \mathsf{ch}(\mathsf{w}) \qquad \textit{capabilities}$$
$$\mid \quad \top \qquad \textit{top}$$

Subtyping

$$\frac{}{\mathsf{ch}(\mathsf{rw}) <: \mathsf{ch}(\mathsf{r})} \qquad \frac{}{\mathsf{ch}(\mathsf{rw}) <: \mathsf{ch}(\mathsf{w})} \qquad \frac{A \in \{\mathsf{ch}(\mathsf{rw}), \mathsf{ch}(\mathsf{w}), \mathsf{ch}(\mathsf{r}), \top\}}{A <: \top}$$

Typing Rules

(T-Pro)
$$\frac{\Gamma(u) <: A}{\Gamma \vdash u : A}$$

(T-Nil)
$$\frac{}{\Gamma \vdash \mathbf{0}}$$

(T-New)
$$\frac{\Gamma, n : A \vdash P}{\Gamma \vdash (\mathsf{new}\ n : A)P}$$

(T-Par)
$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \,|\, Q}$$

(T-Repl)
$$\frac{\Gamma \vdash P}{\Gamma \vdash\, !P}$$

(T-Match)
$$\frac{\Gamma \vdash Q \quad \Gamma \vdash u : A \quad \Gamma \vdash v : B \quad \Gamma \sqcap u : B \sqcap v : A \vdash P}{\Gamma \vdash [u = v]\, P; Q}$$

(T-Out@)
$$\frac{\Gamma^{\mathsf{w}}(u) \downarrow \quad \Gamma \vdash v_1 : B_1, \ldots, \Gamma \vdash v_n : B_n}{\Gamma \vdash u\langle \tilde{v} @ \tilde{B} \rangle}$$

(T-In@)
$$\frac{\Gamma^{\mathsf{r}}(u) \downarrow \quad \Gamma, \tilde{x} : \tilde{A} \vdash P}{\Gamma \vdash u(\tilde{x} @ \tilde{A}).P}$$

operator $\sqcap$ as follows:

$$\begin{array}{lll} \mathsf{rw} \sqcap \mathsf{r} \triangleq \mathsf{rw} & \mathsf{rw} \sqcap \mathsf{w} \triangleq \mathsf{rw} & \mathsf{w} \sqcap \mathsf{r} \triangleq \mathsf{rw} \\ A \sqcap \top \triangleq A & A \sqcap A \triangleq A & A \sqcap B \triangleq B \sqcap A \end{array}$$

Proposition 1. *The operator $\sqcap$ defined above is a meet operator.*

Proof. Follows easily by a case analysis. □

The following propositions illustrate easy but useful results for the meet operator $\sqcap$.

Proposition 2. *If $A <: B$ then $A \sqcap B = A$.*

Proposition 3. *$A <: B$ implies $(A \sqcap C) <: (B \sqcap C)$.*

Typing System

We introduce a few preliminary definitions to formalize the structure of typing environments. Type environments, ranged over by Γ, Δ, are finite mappings from names and variables to types. We use the notation $\Gamma, u : A$ to indicate the mapping $\Gamma \cup (u, A)$. Following [60], we extend type environments by using the meet operator: $\Gamma \sqcap u : A = \Gamma, u : A$ if $u \notin \mathrm{dom}(\Gamma)$, otherwise $\Gamma \sqcap u : A = \Gamma'$ with Γ' differing from Γ only at u: $\Gamma'(u) = \Gamma(u) \sqcap A$. We let $(\Gamma, u : A) \setminus u \triangleq \Gamma$.

Differently from [60], extending a type environment is always a defined operation since in our subtyping relation the meet is not partial (as in [60]). When extending a type environment with tuples, we often omit to indicate parentheses and write $\Gamma \sqcap \tilde{v} : \tilde{A}$ to indicate $((\Gamma \sqcap v_1 : A_1) \sqcap \ldots) \sqcap v_n : A_n$, and in case $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \varnothing$, we write Γ, Γ' to indicate the type environment containing all mappings in Γ and in Γ'. Subtyping is extended to type environments as expected: $\Gamma <: \Delta$ whenever $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Delta)$ and for all $a \in \mathrm{dom}(\Gamma)$ we have $\Gamma(a) <: \Delta(a)$. If $\Gamma(n) <: \mathsf{r}$, we say that $\Gamma^{\mathsf{r}}(n)$ is defined, written $\Gamma^{\mathsf{r}}(n) \downarrow$. Correspondely, we let $\Gamma^{\mathsf{w}}(n) \downarrow$ whenever $\Gamma(n) <: \mathsf{w}$.

Most of the typing rules in Table 2.1 are standard and self explained. Rule (T-Pro) ensures that environments type values at supertypes of their actual types. Rule (T-Par) says that the composition of processes typed independently by the same environment is well-typed. The typing of matching process $[u = v]\, P; Q$ is inherited from [60]. Differently from companion type systems for pi calculi with matching (see [104]), we do not require the environment to possess both the read and the write capability in order to test a value, i.e. u, v may be known at types higher than rw. Similarly, we do not insist that u, v have identical types to infer the well-typeness of the matching process. Indeed u, v may be bound to the same value and represent distinct capabilities; in such case the matching process should continue as P and collate the type information about the values. This is done in rule (T-Match) by implicitly casting the types of u, v to their greatest common

subtype. To illustrate, as a result of that rule, the following judgment is derivable: $x : \mathsf{r}, y : \mathsf{w} \vdash [x = y]\, x\langle\rangle; \mathbf{0}$.

The typing of input/output is characteristic of our present calculus, and is a direct consequence of the structure of the channel types. In particular, notice that the rules (T-OUT@) and (T-IN@) do not expect/impose any relationship between the type of the channel u and the types associated with the values transmitted (in (T-OUT@)) or expected (in (T-IN@)). The only constraint, at the output sites, is that the types at which the emitted values are coerced must be valid. As we mentioned earlier, this rather loose form of static typing is complemented by dynamic type checks to be performed upon synchronization.

The following propositions state useful properties of the typing of values.

Proposition 4 (Weakening). *If* $\Gamma \vdash u : A$ *then* $\Gamma, v : B \vdash u : A$.

Proof. The typing $\Gamma \vdash u : A$ has been inferred from (T-PRO); from this we infer $\Gamma(u) <: A$. From $(\Gamma, v : B)(u) <: A$ and a further application of (T-PRO) we obtain the desired result.

Proposition 5 (Strengthening). *If* $\Gamma, v : B \vdash u : A$ *and* $u \neq v$ *then* $\Gamma \vdash u : A$.

Proof. In this case $\Gamma, v : B \vdash u : A$ has been inferred from $\Gamma(u) <: A$, because $u \neq v$; we apply (T-PRO) and obtain $\Gamma \vdash u : A$, as desired.

Proposition 6 (Subtyping). *If* $\Gamma \vdash v : A$ *and* $\Gamma' <: \Gamma$ *then* $\Gamma' \vdash v : A$.

Proof. From (T-Pro) we know that $\Gamma(v) <: A$. From $\Gamma' <: \Gamma$ we easily obtain $\Gamma'(v) <: \Gamma(v)$ since or $v = a$ and $\Gamma'(a) <: \Gamma(a)$ holds by definition, or $v = x$ and $\Gamma'(x) = \Gamma(x)$. From the transitivity of $<:$ we infer $\Gamma'(v) <: A$. We apply (T-Pro) obtaining $\Gamma' \vdash v : A$. □

Proposition 7. *If* $\Gamma \vdash P$ *then* $\Gamma \sqcap \tilde{v} : \tilde{B} \vdash P$.

Proof. The proof proceeds by induction on the type inference and use weakening and subtyping for values. Indeed $\Gamma \sqcap \tilde{v} : \tilde{B} = \Gamma', \Delta$ for some (possibly empty) type environment Δ and for some $\Gamma' <: \Gamma$. We leave the considerable proof details to the reader. □

An important property of the type inference system is that typings are preserved by the substitution of values into variables. We let $\Gamma\langle v/x\rangle$ be Γ whenever $x \notin \mathrm{dom}(\Gamma)$ and be $\Delta \sqcap v : A$ whenever $\Gamma = \Delta, x : A$; $\Gamma\langle\tilde{v}/\tilde{x}\rangle$ is defined similarly.

Lemma 8 (Substitution). *Suppose* $\Gamma \vdash v : A$.

1. *(Values) If* $\Gamma, x : A \sqcap \Delta \vdash u : B$ *then* $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u\{v/x\} : B$
2. *(Processes) If* $\Gamma, x : A \sqcap \Delta \vdash P$ *then* $\Gamma \sqcap \Delta\langle v/x\rangle \vdash P\{v/x\}$.

Proof. The first part is proved by a case analysis, the second part is proved by induction on length of the inference of the type judgment.

Values For the first part of the statement, we analyze the rule for typing values.

From (T-Pro) we have that $(\Gamma, x : A \sqcap \Delta)(u) <: B$. There are two cases corresponding to (a) $v \in \mathrm{dom}(\Delta\langle v/x\rangle)$ or (b) not. In case (b) we easily infer $x \notin \mathrm{dom}(\Delta)$ and in turn $\Gamma \sqcap \Delta\langle v/x\rangle = \Gamma \sqcap \Delta$. There are two sub-cases corresponding to (i) $x \neq u$ or (ii) $x = u$. In sub-case (i) the substitution is vacuous. From the hypothesis $\Gamma, x : A \sqcap \Delta \vdash u : B$ and strengthening we infer $\Gamma \sqcap \Delta \vdash u : B$, as desired. In sub-case (ii) the type inference has the form $\Gamma, x : A \sqcap \Delta \vdash x : B$; from (T-Pro) we deduce $A <: B$ since we showed that in the current case (b) under analysis we have $x \notin \mathrm{dom}(\Delta)$. From $\Gamma(v) <: A$ we infer $\Gamma(v) <: B$. From $\sqcap$ being a meet operator we infer $(\Gamma \sqcap \Delta)(v) <: A$. We apply (T-PRo) and infer the desired result, $\Gamma \sqcap \Delta \vdash v : B$.

Suppose case (a) holds. There are two sub-cases corresponding to (i) $u \neq x$ or (ii) $u = x$. In sub-case (i) the substitution is vacuous. By applying strengthening (Prop. 5) to the hypothesis we infer $\Gamma \sqcap (\Delta \setminus x) \vdash u : B$. Since $\Delta\langle v/x\rangle$ is equal to Δ whenever $x \notin \mathrm{dom}(\Delta)$ and otherwise it is defined as $(\Delta \setminus x) \sqcap v : \Delta(x)$, we apply Proposition 6 and infer $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u : B$. In sub-case (ii) the type inference has the form $\Gamma, x : A \sqcap \Delta \vdash x : B$. From (T-Pro) we infer $A \sqcap \Delta(x) <: B$ or $A <: B$ (in case $x \notin \mathrm{dom}(\Delta)$). In case $A \sqcap \Delta(x) <: B$ we infer that $(\Gamma \sqcap ((\Delta \setminus x) \sqcap v : \Delta(x)))(v) <: (\Gamma(v) \sqcap \Delta(x))$; from $\Gamma(v) <: A$ we infer $(\Gamma \sqcap ((\Delta \setminus x) \sqcap v : \Delta(x)))(v) <: B$. We apply (T-Pro) and obtain $\Gamma \sqcap \Delta\langle v/x\rangle \vdash v : B$. In the remaining case $x \notin \mathrm{dom}(\Delta)$ from $(\Gamma \sqcap \Delta)(v) <: A$, $A <: B$ and (T-Pro) we easily obtain the desired result, $\Gamma \sqcap \Delta \vdash v : B$.

Processes For the second part of the statement, we analyze the rules for typing processes. (T-Nil) is trivial. (T-New), (T-Par), (T-Repl) are similar; we show (T-New), the remaining ones are easier.

(T-New) We have $\Gamma, x : A \sqcap \Delta \vdash (\mathsf{new}\ n : B)P$ which in turn is derived from $(\Gamma, x : A \sqcap \Delta), n : B \vdash P$ by assuming $n \notin \mathrm{dom}(\Gamma, \Delta)$ (eventually by alpha-renaming n). From $n \notin \mathrm{dom}(\Gamma, \Delta)$ we infer $(\Gamma, x : A \sqcap \Delta), n : B = (\Gamma, n : B, x : A) \sqcap \Delta$. By the hypothesis $\Gamma \vdash v : A$ and weakening (Proposition 4) we obtain $\Gamma, n : B \vdash v : A$ (notice $v \in \mathrm{dom}(\Gamma)$ and therefore $n \neq v$). By induction we have that $\Gamma, n : B \sqcap \Delta\langle v/x\rangle \vdash P\{v/x\}$. From $n \notin \mathrm{dom}(\Gamma, \Delta)$ we infer $\Gamma, n : B \sqcap \Delta\langle v/x\rangle = (\Gamma \sqcap \Delta\langle v/x\rangle), n : B$. We apply (T-New) and infer the desired result, $\Gamma \sqcap \Delta\langle v/x\rangle \vdash (\mathsf{new}\ n : B)P\{v/x\}$.

The interesting case is matching.

(T-Match) We have $\Gamma, x : A \sqcap \Delta \vdash [u_1 = u_2]\, P; Q$ since there are types A_1, A_2 such that

1. $\Gamma, x : A \sqcap \Delta \vdash u_1 : A_1$
2. $\Gamma, x : A \sqcap \Delta \vdash u_2 : A_2$

3. $\Gamma, x : A \sqcap \Delta \vdash Q$

4. $\Gamma, x : A \sqcap \Delta' \vdash P$

where $\Delta' = \Delta \sqcap u_1 : A_1 \sqcap u_2 : A_2$ (the associativity of meet let $\Gamma, x : A \sqcap \Delta'$ be equal to $((\Gamma, x : A \sqcap \Delta) \sqcap u_1 : A_1) \sqcap u_2 : A_2)$). We apply the first part of the proposition or induction and obtain

(i) $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u_1\{v/x\} : A_1$

(ii) $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u_2\{v/x\} : A_2$

(iii) $\Gamma \sqcap \Delta\langle v/x\rangle \vdash Q\{v/x\}$

(iv) $\Gamma \sqcap \Delta'\langle v/x\rangle \vdash P$

Let $\mathrm{I} = ((\Gamma \sqcap \Delta\langle v/x\rangle) \sqcap u_1\{v/x\} : A_2) \sqcap u_2\{v/x\} : A_1$. Notice that $\mathrm{I} = \Gamma \sqcap \Delta'\langle v/x\rangle$. We apply induction and obtain $\Gamma \sqcap \Delta'\langle v/x\rangle \vdash P\{v/x\}$. An application of (T-Match) give us the desired result, $\Gamma \sqcap \Delta\langle v/x\rangle \vdash [u_1\{v/x\} = u_2\{v/x\}]\, P\{v/x\}; Q\{v/x\}$.

The remaining cases are (T-Out@) and (T-In@).

(T-Out@) We have $\Gamma, x : A \sqcap \Delta \vdash u\langle \tilde{w}@\tilde{A}\rangle$. From (T-Out@) we know $\Gamma, x : A \sqcap \Delta \vdash u : \mathsf{w}$ and $\forall i \in 1, \ldots, n$ we have $\Gamma, x : A \sqcap \Delta \vdash w_i : A_i$ where $\tilde{w}@\tilde{A} = w_1@A_1, \ldots w_n@A_n$. We apply the first part of the Lemma and we deduce that both $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u\{v/x\} : \mathsf{w}$ and $\Gamma \sqcap \Delta\langle v/x\rangle \vdash w_i\{v/x\} : A_i$ for all $i \in 1, \ldots, n$. We apply (T-Out@) and infer $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u\{v/x\}\langle \tilde{w}\{v/x\}@\tilde{A}\rangle$.

(T-In@) We have $\Gamma, x : A \sqcap \Delta \vdash u(\tilde{y}@\tilde{B}).P$; we assume $x \notin \tilde{y}$. From (T-In@) we know $\Gamma, x : A \sqcap \Delta \vdash u : \mathsf{r}$ and $(\Gamma, x : A \sqcap \Delta), \tilde{y} : \tilde{B} \vdash P$. From $\tilde{y}$ fresh to the domain of Γ, Δ and $x \notin \tilde{y}$ we infer $(\Gamma, x : A \sqcap \Delta), \tilde{y} : \tilde{B} = (\Gamma, vy : \tilde{B}, x : A) \sqcap \Delta$. We apply the first part of the Lemma or induction and we deduce that $(\Gamma, \tilde{y} : \tilde{B}) \sqcap \Delta\langle v/x\rangle \vdash u\{v/x\} : \mathsf{r}$ and $(\Gamma, \tilde{y} : \tilde{B}) \sqcap \Delta\langle v/x\rangle \vdash P\{v/x\}$. From $(\Gamma, \tilde{y} : \tilde{B}) \sqcap \Delta\langle v/x\rangle = (\Gamma \sqcap \Delta\langle v/x\rangle), \tilde{y} : \tilde{B}$ and (T-In@) we obtain the desired result, $\Gamma \sqcap \Delta\langle v/x\rangle \vdash u\{v/x\}(\tilde{y}@\tilde{B}).P\{v/x\}$. □

The following Corollary generalizes the substitution result for processes.

Corollary 9. *Let $\tilde{v} = v_1, \ldots, v_n$ and $\tilde{A} = A_1, \ldots, A_n$ and assume that for all $i \in 1, \ldots, n$ we have $\Gamma \vdash v_i : A_i$. If $\Gamma, \tilde{x} : \tilde{A} \sqcap \Delta \vdash P$ then $\Gamma \sqcap \Delta\langle \tilde{v}/\tilde{x}\rangle \vdash P\{\tilde{v}/\tilde{x}\}$.*

Proof. By induction on the structure of $\tilde{x}$. Notice that the base coincides with the second part of Lemma 8. □

2.2 Dynamically typed structural operational semantics

The dynamics of the calculus is defined by means of a labelled transition system built around the following actions:

$$\textit{Actions} \quad \alpha \ ::= \ \tau \ \mid \ u(\tilde{v}@\tilde{B}) \ \mid \ (\tilde{c} : \tilde{C})\, u\langle \tilde{v}@\tilde{B} \rangle$$

The output action $(\tilde{c} : \tilde{C})\, u\langle \tilde{v}@\tilde{B} \rangle$ carries a type tag along with the output value: it represents the output of a tuple (possibly including fresh) values $v_1, \ldots, v_n$ respectively at the types $B_1, \ldots, B_n$; we require each bound name in $\tilde{c}$ to be unique. Dually, the input action $u(\tilde{v}@\tilde{B})$ represents the input of $v_1, \ldots, v_n$ respectively at the types $B_1, \ldots, B_n$. Most of the transitions, in Table 2.2, are standard. Rule (Pi-Output@) says that values are emitted together with the tag syntactically occurring in the definition of the output process. Scope extrusion is provided by rule (Pi-Open). The axiom (Pi-Input@) says that input processes with bound variables $x_1, \ldots, x_n$ tagged respectively with type $B_1, \ldots, B_n$ may receive values $v_1, \ldots, v_n$ tagged respectively with type $B_1, \ldots, B_n$, and when values are received they are substituted for the place-holders $x_1, \ldots, x_n$. We assume that the arity of $\tilde{v}$ and $\tilde{x}$ coincide. As anticipated, synchronizing input and output requires a dynamic type check: in (Pi-Com@), complementary labels synchronize only if they agree on the type of the values exchanged. The notation $\tilde{B} <: \tilde{B}'$ is short for the subtyping judgments on the component types. The (Pi-Match) rule say that whenever match succeeds the matching process behaves as the process in the then branch; dually in the (Pi-Mismatch) rule the matching process behaves as the process in the else branch whenever match fails. The remaining rules are the standard rules for parallel composition, replication and new of the pi calculus. For the sake of readability, in the remainder of the section we use the notation $\mathrm{I} \vdash \tilde{v} : \tilde{A}$ to indicate the judgments $\mathrm{I} \vdash v_1 : A_1, \ldots, \mathrm{I} \vdash v_n : A_n$ whenever $\tilde{v} = v_1, \ldots, v_n$ and $\tilde{A} = A_1, \ldots, A_n$.

The subject reduction theorem below follows routinely thanks to these dynamic checks. As is often the case its proof heavily relies on a substitution Lemma.

Theorem 10 (Subject Reduction). *Suppose* $\Gamma \vdash P$. *Then*

1. $P \xrightarrow{\tau} P'$ *implies* $\Gamma \vdash P'$
2. $P \xrightarrow{a(\tilde{v}@\tilde{B})} P'$ *implies* $\Gamma^{\mathsf{r}}(a) \downarrow$ *and* $\Gamma \sqcap \tilde{v} : \tilde{B} \vdash P'$
3. $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{B} \rangle} P'$ *implies* $\Gamma^{\mathsf{w}}(a) \downarrow$ *and* $\Gamma, \tilde{c} : \tilde{C} \vdash : \tilde{v} : \tilde{B}$ *and* $\Gamma, \tilde{c} : \tilde{C} \vdash P'$

Proof. By induction on the length of the inference $P \xrightarrow{\alpha} P'$ and by a case analysis of the last rule used. The most interesting cases are input, output, open, communication and matching transitions. In the following we let $\tilde{v} = v_1, \ldots, v_n$ and $\tilde{B} = B_1, \ldots, B_n$.

Table 2.2 API@ Labelled Transition System

(Pi-Output@)

$$\frac{}{a\langle \tilde{v}@\tilde{B}\rangle \xrightarrow{a\langle \tilde{v}@\tilde{B}\rangle} \mathbf{0}}$$

(Pi-Open)

$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{B}\rangle} P' \qquad b \neq a,\ b \in fn(\tilde{v})}{(\mathsf{new}\ b : B)P \xrightarrow{(b:B,\ \tilde{c}:\tilde{C})\, a\langle \tilde{v}@\tilde{B}\rangle} P'}$$

(Pi-Input@)

$$\frac{}{a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}}$$

(Pi-Com@)

$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{B}\rangle} P' \quad Q \xrightarrow{a(\tilde{v}@\tilde{B}')} Q' \quad \tilde{B} <: \tilde{B}' \quad \tilde{c} \cap fn(Q) = \varnothing}{P \,|\, Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \,|\, Q')}$$

(Pi-Match)

$$\frac{P \xrightarrow{\alpha} P'}{[a = a]\, P; Q \xrightarrow{\alpha} P'}$$

(Pi-Mismatch)

$$\frac{a \neq b \quad Q \xrightarrow{\alpha} Q'}{[a = b]\, P; Q \xrightarrow{\alpha} Q'}$$

(Pi-Par)

$$\frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \,|\, Q \xrightarrow{\alpha} P' \,|\, Q}$$

(Pi-Repl)

$$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \,|\, !P}$$

(Pi-Res)

$$\frac{P \xrightarrow{\alpha} P' \qquad a \notin n(\alpha)}{(\mathsf{new}\ a : A)P \xrightarrow{\alpha} (\mathsf{new}\ a : A)P'}$$

(Pi-Input@) In this case the reduction has the form $a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}$ (note that by the usual convention on binders we have that $\tilde{x}$ are not in the domain of Γ and are disjunct from $\tilde{v}$). From $\Gamma \vdash a(\tilde{x}@\tilde{B}).P$ and (T-In@) we infer $\Gamma \vdash a : \mathsf{r}$ and $\Gamma, \tilde{x} : \tilde{B} \vdash P$. Clearly we have $\Gamma \sqcap \tilde{v} : \tilde{B} \vdash v_i : B_i$ for all i in $1, \ldots, n$.. We apply Proposition 7 and infer $(\Gamma, \tilde{x} : \tilde{B}) \sqcap \tilde{v} : \tilde{B} \vdash P$. From $\tilde{x} \notin \mathrm{dom}(\Gamma) \cup \tilde{v}$ we obtain $(\Gamma, \tilde{x} : \tilde{B}) \sqcap \tilde{v} : \tilde{B} = (\Gamma \sqcap \tilde{v} : \tilde{B}), \tilde{x} : \tilde{B}$. An application of Corollary 9 choosing $\Delta = \emptyset$ let us infer $\Gamma \sqcap \tilde{v} : \tilde{B} \vdash P\{\tilde{v}/\tilde{x}\}$, as required.

(Pi-Out@) In this case the reduction has the form $a\langle \tilde{v}@\tilde{B}\rangle \xrightarrow{a\langle \tilde{v}@\tilde{B}\rangle} \mathbf{0}$. From $\Gamma \vdash a\langle \tilde{v}@\tilde{B}\rangle$ and (T-Out@) we infer $\Gamma \vdash a : \mathsf{w}$ and $\Gamma \vdash v_i : B_i$ for all i in $1, \ldots, n$. From (T-Nil) we deduce $\Gamma \vdash \mathbf{0}$ and we conclude.

(Pi-Open@) In this case the reduction has the form $(\mathsf{new}\ b : B)P \xrightarrow{(b:B, \tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{B}\rangle} P'$ with $a \neq b$ and has been inferred from $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{B}\rangle} P'$. From $\Gamma \vdash (\mathsf{new}\ b : B)P$ we infer $\Gamma, b : B \vdash P$. By the inductive hypothesis we have $\Gamma, b : B \vdash a : \mathsf{w}$ and $\Gamma, b : B, \tilde{c} : \tilde{C} \vdash v_i : B_i$ for all $i \in 1, \ldots, n$ and $\Gamma, b : B, \tilde{c} : \tilde{C} \vdash P'$. Since by hypothesis $a \neq b$ we may apply strengthening (Proposition 5) to $\Gamma, b : B \vdash a : \mathsf{w}$ and deduce $\Gamma \vdash a : \mathsf{w}$, and we are done.

(Pi-Com@) In this case the reduction has the form $P \,|\, Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \,|\, Q')$ and has been inferred from $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{B}\rangle} P'$ and from $Q \xrightarrow{a(\tilde{v}@\tilde{B}')} Q'$ and $B_i <: B'_i$ for all $i \in 1, \ldots, n$ with $\tilde{c} \cap fn(Q) = \emptyset$. From $\Gamma \vdash P \,|\, Q$ and (T-Par) we infer $\Gamma \vdash P$ and $\Gamma \vdash Q$. By induction on $\Gamma \vdash P$ we have that $\Gamma \vdash a : \mathsf{w}$ and $\Gamma, \tilde{c} : \tilde{C} \vdash v_i : B_i$ for all $i \in 1, \ldots, n$, and $\Gamma, \tilde{c} : \tilde{C} \vdash P'$. By induction on $\Gamma \vdash Q$ we infer $\Gamma \vdash a : \mathsf{r}$ and $\Gamma \sqcap \tilde{v} : \tilde{B}' \vdash Q'$. Assume $(u_1 : A_1, \ldots, u_m : A_m) = \tilde{v} : \tilde{B}' \setminus \tilde{c} : \tilde{C}$ and let $\tilde{u} : \tilde{A} = u_1 : A_1, \ldots, u_m : A_m$. From $\Gamma, \tilde{c} : \tilde{C} \vdash v_i : B_i$ and the hypothesis of dynamic typing $B_i <: B'_i$ we obtain $\Gamma, \tilde{c} : \tilde{C} \vdash v_i : B'_i$ for all $i \in 1, \ldots, n$. From strengthening we obtain that for all $j \in 1, \ldots, m$ holds $\Gamma \vdash u_j : A_j$ and in turn from (T-Pro) $\Gamma(u_j) <: A_j$. This let us infer $\Gamma \sqcap \tilde{u} : \tilde{A} = \Gamma$ (see Prop. 2). This result implies that $\Gamma \sqcap \tilde{v} : \tilde{B}' = \Gamma, \tilde{c} : \tilde{C}$. By induction we obtain $\Gamma, \tilde{c} : \tilde{C} \vdash Q'$. We apply (T-Par) to $\Gamma, \tilde{c} : \tilde{C} \vdash P'$ and $\Gamma, \tilde{c} : \tilde{C} \vdash Q'$ and we obtain $\Gamma, \tilde{c} : \tilde{C} \vdash P' \,|\, Q'$. An application of (T-New) give us the desired result, $\Gamma \vdash (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \,|\, Q')$.

(Pi-Match) In this case the reduction has the form $[a = a]\, P; Q \xrightarrow{\alpha} P'$ and has been inferred from $P \xrightarrow{\alpha} P'$. From $\Gamma \vdash [a = a]\, P; Q$ and (T-Match) we know that there are types A, B such that $\Gamma \vdash a : A$ and $\Gamma \vdash a : B$ and $\Gamma \sqcap a : B \sqcap a : A \vdash P$. This let us infer $\Gamma(a) <: A$ and $\Gamma(a) <: B$ and in turn $\Gamma \sqcap a : B \sqcap a : B = \Gamma$. We apply induction to $\Gamma \vdash P$ and $P \xrightarrow{\alpha} P'$ and infer that

- if $\alpha = \tau$ then $\Gamma \vdash P'$;
- if $\alpha = a(\tilde{v}@\tilde{B})$ then $\Gamma^{\mathsf{r}}(a) \downarrow$ and $\Gamma \sqcap \tilde{v} : \tilde{B} \vdash P'$;
- if $\alpha = (\tilde{c} : \tilde{C})a\langle\tilde{v}@\tilde{B}\rangle$ then $\Gamma^{\mathsf{w}}(a) \downarrow$ and $\Gamma, \tilde{c} : \tilde{C} \vdash \tilde{v} : \tilde{B}$ and $\Gamma, \tilde{c} : \tilde{C} \vdash P'$.

□

2.3 Typed behavioural equivalence

The notion of observational equivalence, based on weak bisimulation, is inherited almost directly from [60]. As usual in typed equivalences [23, 91, 46], we observe the behavior of processes by means of contexts that have a certain knowledge of the processes, represented by a set of type assumptions contained in a type environment. However, following [60], we take the view that the typing information available to the context may be different (less informative) than the information available to the system. Thus, while the system processes may perform certain action because they posses the required (type) capabilities, the same may not be true of the context. We formalize these intuitions below.

Given two type environments Γ and I such that $\Gamma <: \mathrm{I}$, we will often say that Γ is *compatible* with I.

Definition 1. *A type-indexed relation $\mathscr{R}$ is a family of binary relations between processes (hence closed) indexed by type environments. We write $\mathrm{I} \models P \mathscr{R} Q$ to mean that (i) P and Q are related by $\mathscr{R}$ at I and (ii) there exist Γ and Δ compatible with I such that $\Gamma \vdash P$ and $\Delta \vdash Q$. We often write $\mathrm{I} \models P \mathscr{R} Q$ as $P \mathscr{R}_{\mathrm{I}} Q$.*

Definition 2 (Contextuality). *A type-indexed relation $\mathscr{R}$ is contextual whenever*

(i) $\mathrm{I} \models P \mathscr{R} Q$ *implies* $\mathrm{I}, a : A \models P \mathscr{R} Q$

(ii) $\mathrm{I} \models P \mathscr{R} Q$ *and* $\mathrm{I} \vdash R$ *imply* $\mathrm{I} \models P \mid R \mathscr{R} Q \mid R$

(iii) $\mathrm{I}, a : A \models P \mathscr{R} Q$ *implies* $\mathrm{I} \models (\mathsf{new}\ a : A)P \mathscr{R} (\mathsf{new}\ a : A)Q$.

We define the barb predicate relative to a type environment as follows. Let $\Longrightarrow$ be the reflexive and transitive closure of $\xrightarrow{\tau}$.

Definition 3 (Barbs). *Let I be a type environment and let P be a closed process. We let:*

1. $P \downarrow_a$ *if and only if* $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle\tilde{b}@\tilde{B}\rangle}$;
2. $P \Downarrow_a$ *if and only if* $P \Longrightarrow \downarrow_a$.
3. $\mathrm{I} \models P \downarrow_a$ *if and only if* $\mathrm{I}(a)^{\mathsf{r}} \downarrow \wedge\ P \downarrow_a$;

4. $\mathrm{I} \vDash P \Downarrow_a$ *if and only if* $\mathrm{I}(a)^{\mathsf{r}} \downarrow \wedge\, P \Downarrow_a$.

We define our notion of observational equivalence.

Definition 4 (Typed behavioral equivalence). *Typed behavioral equivalence, noted* $\cong^{\pi}$*, is the largest symmetric and contextual type-indexed relation* $\mathscr{R}$ *such that* $\mathrm{I} \models P \mathscr{R} Q$ *implies*

(i) *if* $\mathrm{I} \vDash P \downarrow_n$ *then* $\mathrm{I} \vDash Q \Downarrow_n$

(ii) *if* $P \xrightarrow{\tau} P'$ *then* $Q \Longrightarrow Q'$ *and* $\mathrm{I} \models P' \mathscr{R} Q'$ *for some* Q' .

Behavioural equivalence is indeed an equivalence relation; we will prove this result in Section 2.4.

Example As a simple illustration of the calculus and its behavioral theory we draw an example inspired from [88]. In that paper the authors argued that some desirable properties of distributes systems, for instance availability of public resources, can be guaranteed in the pi calculus only by using types that control the behavior of processes. The claim is best illustrated by their motivating example:

$$S = (\mathsf{new}\ s)!d\langle s\rangle \,|\, !s(x).print\langle x\rangle \qquad C = d(x).x\langle j\rangle$$

S is a print spooler serving requests from a private channel s that it communicates to its clients via the public channel d. C is one such client, that receives s and uses it to print the job j. While the intention of the specification is clear, one easily sees that, given the initial configuration $S \,|\, C$, is not guaranteed that the jobs sent by C will eventually be received and printed. Formally:

$$S \,|\, C \;\not\cong\; S \,|\, print\langle j\rangle \tag{2.3}$$

Here $\cong$ is the observational equivalence of the untyped pi calculus. The in-equation above is easily proved by exhibiting a context that interferes with the intended protocol between S and C: for instance the context $\mathscr{C}_1[-] = - \,|\, d(x).!x(y).\mathbf{0}$, that initially behaves as a client, to receive s, but then steals the jobs intended for S.

We show how to recover equation (2.3) in the API@ calculus. Letting J be the type of jobs, the two processes S and C may be defined as follows:

$$S = (\mathsf{new}\ s : \mathsf{rw})!d\langle s@\mathsf{w}\rangle \,|\, !s(x@\mathsf{J}).print\langle x@\mathsf{J}\rangle \qquad C = d(x@\mathsf{w}).x\langle j@\mathsf{J}\rangle$$

The important thing to note is the type $s@\mathsf{w}$ chosen by S for the output on d to make sure that the spooling channel will only be received with output capabilities. Then we can prove the desired equivalence by assuming that contexts may only read on d, and have no control on the channel *print*, namely:

$$j : \mathsf{J},\ print : \top,\ d : \mathsf{r} \;\models\; S \mid C \;\cong^{\pi}\; S \,|\, print\langle j@\mathsf{J}\rangle \tag{2.4}$$

As usual [99], proving such equivalences takes some effort and is often not easy, as it requires induction to capture all the typed contexts. Luckily (but not surprisingly), the construction from [60] works just as well for our calculus in providing a purely coinductive characterization for reduction barbed congruence. □

A fundamental difference between our notion of equivalence and that of [60] arises as a consequence of the different typing disciplines. In our system, the rule for typed value exchange guarantees that the context obtains values emitted on a public channel at the static type occurring in the coercion associated with the output. Conversely, in [60], the type at which the context acquires the new name is determined by the type information that the context has on the channel used for output: that, in turn, presupposes that the context "behaves" in that it will not try to acquire from the value emitted more than allowed by the read type of the transmission channel. As a consequence, in [60] reasoning on the access control policies can hardly be carried out without assuming that the context is well-typed, with the same type system. In our solution, instead, an appropriate use of coercion may be put in place to prevent the context from acquiring, upon output by the system, more information than it was intended to. A corresponding mechanism can be realized in terms of a low-level construction in the applied pi calculus. We illustrate how this can be done in Chapter 4.

2.4 A coinductive proof technique

To prove that two processes are typed behavioural equivalent, we need to find a relation which is symmetric, barb preserving, reduction closed and contextual. Particularly, the last condition implies the candidate relation to be closed under parallel composition of processes that are typechecked by the environment. To avoid the difficulties implicitly arising in such quantification, we provide a characterization of $\cong^{\pi}$ in the terms of a bisimulation equivalence that can be proved coinductively. The characterization draws on the definition of a set of typed labelled transitions in which the interaction between a process and its context is mediated by the type capabilities that the context possesses on the shared names.

Definition 5. *A pair* $\mathrm{I} \triangleright P$ *is a configuration if and only if there is a closed type environment* Γ *compatible with* I *such that* $\Gamma \vdash P$.

The typed actions, in Table 2.3, encode transitions over configurations of the form $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ where

$$\alpha ::= \tau \mid (\tilde{b})a\langle \tilde{v}@\tilde{B} \rangle \mid (\tilde{c} : \tilde{C})a(\tilde{v}@\tilde{B}) \ .$$

The types of the bound names in the output action have been dropped as they are not observable (they are used only for the definition of the untyped reduction

Table 2.3 Typed Actions

$$\text{(G-Out@)}\quad \frac{\mathrm{I}^{\mathsf{r}}(a)\downarrow \qquad \tilde{A} <: \tilde{B}}{\mathrm{I} \triangleright a\langle \tilde{v}@\tilde{A}\rangle \xrightarrow{a\langle \tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \triangleright \mathbf{0}}$$

$$\text{(G-In@)}\quad \frac{\mathrm{I}^{\mathsf{w}}(a)\downarrow \quad \mathrm{I} \vdash \tilde{v} : \tilde{A}' \quad \tilde{A}' <: \tilde{A}}{\mathrm{I} \triangleright a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A}')} \mathrm{I} \triangleright P\{\tilde{v}/\tilde{x}\}}$$

$$\text{(G-Open)}\quad \frac{\mathrm{I}, b : \top \triangleright P \xrightarrow{(\tilde{c})a\langle \tilde{v}@\tilde{A}\rangle} \mathrm{I}' \triangleright P' \qquad b \neq a,\ b \in fn(\tilde{v})}{\mathrm{I} \triangleright (\mathsf{new}\ b : B)P \xrightarrow{(b,\ \tilde{c})\, a\langle \tilde{v}@\tilde{A}\rangle} \mathrm{I}' \triangleright P'}$$

$$\text{(G-Weak)}\quad \frac{\mathrm{I}, b : B \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}' \triangleright P' \qquad b \notin \{a, \tilde{c}\}}{\mathrm{I} \triangleright P \xrightarrow{(b:B,\ \tilde{c}:\tilde{C})\, a(\tilde{v}@\tilde{A})} \mathrm{I}' \triangleright P'}$$

$$\text{(G-Match)}\quad \frac{\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'}{\mathrm{I} \triangleright [a = a]\, P; Q \xrightarrow{\alpha} \mathrm{I}' \triangleright P'} \qquad \text{(G-Mismatch)}\quad \frac{a \neq b \qquad \mathrm{I} \triangleright Q \xrightarrow{\alpha} \mathrm{I}' \triangleright Q'}{\mathrm{I} \triangleright [a = b]\, P; Q \xrightarrow{\alpha} \mathrm{I}' \triangleright Q'}$$

$$\text{(G-Reduce)}\quad \frac{P \xrightarrow{\tau} P'}{\mathrm{I} \triangleright P \xrightarrow{\tau} \mathrm{I} \triangleright P'} \qquad \text{(G-Par)}\quad \frac{\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P' \qquad bn(\alpha) \cap fn(Q) = \emptyset}{\mathrm{I} \triangleright P \,|\, Q \xrightarrow{\alpha} \mathrm{I}' \triangleright P' \,|\, Q}$$

$$\text{(G-Res)}\quad \frac{\mathrm{I}, a : \top \triangleright P \xrightarrow{\alpha} \mathrm{I}', a : \top \triangleright P' \qquad a \notin n(\alpha)}{\mathrm{I} \triangleright (\mathsf{new}\ a : A)P \xrightarrow{\alpha} \mathrm{I}' \triangleright (\mathsf{new}\ a : A)P'} \qquad \text{(G-Repl)}\quad \frac{\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'}{\mathrm{I} \triangleright\, !P \xrightarrow{\alpha} \mathrm{I}' \triangleright P' \,|\, !P}$$

relation $\xrightarrow{\tau}$). Conversely input actions now may contain fresh names generated by the environment at given types and assumed to occur in the input. These transitions identify actions by the process P that are only possible if they are allowed by the environment I. Not surprisingly, most of the typed transitions in our system are derived directly from their companion transitions in the system of [60].

The input and output forms are characteristic of our labelled transition system, as these reflect the nature of the interactions with the context distinctive of our calculus. Specifically, the (G-OUT@) rule formalizes the fact that a context willing to observe an output action performed by a process on a channel readable by the environment may only do so by guessing a super-type of the actual type used in the type coercion: that supertype is also the type at which the context acquires the values emitted. The notation $\tilde{A} <: \tilde{B}$ is short for the subtyping judgments on the component types. Dually, the (G-IN@) rule shows that an input by a process on a channel writable by the environment may in general only be observed at a lower type than actually performed by the process. All this is a consequence of the format of our subtype-based synchronization rule (PI-COM@) in Table 2.2.

Rule (G-OPEN) depicts the behaviour of a process sending fresh values to the environment. Note that in the premises b has to be in the domain of the environment since for $\mathrm{I}, b : \top \triangleright P$ to be a configuration it is required that exists $\Delta <: \mathrm{I}, b : \top$ such that $\Delta \vdash P$, and this requires $\Delta \vdash \tilde{v} : \tilde{A}$ with $b \in \tilde{v}$. However, knowing b at top precludes to the environment any access to the channel. Rule (G-WEAK) says that the environment can create new values at given types inside input actions. Rule (G-REDUCE) formalizes the intuition that Table 2.3 depicts interactions among processes and the environment; in such case the internal reduction of the process is not visible to the environment. Rules for replication, parallel composition, matching and mismatching are simple extensions of their correspondent untyped forms. Similarly rule (G-NEW) extends its untyped version; similarly to (G-OPEN), in the premises the name a bound in P needs to be in the domain of the environment to let $\mathrm{I}, a : \top \triangleright P$ be a configuration.

Based on the labelled transition system of Table 2.3, we develop a bisimulation equivalence that can be used as a proof technique for $\cong^{\pi}$, as we will show. The resulting notion of asynchronous bisimilarity arises as expected [15].

We let $\xrightarrow{\hat{\alpha}}$ be the identity whenever $\alpha = \tau$ else $\xrightarrow{\hat{\alpha}} \triangleq \xrightarrow{\alpha}$.

Definition 6 (Typed labelled bisimilarity). *A symmetric type indexed relation $\mathscr{R}$ over processes is an asynchronous bisimulation if whenever $\mathrm{I} \models P \mathscr{R} Q$, one has:*

- *if* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ *and* α *is* $(\tilde{c})\, a\langle \tilde{v}@\tilde{A} \rangle$ *or* τ *then* $\mathrm{I} \triangleright Q \xRightarrow{\widehat{\alpha}} \mathrm{I}' \triangleright Q'$ *with* $\mathrm{I}' \models P' \mathscr{R} Q'$
- *if* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ *and* α *is* $(\tilde{c} : \tilde{C}) a(\tilde{v}@\tilde{A})$ *then*
  - $\mathrm{I} \triangleright Q \xRightarrow{\alpha} \mathrm{I}' \triangleright Q'$ *with* $\mathrm{I}' \models P' \mathscr{R} Q'$ *or*

– $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q'$ *with* $\mathrm{I}' \models P' \mathscr{R} Q' \mid a\langle \tilde{v}@\tilde{A}\rangle$.

Asynchronous labelled bisimilarity, noted $\approx_a^\pi$*, is the largest type indexed asynchronous bisimulation.*

Perhaps interestingly, the reader will notice that the types $\tilde{C}$ and $\tilde{A}$ chosen to match the input transition are, in both cases, exactly the types occurring in the label of the transition to be matched.

In proving that two process are behaviorally equivalent, is often convenient to exhibit a relation that is smaller than the one required by the definition. These so called *up to techniques* (see [103]) need to be carefully analyzed to ensure that they are sound, i.e. we need to prove that the smaller relation induced by the definition is actually contained in behavioural equivalence.

Definition 7 (Up to technique). *A type-indexed relation is an asynchronous bisimulation up to* $\mathscr{R}_1$, $\mathscr{R}_2$ *(noted up to* $\mathscr{R}$ *whenever* $\mathscr{R}_1 = \mathscr{R} = \mathscr{R}_2$*) if* $\mathrm{I} \models P \mathscr{S} Q$ *and* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ *then* $\mathrm{I} \triangleright Q$ *answers with the same weak move of the definition of typed labelled bisimilarity and for the appropriate* Q' *we have that* $\mathrm{I}' \models P' \mathscr{R}_1 \mathscr{S} \mathscr{R}_2 Q'$.

We introduce lemmas that establish a number of useful properties about the typed actions of Table 2.3 and connect them with corresponding untyped actions.

Proposition 11. *If* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ *then* $\mathrm{I} \sqcap \tilde{b} : \tilde{B} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \sqcap \tilde{b} : \tilde{B} \triangleright P'$.

Proof. By straightforward rule induction. In rule (G-Out@) supposing $\alpha = a\langle \tilde{v}@\tilde{A}\rangle$ we exploit the hypothesis $\mathrm{I}(a)^{\mathsf{r}} \downarrow$ and the fact that $\mathrm{I} \sqcap \tilde{b} : \tilde{B} = \mathrm{I}', \Delta$ for type environments I', Δ such that $\mathrm{I}' <: \mathrm{I}$ to infer $(\mathrm{I} \sqcap \tilde{b} : \tilde{B})^{\mathsf{r}}(a) \downarrow$; the thesis then follows easily. Similarly in rule (G-In@) supposing $\alpha = a(\tilde{v}@\tilde{A})$ we infer $(\mathrm{I} \sqcap \tilde{b} : \tilde{B})^{\mathsf{w}}(a) \downarrow$ and $\mathrm{I} \sqcap \tilde{b} : \tilde{B} \vdash v_1 : A_1, \ldots \mathrm{I} \sqcap \tilde{b} : \tilde{B} \vdash v_n : A_n$ that let us infer the desired transition. The remaining cases follow directly by the induction hypothesis.

□

Lemma 12. *Let* I *be a closed type environment and* P *a closed process.*

1. *If* $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle \tilde{v}@\tilde{B}\rangle} \mathrm{I}' \triangleright P'$ *then* $\mathrm{I}^{\mathsf{r}}(a) \downarrow$, $\mathrm{I}' = \mathrm{I} \sqcap \tilde{v} : \tilde{B}$ *and there are types* $\tilde{A} <: \tilde{B}$ *and types* $\tilde{C}$ *such that* $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{A}\rangle} P'$.

   Vice versa if $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{v}@\tilde{A}\rangle} P'$ *and* $\mathrm{I}^{\mathsf{r}}(a) \downarrow$ *then for all types* $\tilde{B} :> \tilde{A}$ *holds* $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle \tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \triangleright P'$.

2. *If* $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}' \triangleright P'$ *then* $\mathrm{I}^{\mathsf{w}}(a)\downarrow$, $\mathrm{I}' = (\mathrm{I}, \tilde{c} : \tilde{C})$, $\mathrm{I}' \vdash \tilde{v} : \tilde{A}$ *and there are types* $A' :> A$ *s.t.* $P \xrightarrow{a(\tilde{v}@\tilde{A}')} P'$.

   Vice versa if $P \xrightarrow{a(\tilde{v}@\tilde{A}')} P'$ *and* $\mathrm{I}(a)^{\mathsf{w}}\downarrow$ *then for all types* $\tilde{A} <: \tilde{A}'$ *such that* $\mathrm{I} \vdash \tilde{v} : \tilde{A}$ *holds* $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}, \tilde{c} : \tilde{C} \triangleright P'$, *for some types* $\tilde{C}$.

3. $\mathrm{I} \triangleright P \xrightarrow{\tau} \mathrm{I}' \triangleright P'$ *iff* $\mathrm{I}' = \mathrm{I}$ *and* $P \xrightarrow{\tau} P'$.

Proof. By rule induction. Most cases in both directions follow directly by the induction hypothesis. We first sketch the main cases of the typed semantics direction.

(G-Out@) In this case the transition has the form $\mathrm{I} \triangleright a\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{a\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \triangleright \mathbf{0}$ where $\tilde{A} <: \tilde{B}$ and $\mathrm{I}^{\mathsf{r}}(a)\downarrow$. By the untyped semantics in Table 2.2 we infer $a\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{a\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}$.

(G-In@) In this case the transition has the form $\mathrm{I} \triangleright a(\tilde{x}@\tilde{A}').P \xrightarrow{a(\tilde{v}@\tilde{A})} \mathrm{I} \triangleright P\{\tilde{v}/\tilde{x}\}$ where $\tilde{A} <: \tilde{A}'$ and $\mathrm{I} \vdash \tilde{v} : \tilde{A}$ and $\mathrm{I}^{\mathsf{w}}(a)\downarrow$. By the untyped semantics in Table 2.2 we infer $a(\tilde{x}@\tilde{A}').P \xrightarrow{a(\tilde{v}@\tilde{A}')} P\{\tilde{v}/\tilde{x}\}$.

(G-Weak) In this case the transition has the form $\mathrm{I} \triangleright P \xrightarrow{(b:B,\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}' \triangleright P'$ that is inferred from $\mathrm{I}, b : B \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}' \triangleright P'$ with b not occurring in $a, \tilde{c}$. The inductive hypothesis is that $(\mathrm{I}, b : B)(a)^{\mathsf{w}}\downarrow$, $\mathrm{I}' = \mathrm{I}, b : B, \tilde{c} : \tilde{C}$, $\mathrm{I}' \vdash \tilde{v} : \tilde{A}$, and $P \xrightarrow{a(\tilde{v}@\tilde{A}')} P'$ for types $\tilde{A}' :> \tilde{A}$. From $a \neq b$, $(\mathrm{I}, b : B)(a)^{\mathsf{w}}\downarrow$ and strengthening (Proposition 5) we easily infer $\mathrm{I}(a)^{\mathsf{w}}\downarrow$, and we are done.

(G-Match) In this case the transition has the form $\mathrm{I} \triangleright [a = a]\, P; Q \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ that is inferred from $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. For each α, the inductive hypothesis give us the desired conditions on the environment typings. It's left to check the condition for the untyped transitions. Let α' be the action executed in the untyped semantics correspondent to α in the statement of the Lemma. The inductive hypothesis is $P \xrightarrow{\alpha'} P'$. This let us apply rule (Pi-Match) in Table 2.2 and infer $[a = a]\, P; Q \xrightarrow{\alpha'} P'$, as requested.

We now prove the most interesting cases whether the hypothesis involves the untyped semantics.

(Pi-Output@) In this case the transition has the form $a\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{a\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}$. Let $\tilde{B}$ be types such that $\tilde{A} <: \tilde{B}$. If $\mathrm{I}(a)^{\mathsf{r}} \downarrow$ then for rule (G-Out@) we have $\mathrm{I} \triangleright \xrightarrow{a\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \triangleright \mathbf{0}$, as desired.

(Pi-Input@) In this case the transition has the form $a(x@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}$. Let $\tilde{A}$ be types such that $\tilde{A} <: \tilde{B}$. If $\mathrm{I}(a)^{\mathsf{w}} \downarrow$ and $\mathrm{I} \vdash \tilde{v} : \tilde{A}$, for types $\tilde{A} <: \tilde{B}$, then by (G-In@) we infer $\mathrm{I} \triangleright a(x@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{A})} \mathrm{I} \triangleright P\{\tilde{v}/\tilde{x}\}$. We apply Proposition 11 and we infer $\mathrm{I}, \tilde{c} : \tilde{C} \triangleright a(x@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{A})} \mathrm{I}, \tilde{c} : \tilde{C} \triangleright P\{\tilde{v}/\tilde{x}\}$ for types $\tilde{C}$. We have the valid hypothesis to apply (G-Weak@) and obtain $\mathrm{I} \triangleright a(x@\tilde{B}).P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}, \tilde{c} : \tilde{C} \triangleright P\{\tilde{v}/\tilde{x}\}$, as desired.

(Pi-Open) In this case the transition has the form $(\mathsf{new}\ b : B)P \xrightarrow{(b:B,\tilde{c}:\tilde{C})a\langle\tilde{v}@\tilde{A}\rangle} P'$ and has been inferred from $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle\tilde{v}@\tilde{A}\rangle} P'$ with b distinct from a and occurring in $\tilde{v}$. By the inductive hypothesis we know that if $\mathrm{I}(a)^{\mathsf{r}} \downarrow$ then for all types $\tilde{B} :> \tilde{A}$ holds $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \triangleright P'$. We apply Proposition 11 and we infer $\mathrm{I}, b : \top \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@\tilde{B}\rangle} (\mathrm{I}, b : \top) \sqcap \tilde{v} : \tilde{B} \triangleright P'$. From $b \in \tilde{v}$ it's easy to see that $(\mathrm{I}, b : \top) \sqcap \tilde{v} : \tilde{B} = \mathrm{I} \sqcap \tilde{v} : \tilde{B}$. By rule (G-Open) we infer $\mathrm{I} \triangleright (\mathsf{new}\ b : B)P \xrightarrow{(b,\tilde{c})a\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \triangleright P'$, as desired.

(Pi-Com@) In this case the transition has the form $P \mid Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \mid Q')$. We apply (G-Reduce) and obtain $\mathrm{I} \triangleright P \mid Q \xrightarrow{\tau} \mathrm{I} \triangleright (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \mid Q')$.

(Pi-Res) Similar to case (Pi-Open).

The remaining cases are directly obtained by the induction hypothesis.

□

By the definition of type-indexed relation and configuration we directly infer the following proposition.

Proposition 13. *If $\mathscr{R}$ is type-indexed relation such that $\mathrm{I} \models P \mathscr{R} Q$, then $\mathrm{I} \triangleright P$ and $\mathrm{I} \triangleright Q$ are configurations.*

The next lemma ensures that configurations progress to valid configurations.

Lemma 14. *$\mathrm{I} \triangleright P$ is a configuration and $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ imply that $\mathrm{I}' \triangleright P'$ is a configuration.*

Proof. Assume $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ and let Γ be a type environment compatible with I such that $\Gamma \vdash P$. We need to show that there exists Γ' compatible with I' such that $\Gamma' \vdash P'$. The proof is by induction on length of the derivation of $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. Most cases follow directly by the induction hypothesis. We look at the cases (G-IN) and (G-OPEN).

(G-IN@) In this case the transition has the form $\mathrm{I} \triangleright a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A}')} \mathrm{I} \triangleright P\{\tilde{v}/\tilde{x}\}$ where $\tilde{A}' <: \tilde{A}$. Moreover, we know $\mathrm{I}(a)^{\mathsf{w}} \downarrow$ and $\mathrm{I} \vdash \tilde{v} : \tilde{A}'$. By the untyped semantics we know $a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A})} P\{\tilde{v}/\tilde{x}\}$. From $\Gamma \vdash P$ and subject reduction we obtain $\Gamma \sqcap \tilde{v} : \tilde{A} \vdash P\{\tilde{v}/\tilde{x}\}$. From $\mathrm{I}(\tilde{v}) <: \tilde{A}'$ and $\tilde{A}' <: \tilde{A}$ and transitivity of $<:$ we infer $\mathrm{I}(\tilde{v}) <: \tilde{A}$. Since $\Gamma(\tilde{v}) <: \mathrm{I}(\tilde{v})$, these results let us infer $\Gamma \sqcap \tilde{v} : \tilde{A} = \Gamma$, and in turn $\Gamma \vdash P\{\tilde{v}/\tilde{x}\}$, as requested.

(G-OPEN) In this case, the the transition in question has the form $\mathrm{I} \triangleright P \xrightarrow{(b,\tilde{c})a\langle\tilde{v}@A\rangle} \mathrm{I}' \triangleright P'$ derived from $\mathrm{I}, b : \top \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@A\rangle} \mathrm{I}' \triangleright P'$. By the induction hypothesis we know that $\mathrm{I}' \triangleright P'$ is a configuration which is all we need. □

The next Lemma formalizes the intuition that in a configuration the environment plays the role of permitting labelled transitions and collecting the type information on the received capabilities, and that these tasks are fulfilled due to the observation of labels.

Lemma 15. *Let I, J be closed type environments and P, Q be closed processes. Assume $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. If $\mathrm{J} \triangleright Q \xrightarrow{\alpha} \mathrm{J}' \triangleright Q'$ then also $\mathrm{I} \triangleright Q \xrightarrow{\alpha} \mathrm{I}' \triangleright Q'$.*

Proof. Follows directly by Lemma 12, reasoning by cases on the shape of α. □

Lemma 16. *If $\mathrm{I} \vdash P$ and $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$, then there exists I'' such that $\mathrm{I}' = \mathrm{I}, \mathrm{I}''$.*

Proof. The intuition is that processes that typechecks at I may contribute new names to the information conveyed by I, whereas they may refine the knowledge about (i.e. lower the types of any of the) existing names in I. Formally, the proof is by a cases analysis on the form of α. If α is an input or silent action the proof follows directly by Lemma 12. Instead, when α has the form $(\tilde{c})a\langle\tilde{v}@A\rangle$, by Lemma 12 we know that $\mathrm{I}' = \mathrm{I} \sqcap \tilde{v} : A$, and $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle\tilde{v}@A'\rangle} P'$ for suitable types $\tilde{C}$ and $\tilde{A}' <: \tilde{A}$. Then, from $\mathrm{I} \vdash P$, by Subject Reduction, we have $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : A'$ and $\mathrm{I}, \tilde{c} : \tilde{C} \vdash P'$. We have found the required $\mathrm{I}'' = \tilde{c} : \tilde{C}$. □

Lemma 17. *Let $\Gamma \vdash P$ with Γ compatible with* I. *If $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle\tilde{v}@\tilde{A}\rangle} P'$ then $\Gamma, \tilde{c} : \tilde{C}$ is compatible with $\mathrm{I} \sqcap \tilde{v} : \tilde{A}$.*

Proof. From the transition of the hypothesis and Lemma 12 we deduce $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle \tilde{v} @ \tilde{A} \rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{A} \triangleright P'$. From $\Gamma \vdash P$ and subject reduction we infer $\Gamma, \tilde{c} : \tilde{C} \vdash \tilde{v} : \tilde{A}$. From the judgement of values we obtain (i) $\tilde{v} \in \mathrm{dom}(\Gamma, \tilde{c} : \tilde{C})$ and (ii) $(\Gamma, \tilde{c} : \tilde{C})(\tilde{v}) <: \tilde{A}$. From (i) and $\tilde{c} \in \tilde{v}$ we infer $\mathrm{dom}(\Gamma, \tilde{c} : \tilde{C}) = \mathrm{dom}(\mathrm{I} \sqcap \tilde{v} : \tilde{A})$. From (ii) and the hypothesis Γ compatible with I we infer $\Gamma, \tilde{c} : \tilde{C} <: \mathrm{I} \sqcap \tilde{v} : \tilde{A}$. The two inferences give us the desired result. □

We define an up to technique which will be useful to ease the proof of soundness of typed labelled bisimilarity (see Proposition 24). To this aim we introduce the usual notion of structural congruence of the pi calculus.

Definition 8 (Structural congruence). *Structural congruence, noted* $\equiv$*, is the least relation that is closed under* α*-conversion, preserved by the operators of the language, and that satisfies the following axioms:*

$$\begin{array}{rcll} P \mid (\mathsf{new}\ a : A)Q & \equiv & (\mathsf{new}\ a : A)(P \mid Q) & \text{if } a \notin fn(P) \\ [a = a]\, P; Q & \equiv & P & \\ [a = b]\, P; Q & \equiv & Q & \text{if } a \neq b \\ P \mid Q & \equiv & Q \mid P & \\ !P & \equiv & !P \mid P & \\ P \mid \mathbf{0} & \equiv & P\ . & \end{array}$$

To prove that asynchronous bisimulation up to $\equiv$ is a sound technique we use the following result stating that $\equiv$ commutes with the typed dynamics of Table 2.2, in the following sense.

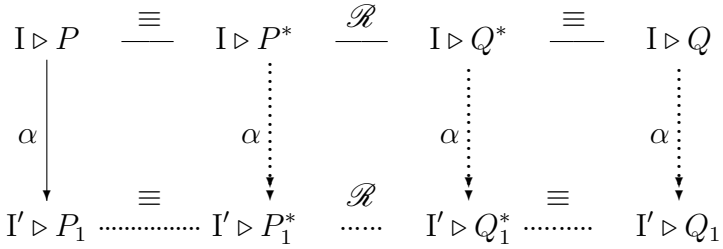
Proposition 18. *If* $P \equiv Q$ *and* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ *then there is* Q' *s.t.* $\mathrm{I} \triangleright Q \xrightarrow{\alpha} \mathrm{I}' \triangleright Q'$ *and* $P' \equiv Q'$*.*

Proof. Let $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. We proceed by case analysis on α and for each α by using the left to right direction of Lemma 12 we find an appropriate untyped action α' such that $P \xrightarrow{\alpha'} P'$. We then show that $P \equiv Q$ and $P \xrightarrow{\alpha'} P'$ implies that there is Q' s.t. $Q \xrightarrow{\alpha'} Q'$ and $P' \equiv Q'$. The proof of this result is similar to the demonstrations available in the literature (cf. [104]), and proceeds by induction on the number of applications of the rewriting rules of Definition 8. Finally for each untyped action α' we apply the right to left direction of Lemma 12 to $Q \xrightarrow{\alpha'} Q'$ and we infer $\mathrm{I} \triangleright Q \xrightarrow{\alpha} \mathrm{I}' \triangleright Q'$, as needed.

Proposition 19. *If* $\mathscr{R}$ *is an an asynchronous bisimulation up to* $\equiv$ *then* $\mathscr{R} \subseteq \approx_a^\pi$*.*

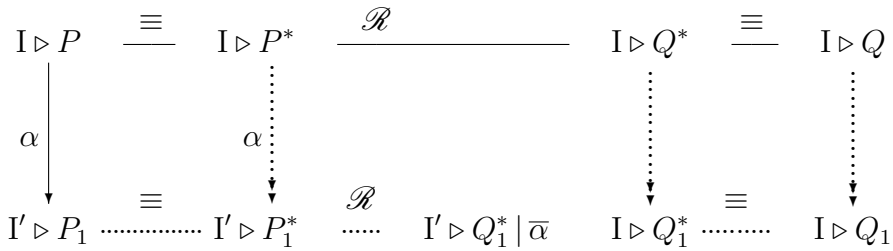
Proof. The proof consists in showing with a chasing-diagram argument that $\equiv \mathscr{R} \equiv$ is an asynchronous bisimulation. We use the convention to indicate existential relations with dotted lines; we let $\twoheadrightarrow$ represent $\Longrightarrow$. We first sketch the case whether

α is an output action.



The first inference on the left is obtained from Proposition 18. Now from $\mathrm{I} \models P^* \mathscr{R} Q^*$ and $\mathrm{I} \triangleright P^* \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright P_1^*$ it's easy to see that whenever α is an output action there is Q_1^* s.t. $\mathrm{I} \triangleright Q^* \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q_1^*$ and $\mathrm{I}' \models P_1^* \mathscr{R} Q_1^*$. Similarly one proves that Proposition 18 still holds whenever the strong move of the hypothesis is substituted with a weak move. This let us conclude that there is Q' s.t. $\mathrm{I} \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q_1$ with $Q_1 \equiv Q_1^*$.

We now draw the case whether α is an input action and $\mathrm{I} \triangleright Q^*$ answers with a silent action; the case whether the configuration matches the move with the same input move is analogous to the previous output case. To ease the notation we indicate with $\overline{\alpha}$ the asynchronous output derived from the input action α in the definition of asynchronous bisimilarity.



From $\mathrm{I} \models P^* \mathscr{R} Q^*$ and $\mathrm{I} \triangleright P^* \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright P_1^*$ we have that there is Q_1^* s.t. $\mathrm{I} \triangleright Q^* \Longrightarrow \mathrm{I} \triangleright Q_1^*$ and $\mathrm{I}' \models P_1^* \mathscr{R} Q_1^* \mid \overline{\alpha}$. As in the previous case from $Q^* \equiv Q$ we infer that there is Q' s.t. $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q_1$ and $Q_1^* \equiv Q_1$, as needed.

The case $\alpha = \tau$ is trivial and we omit it.

□

Soundness of typed labelled bisimilarity

We prove a series of lemmas and propositions that imply that $\approx_a^\pi$ is contained in $\cong^\pi$.

Proposition 20. *$\approx_a^\pi$ is (i) reduction closed and (ii) barb preserving.*

Proof. Let $\mathrm{I} \models P \approx_a^\pi Q$. To see (i), let $\mathrm{I} \triangleright P \stackrel{\tau}{\longrightarrow} \mathrm{I}' \triangleright P'$. By definition of $\approx_a^\pi$ there is Q' s.t. $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I}' \triangleright Q'$ and $\mathrm{I} \models P' \approx_a^\pi Q'$, that is $\approx_a^\pi$ is reduction closed. To see

(ii), from the definition of $\mathrm{I} \models P\downarrow_a$ we obtain that I may read a, that is $\mathrm{I}(a)^{\mathsf{r}}\downarrow$,and there is P' and types $\tilde{A}, \tilde{C}$ such that $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle\tilde{v}@\tilde{A}\rangle} P'$. We apply Lemma 12 and we deduce that there is I' such that $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I}' \triangleright P'$. Thus there is Q' s.t. $\mathrm{I} \triangleright Q \xRightarrow{(\tilde{c})a\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I}' \triangleright Q'$ with $\mathrm{I}' \models P' \approx_a^\pi Q'$. Since $\mathrm{I}(a)^{\mathsf{r}}\downarrow$, this implies $\mathrm{I} \models Q\Downarrow_a$. □

In the the rest of this section we focus on the proof that $\approx_a^\pi$ is contextual.

Lemma 21 (Subtype closure). *Assume* $\mathrm{I} \models P \approx_a^\pi Q$. *If* $\mathrm{I} <: \mathrm{I}'$ *then* $\mathrm{I}' \models P \approx_a^\pi Q$.

Proof. Let J be a type environment: define $\mathscr{R}$ to be the type-indexed relation such that $\mathrm{J} \models P\mathscr{R}Q$ whenever $\mathrm{I} <: \mathrm{J}$ and $\mathrm{I} \models P \approx_a^\pi Q$.
Assume $\mathrm{J} \triangleright P \xrightarrow{\alpha} \mathrm{J}' \triangleright P'$. An inspection of the typed transitions shows that for $\mathrm{I} <: \mathrm{J}$ we have $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ with $\mathrm{I}' <: \mathrm{J}'$. If α is a silent or output transition, by the hypothesis $\mathrm{I} \models P \approx_a^\pi Q$ we know that $\mathrm{I} \triangleright Q \xRightarrow{\widehat{\alpha}} \mathrm{I}' \triangleright Q'$ and $\mathrm{I}' \models P' \approx_a^\pi Q'$. By Lemma 15 we know that $\mathrm{J} \triangleright Q \xRightarrow{\widehat{\alpha}} \mathrm{J}' \triangleright Q'$: by this we can conclude because $\mathrm{I}' \models P' \approx_a^\pi Q'$ implies $\mathrm{J}' \models P'\mathscr{R}Q'$.
The case when α is an input transition is proved in the same way. □

The next lemma proves that labelled bisimilarity respects the first clause of contextuality (Definition 2).

Proposition 22 (Weakening closure). *Assume* $\mathrm{I} \models P \approx_a^\pi Q$. *If* I, I' *is an environment then* $\mathrm{I}, \mathrm{I}' \models P \approx_a^\pi Q$.

Proof. The proof follows by co-induction. We need the following simple properties of typed actions . Assume $\mathrm{I} \triangleright P$ is a configuration. Then, for every I' such that that $\mathrm{I}, \mathrm{I}' \triangleright P \xrightarrow{\alpha} \mathrm{J} \triangleright P'$, with α a silent or output transition it must be the case that $fn(\alpha) \subseteq \mathrm{dom}(\mathrm{I})$. From this we have:

(i) If $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@A\rangle} \mathrm{I} \sqcap \tilde{v} : A \triangleright P'$ then $\mathrm{I}, \mathrm{I}' \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@A\rangle} \mathrm{J} \triangleright P'$ with $\mathrm{J} = (\mathrm{I} \sqcap \tilde{v} : A), \mathrm{I}'$.

(ii) Conversely, if $\mathrm{I} \triangleright P$ is a configuration, and $\mathrm{I}, \mathrm{I}' \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@A\rangle} \mathrm{J} \triangleright P'$, then $\mathrm{J} = (\mathrm{I} \sqcap \tilde{v} : A), \mathrm{I}'$ and $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}@A\rangle} \mathrm{I} \sqcap \tilde{v} : A \triangleright P'$

Notice (i) specializes Proposition 11 for the case of output actions and environment I' such that the domains of I, I' do not intersect.

Let I' be a type environment: define $\mathscr{R}$ to be the type-indexed relation such that $\mathrm{I}, \mathrm{I}' \models P\mathscr{R}Q$ whenever I, I' is a legal type environment and $\mathrm{I}, \mathrm{I}' \models P \approx_a^\pi Q$ or $\mathrm{I} \models P \approx_a^\pi Q$.
We show that $\mathscr{R}$ is an asynchronous bisimulation: from this the result follows directly as $\mathscr{R}$ being a bisimulation implies that, pointwise, it is contained in $\approx_a^\pi$.

Assume $\mathrm{I}, \mathrm{I}' \models P \mathscr{R} Q$ and $\mathrm{I}, \mathrm{I}' \triangleright P \xrightarrow{\alpha} \mathrm{J} \triangleright P'$: we must find Q' such that $\mathrm{I}, \mathrm{I}' \triangleright Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{J} \triangleright Q'$ and $\mathrm{J} \models P' \mathscr{R} Q'$. If $\mathrm{I}, \mathrm{I}' \models P \mathscr{R} Q$ because $\mathrm{I}, \mathrm{I}' \models P \approx_a^\pi Q$ the claim follows trivially. We examine the case when $\mathrm{I} \models P \approx_a^\pi Q$, reasoning by cases on the shape of α.

($\alpha = (\tilde{c}) a\langle \tilde{v} @ A \rangle$) From $\mathrm{I} \models P \approx_a^\pi Q$ we know that $\mathrm{I} \triangleright P$ is a configuration. From (ii) above, we know that $\mathrm{J} = (\mathrm{I} \sqcap \tilde{v} : A), \mathrm{I}'$ and also that

$$\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}) a\langle \tilde{v} @ A \rangle} \mathrm{I} \sqcap \tilde{v} : A \triangleright P'.$$

From $\mathrm{I} \models P \approx_a^\pi Q$, we find Q' such that

$$\mathrm{I} \triangleright Q \stackrel{(\tilde{c}) a\langle \tilde{v} @ A \rangle}{\Longrightarrow} \mathrm{I} \sqcap \tilde{v} : A \triangleright Q'$$

and $\mathrm{I} \sqcap \tilde{v} : A \models P' \approx_a^\pi Q'$. From (i) we know that

$$\mathrm{I}, \mathrm{I}' \triangleright Q \stackrel{(\tilde{c}) a\langle \tilde{v} @ A \rangle}{\Longrightarrow} \mathrm{J} \triangleright Q'$$

and we are done because $\mathrm{I} \sqcap \tilde{v} : A \models P' \approx_a^\pi Q'$ implies that $\mathrm{J} \models P' \mathscr{R} Q'$.

($\alpha = \tau$) In this case the hypothesis is $\mathrm{I}, \mathrm{I}' \triangleright P \xrightarrow{\tau} \mathrm{I}, \mathrm{I}' \triangleright P'$. By Lemma 12 (in the 'only if' direction) we know that $P \xrightarrow{\tau} P'$ and from this, again by Lemma 12 (in the 'if' direction) we have $\mathrm{I} \triangleright P \xrightarrow{\tau} \mathrm{I} \triangleright P$. Because $\mathrm{I} \models P \approx_a^\pi Q$, we find Q' such that $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q'$ and $\mathrm{I} \models P' \approx_a^\pi Q'$. By two further applications of Lemma 12 we obtain $\mathrm{I}, \mathrm{I}' \triangleright Q \Longrightarrow \mathrm{I}, \mathrm{I}' \triangleright Q'$, by which we conclude as $\mathrm{I} \models P' \approx_a^\pi Q'$ implies $\mathrm{I}, \mathrm{I}' \models P' \mathscr{R} Q'$ as desired.

($\alpha = (\tilde{c} : \tilde{C}) a(\tilde{v} @ A)$). From $\mathrm{I}, \mathrm{I}' \triangleright P \xrightarrow{\alpha} \mathrm{J} \triangleright P'$, by (G-Weak), we obtain $\mathrm{I} \triangleright P \xrightarrow{(\mathrm{I}')\alpha} \mathrm{J}' \triangleright P'$. By Lemma 12, we know that $\mathrm{J}' = \mathrm{J} = \mathrm{I}, \mathrm{I}', \tilde{c} : \tilde{C}$. From $\mathrm{I} \models P \approx_a^\pi Q$ we find Q' such that either (a) $\mathrm{I} \triangleright Q \xrightarrow{(\mathrm{I}')\alpha} \mathrm{J} \triangleright Q'$ and $\mathrm{J} \models P' \approx_a^\pi Q'$, or (b) $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q'$ with $\mathrm{J} \models P' \approx_a^\pi (Q' \mid a\langle \tilde{v} @ A \rangle)$. In case (a), an inspection of the typed transitions shows that $\mathrm{I}, \mathrm{I}' \triangleright Q \xrightarrow{\alpha} \mathrm{J} \triangleright Q'$: we can conclude because $\mathrm{J} \models P' \approx_a^\pi Q'$ implies $\mathrm{J} \models P' \mathscr{R} Q'$ as desired. We can conclude case (b) as well because we have found Q' such that $\mathrm{I}, \mathrm{I}', \tilde{c} : \tilde{C} \triangleright Q \Longrightarrow \mathrm{J} \triangleright Q'$ with $\mathrm{J} \models P' \mathscr{R} (Q' \mid a\langle \tilde{v} @ A \rangle)$

□

Next we tackle closure of bisimilarity under new.

Proposition 23. *If* $\mathrm{I}, a : A \models P \approx_a^\pi Q$ *then* $\mathrm{I} \models (\mathsf{new}\ a : A) P \approx_a^\pi (\mathsf{new}\ a : A) Q$.

Proof. It is simpler to show that $\mathrm{I}, a : \top \models P \approx_a^\pi Q$ implies $\mathrm{I} \models (\mathsf{new}\ a : A) P \approx_a^\pi (\mathsf{new}\ a : A) Q$. From this, the desired result follows by Lemma 21 as $\mathrm{I}, a : A \models P \approx_a^\pi Q$ implies $\mathrm{I}, a : \top \models P \approx_a^\pi Q$.

Let $\mathscr{R}$ be the type indexed relation such that $\mathrm{I} \models P \mathscr{R} Q$ whenever (i) $\mathrm{I} \models P \approx_a^\pi Q$ or (*ii*) P and Q have the form $(\mathsf{new}\ a : A) P'$ and $(\mathsf{new}\ a : A) Q'$ and $\mathrm{I}, a : \top \models P' \approx_a^\pi Q'$.

That $\mathscr{R}$ is a type indexed relation follows from these facts. Let $\Gamma = \Gamma', a : A$. From $\Gamma \vdash P$ and (T-New) we have $\Gamma' \vdash (\mathsf{new}\ a : A)P'$. From $\Gamma <: \mathrm{I}, a : \top$ we obtain $\Gamma' <: \mathrm{I}$. The same reasoning holds for $(\mathsf{new}\ a : A)Q'$ and hence the claim.

We show that $\mathscr{R}$ is a type indexed bisimulation: specifically, we show that every transition from $\mathrm{I} \triangleright P$ can be matched by $\mathrm{I} \triangleright Q$.Case (i) is trivial; we consider (ii).

Assume $\mathrm{I}, a : \top \models P \approx_a^\pi Q$ and $\mathrm{I} \triangleright (\mathsf{new}\ a : A)P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. We reason by rule induction, by cases on the last rule in the derivation.

(G-Open) Then the transition in question has the form $\mathrm{I} \triangleright (\mathsf{new}\ a : A)P \xrightarrow{(a)\alpha} \mathrm{I}' \triangleright P'$, and is derived from $\mathrm{I}, a : \top \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. By the hypothesis $\mathrm{I}, a : \top \models P \approx_a^\pi Q$, we know that there exists Q' such that $\mathrm{I}, a : \top \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$ and $\mathrm{I}' \models P' \approx_a^\pi Q'$. From the last transition, by (G-Open), we have $\mathrm{I} \triangleright (\mathsf{new}\ a : A)Q \stackrel{(a)\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$. This is enough to to conclude because $\mathrm{I}' \models P' \approx_a^\pi Q'$ implies, by definition, $\mathrm{I}' \models P' \mathscr{R} Q'$.

(G-Res) Then the transition in question has the form $\mathrm{I} \triangleright (\mathsf{new}\ a : A)P \xrightarrow{\alpha} \mathrm{I}' \triangleright (\mathsf{new}\ a : A)P'$ with $a \notin n(\alpha)$, and is derived from $\mathrm{I}, a : \top \triangleright P \xrightarrow{\alpha} \mathrm{I}', a : \top \triangleright P'$. From the hypothesis that $\mathrm{I}, a : \top \models P \approx_a^\pi Q$, it follows that there exists Q' such that $\mathrm{I}, a : \top \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}', a : \top \triangleright Q'$ and with $\mathrm{I}, a : \top \models P' \approx_a^\pi Q'$. Thus, by applying (G-Res) together with the hypothesis $a \notin n(\alpha)$, we infer $\mathrm{I} \triangleright (\mathsf{new}\ a : A)Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright (\mathsf{new}\ a : A)Q'$. By definition $\mathrm{I}, a : \top \models P' \approx_a^\pi Q'$ implies $\mathrm{I} \models (\mathsf{new}\ a : A)P' \mathscr{R} (\mathsf{new}\ a : A)Q'$, and we are done.

(G-Reduce) Then the transition in question is $\mathrm{I} \triangleright (\mathsf{new}\ a :)P \xrightarrow{\tau} \mathrm{I}' \triangleright (\mathsf{new}\ a : A)P'$, derived from $(\mathsf{new}\ a : A)P \xrightarrow{\tau} (\mathsf{new}\ a : A)P'$, which in turn derives from $P \xrightarrow{\tau} P'$. We show that there exists Q' such that $\mathrm{I} \triangleright (\mathsf{new}\ a : A)Q \Longrightarrow \mathrm{I}' \triangleright (\mathsf{new}\ a : A)Q'$ and $\mathrm{I}, a : \top \models P' \approx_a^\pi Q'$. From $P \xrightarrow{\tau} P'$, by Lemma 12, we know that $\mathrm{I}, a : \top \triangleright P \xrightarrow{\tau} \mathrm{I}, a : \top \triangleright P'$. Since $\mathrm{I}, a : \top \models P \approx_a^\pi Q$ there exists Q' such that $\mathrm{I}, a : \top \triangleright Q \Longrightarrow \mathrm{I}, a : \top \triangleright Q'$ and $\mathrm{I}, a : \top \models P' \approx_a^\pi Q'$. We have found the desired Q' because from the last weak transition, we easily see that $\mathrm{I} \triangleright (\mathsf{new}\ a : A)Q \Longrightarrow \mathrm{I} \triangleright (\mathsf{new}\ a : A)Q'$ (by repeated applications of (G-Reduce) followed by (G-Res) followed by repeated applications of (G-Reduce)).

(G-Weak) Then the transition in question has the form $\mathrm{I} \triangleright (\mathsf{new}\ a : A)P \xrightarrow{(b)\alpha} \mathrm{I}' \triangleright P'$, and is derived from $\mathrm{I}, b : D \triangleright (\mathsf{new}\ a : A)P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. From $\mathrm{I}, a : \top \models P \approx_a^\pi Q$, by Proposition 22 we know that $\mathrm{I}, b : D, a : \top \models P \approx_a^\pi Q$. By the induction hypothesis, we find a weak transition $\mathrm{I}, b : D \triangleright (\mathsf{new}\ a : A)Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$ (or else the weak transition corresponding to case (b) in the definition of asynchronous bisimulation), with $\mathrm{I}' \models P' \approx_a^\pi Q'$, and hence $\mathrm{I}' \models P' \mathscr{R} Q'$. It is not difficult to see that from the last transition we obtain $\mathrm{I} \triangleright (\mathsf{new}\ a : A)Q \stackrel{(b)\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$, as desired. □

The most difficult case is the closure of labelled bisimilarity under parallel composition.

Proposition 24. *If* $\mathrm{I} \models P \approx_a^\pi Q$ *and* $\mathrm{I} \vdash R$ *then* $\mathrm{I} \models (P \mid R) \approx_a^\pi (Q \mid R)$.

Proof. To ease the notation, we write $(\mathsf{new}\ \Delta)P$ whenever Δ has the form $\tilde{a} : \tilde{A}$ with $\tilde{a}$ names; we write $|\Delta|$ for the names $\tilde{a}$. With this understanding, let $\mathscr{R}$ be the type-indexed relation defined as follows:

> $\mathrm{I} \models (\mathsf{new}\ \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ \Delta')(Q \mid R)$ iff $\mathrm{dom}(\Delta) = \mathrm{dom}(\Delta')$ and there exists J such that:
>
> - $\mathrm{I} \sqcap \mathrm{J} \models P \approx_a^\pi Q$ and $\mathrm{I} \sqcap \mathrm{J} \vdash R$
> - there exist Γ and Γ' such that Γ, Δ and Γ', Δ' are compatible with $\mathrm{I} \sqcap \mathrm{J}$, and one has $\Gamma, \Delta \vdash P$ and $\Gamma', \Delta' \vdash Q$

We show that $\mathscr{R}$ is an asynchronous bisimulation up to $\equiv$. By Proposition 19 we obtain that $\mathscr{R}$ is contained in $\approx_a^\pi$. Then, since $\mathrm{I} \models (P \mid R)\mathscr{R}(Q \mid R)$, the proposition follows by coinduction. To see that $\mathrm{I} \models (P \mid R)\mathscr{R}(Q \mid R)$, let $\Delta = \Delta' = \varnothing$. Then, choosing $\mathrm{J} = \varnothing$ we have $\mathrm{I} \models P \approx_a^\pi Q$ and $\mathrm{I} \vdash R$ by our hypothesis. That there exist Γ, Γ' as required follows by the fact that $\mathrm{I} \models P \approx_a^\pi Q$ implies that $\mathrm{I} \triangleright P$ and $\mathrm{I} \triangleright Q$ are configurations.

First we observe that $\mathrm{I} \models (\mathsf{new}\ b : B, \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ b : B', \Delta')(Q \mid R)$ implies $\mathrm{I}, b : \top \models (\mathsf{new}\ \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ \Delta')(Q \mid R)$. To see that, assume $\mathrm{I} \models (\mathsf{new}\ b : B, \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ b : B', \Delta')(Q \mid R)$ and let J be the environment such that $\mathrm{I} \sqcap \mathrm{J} \models P \approx_a^\pi Q$ and $\mathrm{I} \sqcap \mathrm{J} \vdash R$. In addition we know that there exists Γ and Γ' such that $\Gamma, b : B, \Delta \vdash P$, $\Gamma', b : B', \Delta' \vdash Q$ and $\Gamma, b : B, \Delta$ and $\Gamma', b : B', \Delta'$ are compatible with $\mathrm{I} \sqcap \mathrm{J}$. But then $b \in \mathrm{dom}(\mathrm{I} \sqcap \mathrm{J})$, from which we easily see that $\mathrm{I}, b : \top \sqcap \mathrm{J} = \mathrm{I} \sqcap \mathrm{J}$. Thus we have found J that satisfies both the conditions required to say $\mathrm{I}, b : \top \models (\mathsf{new}\ \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ \Delta')(Q \mid R)$.

Now assume $\mathrm{I} \models (\mathsf{new}\ \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ \Delta')(Q \mid R)$ and $\mathrm{I} \triangleright (\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\alpha} \mathrm{I}' \triangleright S$: we proceed by rule induction to find a matching transition for $\mathrm{I} \triangleright (\mathsf{new}\ \Delta')(Q \mid R)$. The reasoning is by cases on the last rule applied in the derivation.

(G-Open) Our hypothesis is $\mathrm{I} \models (\mathsf{new}\ b : B, \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ b : B', \Delta')(Q \mid R)$, and the transition is $\mathrm{I} \triangleright (\mathsf{new}\ b : B, \Delta)(P \mid R) \xrightarrow{(b)\alpha} \mathrm{I}' \triangleright S$, derived from $\mathrm{I}, b : \top \triangleright (\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\alpha} \mathrm{I}' \triangleright S$. By the property we just showed, we know that $\mathrm{I}, b : \top \models (\mathsf{new}\ \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ \Delta')(Q \mid R)$. Thus, by the induction hypothesis we find a matching transition $\mathrm{I}, b : \top \triangleright (\mathsf{new}\ \Delta)(Q' \mid R) \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright S'$ with $\mathrm{I}' \models S\mathscr{R}S'$ (notice that only this case applies because G-Open moves involve output actions). From the last weak transition we have $\mathrm{I} \triangleright (\mathsf{new}\ b : B', \Delta)(Q \mid R) \stackrel{(b)\alpha}{\Longrightarrow} \mathrm{I}' \triangleright S'$ as desired, by (G-open).

(G-Weak) The proof is similar to the case (G-Open).

(G-Res) Assume $\mathrm{I} \models (\mathsf{new}\ a : A, \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ a : A', \Delta')(Q \mid R)$, and consider the transition $\mathrm{I} \triangleright (\mathsf{new}\ a : A, \Delta)(P \mid R) \xrightarrow{\alpha} \mathrm{I}' \triangleright (\mathsf{new}\ a : A)S$, derived from $\mathrm{I}, a : \top \triangleright (\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\alpha} \mathrm{I}', a : \top \triangleright S$. As in the previous case, we know that $\mathrm{I}, a : \top \models (\mathsf{new}\ \Delta)(P \mid R)\mathscr{R}(\mathsf{new}\ \Delta')(Q \mid R)$. Thus, by the induction hypothesis we find the desired matching transition, namely: either $\mathrm{I}, a : \top \triangleright (\mathsf{new}\ \Delta')(Q \mid R) \stackrel{\alpha}{\Longrightarrow} \mathrm{I}', a : \top \triangleright S'$ with $\mathrm{I}', a : \top \models S\mathscr{R}S'$, when α is a silent or output action, or $\mathrm{I}, a : \top \triangleright (\mathsf{new}\ \Delta')(Q \mid R) \Longrightarrow \mathrm{I}, a : \top \triangleright S'$ with $\mathrm{I}', a : \top \models S\mathscr{R}(S' \mid b\langle\tilde{v}@\tilde{B}\rangle)$, when α is an input action.

In the first case we find the transition $\mathrm{I} \triangleright (\mathsf{new}\ a : A', \Delta')(Q \mid R) \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright (\mathsf{new}\ a : A')S'$, in the second the transition $\mathrm{I}' \triangleright (\mathsf{new}\ a : A', \Delta')(Q \mid R) \Longrightarrow \mathrm{I}', \triangleright (\mathsf{new}\ a : A')S'$. To conclude, we need to show $\mathrm{I}' \models (\mathsf{new}\ a : A)S\mathscr{R}(\mathsf{new}\ a : A')S'$ and $\mathrm{I}' \models (\mathsf{new}\ a : A)S\mathscr{R}(b\langle\tilde{v}@\tilde{B}\rangle \mid (\mathsf{new}\ a : A')S')$. The first relation follows from $\mathrm{I}', a : \top \models S\mathscr{R}S'$, by applying the definition of the relation $\mathscr{R}$. The second relation follows similarly.

(G-Par) Our hypothesis is $\mathrm{I} \models (P \mid R)\mathscr{R}(Q \mid R)$, and we have two sub-cases depending on whether the move comes from P or from R.

If P moves, then the transition is $\mathrm{I} \triangleright (P \mid R) \xrightarrow{\alpha} \mathrm{I}' \triangleright (P' \mid R)$, derived from $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$. From our hypothesis we know that $\mathrm{I} \models P \approx_a^\pi Q$. Then, if α is a silent or output action, we find Q' such that $\mathrm{I} \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$ and $\mathrm{I}' \models P' \approx_a^\pi Q'$, and from the last weak transition we have $\mathrm{I} \triangleright Q \mid R \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q' \mid R$. We can conclude because from $\mathrm{I}' \models P' \approx_a^\pi Q'$ and from $\mathrm{I} \vdash R$ we have $\mathrm{I}' \models (P' \mid R)\mathscr{R}(Q' \mid R)$ by definition. If $\alpha = (\Delta)a(\tilde{v}@A)$ we either reason as before, or find Q' such that $\mathrm{I}, \Delta \triangleright Q \Longrightarrow \mathrm{I}' \triangleright Q'$ and $\mathrm{I}' \models P' \approx_a^\pi (Q' \mid a\langle\tilde{v}@A\rangle)$. From the last weak transition, we have $\mathrm{I}, \Delta \triangleright Q \mid R \Longrightarrow \mathrm{I}' \triangleright Q' \mid R$. We can conclude because from $\mathrm{I}' \models P' \approx_a^\pi (Q' \mid a\langle\tilde{v}@A\rangle)$ and $\mathrm{I} \vdash R$ we have $\mathrm{I}' \models (P' \mid R)\mathscr{R} \equiv (Q' \mid R \mid a\langle\tilde{v}@A\rangle)$ by definition, as desired.

If R moves, then the transition is $\mathrm{I} \triangleright (P \mid R) \xrightarrow{\alpha} \mathrm{I}' \triangleright P \mid R'$, derived from $\mathrm{I} \triangleright R \xrightarrow{\alpha} \mathrm{I}' \triangleright R'$. From our hypothesis, we know that $\mathrm{I} \models P \approx_a^\pi Q$. By Lemma 16, we know that $\mathrm{I}' = \mathrm{I}, \mathrm{I}''$, for a suitable I''. Thus, by Proposition 22, we have $\mathrm{I}' \models P \approx_a^\pi Q$. Now we can proceed as in the previous case.

(G-Reduce) In this case, the transition has the form $\mathrm{I} \triangleright (\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\tau} \mathrm{I} \triangleright S$, derived from $(\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\tau} S$. An inspection of the labelled transitions shows that the last transition has, in fact, the form $(\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\tau} (\mathsf{new}\ \Delta)T$, derived from $P \mid R \xrightarrow{\tau} T$. This transition may be the result of of P or R making an autonomous silent transition: these two cases are similar to the two sub-cases we just worked out above. In particular, if $P \xrightarrow{\tau} P'$, then $T = P' \mid R$. Furthermore, from $P \xrightarrow{\tau} P'$, by Lemma 12, we know that $\mathrm{I} \sqcap \mathrm{J} \triangleright P \xrightarrow{\tau} \mathrm{I} \sqcap \mathrm{J} \triangleright P'$. Then a matching transition can be found by observing that $\mathrm{I} \sqcap \mathrm{J} \models P \approx_a^\pi Q$ implies $\mathrm{I} \sqcap \mathrm{J} \triangleright Q \Longrightarrow \mathrm{I} \sqcap \mathrm{J} \triangleright Q'$ and $\mathrm{I} \sqcap \mathrm{J} \models P' \approx_a^\pi Q'$. Then, again by Lemma 12, we also have $Q \Longrightarrow Q'$, and we

can construct the weak transition $\mathrm{I} \triangleright (\mathsf{new}\ \Delta)(Q \mid R) \Longrightarrow \mathrm{I} \triangleright (\mathsf{new}\ \Delta)(P' \mid R)$ which proves this case. The case when R moves is similar.

We have two further sub-cases, that arise when the transition is the result of the interaction between the two processes.

1. $P \mid R \xrightarrow{\tau} (\mathsf{new}\ \Delta_P)(P' \mid R')$ because (i) $P \xrightarrow{(\Delta_P)a\langle\tilde{v}@\tilde{B}\rangle} P'$, (ii) $R \xrightarrow{a(\tilde{v}@\tilde{B}')} R'$ and $\tilde{B} <: \tilde{B}'$

2. $P \mid R \xrightarrow{\tau} (\mathsf{new}\ \Delta_R)(P' \mid R')$ because (i) $P \xrightarrow{a(\tilde{v}@\tilde{B}')} P'$, (ii) $R \xrightarrow{(\Delta_R)a\langle\tilde{v}@\tilde{B}\rangle} R'$ and $\tilde{B} <: \tilde{B}'$

In case (1) the transition in our hypothesis is $\mathrm{I} \triangleright (\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\tau} \mathrm{I} \triangleright (\mathsf{new}\ \Delta, \Delta_P)(P' \mid R')$, and we need to find a matching transition from $\mathrm{I} \triangleright (\mathsf{new}\ \Delta')(Q \mid R)$. From (ii), and $\mathrm{I} \sqcap \mathrm{J} \vdash R$, by Subject Reduction, $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B}' \vdash R'$. From (i), by Lemma 12, we infer $\mathrm{I} \sqcap \mathrm{J} \triangleright P \xrightarrow{(|\Delta_P|)a\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \triangleright P'$. From the hypothesis $\mathrm{I} \sqcap \mathrm{J} \models P \approx_a^\pi Q$, we have $\mathrm{I} \sqcap \mathrm{J} \triangleright Q \xRightarrow{(|\Delta_Q|)a\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \triangleright Q'$ with $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \models P' \approx_a^\pi Q'$. From $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : B' \vdash R'$, by weakening $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : B \vdash R'$. Furthermore, from the last weak transition, by Lemma 12, we know that there exists $\tilde{A} <: \tilde{B}$ and Δ_Q such that $Q \xRightarrow{(\Delta_Q)a\langle\tilde{v}@\tilde{A}\rangle} Q'$. From this, from $\tilde{B} <: \tilde{B}'$, from transitivity of $<:$, and from (ii) we find the weak transition $Q \mid R \Longrightarrow (\mathsf{new}\ \Delta_Q)(Q' \mid R')$, and from this the typed transition $\mathrm{I} \triangleright (\mathsf{new}\ \Delta')(Q \mid R) \Longrightarrow \mathrm{I} \triangleright (\mathsf{new}\ \Delta, \Delta_Q)(Q' \mid R')$. Having proved that $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : B \models P' \approx_a^\pi Q'$ and $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : B \vdash R'$, it remains to show that Γ, Δ, Δ_P and $\Gamma, \Delta', \Delta_Q$ are compatible with $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : B$. This follows by Lemma 17, from the hypothesis that Γ, Δ and Γ, Δ' are compatible with $\mathrm{I} \sqcap \mathrm{J}$.

In case (2) the transition in our hypothesis is $\mathrm{I} \triangleright (\mathsf{new}\ \Delta)(P \mid R) \xrightarrow{\tau} \mathrm{I} \triangleright (\mathsf{new}\ \Delta, \Delta_R)(P' \mid R')$, and we need to find a matching transition from $\mathrm{I} \triangleright (\mathsf{new}\ \Delta')(Q \mid R)$. From (ii), and $\mathrm{I} \sqcap \mathrm{J} \vdash R$, by Subject Reduction, $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash R'$, and $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash v : B$. Since $B <: B'$, this implies $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash v : B'$. From this and from (i), by Lemma 12, $\mathrm{I} \sqcap \mathrm{J} \triangleright P \xrightarrow{(\Delta_R)a(\tilde{v}@\tilde{B}')} \mathrm{I} \sqcap \mathrm{J}, \Delta_R \triangleright P'$. From the hypothesis $\mathrm{I} \sqcap \mathrm{J} \models P \approx_a^\pi Q$, we have:

(a) $\mathrm{I} \sqcap \mathrm{J} \triangleright Q \xRightarrow{(\Delta_R)a(\tilde{v}@\tilde{B}')} \mathrm{I} \sqcap \mathrm{J}, \Delta_R \triangleright Q'$, with $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models P' \approx_a^\pi Q'$, or

(b) $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \triangleright Q \Longrightarrow \mathrm{I} \sqcap \mathrm{J}, \Delta_R \triangleright Q'$, with $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models P' \approx_a^\pi (Q' \mid a\langle\tilde{v}@\tilde{B}''\rangle)$, for all $B'' <: B'$ such that $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models \tilde{v} : B''$.

In case (a) we find the desired matching transition (a) reasoning as in case (1) above. We examine case (b). From the weak transition $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \triangleright Q \Longrightarrow \mathrm{I} \sqcap \mathrm{J}, \Delta_R \triangleright Q'$, by Lemma 12, we obtain $Q \Longrightarrow Q'$. Now, an analysis of transition $R \xrightarrow{(\Delta_R)a\langle\tilde{v}@\tilde{B}\rangle} R'$ shows that $R \equiv (\mathsf{new}\ \Delta_R)(R' \mid a\langle\tilde{v}@\tilde{B}\rangle)$. From this and $Q \Longrightarrow Q'$, we obtain $Q \mid R \Longrightarrow\equiv (\mathsf{new}\ \Delta_R)(Q' \mid a\langle\tilde{v}@\tilde{B}\rangle \mid R')$ from which we may construct the typed

action $\mathrm{I} \triangleright (\mathsf{new}\ \Delta')(Q \mid R) \Longrightarrow \mathrm{I} \triangleright (\mathsf{new}\ \Delta', \Delta_R)(Q' \mid a\langle \tilde{v}@\tilde{B}\rangle \mid R')$. Now we observe that $\mathrm{I} \sqcap \mathrm{J} \vdash R$ implies $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash R'$ and $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash \tilde{v} : B$. Hence $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models P' \approx_a^\pi (Q' \mid a\langle \tilde{v}@\tilde{B}\rangle)$ and it only remains to be proved that Γ, Δ, Δ_R and $\Gamma, \Delta', \Delta_R$ are compatible with $\mathrm{I} \sqcap \mathrm{J}, \Delta_R$. But this follows directly by the hypothesis that Γ, Δ and Γ, Δ' are compatible with $\mathrm{I} \sqcap \mathrm{J}$.

□

Remark The hypothesis that $\mathrm{I} \vdash R$ is crucial for the proof. Furthermore, we observe that some of the standard properties that hold for asynchronous bisimilarity, do not hold for its type indexed counterpart. In particular, $\mathrm{I} \models P \approx_a^\pi Q$ does not imply $\mathrm{I} \models P \mid P \approx_a^\pi Q \mid Q$. To see that, consider the following counter-example:

$$P = n().n().m\langle\rangle \mid n\langle\rangle, \quad Q = n().n() \mid n\langle\rangle, \quad \mathrm{I} = n : \top, m : \mathsf{r}$$

That $\mathrm{I} \models P \approx_a^\pi Q$ follows by observing that the only available action for P and Q is the internal synchronization on n: in particular, the environment may not observe any action on n because it does not have capabilities on n. On the other hand, $\mathrm{I} \not\models P \mid P \approx_a^\pi Q \mid Q$ as in this case there exists a transition $\mathrm{I} \triangleright P \mid P \stackrel{m\langle\rangle}{\Longrightarrow} \mathrm{I} \triangleright P'$ that cannot be matched by $\mathrm{I} \triangleright Q \mid Q$.

Reasoning in the same way, we also see that $\mathrm{I} \models P \approx_a^\pi Q$ does not imply $\mathrm{I} \not\models !P \approx_a^\pi !Q$.

We have thus proved all that is needed to show the desired soundness result.

Theorem 25 (Soundness). *The relation $\approx_a^\pi$ is barb preserving, reduction closed and contextual, hence contained in $\cong^\pi$.*

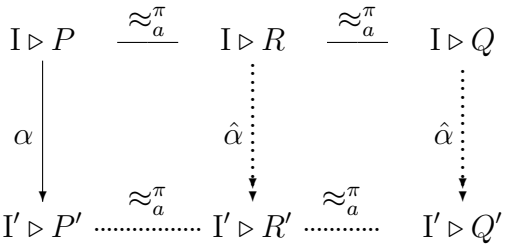
Proof. Apply Propositions 20, 22, 23 and 24. □

Finally we prove that $\approx_a^\pi$ is an equivalence relation. This and the soundness and completeness results guarantee that $\cong^\pi$ is an equivalence relation.

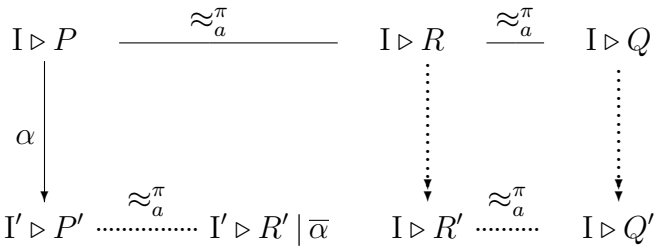
Proposition 26. *$\approx_a^\pi$ is an equivalence relation.*

Proof. The reflexivity of $\approx_a^\pi$ follows trivially by verifying that Id is a bisimulation. Since $\approx_a^\pi$ is symmetric by definition, it remains to prove transitivity. We prove by chasing-diagrams arguments that $\approx_a^\pi \approx_a^\pi$ is indeed a bisimulation. We first need the following result: if $\mathrm{I} \models P \approx_a^\pi Q$ and $\mathrm{I} \triangleright P \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright P'$ then there is Q' such that (i) $\mathrm{I} \triangleright Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{I}' \triangleright Q'$ with $\mathrm{I}' \models P' \approx_a^\pi Q'$ or (ii) $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q'$ with $\mathrm{I}' \models P' \approx_a^\pi Q' \mid \overline{\alpha}$ where α is an input action and $\overline{\alpha}$ is its output co-action. The proof of this result proceeds by straightforward induction on the number of τ reductions.

Let $\mathrm{I} \triangleright P \approx_a^\pi \mathrm{I} \triangleright R \approx_a^\pi \mathrm{I} \triangleright Q$. From $\mathrm{I} \models P \approx_a^\pi R$ and $\mathrm{I} \models R \approx_a^\pi Q$ we infer that there are Γ and Γ' such that $\Gamma <: \mathrm{I}, \Gamma' <: \mathrm{I}$ and $\Gamma \vdash P$, $\Gamma' \vdash Q$. Next suppose $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I} \triangleright P'$. When α is an output action or τ we have the following diagram:



From $\mathrm{I}' \models P' \approx_a^\pi R'$ and $\mathrm{I} \models R' \approx_a^\pi Q'$ we infer that there are Δ and Δ' that respectively define P' and Q' w.r.t. I'.
Now let α be an input action. We have:



To conclude we need to show that $\mathrm{I}' \models P' \approx_a^\pi \approx_a^\pi Q' \,|\, \overline{\alpha}$, that is we have to prove that $\mathrm{I}' \models R' \,|\, \overline{\alpha} \approx_a^\pi Q' \,|\, \overline{\alpha}$. Let $\alpha = (\tilde{c} : \tilde{C})a(\tilde{v}@\tilde{A})$; thus $\overline{\alpha} = a\langle \tilde{v}@\tilde{A} \rangle$. By Lemma 12 we know that $\mathrm{I}' = \mathrm{I}, \tilde{c} : \tilde{C}$ and $\mathrm{I}(a)^{\mathsf{w}} \downarrow$ and $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v}@\tilde{A}$. From these results it' easy to infer that $\mathrm{I}' \vdash \overline{\alpha}$. We apply Proposition 22 to $\mathrm{I} \models R' \approx_a^\pi Q'$ obtaining $\mathrm{I}' \models R' \approx_a^\pi Q'$. Then by Proposition 24 we obtain $\mathrm{I}' \models R' \,|\, \overline{\alpha} \approx_a^\pi Q' \,|\, \overline{\alpha}$, as needed. By subject reduction we infer that there are Δ, Δ' such that $\Delta <: \mathrm{I}', \Delta' <: \mathrm{I}'$ and $\Delta \vdash P'$ and $\Delta' \vdash Q' \,|\, \overline{\alpha}$; this together with the previous results implies that $\approx_a^\pi \approx_a^\pi$ is a type-indexed relation.

By repeating the same proof for the hypothesis $\mathrm{I} \triangleright Q \xrightarrow{\alpha} \mathrm{I}' \triangleright Q'$ one obtains the symmetry of $\approx_a^\pi \approx_a^\pi$ and in turn that $\approx_a^\pi \approx_a^\pi$ is an asynchronous bisimulation.

□

Completeness

We conclude this part by showing that bisimulation coincides with behavioral equivalence. As in [60] the completeness result relies on that the observations of processes we make in typed actions in Table 2.3 are contextually valid. The behaviour of the environment that interacts with processes through typed actions and learn capabilities from or send (possibly fresh) capabilities to them, can be actually represented

through well-typed contexts. We formalize these intuitions by associating each typed input and output action to a typed context that mimics the behaviour of the environment in the typed transition. The results we gain say that labelled transitions induce reductions in their correspective label contexts and vice versa.

We will use the following abbreviations. Let $\mathrm{I} = \tilde{a} : \tilde{T}$. We write $n\langle \mathrm{I} \rangle$ for the process $n\langle \tilde{a}@\tilde{T} \rangle$; we indicate with (I) the types $\tilde{T}$ and with a_{I} the values $\tilde{a}$. We let $\mathrm{I}\,\mathsf{after}\,(\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle$ be $\mathrm{I} \sqcap \tilde{b}@\tilde{B}$, $\mathrm{I}\,\mathsf{after}\,(\tilde{c} : \tilde{C})a(\tilde{b} : \tilde{B})$ be $(\mathrm{I}, \tilde{c} : \tilde{C})$ and $\mathrm{I}\,\mathsf{after}\,\tau$ be I.

Proposition 27. *If* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$, *then* $\mathrm{I}' = \mathrm{I}\,\mathsf{after}\,\alpha$.

Proof. By induction on the length of the inference $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$ and case analysis on the last rule used. □

Proposition 28 (Contextuality of labels). *For each label* $\alpha \neq \tau$ *and environment* I, *there exists a process* C^{I}_{α} *and a name* e *fresh to* I *such that* $\mathrm{I}, e : \mathsf{rw} \vdash C^{\mathrm{I}}_{\alpha}$ *and for all* P *for which* $\mathrm{I} \triangleright P$ *is a configuration one has:*

1. *If* $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$, *then* $C^{\mathrm{I}}_{\alpha} \mid P \Longrightarrow (\mathsf{new}\ \Delta)\ P' \mid e\langle \mathrm{I}' \rangle$ *where* $\mathrm{dom}(\Delta) = bn(\alpha)$
2. *Conversely, let* $C^{\mathrm{I}}_{\alpha} \mid P \Longrightarrow (\mathsf{new}\ \Delta)\ P' \mid e\langle \tilde{v}@\tilde{T} \rangle$, *for* Δ *such that* $\mathrm{dom}(\Delta) = bn(\alpha)$ *and* $e \notin \mathrm{dom}(\Delta)$. *Then*
   - *if* α *is* $(\tilde{c})\, a\langle \tilde{b}@\tilde{B} \rangle$ *or* τ, *then* $\mathrm{I} \triangleright P \overset{\alpha}{\Longrightarrow} (\mathrm{I}\,\mathsf{after}\,\alpha) \triangleright P'$
   - *if* α *is* $(\tilde{c} : \tilde{C})a(\tilde{b}@\tilde{B})$ *then* $\mathrm{I} \triangleright P \overset{\alpha}{\Longrightarrow} (\mathrm{I}\,\mathsf{after}\,\alpha) \triangleright P'$ *or* $P' \equiv P'' \mid a\langle \tilde{b}@\tilde{B} \rangle$ *where* $P \Longrightarrow P''$.

Proof. First note, for future reference, that the hypotheses that $\mathrm{I} \triangleright P$ is a configuration and e is fresh to I imply that $e \notin fn(P)$. Then, we define the testing processes C^{I}_{α} as follows:

$$\begin{array}{lcl} C^{\mathrm{I}}_{(\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle} & = & a(\tilde{x}@\tilde{B}).Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}} \\ C^{\mathrm{I}}_{(\tilde{c}:\tilde{C})\ a(\tilde{b}@\tilde{B})} & = & (\mathsf{new}\ \tilde{c} : \tilde{C})\ (a\langle \tilde{b}@\tilde{B} \rangle \mid e\langle \mathrm{I}, \tilde{c} : \tilde{C} \rangle) \end{array}$$

The process $Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}}$ is defined as follows. Let $\tilde{b} = b_1, \ldots, b_n$, and let $\tilde{j}, \tilde{k}$ be the subindexes of $1, \ldots, n$ such that for each $j \in \tilde{j}$ and for each $k \in \tilde{k}$ we have respectively that b_j is free and b_k is bound in $(\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle$. Then $Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}}$ is so defined as to ensure that $Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}}\{\tilde{b}/\tilde{x}\} \Longrightarrow e\langle \mathrm{I} \sqcap \tilde{b} : \tilde{B} \rangle$ iff $\forall j \in \tilde{j}.(x_j = b_j) \wedge \forall k \in \tilde{k}.(x_k \notin \mathrm{dom}(\mathrm{I})) \wedge$ the name equalities of fresh names are respected. For instance the definition for the tuple $(c)(b, c, c)$ and $\tilde{x} = x_1, x_2, x_3$ is $Q^{\mathrm{I}}_{(c)(b,c,c)} \triangleq [x_1 = b]\,([x_2 \notin \mathrm{I}]\,([x_3 = x_2]\, e\langle \mathrm{I} \sqcap \tilde{x} : \tilde{B} \rangle))$. Here $x \notin \mathrm{I}$ is coded by using nested conditionals to check that x is not equal to the names in I.

Proof of (1.) First consider the case when α is $(\tilde{c}:\tilde{C})\ a(\tilde{b}@\tilde{B})$. From the hypothesis $\mathrm{I}\triangleright P \xrightarrow{\alpha} \mathrm{I}'\triangleright P'$, by Lemma 12, we know that $P \xrightarrow{a(\tilde{b}@\tilde{D})} P'$ for types $\tilde{D}$ such that $\tilde{B} <: \tilde{D}$. By Prop. 27 we know $\mathrm{I}' = \mathrm{I}, \tilde{c}:\tilde{C}$. Then, by the (Pi-Com@) rule in Table 2.2, under the condition $\tilde{c} \cap fn(P) = \emptyset$, we obtain $C^{\mathrm{I}}_{\alpha} \mid P \xrightarrow{\tau} (\mathsf{new}\ \tilde{c}:\tilde{C})\ (P' \mid e\langle \mathrm{I}' \rangle)$, as desired.

Now take $\alpha = (\tilde{c})\ a\langle \tilde{b}@\tilde{B} \rangle$. From the hypothesis $\mathrm{I}\triangleright P \xrightarrow{\alpha} \mathrm{I}'\triangleright P'$, again by Lemma 12, we know that $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{b}@\tilde{D} \rangle} P'$ for types $\tilde{D} <: \tilde{B}$ and $\tilde{C}$, and by Prop. 27 we know $\mathrm{I}' = \mathrm{I} \sqcap \tilde{b}:\tilde{B}$. From this, by (Pi-Com@) rule, we derive $C_{\alpha} \mid P \xrightarrow{\tau} (\mathsf{new}\ \tilde{c}:\tilde{C})\ (P' \mid Q^{\mathrm{I}}_{\tilde{b}}\{\tilde{b}/\tilde{x}\})$ and the claim follows by the properties ensured by the definition of $Q^{\mathrm{I}}_{\tilde{b}}$.

Proof of (2.) First consider the case when α is $(\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle$. Since e is fresh to P, the transition $C^{\mathrm{I}}_{\alpha} \mid P \Longrightarrow (\mathsf{new}\ \Delta)\ P' \mid e\langle \tilde{v}@\tilde{T} \rangle$ in the hypothesis must have involved both P and the process C^{I}_{α}. In particular, the transition must have been derived from:

$$P \xRightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{b}@\tilde{D} \rangle} P' \quad \text{and} \quad C^{\mathrm{I}}_{\alpha} \xrightarrow{a(\tilde{b}@\tilde{B})} Q^{\mathrm{I}}_{\tilde{b}}\{\tilde{b}/\tilde{x}\} \Longrightarrow e\langle \tilde{v}@\tilde{T} \rangle$$

with $\tilde{D} <: \tilde{B}$ and $\Delta = (\tilde{c}:\tilde{C})$ and $\tilde{v}@\tilde{T} = \mathrm{I} \sqcap \tilde{b}:\tilde{B}$. From Prop. 27 we infer $\mathrm{I} \sqcap \tilde{b}:\tilde{B} = \mathrm{I}\ \mathsf{after}\ \alpha$. From the left transition above, and the fact that $\tilde{D} <: \tilde{B}$, we derive $\mathrm{I}\triangleright P \xRightarrow{(\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle} (\mathrm{I}\ \mathsf{after}\ \alpha)\triangleright P'$ directly by Lemma 12.

Now take the case when α is $(\tilde{c}:\tilde{C})a(\tilde{b}@\tilde{B})$. Here, from the structure of C^{I}_{α} and the hypothesis $C^{\mathrm{I}}_{\alpha} \mid P \Longrightarrow (\mathsf{new}\ \Delta)\ P' \mid e\langle \tilde{v}@\tilde{T} \rangle$, we have two possible sub-cases:

- either $P \xRightarrow{a(\tilde{b}@\tilde{D})} P'$, $C^{\mathrm{I}}_{\alpha} \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{b}@\tilde{B} \rangle} e\langle \tilde{v}@\tilde{T} \rangle$, with $\tilde{B} <: \tilde{D}$ and $\tilde{v}@\tilde{T} = \mathrm{I}.\tilde{c}:\tilde{C}$,
- or $P' \equiv P'' \mid a\langle \tilde{b}@\tilde{B} \rangle$ with $P \Longrightarrow P''$,

In the second sub-case we are done. In the first, the hypothesis that $\mathrm{I}, e:\mathsf{rw} \vdash C^{\mathrm{I}}_{\alpha}$ and rules (T-New) and (T-Out@) of Table 2.1 imply that $\mathrm{I}, \tilde{c}:\tilde{C} \vdash \tilde{b}:\tilde{B}$ and $\mathrm{I}^{\mathsf{w}}(a)\downarrow$. By Prop. 27 we infer $\mathrm{I} \sqcap \tilde{c}:\tilde{C} = \mathrm{I}\ \mathsf{after}\ \alpha$. Hence, by Lemma 12, we conclude $\mathrm{I}\triangleright P \Longrightarrow (\mathrm{I}\ \mathsf{after}\ \alpha)\triangleright P'$ as desired.

□

To adequately use the information on exported names carried on the fresh channel e, we need an intermediate result. The following proposition says that the capabilities exported through the lts can be actually collected by the environment.

Proposition 29. *Let $\Delta, \tilde{c}:\tilde{C}$ and $\Delta', \tilde{c}:\tilde{C}'$ be compatible with I', let $\Delta, \tilde{c}:\tilde{C} \vdash P$ and $\Delta', \tilde{c}:\tilde{C}' \vdash Q$, and let e be a name fresh to P and Q. Then*
$\mathrm{I}, e:\mathsf{rw} \models (\mathsf{new}\ \tilde{c}:\tilde{C})\ P \mid e\langle \mathrm{I}' \rangle \cong^{\pi} (\mathsf{new}\ \tilde{c}:\tilde{C}')\ Q \mid e\langle \mathrm{I}' \rangle$ for any environment I implies $\mathrm{I}' \models P \cong^{\pi} Q$.

Proof. The proof proceeds by co-induction. Let $\mathscr{R}$ be the type-indexed relation defined as follows.

$\mathrm{I}' \models P \mathscr{R} Q$ whenever

- $\Delta, \tilde{c} : \tilde{C}$ and $\Delta', \tilde{c} : \tilde{C}'$ are compatible with I'
- there is I and $e : \mathsf{rw}$ fresh to P, Q s.t.
  $\mathrm{I}, e : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e\langle \mathrm{I}' \rangle) \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}')(Q \mid e\langle \mathrm{I}' \rangle)$

We prove that $\mathscr{R}$ is is a typed behavioural equivalence, hence $\mathscr{R}$ is included in $\cong^{\pi}$.

Barb preservation To see that $\mathscr{R}$ preserve barbs, suppose $\mathrm{I}' \models P \downarrow_a$. We choose a process R as testing context to receive the environment I' from e and subsequently unblock the barb $P \downarrow_a$ by executing an input on a; a fresh channel is used to signal the success of the operation. To ease the notation, we write $a().P$ to indicate the process $a(\tilde{x}@\top).P$ whenever the bound variables $\tilde{x}$ do not occur in P and the arity of $\tilde{x}$ is clear from the context.

Take $e' : \mathsf{rw}$ fresh to $\mathrm{I}, e : \mathsf{rw}$ and consider the process $R = e(\tilde{x}@(\mathrm{I}')).x_a().e'\langle\rangle$ where x_a is the component of $\tilde{x}$ that will become bound to a whenever R will receive the capabilities $v_{\mathrm{I}'}@(\mathrm{I}')$.

From the contextuality of $\cong^{\pi}$, particularly from the weakening clause, we infer $\mathrm{I}, e : \mathsf{rw}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e\langle \mathrm{I}' \rangle) \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}')(Q \mid e\langle \mathrm{I}' \rangle)$. Next, we use the closure under parallel composition clause to compose both sides with R.

To see that $e : \mathsf{rw}, e' : \mathsf{rw} \vdash R$, note that from $\mathrm{I}' \models P \downarrow_a$ we obtain $\mathrm{I}' \vdash a : \mathsf{r}$. From $\mathrm{I}'(a) <: \mathsf{r}$ we infer $\mathrm{I}', e : \mathsf{rw}, e' : \mathsf{rw}, \tilde{x} : (\mathrm{I}') \vdash x_a().e'\langle\rangle$ since $\mathrm{I}'(x_a)^{\mathsf{r}} \downarrow$; then the thesis easily follows.

We obtain

$$\mathrm{I}, e : \mathsf{rw}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e\langle \mathrm{I}' \rangle) \mid R \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}')(Q \mid e\langle \mathrm{I}' \rangle) \mid R\ .$$

From the untyped semantics (rule (Pi@-Com) in Tab. 2.2) we obtain

$$e\langle \mathrm{I}' \rangle \mid R \xrightarrow{\tau} a().e'\langle\rangle$$

From $P \downarrow_a$ and (Pi-Com@) and the previous result we deduce

$$P \mid e\langle \mathrm{I}' \rangle \mid R \Longrightarrow \downarrow_{e'}$$

From Lemma 12 we easily infer

$$\mathrm{I}, e : \mathsf{rw}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e\langle \mathrm{I}' \rangle) \mid R \Downarrow_{e'}\ .$$

Since $\cong^{\pi}$ preserve barbs, it needs to be

$$\mathrm{I}, e : \mathsf{rw}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(Q \mid e\langle \mathrm{I}' \rangle) \mid R \Downarrow_{e'}\ .$$

Since e' is fresh to Q, the barb e' come from R. To unblock this barb, it's needed to unblock the channel a. Since e' is fresh to Q, the only possibility for Q to unblock R is that $Q \Downarrow_a$. This and $\mathrm{I}(a)^{\mathsf{r}} \downarrow$ let us conclude $\mathrm{I} \models Q \Downarrow_a$.

Reduction closure of $\mathscr{R}$ is trivial. We concentrate on contextuality, namely closure under parallel composition and new. Weakening is straightforward.

Closure under parallel composition Suppose $\mathrm{I}' \models P \mathscr{R} Q$ and $\mathrm{I}' \vdash R$. We need to show that $\mathrm{I}' \models P \mid R \mathscr{R} Q \mid R$.

By hypothesis $\mathrm{I}, e : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e\langle \mathrm{I}' \rangle) \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}')(Q \mid e\langle \mathrm{I}' \rangle)$. We choose $e' : \mathsf{rw}$ fresh to P and Q and R (we also assume that e is fresh to R) and we define $R' = e(\tilde{x}@(\mathrm{I}')).(R \mid e'\langle v_{\mathrm{I}'}@(\mathrm{I}') \rangle)[\tilde{x}/v_{\mathrm{I}'}]$. Since the free names occurring in $e(\tilde{x}@(\mathrm{I}')).(R \mid e'\langle v_{\mathrm{I}'}@(\mathrm{I}') \rangle)$ are e, e' and $v_{\mathrm{I}'}$ (this follows from $\mathrm{I}' \vdash R$) we easily obtain $e : \mathsf{rw}, e' : \mathsf{rw} \vdash R'$.

We exploit contextuality of $\cong^{\pi}$ (namely closure under weakening, new names and parallel composition) and obtain

$$\mathrm{I}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C}, e : \mathsf{rw})(P \mid e\langle \mathrm{I}' \rangle \mid R') \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}', e : \mathsf{rw})(Q \mid e\langle \mathrm{I}' \rangle \mid R') \ .$$

Since e is fresh to P it's easy to check that

$$\mathrm{I}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C}, e : \mathsf{rw})(P \mid e\langle \mathrm{I}' \rangle \mid R') \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e'\langle \mathrm{I}' \rangle) \mid R)$$

Note indeed that the term on the left side reach in one reduction the term on the right, and that the reduction is non-blockable as P cannot interact with R'.
Similarly from e fresh to Q we have

$$\mathrm{I}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C}, e : \mathsf{rw})(Q \mid e\langle \mathrm{I}' \rangle \mid R') \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C})(Q \mid e'\langle \mathrm{I}' \rangle) \mid R)$$

These equalities let us infer

$$\mathrm{I}, e' : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e'\langle \mathrm{I}' \rangle) \mid R) \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C})(Q \mid e'\langle \mathrm{I}' \rangle) \mid R) \ .$$

Since $e' : \mathsf{rw}$ is fresh to P, Q, and by hypothesis on I, this let us infer $\mathrm{I} \models P \mid R \mathscr{R} Q \mid R$, as desired.

Closure under new Let $\mathrm{I}' = \mathrm{J}, a : A$ and $\mathrm{I}' \models P \mathscr{R} Q$, and let $(\Delta, \tilde{c} : \tilde{C})(a) = B$ and $(\Delta', \tilde{c} : \tilde{C}')(a) = C$. We need to show that $\mathrm{J} \models (\mathsf{new}\ a : B)P \mathscr{R} (\mathsf{new}\ a : C)Q$.

Let $\Gamma = \Delta, \tilde{c} : \tilde{C} \setminus a : B$ and $\Gamma' = \Delta', \tilde{c} : \tilde{C} \setminus a : C$. From the hypotheses $\Delta, \tilde{c} : \tilde{C} \vdash P$ and $\Delta', \tilde{c} : \tilde{C}' \vdash Q$ and with $\Delta, \tilde{c} : \tilde{C}$, $\Delta', \tilde{c} : \tilde{C}'$ compatible with I' we infer

$$\Gamma \vdash (\mathsf{new}\ a : B)P \quad \text{and} \quad \Gamma' \vdash (\mathsf{new}\ a : C)Q$$

From the compatibility hypotheses we easily obtain that Γ and Γ' are compatible with J.

Next, consider the hypothesis $\mathrm{I}, e : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C})(P \mid e\langle \mathrm{I}' \rangle) \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}')(Q \mid e\langle \mathrm{I}' \rangle)$. There are two cases corresponding to (i) $a \in \tilde{c}$ and (ii) $a \in \mathrm{dom}(\mathrm{I})$; this follows from $a \in \mathrm{dom}(\mathrm{I}')$ and that $\cong^{\pi}$ is a type indexed relation. In case (i) we easily infer $\mathrm{J} \models (\mathsf{new}\ a : B)P \mathscr{R} (\mathsf{new}\ a : C)Q$ as $\Gamma \vdash (\mathsf{new}\ a : B)P$ and $\Gamma \vdash (\mathsf{new}\ a : C)Q$ and Γ, Γ' compatible with J. In case (ii) we use contextuality of $\cong^{\pi}$ to infer

$$(\mathrm{I} \setminus a), e : \mathsf{rw} \models (\mathsf{new}\ \tilde{c} : \tilde{C}, a : B)(P \mid e\langle \mathrm{I}' \rangle) \cong^{\pi} (\mathsf{new}\ \tilde{c} : \tilde{C}', a : C)(Q \mid e\langle \mathrm{I}' \rangle)\ .$$

Again from $\Gamma \vdash (\mathsf{new}\ a : B)P$ and $\Gamma \vdash (\mathsf{new}\ a : C)Q$ and Γ, Γ' compatible with J we obtain $\mathrm{J} \models (\mathsf{new}\ a : B)P \mathscr{R} (\mathsf{new}\ a : C)Q$, as desired.

□

Based on the contextuality of labels (Proposition 28) and on the previous Proposition we show that $\approx_a^{\pi}$ provides a characterization of typed behavioural equivalence.

Theorem 30 (Completeness). *If* $\mathrm{I} \models P \cong^{\pi} Q$, *then* $\mathrm{I} \models P \approx_a^{\pi} Q$.

Proof. Let $\mathscr{R}$ be the type-indexed relation defined as follows:

$$\mathrm{I} \models P \mathscr{R} Q \text{ whenever } \mathrm{I} \models P \cong^{\pi} Q$$

We prove that $\mathscr{R}$ is an asynchronous bisimulation up to $\equiv$. Let $\mathrm{I} \models P \mathscr{R} Q$ and suppose $\mathrm{I} \triangleright P \xrightarrow{\alpha} \mathrm{I}' \triangleright P'$.

If α is an output or τ transition, we must find Q' such that $\mathrm{I} \triangleright Q \xRightarrow{\widehat{\alpha}} \mathrm{I}' \triangleright Q'$ and $\mathrm{I} \models P' \cong^{\pi} Q'$.

The case when $\alpha = \tau$ is easy. From $\mathrm{I} \triangleright P \xrightarrow{\tau} \mathrm{I}' \triangleright P'$ and Lemma 12 we know that $P \xrightarrow{\tau} P'$ and $\mathrm{I}' = \mathrm{I}$. Moreover from this transition, since $\mathrm{I} \models P \cong^{\pi} Q$ we find Q' such that $Q \Longrightarrow Q'$, hence by Lemma 12 we have $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q'$, and $\mathrm{I} \models P' \cong Q'$, as desired.

Let then $\alpha = (\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle$, and take e fresh to I such that $\mathrm{I}, e : \mathsf{rw} \vdash C_{\alpha}^{\mathrm{I}}$. By Lemma 12 we know $\mathrm{I}' = \mathrm{I} \sqcap \tilde{b}@\tilde{B}$ and that there are types $\tilde{A} <: \tilde{B}$ and $\tilde{C}$ s.t. $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle \tilde{b}@\tilde{A} \rangle} P'$. Let Δ be the an environment such that $\Delta \vdash P$ and $\Delta <: \mathrm{I}$. By subject reduction $\Delta, \tilde{c} : \tilde{C} \vdash P'$. Since $\tilde{c} \subseteq \tilde{b}$, from $\Delta <: \mathrm{I}$ we infer $\Delta, \tilde{c} : \tilde{C} <: \mathrm{I}'$.

By the hypothesis $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle \tilde{b}@\tilde{B} \rangle} \mathrm{I}' \triangleright P'$ and Proposition 28.1, and previous results for types $\tilde{C}$, we know that $C_{\alpha}^{\mathrm{I}} \mid P \Longrightarrow (\mathsf{new}\ \tilde{c} : \tilde{C})\ P' \mid e\langle \mathrm{I}' \rangle$. Now let $C_f = g\langle\rangle \mid e(\tilde{x}@(\mathrm{I}')).g().f\langle \tilde{x}@(\mathrm{I}') \rangle$, for f and g two names fresh to I. By contextuality, from $\mathrm{I} \models P \cong^{\pi} Q$, we have that

$$\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \models (\mathsf{new}\ e)C_f \mid C_{\alpha}^{\mathrm{I}} \mid P \cong^{\pi} (\mathsf{new}\ e)C_f \mid C_{\alpha}^{\mathrm{I}} \mid Q \tag{2.5}$$

For the left-hand side we may derive the following sequence of reductions (up to structural congruence)

$$(\mathsf{new}\ e)C_f \mid C_{\alpha}^{\mathrm{I}} \mid P \Longrightarrow (\mathsf{new}\ e)C_f \mid (\mathsf{new}\ \tilde{c} : \tilde{C})\ P' \mid e\langle \mathrm{I}' \rangle \Longrightarrow (\mathsf{new}\ \tilde{c} : \tilde{C})\ P' \mid f\langle \mathrm{I}' \rangle$$

We refer to the process reached as to C^P. Notice that $C^P \not\downarrow_g$, since g is not in the free names of P; on the other hand, clearly, $C^P \Downarrow_f$. From repeated applications of (G-Reduce) we infer

$$\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \triangleright (\mathsf{new}\ e)C_f \,|\, C^{\mathrm{I}}_\alpha \,|\, P \Longrightarrow \mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \triangleright C^P$$

From (2.5) and and the last sequence of transitions, we find C^Q such that

$$\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \triangleright (\mathsf{new}\ e)\ C_f \,|\, C_\alpha \,|\, Q \Longrightarrow \mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \triangleright C^Q$$

and $\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \models C^P \cong^\pi C^Q$. This, in particular, means that $\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \not\models C^Q \Downarrow_g$, $\mathrm{I}, f : \mathsf{rw} \models C^Q \Downarrow_f$, and hence implies that the prefix $e(\tilde{x}@(\mathrm{I}')).g()$ must have been consumed in C^Q. Using minor structural rearrangements, we may thus find Q' such that $C^Q \equiv (\mathsf{new}\ \tilde{c} : \tilde{C}')Q' \,|\, f\langle \tilde{v}@\tilde{T}\rangle$. Furthermore, by the construction of C_f, we see that to consume the prefix $e(\tilde{x}@(\mathrm{I}')).g()$ and the output $g\langle\rangle$, the sequence of transitions leading to C^Q must have been derived from the following untyped transition sequence from $C^{\mathrm{I}}_\alpha \,|\, Q$

$$C^{\mathrm{I}}_\alpha \,|\, Q \Longrightarrow (\mathsf{new}\ \tilde{c} : \tilde{C}')\ Q' \,|\, e\langle \mathrm{I}'\rangle$$

and thus we deduce that

$$C^Q \equiv (\mathsf{new}\ \tilde{c} : \tilde{C}')Q' \,|\, f\langle \mathrm{I}'\rangle\ .$$

Now we apply Proposition 28.2 to obtain $\mathrm{I} \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$.

Similarly to the case $\Delta \vdash P$, if Δ' is a type environment such that $\Delta' \vdash Q$ and $\Delta' <: \mathrm{I}$ by subject reduction we obtain $\Delta', \tilde{c} : \tilde{C}' \vdash Q'$. We know $\Delta', \tilde{c} : \tilde{C}' <: \mathrm{I}'$. By $g \notin fn(C^P, C^Q)$ and $\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \models C^P \cong^\pi C^Q$ it's not hard to see that $\mathrm{I}, f : \mathsf{rw} \models C^P \cong^\pi C^Q$. We may then apply Proposition 29 to $\mathrm{I}, f : \mathsf{rw} \models C^P \cong^\pi C^Q$, and we obtain $\mathrm{I}' \models P' \mathscr{R} Q'$, as desired.

We conclude by examining the case when α is an input transition. Let then $\alpha = (\tilde{c} : \tilde{C})\ a(\tilde{b}@\tilde{B})$. The proof proceeds exactly as in the output case (with the difference that here we know that $\tilde{C}' = \tilde{C}$), and leads us find a sequence of transitions

$$C^{\mathrm{I}}_\alpha \,|\, Q \Longrightarrow (\mathsf{new}\ \tilde{c} : \tilde{C})\ Q' \,|\, e\langle \mathrm{I}'\rangle.$$

Now Proposition 28.2 gives us two sub-cases, corresponding to the definition of asynchronous bisimulation: either $\mathrm{I} \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$ or $Q' \equiv Q'' \,|\, a\langle \tilde{b}@\tilde{B}\rangle$ where $Q \Longrightarrow Q''$.

In the first sub-case we reason exactly as for output transitions. In the second, from $Q \Longrightarrow Q''$, by Lemma 12, we know that $\mathrm{I}, \tilde{c} : \tilde{C} \triangleright Q \Longrightarrow \mathrm{I}, \tilde{c} : \tilde{C} \triangleright Q''$. Then $\mathrm{I}' \models P' \mathscr{R} Q'$ follows by Proposition 29 as in the previous cases.

□

Theorem 31 (Soundness and Completeness). $\mathrm{I} \models P \approx_a^\pi Q$ *if and only if* $\mathrm{I} \models P \cong^\pi Q$.

Using this characterization, our equation (2.4) in page 24 can now be proved coinductively. We find a relation containing the process equated by (2.4) and we show that is a bisimulation. To ease the notation, let $S' = !d\langle s@\mathsf{w}\rangle \mid !s(x@\mathsf{J}).print\langle x@\mathsf{J}\rangle$. We have $S = (\mathsf{new}\ s : \mathsf{rw})S'$. By verifying that the relation $\mathscr{R}$ below is a bisimulation and by Theorem 31 one obtains the proof of (2.4).

$$\begin{array}{rcl}\mathscr{R} \triangleq Id & \cup & \{j : \mathsf{J},\, print : \top,\, d : \mathsf{r} \models S \mid C \approx S \mid print\langle j@\mathsf{J}\rangle\} \\ & \cup & \{j : \mathsf{J},\, print : \top,\, d : \mathsf{r} \models (\mathsf{new}\ s : \mathsf{rw})S' \mid s\langle j@\mathsf{J}\rangle \approx S \mid print\langle j@\mathsf{J}\rangle\} \\ & \cup & \{j : \mathsf{J},\, print : \top,\, d : \mathsf{r}, s : T \models S' \mid C \approx S' \mid print\langle j@\mathsf{J}\rangle \mid \mathsf{w} <: T\} \\ & \cup & \{j : \mathsf{J},\, print : \top,\, d : \mathsf{r}, s : T \models S' \mid s\langle j@\mathsf{J}\rangle \approx S' \mid print\langle j@\mathsf{J}\rangle \\ & & \qquad\qquad \mid \mathsf{w} <: T\}\end{array}$$

Next we focus on some distinguishing equations for API@. As we show below, the presence of typed synchronization has some noticeable consequences on the behavioral theory.

A first law we examine is the following generalization of a standard fact about replication.

$$a : \mathsf{r} \models a\langle n@T\rangle \mid !a\langle n@S\rangle \cong !a\langle n@S\rangle \qquad \text{whenever } S <: T \tag{2.6}$$

That this law holds also in case $S \neq T$ is a consequence of our subtype-based synchronization rule. The proof of this equivalence follows by coinduction by verifying that the relation $\mathscr{R}$ below is a bisimulation:

$$\begin{array}{rcl}\mathscr{R} \triangleq Id & \cup & \{a : \mathsf{r} \models_a a\langle n@T\rangle \mid !a\langle n@S\rangle \approx !a\langle n@S\rangle\} \\ & \cup & \{a : \mathsf{r}, n : S' \models_a a\langle n@T\rangle \mid !a\langle n@S\rangle \approx !a\langle n@S\rangle \mid S <: S'\}\end{array}$$

Another interesting consequence of the typed semantics is observed in the behavior of *forwarder* processes like $a(x).b\langle x\rangle$. In particular, we show that one of the distinguishing equations of the asynchronous pi calculus holds only in very specific cases in API@. Let T be a type; we say that T is minimal iff $T <: S$ for every type S. Specifically, we have:

$$a : \mathsf{rw} \models a(x@T).a\langle x@T\rangle \cong \mathbf{0} \qquad \text{iff } T \text{ is minimal}$$

The "if" direction can be proved by co-induction, showing that $\mathscr{R}$ below is an asynchronous bisimulation.

$$\begin{array}{rcl}\mathscr{R} & \triangleq & \{a : \mathsf{rw} \models_a a(x@T).a\langle x@T\rangle \approx \mathbf{0}\} \cup \\ & & \{a : \mathsf{rw}, n : T \models_a a\langle n@T\rangle \approx a\langle n@T\rangle \mid n \text{ name}\}\end{array}$$

Note that n may only be received at T as T has no proper subtypes. For the "only if" direction, it is enough to exhibit a distinguishing process:

$$C_S \triangleq a\langle v@S\rangle \,|\, a(x@S).\omega\langle\rangle$$

It is easy to see that this process tells the two processes apart if S is a proper subtype of T. Let $P \triangleq a(x@T).a\langle x@T\rangle$. For all $S <: T$, we have

$$C_S \,|\, P \longrightarrow a\langle x@T\rangle \,|\, a(x@S).\omega\langle\rangle$$

Now $C_S \,|\, \mathbf{0}$ have two possibilities to match this move:

$$\begin{array}{ll} \text{(i)} & C_S \,|\, \mathbf{0} \xrightarrow{\tau} \omega\langle\rangle \,|\, \mathbf{0} \\ \text{(ii)} & C_S \,|\, \mathbf{0} \Longrightarrow C_S \,|\, \mathbf{0} \end{array}$$

Suppose (i) holds. The couple $(a\langle x@T\rangle \,|\, a(x@S).\omega\langle\rangle, \omega\langle\rangle \,|\, \mathbf{0})$ cannot be in behavioural equivalence as $\omega\langle\rangle \,|\, \mathbf{0} \downarrow_\omega$ and $a\langle x@T\rangle \,|\, a(x@S).\omega\langle\rangle \not\downarrow_\omega$. In case (ii) the same arguments apply for the couple $(a\langle x@T\rangle \,|\, a(x@S).\omega\langle\rangle, C_S \,|\, \mathbf{0})$, and the result follows.

2.5 Relationships with statically typed pi calculi

Having presented our calculus in detail, we are in a position to draw more precise comparisons with the statically typed pi calculus of [60] to which we have referred throughout.

It is a very simple observation that the typed pi calculus can be encoded in API@. Below we give the relevant clauses of a type-directed translation from well-typed pi processes to processes of API@. If A is a fully fledged capability type, we let $|A|$ denote the outermost capability in A, and define:

$$\begin{array}{rcl} [\![\, u\langle\tilde{v}\rangle \,]\!]_\Gamma & = & u\langle\tilde{v}@|\Gamma^{\mathsf{w}}(u)|\rangle \\ [\![\, u(\tilde{x}).P \,]\!]_\Gamma & = & u(\tilde{x}@|\Gamma^{\mathsf{r}}(u)|).[\![\, P \,]\!]_{\Gamma,\tilde{x}:\Gamma^{\mathsf{r}}(u)} \end{array}$$

This encoding has some of the good properties one expects: it is type-preserving, and sound, in the following sense. Let $\Gamma \vdash_\pi P$ and $\mathrm{I} \models_\pi P \cong Q$ denote the typability relation and the asynchronous version of the typed congruence of [60], respectively. Then we have:

Theorem 32. *Let Γ, Γ' be two type environments compatible with I and such that $\Gamma \vdash_\pi P$ and $\Gamma' \vdash_\pi Q$. Then $|\mathrm{I}| \models [\![\, P \,]\!]_\Gamma \cong^\pi [\![\, Q \,]\!]_{\Gamma'}$ implies $\mathrm{I} \models_\pi P \cong Q$.*

Proof. Let $\xrightarrow{\alpha}_{HR}$ denote the transition relation over configurations defined in [60]. It is easy to verify the operational correspondence of the encoding:

- If $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}\rangle}_{HR} \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \triangleright P'$ then $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c})a\langle\tilde{v}@|\Gamma^{\mathsf{w}}(a)|\rangle} |\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \triangleright [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$ for HR types $\tilde{C}$.
  If $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c})a\langle\tilde{v}@\tilde{A}\rangle} |\mathrm{I}| \sqcap \tilde{v} : \tilde{A} \triangleright R$ and $|\Gamma, \tilde{c} : \tilde{C}| \vdash \tilde{v} : \tilde{A}$ then $\exists P'$ s.t. $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}\rangle}_{HR} \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \triangleright P'$ and $R \equiv [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$.

- If $\mathrm{I} \triangleright P \xrightarrow{\tau}_{HR} \mathrm{I} \triangleright P'$ then $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{\tau} |\mathrm{I}| \triangleright [\![P']\!]_\Gamma$.
  If $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{\tau} |\mathrm{I}| \triangleright R$ then $\exists P'$ s.t. $\mathrm{I} \triangleright P \xrightarrow{\tau}_{HR} \mathrm{I} \triangleright P'$ and $R \equiv [\![P']\!]_\Gamma$.

- If $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})}_{HR} \mathrm{I}, \tilde{c} : \tilde{C} \triangleright P'$ then $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \triangleright [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$.
  If $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@\tilde{A})} |\mathrm{I}, \tilde{c} : \tilde{C}| \triangleright R$ and $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \mathrm{I}^{\mathsf{w}}(a)$ then $\exists P'$ s.t. $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})}_{HR} \mathrm{I}' \triangleright P'$ and $R \equiv [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$.

The only subtlety arises in the input case. Suppose $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})}_{HR} \mathrm{I}, \tilde{c} : \tilde{C} \triangleright P'$. From the typed semantics of [60] we infer $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \mathrm{I}^{\mathsf{w}}(a)$. Let $<:_\pi$ be the subtyping preorder on types defined in [60]. By $\Gamma <:_\pi \mathrm{I}$ and contra-variance we have $\mathrm{I}^{\mathsf{w}}(a) <:_\pi \Gamma^{\mathsf{w}}(a)$. We know that $\Gamma^{\mathsf{r}}(a) \downarrow$. By definition of types $\Gamma^{\mathsf{w}}(a) <:_\pi \Gamma^{\mathsf{r}}(a)$. By transitivity of $<:_\pi$ we have $\mathrm{I}^{\mathsf{w}}(a) <:_\pi \Gamma^{\mathsf{r}}(a)$ and by projection $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \Gamma^{\mathsf{r}}(a)$. By type preservation $|\mathrm{I}, \tilde{c} : \tilde{C}| \vdash \tilde{v} : |\Gamma^{\mathsf{r}}(a)|$ and the thesis follows easily by rules (G-IN),(G-WEAK). The other direction is clear.

Now, defined $\mathscr{R}$ to be the type indexed relation such that

$$\mathrm{I} \models_{HR} P \mathscr{R} Q \text{ iff } |\mathrm{I}| \models [\![P]\!]_\Gamma \approx_a [\![Q]\!]_{\Gamma'} \text{ for } \Gamma, \Gamma' \text{ compatible with I and such that } \Gamma \vdash P, \Gamma \vdash Q.$$

The proof follows by showing that $\mathscr{R}$ is an asynchronous bisimulation. The soundness and completeness results of [60] provide the desired result.

Let $\mathrm{I} \triangleright P \xrightarrow{\alpha}_{HR} \mathrm{I}' \triangleright P'$. We show the input and output case; the reduction case is trivial.

(Output) The reduction under analysis has the form $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c})a\langle\tilde{v}\rangle}_{HR} \mathrm{I}' \triangleright P'$. From [60, Lemma 4.3] (that corresponds to Proposition 27) we obtain that $\mathrm{I}' = \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a)$. We apply operational correspondence obtaining: $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c})a\langle\tilde{v}@|\Gamma^{\mathsf{w}}(a)|\rangle} |\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \triangleright [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$ for types $\tilde{C}$. By hypothesis there is R such that $|\mathrm{I}| \triangleright [\![Q]\!]_{\Gamma'} \xRightarrow{(\tilde{c})a\langle\tilde{v}@|\Gamma^{\mathsf{w}}(a)|\rangle} |\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \triangleright R$ and $|\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx^\pi_a R$. By the typing rules in Table 2.1 it easily follows that $|\Gamma', \tilde{c} : \tilde{C}| \vdash \tilde{v} : |\Gamma^{\mathsf{w}}(a)|$.

By operational correspondence we infer that

$$\mathrm{I} \triangleright Q \xRightarrow{(\tilde{c})a\langle\tilde{v}\rangle}_{HR} \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \triangleright Q'$$

and $R \equiv [\![Q']\!]_{\Gamma', \tilde{c}:\tilde{C}}$. From these results we obtain $\mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \models P' \mathscr{R} Q'$, as requested.

(Input) The reduction under analysis has the form $\mathrm{I} \triangleright P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})}_{HR} \mathrm{I}' \triangleright P'$. From [60, Lemma 4.3] we infer $\mathrm{I}' = \mathrm{I}, \tilde{c} : \tilde{C}$. By operational correspondence we have $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \triangleright [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}}$. There are two cases corresponding to $\Gamma'^{\mathsf{r}}(a)$ proper subtype of $\Gamma^{\mathsf{r}}(a)$ or not. Suppose not. From the hypothesis $|\mathrm{I}| \models [\![P]\!]_\Gamma \approx_a [\![Q]\!]_{\Gamma'}$ we infer that two possible matching moves arise for $|\mathrm{I}| \triangleright [\![Q]\!]_{\Gamma'}$. In case the move is matched with a weak input we have $|\mathrm{I}| \triangleright [\![Q]\!]'_\Gamma \xRightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \triangleright R$ with $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}} \approx^\pi_a R$. Since we know $\mathrm{I} \vdash \tilde{v} : \mathrm{I}^{\mathsf{w}}(a)$ by hypothesis on the transition of $\mathrm{I} \triangleright P$, by operational correspondence we infer

$$\mathrm{I} \triangleright Q \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})}_{HR} \mathrm{I}, \tilde{c} : \tilde{C} \triangleright Q'$$

and $R \equiv [\![Q']\!]_{\Gamma', \tilde{c}:\tilde{C}}$. By $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}} \approx^\pi_a R$ we infer $\mathrm{I}, \tilde{c} : \tilde{C} \models P' \mathscr{R} Q'$, as needed. So let consider when we have an asynchronous answer. The matching move is

$$|\mathrm{I}| \triangleright [\![Q]\!]_{\Gamma'} \Longrightarrow |\mathrm{I}| \triangleright R$$

with $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}} \approx^\pi_a R \,|\, a\langle \tilde{v}@|\Gamma^{\mathsf{r}}(a)| \rangle$. From $\Gamma', \tilde{c} : \tilde{C} \vdash \tilde{v}@|\Gamma^{\mathsf{r}}(a)|$ and the hypothesis above on $\Gamma'^{\mathsf{r}}(a)$ we infer $\Gamma'^{\mathsf{r}}(a) = \Gamma^{\mathsf{r}}(a)$. Thus $R \,|\, a\langle \tilde{v}@|\Gamma^{\mathsf{r}}(a)| \rangle \equiv [\![Q' \,|\, a\langle v \rangle]\!]_{\Gamma', \tilde{c}:\tilde{C}}$ and we can infer $\mathrm{I}, \tilde{c} : \tilde{C} \models P' \mathscr{R} Q' \,|\, a\langle v \rangle$ as needed.

Now suppose $\Gamma'^{\mathsf{r}}(a)$ proper subtype of $\Gamma^{\mathsf{r}}(a)$. From the proof of the operational correspondence we know that $\mathrm{I}^{\mathsf{w}}(a) \downarrow$ and $\mathrm{I}^{\mathsf{w}}(a) <:_\pi \Gamma'^{\mathsf{w}}(a) <: \Gamma'^{\mathsf{r}}(a)$. Thus $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \Gamma'^{\mathsf{r}}(a)$ and in turn $|\mathrm{I}| \triangleright [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma'^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \triangleright [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}}$. We proceed as in the previous case and we infer that one of the following cases holds:

$$\text{(i)} \quad |\mathrm{I}| \triangleright [\![Q]\!]_{\Gamma'} \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma'^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \triangleright R \;\wedge\; |\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}} \approx^\pi_a R$$

$$\text{(ii)} \quad |\mathrm{I}| \triangleright [\![Q]\!]_{\Gamma'} \Longrightarrow |\mathrm{I}| \triangleright R \wedge\; |\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma, \tilde{c}:\tilde{C}} \approx^\pi_a R \,|\, a\langle \tilde{v}@|\Gamma'^{\mathsf{r}}(a)| \rangle$$

Case (i) proceeds as above by using operational correspondence. In case (ii) by operational correspondence we infer $\mathrm{I} \triangleright Q \Longrightarrow \mathrm{I} \triangleright Q'$ and $R \equiv [\![Q']\!]_{\Gamma'}$. From $R \,|\, a\langle \tilde{v}@|\Gamma'^{\mathsf{r}}(a)| \rangle \equiv [\![Q' \,|\, a\langle \tilde{v} \rangle]\!]_{\Gamma', \tilde{c}:\tilde{C}}$ we deduce $\mathrm{I}, \tilde{c} : \tilde{C} \models P' \mathscr{R} Q' \,|\, a\langle v \rangle$, as needed. □

Not surprisingly, however, the translation is not fully abstract. To see that, simply take $Q = \mathbf{0}$, and $P = a(x).a\langle x \rangle$ with $\Gamma = a : ((T)^{\mathsf{r}})^{\mathsf{rw}}$, so that $[\![P]\!]_\Gamma = a(x@\mathsf{r}).a\langle x@\mathsf{r} \rangle$. Then, $a : ((T)^{\mathsf{r}})^{\mathsf{rw}} \models_\pi P \cong Q$ while $a : \mathsf{rw} \not\models a(x@\mathsf{r}).a\langle x@\mathsf{r} \rangle \cong \mathbf{0}$, as we have showed previously.

While we do not have a formal separation result between the two calculi, it appears that achieving a fully abstract encoding is just as hard as giving a fully abstract implementation of the pi calculus. In fact, as we observed, the flat capability types of API@ provide much looser control over the dynamic invariants of execution than the fully fledged capability types of [60]. Clearly, this affects the notion of typed equivalence, as the representation of contexts in terms of the typing assumptions they satisfy is much less informative on the behavior of those contexts that it is with traditional typing systems. This loss of control is compensated by the type coercions available for API@ processes to determine the types at which a context receives the emitted values: still, as the example above shows, the underlying equational theory remains affected.

On the other hand, just because its typing system makes looser assumptions on the structure of the typed contexts, API@ lends itself to be implemented into low-level calculi while preserving the typed behavior. We will show how that can be achieved in the next chapters.

2.6 Related work

The asynchronous pi calculus was proposed independently by Honda and Tokoro [65] and by Boudol [26]; more generally, properties of asynchronous labelled transition systems wit input and output capabilities are studied in [105]. Although distinguishing features of the pi calculus can be simulated in its asynchronous fragment, for instance output-prefixing [64, 26], and input-guarded choice [84], in [85] it is elegantly shown that no "reasonable" encoding of the pi calculus into its asynchronous fragment exists , i.e. any possible uniform, compositional encoding is not correct w.r.t. most important observational theories. Behavioral theories for the asynchronous pi calculus and its variants have been studied in [15, 52, 70, 38]. A programming language based on the asynchronous pi calculus is presented in [92]; asynchronous theories for CCS languages have been studied in [95].

Types and advanced techniques for equational reasoning are emerging as powerful tools for the analysis of distributed computations and open systems. Areas of particular relevance to our work are related to the ability of types to talk about resources and their usage [40, 41, 56, 44, 43, 62] and of typed equational theories to characterize observational properties of processes [60, 23, 91, 97, 46, 66]. Particularly, [40, 41] and [56] define type systems respectively for Ambient calculus and Klaim where values emitted in the output constructs are coerced with the intended types to be released and type soundness derives by a combination of static and dynamic typing, as in API@ .

Resource control, in diverse incarnations, has been the focus of extensive foundational research in type theory. Topics considered relevant to our work include the ability to read from and to write to a channel via subtyping [88], the guarantee of secrecy within local computational environments [37, 2, 28], the control on the flow

and sharing of information [81, 78, 79, 80, 61, 58, 42, 31, 49]. An orthogonal approach to types is the use of logics that describe policies for the access to resources, e.g. [5, 3, 57].

With few exceptions, most of these systems rely on classical typing techniques whereby all the system components are type-checked under the same global typing assumptions. This is clearly unrealistic in distributed and dynamic environments as the ones of interest to us, and represent the starting point of this work.

Besides the relations with [60] that we have extensively discussed all along, the typed theory presented in this chapter have several analogies with [23]. In that paper a coinductive characterization of an observational equivalence is given for the pi calculus without matching; as in our case, their bisimulation relies on a labelled transition system recording type information on the types that the observer has on names. However, the absence of matching comports notable differences among our definition of typed bisimulation and the one of [23]. Specifically, in [23] the labels of matching transitions of bisimilar processes may be syntactically different; for instance it can be proved that the processes $a\langle b\rangle.L$ and $a\langle c\rangle.L$, where L is an equator [65] of a and b, are behaviorally equivalent. For this reason the observer's view on the identity of names is separated from their real identity by considering a *typed closure* of the form Δ, σ where Δ is a type environment that collects typed information on the local variables of the observer and σ is a substitution that says the real value of the variables in Δ. The LTS embeds transitions of the form $\Delta, \sigma \triangleright \xrightarrow[\delta]{\alpha} \Delta', \sigma'$ where α is executed by the process and has a standard syntax for the pi calculus, and δ is an environment co-action such that $\delta\sigma$ is the dual of α. For instance when $\alpha = (b)a\langle b\rangle$ we have $\delta\sigma = a(x)$ since there is z such that $\Delta \vdash z : T^{\mathsf{r}}$, $\delta = z(x)$ and $z\sigma = a$.

The bisimulation defined in [23] relates closures $(\Delta, \sigma \triangleright P, \Delta, \rho \triangleright Q)$ such that whenever $\Delta, \sigma \triangleright P$ executes a transition with environment action δ we have that $\Delta, \rho \triangleright Q$ answers with a weak transition labelled with environment action δ, and the converse, but for the case whether δ represents an environment internal synchronization. The synchronization arises among two complementary actions bound to same name as in $\delta = \langle x\langle z\rangle, y(w)\rangle$ with $x\sigma = y\sigma$. Since different matching labels may lead the name $x\sigma$ to be unknown in Δ, ρ the process $\Delta, \rho \triangleright Q$ may answer to this move also by interacting with the process as in $\Delta, \rho \triangleright Q \xrightarrow[x\langle z\rangle]{\alpha'} \xrightarrow[y\langle w\rangle]{\alpha''}$. The internal synchronization is necessary to represent context interactions. For instance the configurations $(\Delta, x : T^{\mathsf{rw}}), \sigma\{b/x\} \triangleright a\langle b\rangle$ and $(\Delta, x : T^{\mathsf{rw}}), \rho \triangleright a\langle c\rangle$ would be possibly equated for some Δ, σ and ρ, while clearly they are not equivalent in contexts where the read and/or the write capability of b are available; a distinguishing context is $\mathscr{C}[-] = a(x).x\langle\rangle \mid b().e\langle\rangle$ whenever we have the typed assumptions $\Delta\sigma \vdash a : (T^{\mathsf{rw}})^{\mathsf{rw}}$ and $\Delta\sigma \vdash e : S^{\mathsf{rw}}$. It is worth nothing that the presence of matching in our high level calculus permits to coinductively rule out the two configurations above by requiring that bisimilar processes exhibit syntactically equal labels.

3

The implementation language

This chapter reviews the language we use to compile our high level processes. The language is a variant of the applied pi calculus [7] and differs from the pi calculus in that the expressions of the language are built around terms generated from applications of functions to names. We use low-level functions implementing list representations, hashing, symmetric/asymmetric encryption, and digital signatures; we also use functions for testing equality of terms and to project elements in tuples of terms. We define labelled semantics for the calculus, and on top a notion of behavioral equivalence indexed by term-environments (cf. [22]). A term environment is an association from variables to terms that formally describes the knowledge of terms by the context. The discriminating power of low-level observers basically correspond to their ability to destruct terms by using (i) matching and (ii) decryption schemes mentioned above. In the next chapter, we will use term environments to represent the low level capabilities corresponding to the high level type assumptions that index typed behavioral equivalences.

3.1 Background on applied pi calculus

The applied pi calculus we use is an asynchronous version of the original calculus of [7], in which we assume that destructors are only used in let-expressions and may not occur in arbitrary terms. This is becoming common practice in the presentations of the applied pi calculus [19, 20, 17].

As for the high-level calculus, we presuppose countable sets of names and variables, under the same notational conventions. In addition the calculus is characterized by a finite set of function symbols Σ from which terms may be formed (see Table 3.1). As in [20], we distinguish constructors and destructors, and use the former to build terms, the latter in let expressions to take terms apart. Constructors are typically ranged over by f, destructors by d. Terms are built around variables and

constructors, expressions correspond to destructor application:

$$\begin{array}{rcll} M, N & ::= & a, b, ... & \textit{channel names} \\ & & x, y, ... & \textit{variables} \\ & & f(M_1, ..., M_n) & \textit{constructor application} \\ E & ::= & d(\tilde{M}) & \textit{expression} \end{array}$$

A value is a term without variables. We always assume that constructors are applied consistently with their arity. Processes are defined as follows:

$$\begin{array}{rcl} P, Q, & ::= & \mathbf{0} \mid M\langle \tilde{N} \rangle \mid M(\tilde{x}).P \mid P \,|\, Q \mid (\mathsf{new}\ n)P \mid \\ & & \mathbf{!}\, P \mid \mathsf{let}\ x = E\ \mathsf{in}\ P\ \mathsf{else}\ Q \end{array}$$

Input prefixing, let, and restriction are binders: $M(x).P$ and let $x = E$ in P else Q bind the variable x in P, (new n)P binds the name n in P. The notions of free and bound names/variables arise as expected. The process let $x = E$ in P else Q tries to evaluate E; if that succeeds x is bound to the resulting term and the process continues as P (with the substitution in place). Otherwise the process reduces to Q. The evaluation of E is governed by a set of definitions which give semantics to the destructor. Each definition has the form $d(\tilde{M}) \doteq N$ where the terms $\tilde{M}$ and N have no free names and $fv(N) \subseteq fv(\tilde{M})$. Then $d(\tilde{M})$ is defined only if there is a definition $d(\tilde{M'}) \doteq N$ and a substitution σ such that $\tilde{M} = \tilde{M'}\sigma$, in which case $d(\tilde{M})$ evaluates to the term $N\sigma = N^*$, noted $d(\tilde{M}) \to N^*$. Conversely, we note $d(\tilde{M}) \not\to$ whenever there is no $\tilde{M'}\sigma = \tilde{M}$ with a defining equation. The destructors under consideration together with their defining equations are listed in Table 3.1. Most of them are inherent to cryptography and will be presented in Section 3.1; rules (3.8)-(3.11) will be introduced in the next Chapter. We say that a constructor f is *one-way* if no destructor application ever returns its argument(s).

We always omit trailing nil processes and similarly write let $x = E$ in P instead of let $x = E$ in P else $\mathbf{0}$. We also use multiple let-bindings instead of writing the corresponding nested definitions.

Evaluation contexts are processes with one hole that are built apart the following grammar:

$$C, D \ ::= \ - \mid (\mathsf{new}\ a)C \mid C \,|\, P \mid P \,|\, C$$

We will often write $C[-]$ to explicit that C is a context; we indicate with $C[P]$ the process obtained by replacing the hole $-$ with the process P.

We implement recursion by blocking/unblocking replications prefixed by a private input:

$$\mathsf{rec}\, X.P \triangleq (\mathsf{new}\ a)\ (a\langle\rangle \,|\, !a().P\{a\langle\rangle/X\}) \quad a \notin fn(P)$$

Following [20], we define an if $-$ then $-$ else construct in terms of let as shown below:

$$\begin{array}{rcl} \mathsf{if}\ M = N\ \mathsf{then}\ P\ \mathsf{else}\ Q & \triangleq & \mathsf{let}\ x = equals(M, N)\ \mathsf{in}\ P \\ & & \mathsf{else}\ Q \quad (x \notin fv(P, Q)) \end{array}$$

Table 3.1 Constructors and Destructors

Constructors

$$:: (M, N), \emptyset, sk(M), ek(M), dk(M), rd(M), wr(M), p(M), q(M),$$
$$hash(M), pub(M), priv(M), \mathsf{rw}, \mathsf{w}, \mathsf{r}, \top$$

Destructors and their defining equations

$$\pi_i(x_1, \ldots, x_n) \doteq x_i \tag{3.1}$$
$$equals(x, x) \doteq x \tag{3.2}$$
$$hd(x :: y) \doteq x \tag{3.3}$$
$$tl(x :: y) \doteq y \tag{3.4}$$
$$decipher(cipher(x, ek(y)), dk(y)) \doteq x \tag{3.5}$$
$$decipher(cipher(x, sk(y)), sk(y)) \doteq x \tag{3.6}$$
$$verify(sign(x, priv(y)), pub(y)) \doteq x \tag{3.7}$$
$$\Uparrow((x, y, z, w), \mathsf{rw}) \doteq (x, y, z, w) \tag{3.8}$$
$$\Uparrow((x, y, z, w), \mathsf{r}) \doteq (x, \top, z, w) \tag{3.9}$$
$$\Uparrow((x, y, z, w), \mathsf{w},) \doteq (x, y, \top, w) \tag{3.10}$$
$$\Uparrow((x, y, z, w), \top) \doteq (x, \top, \top, w) \tag{3.11}$$

Operations on lists

Our applied pi calculus includes destructors to project the elements of tuples (noted π_i), as well as two constructors for lists, :: (cons) and $\emptyset$ (nil), together with the standard destructors for retrieving the head (hd) and the tail (tl) of a list .

In the following we program some code to fulfill useful tasks on lists. We use the process $\mathsf{if}\ M \notin N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ to test if a term M does not occur in a list N and in case continue as P, else as Q:

$$\begin{aligned}\mathsf{if}\ M \notin N\ \mathsf{then}\ P\ \mathsf{else}\ Q \triangleq\ & (\mathsf{new}\ a)\ a\langle N\rangle \mid !a(z).\mathsf{let}\ h = hd(z), t = tl(z)\ \mathsf{in}\\ & (\mathsf{if}\ h = M\ \mathsf{then}\ Q\ \mathsf{else}\ a\langle t\rangle)\\ & \mathsf{else}\ (\mathsf{if}\ z = M\ \mathsf{then}\ Q\ \mathsf{else}\ P)\end{aligned}$$

Similarly process $\mathsf{let}\ \tilde{y} = ?(M, N)\ \mathsf{in}\ P\ \mathsf{else}\ Q$ retrieves from a list N of associations $(M_I, \tilde{M})$ (where we assume that $\tilde{M}$ has the same arity as $\tilde{y}$) the terms $\tilde{M}$ associated to $M = M_I$ and in case continues as $P\{\tilde{M}/\tilde{y}\}$; if there is not an M_I such that $M = M_I$ we continue as Q. We informally call the list N a *table* and the term M_I an index of $\tilde{M}$.

$$\begin{aligned}&\mathsf{let}\ \tilde{y} = ?(M, N)\ \mathsf{in}\ P\ \mathsf{else}\ Q \triangleq\\ &\quad (\mathsf{new}\ a)a\langle N\rangle \mid !a(w).\mathsf{let}\ h = hd(w), t = tl(w), h_I = \pi_1(h), \tilde{y} = \pi_2(h)\ \mathsf{in}\\ &\qquad (\mathsf{if}\ h_I = M\ \mathsf{then}\ P\ \mathsf{else}\ a\langle t\rangle)\ \mathsf{else}\ R(w, \tilde{y})\\ &R(x, \tilde{y}) \triangleq \mathsf{let}\ h_I = \pi_1(x), \tilde{y} = \pi_2(x)\ \mathsf{in}\ (\mathsf{if}\ h_I = M\ \mathsf{then}\ P\ \mathsf{else}\ Q)\\ &\qquad\qquad \mathsf{else}\ Q\end{aligned}$$

We write $\mathsf{if}\ M \notin Set_n\ \mathsf{then}\ P$, for a process that adds M to a list carried on the channel n, and, in case M does not belong to the list, continues as P .

$$\mathsf{if}\ M \notin Set_n\ \mathsf{then}\ P \triangleq n(y).(\mathsf{if}\ M \notin y\ \mathsf{then}\ n\langle y :: M\rangle \mid P\ \mathsf{else}\ n\langle y\rangle)$$

Constructors and destructors for cryptography

We introduce a few useful function symbols that provide the structure of the cryptosystem employed in the implementation.

The one-way unary constructor *hash* generate a hash from the seed M. The two unary one-way constructors *ek* and *dk* generate encryption and decryption keys $ek(M)$ and $dk(M)$ from a seed M. We often abbreviate $ek(M)$ to M^+ and $dk(M)$ to M^-. We use unary one-way constructors rd, wr, p, q to generate seeds for encryption and decryption keys.

A unary one-way constructor *sk* generates a shared key $sk(M)$ from the seed M. The one-way constructor *pub* generate a public key $pub(M)$ from the seed M while *priv* generates a private key $priv(M)$ from M. We often abbreviate $pub(M)$ with M_{ID}.

Digital signatures are built using the binary constructor *sign* and checked by using the destructor *verify* [19, 20] (Tab. 3.1(3.7)); for instance $(sign(M, priv(N)), pub(N))$ certifies that M has been signed by the identity N. Encrypted packets are formed around the binary constructor *cipher*, and taken apart by using the destructor *decipher*. The behavior of the destructor is given by the two defining equations Tab. 3.1(3.5) and Tab. 3.1(3.6) capturing asymmetric and symmetric encryption, respectively. In the implementation, we use four further constructors, rd, wr, p, q, to construct different sets of keys associated with the same seed (cf. Chapter 4).

We often use the conventional spi calculus notation $\{\tilde{M}\}_N$ for the encrypted packet $cipher(\tilde{M}, N)$; we often overload the notation and write $\{\tilde{M}\}_{priv(N)}$ to indicate the signed packet $sign(\tilde{M}, priv(N))$. We also introduce an explicit form for decryption, that binds multiple variables, as in the original spi calculus.

$$\begin{array}{l}\mathsf{decrypt}\ M\ \mathsf{as}\ \{\tilde{y}\}_N\ \mathsf{in}\ P\ \mathsf{else}\ Q\ \triangleq \\ \quad \mathsf{let}\ x = decipher(M, N)\ \mathsf{in} \qquad (x \notin fv(P, Q)) \\ \quad\quad \mathsf{let}\ y_1 = \pi_1(x), \ldots, y_n = \pi_n(x)\ \mathsf{in}\ P\ \mathsf{else}\ Q\end{array}$$

3.2 Operational semantics and term-indexed behavioural equivalence

The operational semantics is defined in terms of labelled transitions. We chose to not base it on active substitutions as in the original formulation (e.g., similarly to the reduction semantics in [20, 17]). Although several proofs contained in this thesis would be probably helped by using active substitutions (as in proofs for full abstraction for security in the join calculus for which they were designed), we prefer to use a separate environment that represents a nice counterpart to the typing envinroments that index equivalent processes in the source calculus (see Chapter 4), and that are not represented by active subtitutions.

The labelled transitions in Table 3.2 are standard. For simplicity, we require that all synchronizations occur on channel names, rather then arbitrary terms. As usual we require the bound names $\tilde{c}$ in the output action $(\tilde{c})\, a\langle\tilde{M}\rangle$ to be unique. The treatment of let is taken from [20]. The transitions are only defined over closed processes (without free variables).

Behavioural equivalence As in Chapter 2, we rely on a notion of behavioral equivalence based on weak bisimulation, and relative to contexts with a certain knowledge about names and terms. The formal definitions are mostly based on the work of Boreale, De Nicola and Pugliese in [22].

Definition 9 (Term environment). *A term environment ρ is a finite substitution from variables to terms. We write $fn(\rho)$ to mean $fn(\mathrm{Range}(\rho))$. Substitutions may*

Table 3.2 Labelled transitions for the applied pi

$$\frac{}{a\langle \tilde{M} \rangle \xrightarrow{a\langle \tilde{M} \rangle} \mathbf{0}} \quad (\textsc{out})$$

$$\frac{P \xrightarrow{(\tilde{b})\, a\langle \tilde{M} \rangle} P' \qquad c \neq a,\ c \in fn(\tilde{M})}{(\mathsf{new}\ c)P \xrightarrow{(\tilde{b}, c)\ a\langle \tilde{M} \rangle} P'} \quad (\textsc{open})$$

$$\frac{}{a(\tilde{x}).P \xrightarrow{a(\tilde{M})} P\{\tilde{M}/\tilde{x}\}} \quad (\textsc{in})$$

$$\frac{P \xrightarrow{(\tilde{b})\, a\langle \tilde{M} \rangle} P' \quad Q \xrightarrow{a(\tilde{M})} Q' \qquad \tilde{b} \notin fn(Q)}{P \mid Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{b})(P' \mid Q')} \quad (\textsc{close})$$

$$\frac{d(\tilde{M}) \to N \quad P\{N/x\} \xrightarrow{\alpha} P'}{\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \xrightarrow{\alpha} P'} \quad (\textsc{let})$$

$$\frac{d(\tilde{M}) \not\to \quad Q \xrightarrow{\alpha} Q'}{\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \xrightarrow{\alpha} Q'} \quad (\textsc{let-else})$$

$$\frac{P \xrightarrow{\alpha} P' \quad n \notin n(\alpha)}{(\mathsf{new}\ n)P \xrightarrow{\alpha} (\mathsf{new}\ n)P'} \quad (\textsc{New})$$

$$\frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \quad (\textsc{Par})$$

$$\frac{P \xrightarrow{\alpha} P'}{!\,P \xrightarrow{\alpha} !\,P \mid P'} \quad (\textsc{Repl})$$

only be extended with new bindings for fresh variables: $\rho, M/x$ *indicates the extension of* ρ *with* $x \notin \mathrm{dom}(\rho)$.

Given a term environment ρ, we let $A(\rho)$ be the analysis of ρ, that is, the environment obtained by extending ρ with new bindings for the terms resulting from the application of destructors to the range of ρ. Formally:

Definition 10. *The analysis* $A(\rho)$ *of* ρ *is the smallest substitution* σ *extending* ρ *that is closed by the following rule:*

$$\frac{d(\tilde{N}) \doteq N \quad \exists \sigma' \,.\, \tilde{N}\sigma' \subseteq \mathrm{Range}(\sigma) \quad (N\sigma' \notin \mathrm{Range}(\sigma) \wedge z \notin \mathrm{dom}(\sigma))}{N\sigma'/z \in \sigma}$$

Abusing the notation we often write $N \in A(\rho)$ to mean $N \in \mathrm{Range}(A(\rho))$.

Example We find the analysis of the substitution $\rho = \{\top\}_{sk(k)}/x, k/y$. The projection case (3.1) leads to bindings $\{\top\}_{sk(k)}/z$ or k/z that are not introduced since the side condition requires new terms to be not in the range of σ. Similarly (3.2) does not introduce new bindings. Equations (3.3) and (3.4) do not apply as no terms in the range of σ are lists. Similarly (3.7),(3.5) do not apply as no terms in $\mathrm{Range}(\sigma)$ have the required form. The relevant case is the equation (3.6) for symmetric cryptography: $\mathit{decipher}(\mathit{cipher}(x_1, sk(x_2)), sk(x_2)) \doteq x_1$. We let $\sigma' = \top/x_1, k/x_2$. From $\mathit{cipher}(x_1, sk(x_2))\sigma' \in \mathrm{Range}(\rho)$ and $x_2\sigma' \in \mathrm{Range}(\rho)$ we obtain $x_1\sigma'/z \in \sigma$, that is $\top$ is in the range of σ as the environment has built the key $sk(k)$ apart the name k and has used it to open the crypto-packet $\{\top\}_{sk(k)}$. These results let us infer $A(\rho) = \rho, \top/z$.

Definition 11. *Given a process* P *we say that* ρ *defines* P*, written* $\rho \vdash P$*, if* $fv(P) \subseteq \mathrm{dom}(\rho)$ *and* $fn(P) \cap fn(\rho) = \emptyset$*. Given a term* M*, we say that* ρ *defines* M*, written* $\rho \vdash M$*, if* $fv(M) \subseteq \mathrm{dom}(\rho)$ *and* $fn(M) \cap fn(\rho) = \emptyset$.

Definition 12 (Term-indexed relation). *A term-indexed relation* $\mathscr{R}$ *is a family of binary relations between closed processes indexed by term environments. We write* $\rho \models P \mathscr{R} Q$ *(or equivalently* $P \mathscr{R}_\rho Q$*) to mean that* P *and* Q *are related by* $\mathscr{R}$ *at* ρ *and that* $fn(P, Q) \subseteq fn(\rho)$.

We have a notion of contextuality corresponding to that given in Chapter 2(Def. 2). We denote $\rho \setminus n$ the term environment resulting from erasing all bindings M/x such that $n \in fn(M)$.

Definition 13 (Contextuality). *A term-indexed relation* $\mathscr{R}$ *is contextual whenever* $\rho \models P \mathscr{R} Q$ *implies*

(i) if $\rho \vdash R$ *then* $\rho \models (P \mid R\rho)\, \mathscr{R}\, (Q \mid R\rho)$,

(ii) $\rho, n/x \models P \mathscr{R} Q$ *with* $n \notin fn(\rho)$

(iii) $\rho \setminus n \models ((\mathsf{new}\ n)P)\mathscr{R}((\mathsf{new}\ n)Q)$.

The barb predicate is defined relative to a term-environment, as expected:

Definition 14 (Barbs). *Let ρ be a term environment and let P be a closed process. We have that*

- $P\downarrow_a$ *if and only if* $\exists P'$ *such that* $P \xrightarrow{(\tilde{n})a\langle\tilde{M}\rangle} P'$;
- $P\Downarrow_a$ *if and only if* $\exists P'$ *such that* $P \Longrightarrow P' \wedge P'\downarrow_a$;
- $\rho \models P\downarrow_a$ *if and only if* $a \in A(\rho) \wedge\ P\downarrow_a$;
- $\rho \models P\Downarrow_a$ *if and only if* $a \in\ A(\rho) \wedge\ P\Downarrow_a$.

The definition of behavioural equivalence reflects the definition of typed behavioural equivalence for API@ (see Definition 4).

Definition 15 (Behavioural equivalence). *Behavioural equivalence, noted $\cong^{A\pi}$, is the largest symmetric and contextual term-indexed relation $\mathscr{R}$ such that $\rho \models P\mathscr{R}Q$ implies*

(i) *if* $\rho \models P\downarrow_n$ *then* $\rho \models Q\Downarrow_n$

(ii) *if* $P \xrightarrow{\tau} P'$ *then* $\exists Q'.\ Q \Longrightarrow Q'$ *and* $\rho \models P'\mathscr{R}Q'$.

In the low-level setting it is often convenient to consider a stronger version of behavioural equivalence.

Definition 16 (Strong behavioural equivalence). *Strong behavioural equivalence, noted $\simeq^{A\pi}$, is the largest symmetric and contextual term-indexed relation $\mathscr{R}$ such that $\rho \models P\mathscr{R}Q$ implies*

(i) *if* $\rho \models P\downarrow_n$ *then* $\rho \models Q\downarrow_n$

(ii) *if* $P \xrightarrow{\tau} P'$ *then* $\exists Q'.\ Q \xrightarrow{\tau} Q'$ *and* $\rho \models P'\mathscr{R}Q'$.

Example To illustrate the behavioral theory of the calculus, we present a result that shows that the environment do not distinguish between ciphertexts encrypted with keys it does not know.
(Claim) *For all ρ holds* $\rho \models (\mathsf{new}\ k)net\langle\{M\}_{sk(k)}\rangle \simeq^{A\pi} (\mathsf{new}\ n)net\langle\{n\}_n\rangle$.
We give some intuitions that motivate the claim; a formal proof is available in Chapter 5. Indeed it's easy to see that the term-indexed relation containing the processes above is strong barb preserving and strong reduction closed; the main obstacle is to prove that is contextual. To this aim, consider the terms $N_1 = \{M\}_{sk(k)}$ and $N_2 = \{n\}_n$ respectively emitted on the left and on the right side of the equation.

Since N_1 and N_2 are encrypted respectively with terms $sk(k)$ and n containing fresh names, we infer that there is not a term N' such that:

$$decipher(N_1, N'\rho) \to \ \vee \ decipher(N_2, N'\rho) \to$$

where $d(\tilde{M}) \to$ is short for $\exists N \,.\, d(\tilde{M}) \to N$. Similarly one infers that $\not\exists N'$ such that

$$equals(N_1, N'\rho) \to \ \vee \ equals(N_2, N'\rho) \to$$

The application of the remaining destructors or trivially reduce in both sides or does not reduce in both sides. Intuitively, from these results we infer that a process closed by ρ that interacts with both sides and receives these packets and continue as with a let application does not see any difference among the emissions. From this result the claim follows easily by finding an appropriate relation that contains the related processes immersed in applied pi contexts that in the left side are built around $\rho, \{M\}_{sk(k)}/x$ while in the right side are built around $\rho, \{n\}_n/x$ (see Lemma 69).

Strengthening the result by removing the condition on fresh names is particularly delicate. For instance consider the environment $\rho = \rho', e/w, net/x, n/y$ where $e \neq net$. We have the following inequation:

$$\rho \not\models net\langle\{M\}_{sk(k)}\rangle \simeq^{A\pi} net\langle\{n\}_n\rangle \tag{3.12}$$

Indeed the environment knows the seed n and can execute cryptanalysis attacks. As a counter-example based on this, consider the context $R = x(z).\mathsf{if}\ z = \{y\}_y\ \mathsf{then}\ w\langle\rangle$. We have $\rho \models net\langle\{n\}_n\rangle \,|\, R\rho \Downarrow_e$ while $\rho \not\models net\langle\{M\}_{sk(k)}\rangle \,|\, R\rho \Downarrow_e$. As we will see in the next chapter, such attacks can be prevented by inserting confounders in the encrypted packets. Similarly, inequation (3.12) holds when the environment knows the symmetric key $sk(k)$: $\sigma \triangleq \sigma', e/w,\ net/x, sk(k)/y$. The the following context tells apart the two emissions once instantiated by σ: $R = x(z).\mathsf{decrypt}\ z\ \mathsf{as}\ \{\}_y\ \mathsf{in}\ w\langle\rangle$.

The following results prove that $\cong^{A\pi}$ and $\simeq^{A\pi}$ are indeed equivalence relations. We prove the former result; the latter is easier. We first establish useful properties of $\cong^{A\pi}$ (cf. [104]).

Proposition 33. *Assume* $\rho \models P \cong^{A\pi} Q$.

(i) if $\rho \models P \Downarrow_a$ *then* $\rho \models Q \Downarrow_a$

(ii) if $P \Longrightarrow P'$ *then* $\exists Q'.\ Q \Longrightarrow Q'$ *and* $\rho \models P' \cong^{A\pi} Q'$.

Proof. (ii) Let $P \stackrel{\tau^n}{\Longrightarrow} P'$ with $n \geq 0$. We proceed by induction on n. The case $n = 0$ is trivially obtained by choosing $Q' = Q$. To see the case $n + 1$ suppose $P \stackrel{\tau^n}{\Longrightarrow} P' \stackrel{\tau}{\longrightarrow} P^*$. By the I.H. $\rho \models P' \cong^{A\pi} Q'$ we infer that there is Q^* such that $Q' \Longrightarrow Q^*$ and $\rho \models P^* \cong^{A\pi} Q^*$. Thus $Q \Longrightarrow \cong^{A\pi}_{\rho} P^*$, as desired.

(i) Assume $\rho \models P \Downarrow_a$. By definition there is P' such that $P \Longrightarrow P' \xrightarrow{(\tilde{c})a\langle \tilde{M}\rangle}$. We apply (ii) to the hypothesis $\rho \models P \cong^{A\pi} Q$ and infer that $Q \Longrightarrow Q'$ for some Q' s.t. $\rho \models Q' \cong^{A\pi} P'$. From this hypothesis we infer $\rho \models Q' \Downarrow_a$. From $Q \Longrightarrow Q'$ we deduce $\rho \models Q \Downarrow_a$, as desired. □

Proposition 34. $\cong^{A\pi}$ *is an equivalence relation.*

Proof. That $\cong^{A\pi}$ is reflexive is straightforward while the relation is symmetric by definition. To see transitivity, we consider the relation $\cong^{A\pi}\cong^{A\pi}$ and we show that it's a behavioural equivalence. Let $P \cong_\rho^{A\pi} R \cong_\rho^{A\pi} Q$. Since $fn(P,R) \subseteq fn(\rho)$ and $fn(R,Q) \subseteq fn(\rho)$, we obtain $fn(P,Q) \subseteq fn(\rho)$. Let $\rho \models P\downarrow_a$; by definition $\rho \models R \Downarrow_a$. By Proposition 33(i) we have $\rho \models Q \Downarrow_a$, as needed. Now let $P \longrightarrow P'$; by hypothesis we have that there is R' such that $R \Longrightarrow R'$ and $P' \cong_\rho^{A\pi} R'$. By the hypothesis $R \cong_\rho^{A\pi} Q$ and Proposition 33(ii) we infer that there is Q' such that $Q \Longrightarrow Q'$ and $R' \cong_\rho^{A\pi} Q'$, as needed. By gluing $fn(P',R') \subseteq fn(\rho)$ and $fn(R',Q')$ we obtain $fn(P',Q') \subseteq fn(\rho)$. That $\cong^{A\pi}\cong^{A\pi}$ is symmetric can be easily viewed by following the same proof when the hypothesis regards Q. Next we prove that $\cong_\rho^{A\pi}\cong_\rho^{A\pi}$ is contextual. Let $\rho \vdash S$. By hypothesis both $\rho \models P \,|\, S \cong^{A\pi} R \,|\, S$ and $\rho \models R \,|\, S \cong^{A\pi} Q \,|\, S$. Thus $\rho \models P \,|\, S \cong^{A\pi}\cong^{A\pi} Q \,|\, S$, as needed. Moreover $fn(P,S,Q) \subseteq fn(\rho)$. Now let $n \notin fn(\rho)$. We have $\rho, n/x \models P \cong^{A\pi} R$ and $\rho, n/x \models R \cong^{A\pi} Q$; thus $\rho, n/x \models P \cong^{A\pi}\cong^{A\pi} Q$, as desired. Trivially we have $fn(P,Q) \subseteq fn(\rho, n/x)$. Finally we know $\rho \setminus n \models (\mathsf{new}\ n)P \cong^{A\pi} (\mathsf{new}\ n)R$ and $\rho \setminus n \models (\mathsf{new}\ n)R \cong^{A\pi} (\mathsf{new}\ n)Q$; this implies $\rho \setminus n \models (\mathsf{new}\ n)P \cong^{A\pi}\cong^{A\pi} (\mathsf{new}\ n)Q$.From these hypotheses we know $fn((\mathsf{new}\ n)P, (\mathsf{new}\ n)Q) \subseteq fn(\rho \setminus n)$. This together the previous results let us infer that $\cong^{A\pi}\cong^{A\pi}$ is a term-indexed relation, and we are done.

□

Proposition 35. $\simeq^{A\pi}$ *is an equivalence relation.*

We obtain that $\simeq^{A\pi}$ is included in $\cong^{A\pi}$, in the following sense.

Proposition 36. *If* $\rho \models P \simeq^{A\pi} Q$ *then* $\rho \models P \cong^{A\pi} Q$.

Proof. Let $\rho \models P\mathscr{R}Q$ whenever $\rho \models P \simeq^{A\pi} Q$. Contextuality follows straightforwardly from the contextuality of $\simeq^{A\pi}$. To see barb preservation, suppose $\rho \models P\downarrow_a$. Therefore we have that $P \xrightarrow{(\tilde{c})a\langle M\rangle}$, for some a and M. From strong barb preservation of $\simeq^{A\pi}$ we infer $Q \xrightarrow{(\tilde{d})a\langle N\rangle}$ and in turn $\rho \models Q \Downarrow_a$, as desired. To see that $\mathscr{R}$ is reduction closed assume $P \xrightarrow{\tau} P'$. By strong reduction closure of $\simeq^{A\pi}$ we infer that there is Q' such that $Q \xrightarrow{\tau} Q'$ and $\rho \models P' \simeq^{A\pi} Q'$. We have thus found Q' such that $Q \Longrightarrow Q'$ and $\rho \models P'\mathscr{R}Q'$, as desired. □

Similarly to the previous chapter we define a notion of up to technique.

Definition 17 (Up to technique). *A term-indexed relation $\mathscr{S}$ is contextual up to $\mathscr{R}_1$, $\mathscr{R}_2$ (noted up to $\mathscr{R}$ whenever $\mathscr{R}_1 = \mathscr{R} = \mathscr{R}_2$) whenever $\rho \models P \mathscr{S} Q$ implies*

(i) if $\rho \vdash R$ then $\rho \models (P \mid R\rho) \mathscr{R}_1 \mathscr{S} \mathscr{R}_2 (Q \mid R\rho)$,

(ii) $\rho, n/x \models P \mathscr{R}_1 \mathscr{S} \mathscr{R}_2 Q$ with $n \notin fn(\rho)$

(iii) $\rho \setminus n \models (\mathsf{new}\ n) P \mathscr{R}_1 \mathscr{S} \mathscr{R}_2 (\mathsf{new}\ n) Q$.

A term-indexed relation is a behavioural equivalence up to $\mathscr{R}_1$, $\mathscr{R}_2$ if is barb preserving, contextual up to $\mathscr{R}_1$, $\mathscr{R}_2$ and if $\rho \models P \mathscr{S} Q$ and $P \xrightarrow{\tau} P'$ then $\exists Q'$ such that $Q \Longrightarrow Q'$ and $\rho \models P' \mathscr{R}_1 \mathscr{S} \mathscr{R}_2 Q'$.

We introduce an up to technique that will be used to ease bisimulation proofs (see Propositions 40,41).

Definition 18 (Structural congruence). *Structural congruence, noted $\equiv$, is the least equivalence relation that is closed under α-conversion, preserved by the operators of the language, and that satisfies the following axioms:*

$$
\begin{array}{rcll}
P \mid (\mathsf{new}\ a) Q & \equiv & (\mathsf{new}\ a)(P \mid Q) & \text{if } a \notin fn(P) \\
(\mathsf{new}\ a) P & \equiv & P & \text{if } a \notin fn(P) \\
\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q & \equiv & P\{N/x\} & \text{if } d(\tilde{M}) \to N \\
\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q & \equiv & Q & \text{if } d(\tilde{M}) \not\to \\
P \mid Q & \equiv & Q \mid P & \\
!P & \equiv & !P \mid P & \\
P \mid \mathbf{0} & \equiv & P\ . &
\end{array}
$$

As in the API@ calculus we obtain that $\equiv$ commutes with reduction, in the following sense:

Proposition 37. *If $P \equiv Q$ and $P \xrightarrow{\alpha} P'$ then there is Q' s.t. $Q \xrightarrow{\alpha} Q'$ and $P' \equiv Q'$.*

Proof. The proof proceeds by induction on the number of applications of the rewriting rules of Definition 18. The only novelties w.r.t. the analogous result for the pi calculus are the let rules. Let $\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \equiv P\{N/x\}$ since $d(\tilde{M}) \to N$ and assume $\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \xrightarrow{\alpha} P'$. Then it must be the case that this reduction has been inferred by using rule (LET) of Table 3.2, and therefore we also have that $P\{N/x\} \xrightarrow{\alpha} P'$, as needed. The case $\mathsf{let}\ x = d(\tilde{M})\ \mathsf{in}\ P\ \mathsf{else}\ Q \equiv Q$ since $d(\tilde{M}) \not\to$ is handled similarly by using rule (LET-ELSE) .

Proposition 38. *Behavioural equivalence up to $\equiv$ is included in $\cong^{A\pi}$.*

Proof. For reduction closure the proof is similar to Proposition 19 and proceeds by showing by chasing diagrams arguments and using Proposition 37 that $\equiv\cong^{A\pi}\equiv$ is reduction closed. Barb preservation follows from the fact that if $P \equiv Q$ then $P\downarrow_a$ iff $Q\downarrow_a$. Contextuality of $\equiv\cong^{A\pi}\equiv$ is obtained straightforwardly by noting that whenever $\rho \models P \equiv Q$ is a term-indexed relation it is contextual; this holds since by definition $\equiv$ is preserved by the operators of the calculus. □

A term environment informally represents the knowledge of the context based on terms and can be used to tell apart processes interacting with the environment. However this distinguishing power should not increase whenever terms built around known ones are added to the environment. The proposition below shows that this intuition is correct. We first need a technical lemma.

Lemma 39. *Let ρ be a a term environment, let M be a term without free names and suppose $\rho \vdash M$. Then $A(\rho, M\rho/x) = A(\rho)$.*

Proof. It follows from $M\rho/x \in A(\rho)$. To see that, it is enough to consider that $equals(M, M)\rho \to M\rho$. □

Proposition 40 (Weakening). *Let ρ be a a term environment, let M be a term without free names and suppose $\rho \vdash M$. Then $\rho \models P \cong^{A\pi} Q$ implies $\rho, M\rho/x \models P \cong^{A\pi} Q$.*

Proof. We proceed by co-induction and we show that the relation $\mathscr{R}$ such that (i) $\rho, M\rho/x \models P\mathscr{R}Q$ whenever $\rho \models P \cong^{A\pi} Q$ and $\rho \vdash M$ or (ii) $\rho \models P\mathscr{R}Q$ whenever $\rho \models P \cong^{A\pi} Q$ is a behavioural equivalence up to $\equiv$. Proposition 38 guarantees the desired result. To ease the notation we let $N \triangleq M\rho$.
Case (ii) is trivial; we show case (i).

To see barb preservation, let $\rho, N/x \models P\downarrow_a$. This means $a \in A(\rho, N/x)$ and $P \xrightarrow{(\tilde{c})a\langle\tilde{M}\rangle}$. By the lemma introduced above we have that $a \in A(\rho)$ and thus $\rho \models P\downarrow_a$. By barb preservation of $\cong^{A\pi}$ we obtain $\rho \models Q\Downarrow_a$ and in turn $\rho, N/x \models Q\Downarrow_a$, as needed.

Reduction closure is straightforward. To see contextuality, let $\rho, N/x \models P\mathscr{R}Q$. We first show that $(\rho, N/x) \setminus a \models (\mathsf{new}\ a)P\mathscr{R}(\mathsf{new}\ a)Q$. Suppose $a \in fn(N)$; thus $(\rho, N/x)\setminus a = \rho\setminus a$. By contextuality of $\cong^{A\pi}$ we have $\rho\setminus a \models (\mathsf{new}\ a)P \cong^{A\pi} (\mathsf{new}\ a)Q$ and by (ii) and the results above $(\rho, N/x) \setminus a \models (\mathsf{new}\ a)P\mathscr{R}(\mathsf{new}\ a)Q$, as needed. Otherwise let $a \notin fn(N)$. Thus $(\rho, N/x)\setminus a = (\rho\setminus a), N/x$. By $\rho\setminus a \models (\mathsf{new}\ a)P \cong^{A\pi} (\mathsf{new}\ a)Q$ we infer $(\rho, N/x)\setminus a \models (\mathsf{new}\ a)P\mathscr{R}(\mathsf{new}\ a)Q$, as needed. Weakening closure of $\mathscr{R}$ is easily obtained by noting that $fn(N) \subseteq fn(\rho)$. To conclude let $\rho, N/x \vdash R$. We need to show that $\rho, N/x \models P \mid R(\rho, N/x) \equiv \mathscr{R} \equiv Q \mid R(\rho, N/x)$. By definition (a) $M = u$ or (b) $M = f(\tilde{M})$. We show by induction on the shape of M that there exists R' closed by ρ such that $R(\rho, N/x) \equiv R'\rho$. Case (a) is trivial as from $u \in \mathrm{dom}(\sigma)$ we easily obtain the claim by considering $R' = R\{u/x\}$. Suppose (b) and consider $R^* = R\{f(\tilde{x})/x\}$. By inductive hypothesis there exists R'' such

that $R^*(\rho, \tilde{M}\rho/\tilde{x}) \equiv R''\rho$. From $R(\rho, N/x) \equiv R^*(\rho, \tilde{M}\rho/\tilde{x})$ and transitivity of $\equiv$ we obtain $R(\rho, N/x) \equiv R''\rho$, as desired. We have all we need to conclude. Let R' be a process closed by ρ such that $R(\rho, N/x) \equiv R'\rho$. By contextuality of $\cong^{A\pi}$ we have $\rho \models P \mid R'\rho \cong^{A\pi} Q \mid R'\rho$ and in turn $\rho, N/x \models P \mid R'\rho \mathscr{R} Q \mid R'\rho$. Thus $\rho, N/x \models P \mid R(\rho, N/x) \equiv \mathscr{R} \equiv Q \mid R(\rho, N/x)$, as desired.

□

The distinguishing power of the environment does not increase whenever we remove entries from the base representing its knowledge. The following proposition formalize this intuition.

Proposition 41 (Erasure). *If* $\rho, M/x \models P \cong^{A\pi} Q$ *then* $\rho \models P \cong^{A\pi} Q$.

Proof. Let $\mathscr{R}$ be the relation such that $\rho \models P\mathscr{R}Q$ whenever (i) $\rho, M/x \models P \cong^{A\pi} Q$ or (ii) $\rho \models P \cong^{A\pi} Q$; we show that $\mathscr{R}$ is a behavioural equivalence up to $\equiv$. We show case (i), (ii) is trivial.

To see that $\mathscr{R}$ preserve barbs, suppose $\rho \models P\downarrow_a$. Of course this implies $\rho, M/x \models P\downarrow_a$ and by hypothesis we obtain $\rho \models Q \Downarrow_a$. Since $a \in A(\rho)$ we deduce $\rho \models Q \Downarrow_a$, as desired.

Reduction closure is straightforward. To see that $\mathscr{R}$ is contextual, let $\rho \models P\mathscr{R}Q$. We first show that $\rho, a/y \models P\mathscr{R}Q$ whenever $a \notin fn(\rho)$. Let $a \notin fn(\rho)$. There are two cases corresponding to $a \in fn(M)$ or not. If not we easily obtain $\rho, a/y, M/x \models P \cong^{A\pi} Q$ and in turn $\rho, a/y \models P\mathscr{R}Q$. Suppose $a \in fn(M)$. Since $\rho \models P\mathscr{R}Q$ is a term-indexed relation we know that $a \notin fn(P, Q)$, that is a is fresh to ρ, P and Q. We therefore alpha-convert the fresh name a in M with a' and we proceed as above . To see that $\rho \setminus a \models (\mathsf{new}\ a)P\mathscr{R}(\mathsf{new}\ a)Q$ we consider cases (a) $a \in fn(M)$ and (b) $a \notin fn(M)$. In case (a) $\rho, M/x \setminus a = \rho \setminus a$ and from $(\rho, M/x) \setminus a \models (\mathsf{new}\ a)P \cong^{A\pi} (\mathsf{new}\ a)Q$ and (ii) we obtain $\rho \setminus a \models (\mathsf{new}\ a)P\mathscr{R}(\mathsf{new}\ a)Q$. In case (b) from $(\rho, M/x) \setminus a = \rho \setminus a, M/x$ and $(\rho, M/x) \setminus a \models (\mathsf{new}\ a)P \cong^{A\pi} (\mathsf{new}\ a)Q$ we obtain $\rho \setminus a \models (\mathsf{new}\ a)P\mathscr{R}(\mathsf{new}\ a)Q$. Finally let $\rho \vdash R$. Since $x \notin fv(R)$ we trivially obtain $\rho, M/x \vdash R$ and $R\rho \equiv R(\rho, M/x)$. By hypothesis we have $\rho, M/x \models P \mid R(\rho, M/x) \cong^{A\pi} Q \mid R(\rho, M/x)$ and in turn $\rho \models P \mid R\rho \equiv \mathscr{R} \equiv Q \mid R\rho$, as desired.

□

3.3 Related work

The spi calculus [12] is a variant of the pi calculus where terms may be built around cryptographic functions and told apart by a decryption process of the form $\mathsf{decrypt}\ M\ \mathsf{as}\ \{x\}_N\ \mathsf{in}\ P$. The dynamics of the process are those of the process $P\{M^*/x\}$ whenever $M = \{M^*\}_N$; otherwise the process is stuck. The spi calculus introduced the idea to interpret low-level contexts as attackers that try to leak secrets by observing the communications of some protocol; invalidate the desired

behavioral equivalences (possibly among two protocol executions) correspond to leak a secret from the execution of the protocol. Typically the protocol's exchanges occur in open channels protected by cryptographic keys; the attacker may therefore intercepted messaged and forge new ones by interposing in all communications. In such sense we may say that the attacker models a Dolev-Yao intruder [47]. The leaking of secrets is particularly critical when the protocol implements high level abstractions; in such case telling apart the implementation of high level equivalent processes may affects the security of the implementation [1]. In this thesis, we follow this approach and devise implementations of typed channel abstractions that do not leak secrets when immersed in arbitrary, possibly hostile applied pi contexts; we show how this may be accomplished in the next chapter.

Behavioral equivalences for pi calculi with low level functions have been recently studied extensively, e.g. [7, 22, 12, 11, 24]; in [25] several notions of observational equivalence for the spi calculus are compared. Besides the relations with the works on the applied pi calculus [7, 19, 20, 4, 17] that we have already analyzed, the behavioral theory we devised for our variant of the applied pi calculus have several analogies with [22].

As in Boreale, De Nicola and Pugliese's work, we represent the knowledge of the environment by means of a substitution from names to terms. We also inherit from them the notion of term-indexed relation and behavioral equivalence; differently from [22], our behavioral equivalence consider a single index. The need of consider different indexes holds for reasons similar to [23], namely bisimilar processes may not exhibit the same labels (see Section 2.6). More in detail, in [22] behavioral equivalence for the spi calculus is indexed by a couple of equivalent substitutions, i.e. substitutions with the same domain that satisfy the same evaluation and matching of terms. The paper provides a coinductive characterization of behavioral equivalence based on a labelled transition system considering configuration's transitions of the form $\sigma \triangleright P \triangleright \xrightarrow[\delta]{\alpha} \sigma' \triangleright P'$ where σ is a substitution from names to terms. As in [23], α is a process actions while δ represents the complementary environment action; bisimilar processes exhibit the same environment actions. Indeed, as the following example shows, insisting that bisimilar processes have the same process actions is too restrictive in the presence of cryptography. Consider processes $P = (\mathsf{new}\ k)p\langle\{M\}_k\rangle$ and $Q = (\mathsf{new}\ k)p\langle\{N\}_k\rangle$ where p is a public channel. Clearly the environment cannot tell apart the message $\{M\}_k$ from $\{N\}_k$ as it does not have access to the key k: indeed it can be proved that $(\rho, \rho) \models P \cong^{Spi} Q$. On the other hand $\rho \triangleright P$ and $\rho \triangleright Q$ exhibit different process labels that lead to different indexes: $\rho \triangleright P \xrightarrow[\delta]{(k)a\langle\{M\}_k\rangle} \rho, \{M\}_k/x \triangleright \mathbf{0}$ and $\rho \triangleright Q \xrightarrow[\delta]{(k)a\langle\{N\}_k\rangle} \rho, \{N\}_k/x \triangleright \mathbf{0}$. Finally to establishing the bisimilarity of $\rho \triangleright P$ and $\rho \triangleright Q$ one needs to check that the substitutions $\rho, \{M\}_k/x$ and $\rho, \{N\}_k/x$ are equivalent. To avoid the quantification over all possible evaluations of terms arising in the definition of equivalent substitutions, an alternative definition of equivalence is given; the new definition can be easily

checked once one has calculated the names in the analysis of each substitution.

4

Secure implementations

In this chapter we devise secure implementations of API@ calculus processes into applied pi calculus processes. We first review the challenges hidden in the implementation of high-level processes into low-level, open environments (see [1]). Then we present the framework of the implementation and we introduce the low-level data structures that represent the names and the capability types of the high level calculus. We give a first implementation based on channel server processes that serve write and read requests for synchronization; high level processes represents the clients running corresponding protocols with the servers. Relying on the operational correspondence of the encoding, we prove that the translation is sound, i.e. if the translations of P and Q are observationally equivalent in the applied pi calculus then P and Q are behaviorally equivalent in the API@ calculus. The opposite direction, instead, is harder to achieve as one needs to build safeguards against attacks that rely on channel servers created by the context that hide malicious code. We enhance the design by setting up the synchronization protocols so as to ensure that all the exchanges occur over system generated (hence secure) channel servers. We finally prove that the resulting implementation is fully abstract (sound and complete) by showing that the untyped equivalences of the low-level cryptographic calculus coincide, via the translation, with the typed equivalences of the high-level calculus.

Remark In this chapter we omit most proofs that are carefully deployed in Chapter 5. Particularly, we establish the soundness of our implementations by relying on a Theorem of operational correspondence (see [21]) stated in terms of weak actions. The proof of this theorem is subtle, because our translations are not "prompt" [84], i.e. a translated process need to execute several steps to be ready for the commit step that matches the high level synchronization. The proof relies on the definition of an observational equivalence smaller than behavioral equivalence, and on a strong version of operational correspondence defined over this smaller equivalence. We omit all details and refer the reader to Chapter 5 for a rigorous construction of the proof. For the sake of readability, we present implementations of the monadic version of the calculus defined in Chapter 2, and we sketch how these translations may be extended to the full polyadic calculus.

4.1 Obstacles to a fully abstract implementation

In the pi calculus communications are protected trough high level abstractions such private channels or sortings that rule out interfering contexts. To illustrate, consider the following untyped example:

$$(\mathsf{new}\ a)a\langle b\rangle \mid a(x).P \qquad a \notin fn(P)$$

In this process a private channel is created to ensure that the name b is sended to P without possible interferences. Indeed we have the equation [71]:

$$(\mathsf{new}\ a)a\langle b\rangle \mid a(x).P \cong^{pi} P\{b/x\} \tag{4.1}$$

The relation $\cong^{pi}$ is the untyped version of behavioral equivalence (hence is contextual). The threads $a\langle b\rangle$, $a(x).P$ can be processes that physically stay in different machines that are connected trough an open communication medium as the Internet. The challenge is to implement such communications in open networks while preserving equations like (4.1). Experience in the developing of network protocols suggest that the way to protect open communications is cryptography (see [68]). In this vision the objective of safely send b to process P is gained by sending it on a public channel encrypted under a key generated apart a:

$$[\![\, a\langle b\rangle \,]\!] = net\langle \{[\![\, b \,]\!]\}_{a^+}\rangle \tag{4.2}$$

The intuition is that no context can open the packet $\{[\![\, b \,]\!]\}_{a^+}$ since the only process knowing the related decryption key is the intended receiver:

$$[\![\, a(x).P \,]\!] = net(y).\mathsf{decrypt}\ y\ \mathsf{as}\ \{x\}_{a^-}\ \mathsf{in}\ [\![\, P \,]\!] \tag{4.3}$$

Each channel is thus represented through a couple of communication keys $[\![\, b \,]\!] = b^+, b^-$ representing respectively the capability of write and read b.
While this representation is appealing in its simplicity, it suffers from a number of shortcomings, first made explicit by Abadi in [1]. Most of these are recoverable by strengthening the implementation, while for the lack of forward secrecy only workarounds are actually available for calculi that support the exchange of write *and/or* read access rights among processes. We outline below the details.

The first attack mounted by the implementation above is traffic analysis. Consider the equation (4.1) for the particular $P = \mathbf{0}$; this result states that the communication of b should be invisible in the implementation. This clearly does not hold as reductions on the network are visible. The context $C[-] = net(x).e\langle\rangle$ breaks the desired equivalence:

$$C[(\mathsf{new}\ a)[\![\, a\langle b\rangle \mid a(x) \,]\!]\,]\downarrow_e \quad \wedge \quad C[\mathbf{0}]\not\downarrow_e$$

A solution could be to inject noise in the communication interface; we let $\cong^{api}$ be the non-indexed version of our behavioral equivalence for the applied pi calculus.

$$!net\langle\{n\}_n\rangle \mid (\mathsf{new}\ a)[\![\, a\langle b\rangle \mid a(x)\,]\!] \cong^{api} !net\langle\{n\}_n\rangle \mid \mathbf{0}$$

Needless to say, this solution is hardly realistic and does not take in account that a determinate amount of traffic should be always be observable on the network. The use of noise could be avoided by considering semantics equations that hold with a certain degree of probability, e.g. [63, 39, 96, 77]; on the other hand, reasoning on the properties of a translation in this setting appears particularly difficult.

Denial of service attacks can prevent P to receive b. In the implementation above such attacks can be carried on by contexts (i) intercepting the message containing b or (ii) making the receiving process stuck. As an example of (i), consider the equation (4.1) for the particular $P = x\langle\rangle$. We choose a distinguishing context that emits a signal whenever a decryption with the key b^- is successful: $C[-] \triangleq -\mid !\, net(x).\mathsf{decrypt}\ x\ \mathsf{as}\ \{\}_{b^-}\ \mathsf{in}\ e\langle\rangle$. By definitions (4.2),(4.3) we have:

$$\begin{array}{rcl} C[[\![\, P\,]\!]] & \xrightarrow{\ \tau\ } & (\mathsf{new}\ a)(C[\mathsf{decrypt}\ \{[\![\, b\,]\!]\}_{a^+}\ \mathsf{as}\ \{\}_{b^-}\ \mathsf{in}\ e\langle\rangle \mid [\![\, a(x).x\langle\rangle\,]\!]]) \\ & \equiv & C[(\mathsf{new}\ a)(\mathbf{0} \mid [\![\, a(x).x\langle\rangle\,]\!])] \end{array}$$

Process $C[[\![\, b\langle\rangle\,]\!]]$ has two possible matching answers:

$$\begin{array}{lll} (i) & C[[\![\, b\langle\rangle\,]\!]] \xrightarrow{\ \tau\ } C[\mathsf{decrypt}\ \{\}_{b^+}\ \mathsf{as}\ \{\}_{b^-}\ \mathsf{in}\ e\langle\rangle] \equiv C[e\langle\rangle] \\ (ii) & C[[\![\, b\langle\rangle\,]\!]] \Longrightarrow C[[\![\, b\langle\rangle\,]\!]] \end{array}$$

Neither the couple $(C[(\mathsf{new}\ a)\mathbf{0} \mid [\![\, a(x).x\langle\rangle\,]\!], C[e\langle\rangle])$ and the couple $(C[(\mathsf{new}\ a)\mathbf{0} \mid [\![\, a(x).x\langle\rangle\,]\!]],\ C[[\![\, b\langle\rangle\,]\!]])$ can be in behavioural equivalence as $C[(\mathsf{new}\ a)\mathbf{0} \mid [\![\, a(x).x\langle\rangle\,]\!]] \not\Downarrow_e$.
To enforce our protocol we add replication to emission and recursion to reception; also, we instrument the receiving process with a mechanism to throws out messages wrongly picked up: this prevents the context to notice the presence of a receiver by checking if it swallows messages [8].

$$\begin{array}{rcl} [\![\, a\langle b\rangle\,]\!] & = & !\, net\langle\{[\![\, b\,]\!]\}_{a^+}\rangle \\ [\![\, a(x).P\,]\!] & = & \mathsf{rec}\, X.net(y).\mathsf{decrypt}\ y\ \mathsf{as}\ \{x\}_{a^-}\ \mathsf{in}\ [\![\, P\,]\!]\ \mathsf{else}\ net\langle y\rangle \mid X \end{array}$$

The replication of emissions clearly exposes to replay attacks. Standard countermeasures apply as using challenge-response, inserting identifiers or timestamps in the emitted messages. The implementation we present below packages each emission with a fresh nonce; receivers on a channel a share a private channel a° to store nonces and discard replication of accepted messages.

$$\begin{array}{rcl} [\![\, a\langle b\rangle \,]\!] & = & (\mathsf{new}\ n)!\, net\langle \{[\![\, b \,]\!], n\}_{a^+}\rangle \\ [\![\, a(x).P \,]\!] & = & \mathsf{rec}\, X.net(y).\mathsf{decrypt}\ y\ \mathsf{as}\ \{x, z\}_{a^-}\ \mathsf{in} \\ & & \quad (\mathsf{if}\ z \notin Set_{a^\circ}\ \mathsf{then}\ [\![\, P \,]\!]\ \mathsf{else}\ net\langle y\rangle \mid X) \\ & & \mathsf{else}\ net\langle y\rangle \mid X \end{array}$$

Saving nonces of all exchanges is unlikely proposable; [8] formalizes the intuition that this mechanism is equivalent to use a session-based protocol.

One advantage of using nonces is that they can represent confounders to prevent cryptanalysis attacks. To illustrate, consider processes $P = (\mathsf{new}\ a, b, c)a\langle b\rangle \mid a\langle c\rangle$, $Q = (\mathsf{new}\ a, b)a\langle b\rangle \mid a\langle b\rangle$. We have $P \cong^{pi} Q$; this can be easily seen by noting that both sides are weakly equivalent to $\mathbf{0}$. On contrast, an applied pi context $C[-] = net(x).net(y).\mathsf{if}\ x = y\ \mathsf{then}\ e\langle\rangle$ can distinguish among implementations (4.2) of P and Q by comparing bit a bit the two emissions:

$$C[net\langle \{[\![\, b \,]\!]\}_{a^+}\rangle \mid net\langle \{[\![\, b \,]\!]\}_{a^+}\rangle] \Downarrow_e$$
$$C[net\langle \{[\![\, b \,]\!]\}_{a^+}\rangle \mid net\langle \{[\![\, c \,]\!]\}_{a^+}\rangle] \not\Downarrow_e$$

The harder obstacle to a fully abstract implementation of the pi calculus is the lack of forward secrecy. Consider the process $P = (\mathsf{new}\ a)a\langle b\rangle \mid a(x).p\langle a\rangle$. Similarly to the introductive example, a channel a is created to safely send b to the receiver. The receiver discards b and releases the communication channel a to the environment through a public port p. The communication of b is secret, in the sense that no pi context can retrieve the content of the exchange happened on the channel a. Formally,we have that any process $Q = (\mathsf{new}\ a)a\langle b'\rangle \mid a(x).p\langle a\rangle$ sending some name b' is equivalent to P: $P \cong^{pi} Q$. As in previous cases, this equation is obtained by noting that contexts cannot interact with the port a and in turn both P and Q are behavioural equivalent to $(\mathsf{new}\ a)p\langle a\rangle$. Conversely, an applied pi context can buffer the emission of b and subsequently, once received the read capability on a, may decrypt the packet and break the secrecy of the communication on the high level channel a. To this aim, consider the context $C[-] = net(y).net(k).\mathsf{decrypt}\ y\ \mathsf{as}\ \{x\}_k\ \mathsf{in}\ \mathsf{if}\ x = b\ \mathsf{then}\ e\langle\rangle$. This context tells apart P and Q:

$$C[[\![\, P \,]\!]] \Downarrow_e \ \wedge\ C[[\![\, Q \,]\!]] \not\Downarrow_e$$

To the best of our knowledge, giving a direct implementation of the distribution of read capabilities in pi calculi is still an open problem due to this attack (see the remark below). The solution we propose in this thesis is based on the representation of a channel as a process that serves input and output requests, so that each exchange of messages is the result of two separate protocols with writer and reader clients.

All channels are associated with two separate key-pairs. The decryption keys are always stored securely at the channel, and never leaked; the encryption keys, in turn, are available to the clients that have read and/or write access to the channel. In the write protocol, the client sends data, and the channel buffers it on private queue; in the read protocol, the client sends a session key and the server returns data encrypted with that key. Publishing a read/write capability on a channel corresponds to publishing the read/write encryption key associated with the channel.

Under appropriate, mostly standard, hypothesis on the properties of the underlying network, we show that a translation based on these ideas is sound. Full abstraction, instead, is harder to achieve as we need to build safeguards against attacks that exploit malformed data or malicious channels that intentionally leak their associated decryption keys. To account for that, we complement the translation with a proxy-service mechanism to ensure that all communication protocols take place via system generated (hence secure) channels. We establish the adequacy and full abstraction of the resulting implementation by contrasting the untyped equivalences of the low-level cryptographic calculus, with the typed equivalences of the high-level source calculus.

Remark In [8] Abadi, Fournet and Gonthier provide a fully abstract implementation of join calculus in a lower calculus equipped with cryptographic primitives. In [8] the problem of releasing read access rights is avoided altogether, as the join calculus does not allow names to be communicated with read capabilities, a feature that instead constitutes one of the fundamental ingredients of our typed calculus. Hence, to recover forward secrecy, we need a more structured representation of type capabilities to make sure that distributing a read capability does not correspond to leaking any decryption key. The existence of a fully abstract encoding of the pi calculus without matching into join [50] provides an indirect fully abstract encoding of pi calculus. However, the solution used in [50] to implement the pi calculus does not extend in the presence of matching, that in contrast is one of the fundamental ingredients of our high level calculus. We will return on this point in Section 4.5.

4.2 The implementation framework

Our implementation is based on a compilation of the high-level processes of the pi calculus into corresponding processes of the spi calculus representing low-level principals running in an open network.

The model of the network

Our assumptions about the low-level communication model are the same as those of [8]. In particular, we presuppose a Needham-Schroeder [83] network model, in which an intruder can interpose a computer in all communication paths and thus alter or

copy parts of messages, replay messages or forge new ones. We also assume that each principal has a secure environment in which to compute and store private data. On the other hand, we assume that the intruder cannot gain control of the whole network, and thus we do not guarantee that it actually will intercept every message. Consequently, message delivery may always be achieved with an adequate degree of redundancy. All processes can send and receive messages through a network interface consisting of a channel *net*. Typically all the exchanges over this channel net are encrypted. As [8] our results about the implementation rely on the presence of noise to prevent traffic analysis.

Data Structures for Names

The design of the implementation is centered around the choice of an appropriate data structure to represent the names and the capability types of the high-level calculus. As we mentioned at the outset, the idea is to represent the capability types with corresponding term capabilities, implemented as encryption keys. Specifically, the representation of the name n as a fully fledged channel includes the name identity and two encryption keys corresponding to the read and write capabilities: $(pub(n), ek(wr(n)), ek(rd(n)))$. The application of the one-way constructors rd and wr provide a way to construct the read and write seeds associated with the key-pairs that implement the capability types associated with n.

The representation of a name is now obtained by introducing self-signed certificates [19, 4] of the form:

$$Cert(n) \triangleq \{pub(n), hash(ek(wr(n))), hash(ek(rd(n)))\}_{priv(n)}$$

The certificates help determine the "type" of the names they are attached to. To illustrate, suppose we receive a tuple formed as $(M_0, M_1, M_2, Cert(n))$, with M_i arbitrary terms. One may first ensure that the certificate corresponds to the correct identity by using the public key $pub(n)$ to verify the certificate, and then match M_0 with the public key. As this stage, one may decide whether or not M_1 and M_2 are valid capabilities for the identity M_0 by calculating $hash(M_1)$ and $hash(M_2)$ and by checking whether they match the two values $hash(ek(wr(n)))$ and $hash(ek(rd(n)))$ respectively: in the former case we certify a write capability, in the latter a read capability.

We henceforth let $\underline{n}$ denote the representation of n together with the associated certificate:

$$\underline{n} \;=\; (pub(n), ek(wr(n)), ek(rd(n)), Cert(n)).$$

All the low-level protocols that implement the synchronization steps in the high-level calculus exchange and manipulate quadruples and expect the components of such tuples to adhere to the format above. For uniformity, we write $\underline{x}$ to note a

quadruple of variables used to store name representations. Further, we introduce the following mnemonic notation for the components of 4-tuple terms:

$$M_{ID} = \pi_1(M) \quad M_w^+ = \pi_2(M)$$
$$M_r^+ = \pi_3(M) \quad M_{CERT} = \pi_4(M)$$

Notice that we make no assumption on the format of the 4-tuple: hence, only when $M = \underline{n}$ for some name n, the projections select the identity, the keys and the certificate associated with n. Abusing the notation, we often write $\underline{n}/\underline{x}$ (and more generally $M/\underline{x}$, when M is a 4-tuple) to indicate the substitution of the components of $\underline{n}$ (or M) for the corresponding variables composing $\underline{x}$.

In the translation, we will find it useful to compute, and manipulate the run-time type of terms. We give a low-level representation of the high-level types by simply treating the latter as nullary constructors: thus the terms $\mathsf{rw}, \mathsf{r}, \mathsf{w}, \top$ in the implementation represent the corresponding high-level types $\mathsf{ch}(\mathsf{rw}), \mathsf{ch}(\mathsf{r}), \mathsf{ch}(\mathsf{w}), \top$. Following the convention adopted for names, we write $\underline{T}$ for the representation of the type T in the low-level calculus. Also, we sometime write $\underline{S} <: \underline{T}$ when $S <: T$.

Given this representation of names, the effect of casting a name at a higher type in the high-level calculus is realized, in the implementation, with a mechanism to mask away the encryption keys corresponding to the type capabilities "lost" in the type cast. This is accomplished with the binary destructor $\Uparrow(M, \underline{T})$ corresponding to the up-cast of the source calculus (see Table 3.1). We use infix notation in the rest of the presentation, writing $M \Uparrow \underline{T}$ instead of $\Uparrow(M, \underline{T},)$. We often use the short $P\{(M \Uparrow \underline{T})/\tilde{x}\}$ for the process $\mathsf{let}\ \tilde{x} = \Uparrow(M, \underline{T})\ \mathsf{in}\ P$.

Based on these conventions, we introduce useful notation to express various operations to manipulate the representation of names. We let $\mathsf{let}\ x = \mathsf{Typeof}(M)\ \mathsf{in}\ P\ \mathsf{else}\ Q$ note the process that verifies that M represents capabilities certified by M_{CERT}, determines the true type of M, and binds that type to x. In such case we informally say that M has such type. If M is not certified the process continues as Q; we will often write $\mathsf{let}\ x = \mathsf{Typeof}(M)\ \mathsf{in}\ P$ to indicate the process $\mathsf{let}\ x = \mathsf{Typeof}(M)\ \mathsf{in}\ P\ \mathsf{else}\ \mathbf{0}$. We let $\mathsf{if}\ WF(M, \underline{T})\ \mathsf{then}\ P\ \mathsf{else}\ Q$ be the process that tests whether M is certified and has type $\underline{S}$, for some $\underline{S} <: \underline{T}$. Finally, the process $\mathsf{let}\ x = \mathsf{Meet}(M_1, \ldots, M_n)\ \mathsf{in}\ P$ computes a new tuple representing the result of merging the capabilities of the M_i's and binds such tuple to x: the result of the meet is defined only if $M^i_{CERT} = M^j_{CERT}$ for all $i, j \in [1..n]$.

All these operations are encoded with little effort by nested applications of projection, equality, *verify* and cast destructors.

$$\begin{aligned}
&\mathsf{let}\ x = \mathsf{Typeof}(M)\ \mathsf{in}\ P\ \mathsf{else}\ Q \triangleq \mathsf{let}\ z = \mathit{verify}(M_{CERT}, M_{ID})\ \mathsf{in}\\
&\quad(\mathsf{if}\ M_{ID} = z_{ID}\ \mathsf{then}\ (\mathsf{if}\ \mathit{hash}(M_w^+) = z_w^+\ \mathsf{then}\\
&\qquad(\mathsf{if}\ \mathit{hash}(M_r^+) = z_r^+\ \mathsf{then}\ (\mathsf{let}\ x = \mathsf{rw}\ \mathsf{in}\ P)\ \mathsf{else}\ \mathsf{let}\ x = \mathsf{w}\ \mathsf{in}\ P))\\
&\quad\mathsf{else}\ (\mathsf{if}\ \mathit{hash}(M_r^+) = z_r^+\ \mathsf{then}\ (\mathsf{let}\ x = \mathsf{r}\ \mathsf{in}\ P)\ \mathsf{else}\ \mathsf{let}\ x = \top\ \mathsf{in}\ P))\\
&\quad\mathsf{else}\ Q
\end{aligned}$$

$$\mathsf{let}\ x = \mathsf{Meet}(M_1, \ldots, M_n)\ \mathsf{in}\ P \triangleq \mathsf{let}\ z = \mathsf{Merge}((M_1, M_2))\ \mathsf{in}$$
$$\mathsf{let}\ y = \mathsf{Merge}((z, M_3))\ \mathsf{in}\ \ldots\ \mathsf{let}\ x = \mathsf{Merge}((w, M_n))\ \mathsf{in}\ P$$

$$\mathsf{let}\ x = \mathsf{Merge}((M, N))\ \mathsf{in}\ P \triangleq \mathsf{if}\ M_r^+ = \top\ \mathsf{then}$$
$$(\mathsf{if}\ M_w^+ = \top\ \mathsf{then}\ (\mathsf{let}\ x = N\ \mathsf{in}\ P)\ \mathsf{else}\ \mathsf{let}\ x = (N_{ID}, N_r^+, M_w^+, N_{CERT})\ \mathsf{in}\ P$$
$$\mathsf{else}\ (\mathsf{if}\ M_r^+ = \top\ \mathsf{then}\ \mathsf{let}\ x = (N_{ID}, M_r^+, N_w^+, N_{CERT})\ \mathsf{in}\ P\ \mathsf{else}\ \mathsf{let}\ x = N\ \mathsf{in}\ P$$

$$\mathsf{if}\ WF(M, \underline{T})\ \mathsf{then}\ P\ \mathsf{else}\ Q \triangleq \mathsf{let}\ x = \mathsf{Typeof}(M)\ \mathsf{in}\ (\mathsf{if}\ SUB(x, \underline{T})\ \mathsf{then}\ P\ \mathsf{else}\ Q)\ \mathsf{else}\ Q$$
$$\mathsf{if}\ SUB(M, N)\ \mathsf{then}\ P\ \mathsf{else}\ Q \triangleq \mathsf{if}\ (M = \mathsf{rw}) \vee (N = \top) \vee (M = N)\ \mathsf{then}\ P\ \mathsf{else}\ Q$$

The process $\mathsf{if}\ (M_1 = N_1) \vee \cdots \vee (M_n = N_n)\ \mathsf{then}\ P\ \mathsf{else}\ Q$ can be coded by using nested conditionals: we omit all the details.

For future reference, we note instead that the implementation is structured so that such processes are activated as part of the communication protocols only in the presence of precise invariants that guarantee that their (the processes') behavior is well-defined. More precisely, the communication protocols guarantee the following conditions: (i) whenever we activate the process $\mathsf{let}\ x = \Uparrow(M, T)\ \mathsf{in}\ P$, M_{CERT} is indeed a certificate that certifies the remaining components of M, and the type of M is $S <: T$, (ii) whenever we evaluate $\mathsf{let}\ x = \mathsf{Meet}(M^1, \ldots, M^n)\ \mathsf{in}\ P$, M^i_{CERT} and M^j_{CERT} are the same certificate which certifies the remaining components of M^i and M^j at their respective types for all $i, j \in [1..n]$.

A further important remark is in order on the nature of the certificates and the guarantees they convey. On the one hand, as we argued, the certificates help extract the capabilities from the representation of names. On the other hand, being self-signed a certificate does not by itself ensure that the capabilities extracted from a term $(M_{ID}, M_w^+, M_w^+, M_{CERT})$ are valid encryption keys. That is indeed the case for names generated by our implementation, but a term with this structure received from an arbitrary context may well be validated at type T without being the representation of any name at any type. To illustrate, the (fake) certificate $C = \{pub(n), hash(M), hash(N)\}_{priv(n)}$ will certify a write capability in $(pub(n), M, \top, C)$ regardless of the structure of M, which can be any term.

Fortunately, one can build more structure in the implementation so as to protect against such threats and give guarantees that any term used as a key is indeed a valid key. We show how that can be accomplished in Section 4.4.

Message Filtering

All the communication protocols underlying our implementation run on the public channel *net*. Consequently, as in [8], our implementation relies on the ability of

processes to filter replays of messages, based on nonces. This is accomplished by using the process $\mathsf{if}\ M \notin Set_n\ \mathsf{then}\ P$ defined in Chapter 3.

The input of a message relies on two filtering protocols. The first picks a packet from the *net* and proceeds with the continuation only if the message is successfully decrypted; otherwise it re-emits the message and retries:

$$\begin{array}{l}\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ N\ \mathsf{in}\ P\ \triangleq \\ \quad \mathsf{rec}\, X.net(x).\mathsf{decrypt}\ x\ \mathsf{as}\ \{\tilde{y}\}_N\ \mathsf{in}\ P\ \mathsf{else}\ (net\langle x\rangle \mid X)\end{array}$$

The second protocol filters messages based on their types, and up-casts them at those types.

$$\begin{array}{l}\mathsf{filter}\ \underline{x}@t\ \mathsf{from}\ c\ \mathsf{in}\ P\ \triangleq \\ \quad \mathsf{rec}\, X.c(\underline{y}).\mathsf{if}\ \ WF(\underline{y},t)\ \mathsf{then}\ (\mathsf{let}\ \underline{x} = \Uparrow(\underline{y},t)\ \mathsf{in}\ P)\ \mathsf{else}\ (c\langle\underline{y}\rangle \mid X)\end{array}$$

The output of a message M on a network is realized by emitting M on the *net* channel, typically encrypted. To ensure delivery, the emission is replicated, and packaged with a fresh nonce, n, to protect against replay attacks. The nonce also acts as a confounder for cryptanalysis attacks.

$$\mathsf{emit}(\{M\}_N)\ \triangleq\ (\mathsf{new}\ n)!\, net\langle\{M,n\}_N\rangle$$

4.3 A sound implementation

We are ready to give a formal definition of the two complementary components of the implementation: the channel infrastructure for communication, and the compilation of high-level processes. As anticipated, the idea is to set up appropriate channel servers to support write and read requests for synchronization and value exchange; the high-level processes of the pi calculus, in turn, are represented as clients running corresponding protocols with the servers.

Channels and communication protocols

Each channel server is associated with two pairs of asymmetric keys employed in the protocols for reading and for writing, respectively. The encryption keys are circulated among clients as components of the tuples that represent the high-level names; the decryption keys, in turn, are stored securely at the channel servers.

Write protocol. On the client side, writing on a channel is accomplished by emitting a packet encrypted under the channel's write encryption key. The message is replicated to ensure delivery, and packaged with a nonce to protect against replay attacks. The server, in turn, uses the write decryption key to receive the message, uses the nonce to filter multiple copies of the message (hardly realistic, of course: indeed, the solution from [8], based on a challenge response mechanisms, would work

just as well here), and stores them into a private queue. It also filters based on the format of the messages received, requiring that they match the format expected of the encoding of names.

Read Protocol. On the client side, the reader process uses the channel's read encryption key to send a request to the server. The request takes the form of pair including a symmetric encryption key that will be used to exchange the message with the server, and (the representation of) a type at which the client expects its input value. The client then waits for a message from the server encrypted under the session key: upon receiving the packet, it proceeds with its continuation. The server, in turn, uses the channel's read decryption key to receive a request from the client. To protect against replays, the server keeps track of the nonces received on a private channel. After checking the freshness of the request, the server uses the type to select one of the messages from its private queue and then encrypts the message with the key. The nonce can be spared on the actual message sent by the server as the key expires at the completion of the protocol (the server may easily filter out replays of the session key).

The definition of the channel servers is reported in Table 4.1. We use the notation n_r^- and n_w^- to refer to $dk(rd(n))$ and $dk(wr(n))$ respectively. The two private names n° and n^* hold, respectively, the message queue and the nonce set associated with n. The definitions, as given, are adequate for monadic exchanges only, but the extension to the polyadic case is relatively straightforward, as one may rely on corresponding polyadic exchanges in the implementation language. Indeed, we could simply associate an arity with each channel, and parameterize definition of the filters (on the server and client sides) on that arity.

Compilation of high-level processes

Having defined the underlying framework, we now illustrate the compilation function.

The translation is given by induction on the typing judgments of the pi calculus, and is only defined for derivable judgments. We remark, however, that the output of the translation only depends on the structure of the processes, as the latter contain enough information to guide the generation of the low-level code. On the other hand, the typing information is useful in stating and proving the properties of the translation, whenever we need to draw precise connections between the static capability types of the high-level processes and the dynamic term capabilities of the low-level principals.

The compilation function takes an extra argument, noted μ, that collects information on the equalities corresponding to the matching prefixes encountered in the traversal of the process being compiled. This information is expressed as a set of bindings $u \leftrightarrow v$, with u and v syntactically different (if u and v are syntactically identical,

Table 4.1 Channels and a compositional translation of client processes

Channels

$$
\begin{array}{rcl}
WS_n & \triangleq & !\,\mathsf{filter}\ (\underline{x}, z)\ \mathsf{with}\ n_w^-\ \mathsf{in}\ \mathsf{if}\ z \notin Set_{n^*}\ \mathsf{then}\ n^\circ\langle \underline{x}\rangle \\
RS_n & \triangleq & !\,\mathsf{filter}\ (y, t, z)\ \mathsf{with}\ n_r^-\ \mathsf{in} \\
 & & \quad \mathsf{if}\ z \notin Set_{n^*}\ \mathsf{then\ filter}\ \underline{x}@t\ \mathsf{from}\ n^\circ\ \mathsf{in}\ !\,net\langle\{\underline{x}\}_y\rangle \\
Chan_n & \triangleq & (\mathsf{new}\ n^*, n^\circ)\ n^*\langle\emptyset\rangle \mid RS_n \mid WS_n
\end{array}
$$

Clients

$$
\begin{array}{rcl}
[\![\,\Gamma \triangleright u\langle v@T\rangle\,]\!]_\mu & \triangleq & \mathsf{let}\ \hat{\underline{u}} = [\![\,u\,]\!]_\mu,\ \hat{\underline{v}} = [\![\,v\,]\!]_\mu\ \mathsf{in}\ \mathsf{emit}(\{\hat{\underline{v}} \Uparrow \underline{T}\}_{\hat{\underline{u}}_w^+}) \\
[\![\,\Gamma \triangleright u(x@T).P\,]\!]_\mu & \triangleq & (\mathsf{new}\ k)\,(\mathsf{let}\ \hat{\underline{u}} = [\![\,u\,]\!]_\mu\ \mathsf{in}\ \mathsf{emit}(\{sk(k), \underline{T}\}_{\hat{\underline{u}}_r^+})) \\
 & & \mid \mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ [\![\,\Gamma, x{:}T \triangleright P\,]\!]_\mu \\
[\![\,\Gamma \triangleright [u = v]\,P; Q\,]\!]_\mu & \triangleq & \mathsf{if}\ \underline{u}_{ID} = \underline{v}_{ID}\ \mathsf{then}\ [\![\,\Gamma \sqcap v{:}\Gamma(u) \sqcap u{:}\Gamma(v) \triangleright P\,]\!]_{\mu[u \leftrightarrow v]} \\
 & & \mathsf{else}\ [\![\,\Gamma \triangleright Q\,]\!]_\mu \\
[\![\,\Gamma \triangleright (\mathsf{new}\ n : A)P\,]\!]_\mu & \triangleq & (\mathsf{new}\ n)(Chan_n \mid [\![\,\Gamma, n : A \triangleright P\,]\!]_\mu) \\
[\![\,\Gamma \triangleright P \mid Q\,]\!]_\mu & \triangleright & [\![\,\Gamma \triangleright P\,]\!]_\mu \mid [\![\,\Gamma \triangleright Q\,]\!]_\mu \\
[\![\,\Gamma \triangleright !\,P\,]\!]_\mu & = & !\,[\![\,\Gamma \triangleright P\,]\!]_\mu \\
[\![\,\Gamma \triangleright \mathbf{0}\,]\!]_\mu & \triangleq & \mathbf{0}
\end{array}
$$

Top−level translation $[\![\,\Gamma \triangleright P\,]\!] \triangleq [\![\,\Gamma \triangleright P\,]\!]_\emptyset$

the binding convey no information). We note $\mu[u{\leftrightarrow}v]$ the extension of μ with a new binding. Also, we write $\mu \vdash u{\leftrightarrow}v$ to indicate that $u{\leftrightarrow}v$ is derivable from the equational theory generated by μ (notice that $\mu \vdash u{\leftrightarrow}v$ is decidable for finite $\mu's$).

The information conveyed by the argument μ is exploited by the compiler to generate code that reconstructs the run-time type of a name, gathering the encryption keys corresponding to the capability types available in the source calculus. More precisely, the term formed by the compiler-generated code represents the run-time counterpart of the meet taken on the types of all the names (or variables) that occur in an match enclosing the process being compiled. In Table 4.1, we express this calculation using the notation $\mathsf{let}\ \underline{x} = [\![\,u\,]\!]_\mu\ \mathsf{in}\ P$, which may be defined formally as the process $\mathsf{let}\ \underline{x} = \mathsf{Meet}\,\{\underline{w} \mid \mu \vdash u \leftrightarrow w\}\ \mathsf{in}\ P$ that computes the expected meet and binds it to the fresh variable $\underline{x}$. Notice that the set $\{\underline{w} \mid \mu \vdash u \leftrightarrow w\}$ is always non-empty (even for an empty μ) as $\mu \vdash u{\leftrightarrow}u$ may be by reflexivity, for all μ and u.

We are now ready to illustrate the clauses of the translation. An output in the source calculus is compiled into a corresponding emission: before that, however, the compiled code computes the appropriate representation for the term to be output

and collects the cryptographic key required to form the packet emitted on the public network. Likewise, an input process is compiled into code that runs the read protocol after having collected the appropriate key. A high-level match is compiled into the corresponding conditional low-level process: the name-equality test in the pi calculus is realized as a corresponding test on the public keys associated with the names involved. In addition, in case the test is successful, the continuation process is given access, via μ, to the new set of capabilities that corresponds to the meet of the two types associated with the names equated in the pi process. As for restriction, the translation of a high-level restriction generates a corresponding restriction together with a new channel to be associated with the newly generated name. The remaining clauses are defined homeomorphically. The definition of the translation function is completed by a clause that introduces the top-level compilation map $[\![\cdot]\!]$.

Properties of the implementation

The format of the communication servers and the structure of the client code resulting from the translation ensure the following properties: (i) each message output by a writer will reach at most one reader, and dually, (ii) a legitimate reader client will complete the protocol provided that a type-compatible message on the same channel has been output by a writer. Thus, if appropriate channels are allocated for the free names of the high-level processes, one can prove that any pi synchronization on a name is simulated by a corresponding reduction sequence in the implementation, and conversely that the synchronizations on the channel queues in the implementation reflect the τ-reductions of the source calculus.

Given the relative complexity of the implementation, the proof of operational correspondence is elaborate. As a first, basic step, one needs a proof that the run-time flow and management of the cryptographic keys representing the high-level type capabilities provides the low-level clients with enough capabilities to simulate all of the synchronizations of the well-typed processes. That can be proved by showing that the translation is closed by substitution, in the sense made precise below.

Given a type environment I, define the term environment corresponding to I, noted $\{\!|\, \mathrm{I} \,|\!\}$, as follows:

$$\{\!|\, \emptyset \,|\!\} = \{net/x_o\}, \quad \{\!|\, \mathrm{I}, a : A \,|\!\} = \{\!|\, \mathrm{I} \,|\!\}, \underline{a}{\Uparrow}\underline{A}/\underline{x}$$

where $\underline{x} \notin \mathrm{dom}(\{\!|\, \mathrm{I} \,|\!\})$. Now we can state the desired closure properties for the translation.

Lemma 42 (Substitution Closure). *Let Γ be a closed type environment such that $\Gamma \vdash v : A$. Then:*

$$[\![\, \Gamma, x{:}A \triangleright P \,]\!]\, \{(\underline{v}{\Uparrow}\underline{A})/\underline{x}\} \cong^{A\pi}_{\{\!|\Gamma|\!\}} [\![\, \Gamma \triangleright P\{v/x\} \,]\!]$$

To show the main result of operational correspondence, as in [8], we must rely on the presence of noise to prevent traffic analysis. Given the simple network interface we have assumed, injecting noise into the network is simply accomplished by the process

$$W \triangleq \,!\,(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle$$

which generates infinitely many copies of infinitely many secret packets. Now, we complete the definition of the computing environment by including a general interface to the network, the noise-generating process and channel support for the free names shared between the processes and the environment.

$$\mathsf{E}_{\mathrm{I}}[-] \;=\; -\mid W \mid \prod_{n\in \mathrm{dom(I)}} Chan_n$$

As is often the case, our soundness result relies on the operational correspondence of the translation (cf. [21], see also Section 5.1). The preservation direction of this result is standard. On the other hand, 'the 'reflection" direction is subtle, as the translation is not "prompt" [84]: in fact, it takes several steps for $\mathsf{E}_{\mathrm{I}}[\![\, P \,]\!]$ to be ready for the commit synchronization step on the channel queue that corresponds to the high-level synchronization on the channel. These preparation steps have the following properties: each reduction conducing to a commit (i) does not preclude any other reduction and (ii) is invisible to the context. As we will show in the next chapter, these reductions preserve behavioural equivalence and can be factored out in the proof of the operational correspondence by resorting to a suitable notion of *administrative* equivalence.

We have finally all we need to establish the result of operational correspondence.

Theorem 43 (Operational Correspondence). *Let* Γ *and* I *be two closed type environments such that* $\Gamma <: \mathrm{I}$. *Then:*

- *If* $P \Longrightarrow P'$ *then* $\mathsf{E}_{\mathrm{I}}[\![\, \Gamma \triangleright P \,]\!] \Longrightarrow \cong^{A\pi}_{\{\mathrm{I}\}} \mathsf{E}_{\mathrm{I}}[\![\, \Gamma \triangleright P' \,]\!]$.
- *Conversely, if* $\mathsf{E}_{\mathrm{I}}[\![\, \Gamma \triangleright P \,]\!] \Longrightarrow K$ *then there exists* P' *s.t.* $P \Longrightarrow P'$ *and* $K \cong^{A\pi}_{\{\mathrm{I}\}} \mathsf{E}_{\mathrm{I}}[\![\, \Gamma \triangleright P' \,]\!]$.

Given a type environment I, to prove preservation and reflection of barbs we define testing processes of the form

$$T^u \;= (\mathsf{new}\ k)\mathsf{emit}(\{sk(k), \top\}_{\underline{u}^+_r}) \,|\, \mathsf{filter}\ \tilde{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \omega\langle\rangle$$

where $\omega \notin fn(\{\!|\, \mathrm{I} \,|\!\})$. The idea is that whenever $\mathrm{I} \models P \Downarrow_a$, we know that $\mathrm{I}(a)^{\mathsf{r}} \downarrow$ and $P \xrightarrow{(\tilde{c})a\langle v@A\rangle}$. Thus both $\{\!|\, \mathrm{I} \,|\!\}$ contains the key a^+_{r} to read from channel a and there is a computation from $\mathsf{E}_{\mathrm{I}}[\![\, P \,]\!]_\Gamma]$ such that $Chan_a$ contains in its queue the message $\underline{v}\Uparrow\underline{A}$. Finally we use a fresh name ω to emit a signal whenever the read request to $Chan_a$ actually succeeds. To this aim we let $\{\!|\, \mathrm{I} \,|\!\}, \omega/z$ be such that

$\omega \notin fn(\{\!|\, \mathrm{I} \,|\!\})$ and $(\{\!|\, \mathrm{I} \,|\!\}, \omega/z)(y) = a_r^+$ and we consider the process T^y closed by $\{\!|\, \mathrm{I} \,|\!\}, \omega/z$: $T^y \triangleq (\mathsf{new}\ k)\mathsf{emit}(\{sk(k), \top\}_{y_r^+}) \,|\, \mathsf{filter}\ \tilde{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ z\langle\rangle$. We obtain $T^a \equiv T^y\{\!|\, \mathrm{I}, \omega/z \,|\!\}$.

The following barb predicate $\mathrm{I} \models P \Downarrow_{n_w^+} \triangleq (n_r^+ \in A(\{\!|\, \mathrm{I} \,|\!\}) \wedge \{\!|\, \mathrm{I} \,|\!\}, \omega/x \models (T^n \,|\, P)\Downarrow_\omega$ correspond to pi calculus barb $\mathrm{I} \models P\Downarrow_n$, in the following sense.

Proposition 44. *The following hold.*

1. $\mathrm{I} \models P\downarrow_a$ *implies* $\mathrm{I} \models \mathsf{E}_\mathrm{I}[\![\, \Gamma \triangleright P \,]\!]\Downarrow_{a_w^+}$
2. $\mathrm{I} \models \mathsf{E}_\mathrm{I}[\![\, \Gamma \triangleright P \,]\!]\Downarrow_{a_w^+}$ *implies* $\mathrm{I} \models P\Downarrow_a$.

We can now prove that the translation $[\![\, \cdot \,]\!]$ is sound, in the following sense.

Theorem 45. *Let* Γ, Δ *and* I *be closed type environments s.t.* $\Gamma, \Delta <: \mathrm{I}$. *If* $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{E}_\mathrm{I}[\![\, \Gamma \triangleright P \,]\!] \cong^{A\pi} \mathsf{E}_\mathrm{I}[\![\, \Delta \triangleright Q \,]\!]$, *then* $\mathrm{I} \models P \cong^\pi Q$.

Proof. We consider the relation defined as $\mathrm{I} \vDash P\mathscr{R}Q$ whenever $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{E}_\mathrm{I}[\![\, \Gamma \triangleright P \,]\!]] \cong^{A\pi} \mathsf{E}_\mathrm{I}[\![\, \Delta \triangleright Q \,]\!]$, and we prove that $\mathscr{R}$ is a typed behavioural equivalence. We prove that $\mathscr{R}$ is reduction closed by using the operational correspondence of the encoding. In the following we omit to indicate the type environments inside the translation as they are irrelevant for the proof. Let $P \xrightarrow{\tau} P'$; we apply preservation of execution steps and find K s.t. $\mathsf{E}_\mathrm{I}[\![\, P \,]\!] \Longrightarrow K$ and $\{\!|\, \mathrm{I} \,|\!\} \models K \cong^{A\pi} \mathsf{E}_\mathrm{I}[\![\, P' \,]\!]$. By reduction closure of $\cong^{A\pi}$ we easily deduce that there exists H s.t. $\mathsf{E}_\mathrm{I}[\![\, Q \,]\!_\Gamma] \Longrightarrow H$ and $\{\!|\, \mathrm{I} \,|\!\} \models K \cong^{A\pi} H$. By reflection of execution steps we infer that there is Q' such that $Q \Longrightarrow Q'$ and $\{\!|\, \mathrm{I} \,|\!\} \models H \approx^A \mathsf{E}_\mathrm{I}[\![\, Q' \,]\!]$. By the results above and transitivity of $\cong^{A\pi}$ we conclude that $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{E}_\mathrm{I}[\![\, P' \,]\!] \cong^{A\pi} \mathsf{E}_\mathrm{I}[\![\, Q' \,]\!_\Gamma]$, which in turn implies $\mathrm{I} \models P'\mathscr{R}Q'$, as needed. That $\mathscr{R}$ is barb preserving is obtained directly by using Proposition 44. Let $\mathrm{I} \models P \downarrow_a$; thus $\mathrm{I}(a) <: \mathsf{r}$ and in turn $a_r^+ \in \mathrm{Range}(\{\!|\, \mathrm{I} \,|\!\})$. By the contextuality of $\cong^{A\pi}$, we find $\omega \notin fn(\{\!|\, \mathrm{I} \,|\!\})$ s.t. $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \models \mathsf{E}_\mathrm{I}[\![\, P \,]\!] \cong^{A\pi} \mathsf{E}_\mathrm{I}[\![\, Q \,]\!]$. Let y be the variable such that $\{\!|\, \mathrm{I} \,|\!\}(y) = a_r^+$. From $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \vdash T^y$ and closure under parallel composition of $\cong^{A\pi}$ we infer $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \models T^a \,|\, \mathsf{E}_\mathrm{I}[\![\, P \,]\!] \,| \cong^{A\pi} T^a \,|\, \mathsf{E}_\mathrm{I}[\![\, Q \,]\!]$. By the preservation direction of Proposition 44 we have $\mathrm{I} \models \mathsf{E}_\mathrm{I}[\![\, P \,]\!]\Downarrow_{a_w^+}$, that is $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \models \mathsf{E}_\mathrm{I}[\![\, P \,]\!] \,|\, T^a \Downarrow_\omega$. By barb preservation of $\cong^{A\pi}$ we easily deduce $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \models \mathsf{E}_\mathrm{I}[\![\, Q \,]\!] \,|\, T^a\Downarrow_\omega$; this and $a_r^+ \in \mathrm{Range}(\{\!|\, \mathrm{I} \,|\!\})$ let us infer $\mathrm{I} \models \mathsf{E}_\mathrm{I}[\![\, Q \,]\!]\Downarrow_{a_w^+}$. We apply reflection of barbs and obtain $\mathrm{I} \models Q\Downarrow_a$, as needed.

Finally we show that $\mathscr{R}$ is contextual. Let $\mathrm{I} = \mathrm{I}', a : A$. To ensure closure under new we need to prove that $\mathrm{I}' \models (\mathsf{new}\ a : A)P\mathscr{R}(\mathsf{new}\ a : A)Q$. By the hypothesis on the shape of I we know that the computing environment $\mathsf{E}_\mathrm{I}[-] \equiv \mathsf{E}_{\mathrm{I}'}[- \,|\, Chan_\mathsf{a}]$. We apply contextuality of $\cong^{A\pi}$ obtaining $\{\!|\, \mathrm{I} \,|\!\} \setminus a \models (\mathsf{new}\ a)\mathsf{E}_\mathrm{I}[\![\, P \,]\!] \cong^{A\pi} (\mathsf{new}\ a)\mathsf{E}_\mathrm{I}[\![\, Q \,]\!]$. It's easy to see that $\{\!|\, \mathrm{I} \,|\!\} \setminus a = \{\!|\, \mathrm{I}' \,|\!\}$. By the result above, by $a \notin fn(\mathsf{E}_{\mathrm{I}'})$ and by definition of $[\![\, (\mathsf{new}\ n : A)P \,]\!]$ we deduce $(\mathsf{new}\ a)\mathsf{E}_\mathrm{I}[\![\, P \,]\!] \equiv \mathsf{E}_{\mathrm{I}'}[\![\, (\mathsf{new}\ a : A)P \,]\!]$. From these results we infer $\{\!|\, \mathrm{I}' \,|\!\} \models \mathsf{E}_{\mathrm{I}'}[\![\, (\mathsf{new}\ a : A)P \,]\!] \cong^{A\pi} \mathsf{E}_{\mathrm{I}'}[\![\, (\mathsf{new}\ a : A)Q \,]\!]$

which in turn implies $\mathrm{I}' \models (\mathsf{new}\ a : A)P\mathscr{R}(\mathsf{new}\ a : A)Q$, as desired. For the second clause of contextuality, we show that if $\mathrm{I} \vdash R$ then $\mathrm{I} \models P \mid R\mathscr{R}Q \mid R$. From $\mathrm{I} \vdash R$ we infer $fn(R) \subseteq \mathrm{I}$. Moreover, we deduce that $fn(\llbracket R \rrbracket) \subseteq \{\!|\, \mathrm{I} \,|\!\}$ and that the encryption keys occurring in the definition of $\llbracket R \rrbracket$ generated around names free in $\llbracket R \rrbracket$ are supplied by $\{\!|\, \mathrm{I} \,|\!\}$, because of the assumption $\mathrm{I} \vdash R$. We find an applied pi process T_R with $bn(T_R) = bn(R)$ such that $\{\!|\, \mathrm{I} \,|\!\} \vdash T_R$ and $\llbracket R \rrbracket \equiv T_R\{\!|\, \mathrm{I} \,|\!\}$. From these and contextuality of $\cong^{A\pi}$ we infer $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{E}_{\mathrm{I}}[\llbracket P \rrbracket] \mid \llbracket R \rrbracket \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[\llbracket Q \rrbracket] \mid \llbracket R \rrbracket$. By definition of the encoding of parallel composition and by using some structural rearrangement we obtain $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{E}_{\mathrm{I}}[\llbracket P \mid R \rrbracket] \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[\llbracket Q \mid R \rrbracket] \mid \llbracket R \rrbracket$ and in turn the desired result, $\mathrm{I} \models P \mid R\mathscr{R}Q \mid R$. The last clause requires that $a \notin \mathrm{dom}(\mathrm{I})$ implies $\mathrm{I}, a : A \models P\mathscr{R}Q$. Assuming $a \neq net$, it's easy to see that the hypothesis above implies $a \notin fn(\{\!|\, \mathrm{I} \,|\!\})$. The contextuality of $\cong^{A\pi}$ let us infer $\{\!|\, \mathrm{I} \,|\!\}, a/s \models \mathsf{E}_{\mathrm{I}}[\llbracket P \rrbracket] \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[\llbracket Q \rrbracket]$; from Proposition 40 we infer $\{\!|\, \mathrm{I} \,|\!\}, \underline{a}{\Uparrow}\underline{A}/\tilde{x}, a/s \models \mathsf{E}_{\mathrm{I}}[\llbracket P \rrbracket] \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[\llbracket Q \rrbracket]$. Let $\sigma = \{\!|\, \mathrm{I} \,|\!\}, \underline{a}{\Uparrow}\underline{A}/\tilde{x}, a/s$. From the seed a we may build a channel for a exploiting closure under parallel composition of $\cong^{A\pi}$: $\sigma \models \mathsf{E}_{\mathrm{I}}[\llbracket P \rrbracket] \mid Chan_s\sigma \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[\llbracket Q \rrbracket] \mid Chan_s\sigma$; notice that $Chan_s\sigma = Chan_a$. We use erasure (Proposition 41) to remove the binding a/s and infer $\{\!|\, \mathrm{I} \,|\!\}, \underline{a}{\Uparrow}\underline{A}/\tilde{x} \models \mathsf{E}_{\mathrm{I},a:A}[\llbracket P \rrbracket] \cong^{A\pi} \mathsf{E}_{\mathrm{I},a:A}[\llbracket Q \rrbracket]$; from this we obtain $\mathrm{I}, a : A \models P\mathscr{R}Q$, as desired. □

The converse direction of Theorem 45 does not hold. In fact, as we noted, the communication protocols presuppose a certain structure associated with names. Indeed, for the names that are statically shared with the context, this structure is easily enforced by allocating the corresponding channels as part of the encoding definition. However, the context may dynamically generate new names that do not satisfy the expected invariants. Notice, for instance, that the client of a reader protocol presupposes a legitimate channel on the other end of the protocol and is not protected against malformed messages received by illegitimate channels: given that, it is easy to find a counter-example to full abstraction. For instance, in the source calculus we have:

$$a : \mathsf{w} \models a(y@\mathsf{rw}).y(x@\mathsf{rw}).y\langle x@\mathsf{rw}\rangle \cong^{\pi} a(y@\mathsf{rw})$$

This is an instance of the API@ calculus law $a : \mathsf{w} \models a(x@\mathsf{rw}).a\langle x@\mathsf{rw}\rangle \cong^{\pi} \mathbf{0}$ that we have seen in Chapter 2. This is an instance of the well-known asynchronous pi calculus law $a(x).a\langle x\rangle \cong \mathbf{0}$, and holds in our pi calculus for similar reasons. On the other hand, one easily sees that for $\mathrm{I} :> \Gamma$ we have

$$a_w^+/z \not\models \quad \mathsf{E}_{\mathrm{I}}[\llbracket \Gamma \triangleright a(y@\mathsf{rw}).y(x@\mathsf{rw}).y\langle x@\mathsf{rw}\rangle \rrbracket] \cong^{A\pi} \mathsf{E}_{\mathrm{I}}[\llbracket \Gamma \triangleright a(y@\mathsf{rw}) \rrbracket]$$

In fact, a context may create the legitimate representation of a full-fledged name $\underline{b}$ and exchange it over a; the subsequent request $\mathsf{emit}(\{sk(k), \mathsf{rw}\}_{b_r^+})$ made by the left process can now be decrypted by the context, which possesses the decryption key b_r^-, and thus the left reduct $\mathsf{E}_{\mathrm{I}}[\llbracket \Gamma, b : \mathsf{rw} \triangleright b(x@\mathsf{rw}).b\langle x@\mathsf{rw}\rangle \rrbracket]$ can be distinguished from the null process $\mathsf{E}_{\mathrm{I}}[\mathbf{0}]$ (i.e., the process environment $\Gamma, b : \mathsf{rw}$ is not compatible

with the context environment I). The following context exploits this attack to tell apart the two implementations:

$$C[-] \triangleq - \mid (\mathsf{new}\ b) net\langle \{\underline{b}\}_{a_w^+} \rangle \mid net(x).\mathsf{decrypt}\ x\ \mathsf{as}\ \{\}_{b_r^-}\ \mathsf{in}\ e\langle\rangle$$

4.4 A fully abstract implementation using a certification authority

To recover full abstraction, we must shield our translated processes from such undesired interactions. That may be achieved by setting up the synchronization protocols so as to ensure that all the exchanges occur over system-generated, trusted channels whose decryption keys remain secret.

The new translation introduces a separation between *client names*, used syntactically by context processes and by translated processes to communicate, and corresponding *server names* generated within the system and associated with system generated channels to be employed in the actual protocols for communication.

A proxy server maintains an association map between client and server names so as to preserve the expected interactions among clients. The map is implemented as a list of entries of the form $(pub(n), \underline{m})$, whose intended invariant is that m is the server counterpart of the client name n. As introduced in Chapter 3 we call this data structure a table. We call $pub(n)$ the index of the entry, and $\underline{m}$ the target. The proxy map is set up to ensure that each index has exactly one target. We represent the public keys used by the proxy service by letting $k_P^+ \triangleq ek(k)$ and $k_p^- \triangleq dk(k)$, with k a seed not known to the environment.

The read/write protocols follow the same rationale as in the previous translation, with the difference that now the clients must first obtain the access to the system channel by contacting the proxy server. The interaction between clients and proxy is as follows: the client presents a name to the proxy and the proxy replies with the corresponding server name cast at the (true) type of the name sent by the client. In case the name received is new, the proxy returns a fresh server name for which it also allocates a system channel. On the client side, the protocol is implemented as shown below:

$$\begin{array}{l}\mathsf{link}\ (\underline{u}, \underline{y})\ \mathsf{in}\ P \triangleq \\ \quad (\mathsf{new}\ h)\mathsf{emit}(\{sk(h), \underline{u},\}_{k_p^+}) \mid \mathsf{filter}\ \underline{y}\ \mathsf{with}\ sk(h)\ \mathsf{in}\ P\end{array}$$

For the proxy side, the definition is found in Table 4.2. The only subtlety is that upon receiving a name that does not occur in the association map, the proxy allocates two indexes for the same target: one index is the name received from the client, the other is the public key of the target itself. This second association is needed to make linking idempotent, so that linking a server name always returns the same name. One may wonder how a client could possibly end up requesting a link for a

server name as (i) server names originate from the proxy, and (ii) are never passed on any exchange by the clients. Notice however, that this invariant is only true of the clients that arise from the translation, not for arbitrary applied pi processes of the context.

Table 4.2 Fully Abstract Translation

Proxy server

$$
\begin{array}{rcl}
P_t & \triangleq & \mathbf{!}\,\mathsf{filter}\ (k, \underline{x}, y)\ \mathsf{with}\ k_p^-\ \mathsf{in\ if}\ y \notin Set_{t^*}\ \mathsf{then} \\
 & & \quad \mathsf{let}\ s = Typeof(\underline{x})\ \mathsf{in} \\
 & & \quad t(z).\mathsf{let}\ \underline{y} = ?(x_{ID}, z)\ \mathsf{in}\ t\langle z\rangle \mid !net\langle\{\underline{y}\Uparrow s\}_k\rangle \\
 & & \qquad \mathsf{else}\ (\mathsf{new}\ n)\, Chan_n \mid\ t\langle(z :: x_{ID}; \underline{n}) :: (n_{ID}; \underline{n})\rangle \\
 & & \qquad\quad \mid !net\langle\{\underline{n}\Uparrow s\}_k\rangle \\
Proxy & \triangleq & (\mathsf{new}\ t, t^*)\ P_t \mid t\langle\emptyset\rangle \mid t^*\langle\emptyset\rangle
\end{array}
$$

Channels

$$
\begin{array}{rcl}
WS_n & \triangleq & \mathbf{!}\,\mathsf{filter}\ (\underline{x}, z)\ \mathsf{with}\ n_w^-\ \mathsf{in\ if}\ z \notin Set_{n^*}\ \mathsf{then}\ n^\circ\langle\underline{x}\rangle \\
RS_n & \triangleq & \mathbf{!}\,\mathsf{filter}\ (y, t, z)\ \mathsf{with}\ n_r^-\ \mathsf{in} \\
 & & \quad \mathsf{if}\ z \notin Set_{n^*}\ \mathsf{then\ filter}\ \underline{x}\ \mathsf{from}\ n^\circ@t\ \mathsf{in}\ \mathbf{!}\, net\langle\{\underline{x}\}_y\rangle \\
Chan_n & \triangleq & (\mathsf{new}\ n^*, n^\circ)\ n^*\langle\emptyset\rangle \mid RS_n \mid WS_n
\end{array}
$$

Clients

$$
\begin{array}{rcl}
\langle\, \Gamma \triangleright u\langle v@T\rangle \,\rangle_\mu & \triangleq & \mathsf{let}\ \hat{\underline{u}} = [\![\, u \,]\!]_\mu,\ \hat{\underline{v}} = [\![\, v \,]\!]_\mu\ \mathsf{in\ link}\ (\hat{\underline{u}}\Uparrow \mathsf{w}, \underline{x})\ \mathsf{in\ emit}(\{\hat{\underline{v}}\Uparrow \underline{T}\}_{\underline{x}_{\mathsf{w}}^+}) \\
\langle\, \Gamma \triangleright u(x@T).P \,\rangle_\mu & \triangleq & (\mathsf{let}\ \hat{\underline{u}} = [\![\, u \,]\!]_\mu\ \mathsf{in\ link}\ (\hat{\underline{u}}\Uparrow \mathsf{r}, \underline{y})\ \mathsf{in}\ (\mathsf{new}\ k)\, \mathsf{emit}(\{sk(k), \underline{T}\}_{\underline{y}_{\mathsf{r}}^+})) \\
 & & \mid \mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle\, \Gamma, x : T \triangleright P \,\rangle_\mu \\
\langle\, \Gamma \triangleright [u = v]\, P; Q \,\rangle_\mu & \triangleq & \mathsf{if}\ \underline{u}_{ID} = \underline{v}_{ID}\ \mathsf{then}\ \langle\, \Gamma \sqcap v{:}\Gamma(u) \sqcap u{:}\Gamma(v) \triangleright P \,\rangle_{\mu[u \leftrightarrow v]} \\
 & & \mathsf{else}\ \langle\, \Gamma \triangleright Q \,\rangle_\mu \\
\langle\, \Gamma \triangleright (\mathsf{new}\ n : A)P \,\rangle_\mu & \triangleq & (\mathsf{new}\ n)\, \langle\, \Gamma, n : A \triangleright P \,\rangle_\mu \\
\langle\, \Gamma \triangleright P \mid Q \,\rangle_\mu & \triangleq & \langle\, \Gamma \triangleright P \,\rangle_\mu \mid \langle\, \Gamma \triangleright Q \,\rangle_\mu \\
\langle\, \Gamma \triangleright \mathbf{!}\, P \,\rangle_\mu & \triangleq & \mathbf{!}\, \langle\, \Gamma \triangleright P \,\rangle_\mu \\
\langle\, \Gamma \triangleright \mathbf{0} \,\rangle_\mu & \triangleq & \mathbf{0}
\end{array}
$$

Top−level translation $\langle\, \Gamma \triangleright P \,\rangle \triangleq \langle\, \Gamma \triangleright P \,\rangle_\emptyset$

Having given the intuitions, the definitions in Table 4.2 should be easily understood. Notice that the clause for restriction is defined homomorphically in this translation, as the creation of the channel is delegated entirely to the proxy server. In the translation of the matching construct, we could test matching on linked names

(which would make the translation more uniform), rather than on client names. While this choice has no consequences in the centralized translation, it does create a problem in the distributed implementation that we will present in Chapter 6. Indeed in that case matching should test names linked on the same proxy and this creates a technical problem as names are known at different proxies, under different linked names.

Full Abstraction

Having set up the underlying infrastructure, we now have the expected protection against hostile contexts. We may therefore strengthen the result of Theorem 45 as desired, provided that we plug our processes in the appropriate computing environment. We first define

$$\mathsf{CE}[-] \;=\; - \mid W \mid \mathit{Proxy}$$

and let the low-level term environments corresponding to the high-level type environment be extended with a new binding expressing the knowledge of the public proxy encryption key k_P^+ needed to interact with the proxy. We obtain that the encoding $\langle\cdot\rangle$ is sound and complete.

Theorem 46. *Let* Γ, Δ *and* I *be closed type environments such that* $\Gamma, \Delta <: \mathrm{I}$. *Then:*

$$\mathrm{I} \models P \cong^{\pi} Q \;\; \textit{iff} \;\; \{\!|\, \mathrm{I} \,|\!\}, k_P^+/y \models \mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \cong^{A\pi} \mathsf{CE}[\,\langle\, \Delta \triangleright Q \,\rangle\,] \;.$$

4.5 Related work

In the literature on process calculi, a standard method to study the expressiveness of a calculus is to provide an encoding in another, possibly well-studied formalism, and to reason on the semantic properties preserved by the encoding. The expressivity of the pi calculus and its fragments has been studied intensively, e.g. [99, 85, 84, 21, 69, 32, 35, 33]; mint process calculi are often related to the pi calculus, e.g. [51, 27, 86, 17, 16, 108, 54].

Translations from high level calculi using abstractions as secure channels and types to lower-level ones using open communications and cryptographic functions are often referred as implementations [12, 1, 10, 8]. Indeed, from an abstract point of view, the low-level language models the open and distributed communications that characterize actual network protocols. In this field, the work closest to ours is [8]. In that paper, a fully abstract encoding of the join calculus into a dialect of join using cryptographic primitives is given. The located nature of channels in the join calculus makes it possible to rely on a very compact representation in which a communication channel a is associated with a low level key name a^+, and to protect communications using an asymmetric cryptosystem. The authors of [8] propose two encodings, with different purposes. A first, compositional encoding translates every

message of the source process in a cryptographic protocol without any assumption on the distribution of processes. The second encoding avoids the use of cryptography for all communications by using a centralized context that acts as a firewall with an encrypting tunnel [9]. Join processes immersed in this context execute internal communications without be disturbed by the firewall, while messages sent outside the context boundary and received from the context are respectively marshalled and unmarshalled by encoding/decoding contents. The authors prove that the two encodings are equivalent, up-to behavioral equivalence. The soundness of the compositional encoding is based on correctness conditions for the protocol; for instance (a) the emission of a message needs to be non-interruptible, (b) a recipient of a message upon picking a "wrong message" needs both to recover it's initial state and to make the picked message available, (c) an emission needs not to interfere with other emissions. We followed these guidelines and devised cryptographic protocols that satisfy the conditions above.

The problem of preserving the forward secrecy of communications (see Section 4.1) in [8] is avoided altogether, because in the join calculus the capability of receiving on a channel can never been transferred. Quoting the authors: "An advantage of the join-calculus as the starting point of our work is that, it relies on asymmetric (one-directional) channels, unlike the standard pi calculus which relies on symmetric (two-directional) channels. We did in fact attempt to use the pi calculus, but we encountered a subtle difficulty [1]". The use of join-calculus instead of pi calculus does not entail a loss of expressiveness, because there is a fully abstract encoding from the pi calculus without matching in the join-calculus [51]; like ours, the translation relies both on the presence of proxy pairs of internal and external names, and on relays among the pairs components. By composing the two encodings [51],[8] one obtains a fully abstract encoding of the pi calculus without matching. However, the approach followed in [51] to encode the pi calculus does not extend in the presence of matching. This is because, as noted in [104], the ability of test syntactic equality on names invalidates the semantic equalities on names provided by the equators [65] used in [51] to merge internal and external names. Therefore, it is not clear how to implement matching in a join calculus extended with a name-matching construct [53, 8].

A different approach to the implementation of high level channel abstractions is followed in [14]; in that paper high level processes are mapped in a computational setting, rather than in an algebraic framework. The authors implement high-level functionalities of a variant of the pi calculus supporting reliable messaging and authentication primitives, by using concrete cryptography based on probabilistic algorithms that operate on bitstrings. This is in contrast with the algebraic view of cryptographic protocols followed in this thesis and in other recent works, e.g. [10, 8]. We agree that the approach we followed is probably too abstract. Providing a cryptographically sound implementation of the type-based management of capabilities we have presented in this thesis seems to us an interesting challenge.

5 Proofs

In this chapter we present the proofs of the results of Chapter 4. We first introduce the outline of the proof of soundness. Then we introduce the definitions of administrative reductions and equivalence and prove some useful properties of administrative equivalence and its variants. Then we prove cryptographic Lemmas that will be useful to treat the leftover of the protocol as noise (cf. [8]). The next sections are dedicated respectively to the proof of the soundness and completeness of the encoding $\langle\cdot\rangle$.

5.1 Outline of the proof of soundness

In programming languages translations, often the soundness of the encoding is a direct consequence of the operational correspondence:

(i) $P \to P'$ implies $\{\!|\, P \,|\!\} \to \mathscr{R} \{\!|\, P' \,|\!\}$

(ii) $\{\!|\, P \,|\!\} \to Q$ implies $\exists P' \,.\, P \to P'$ and $Q \mathscr{R} \{\!|\, P' \,|\!\}$

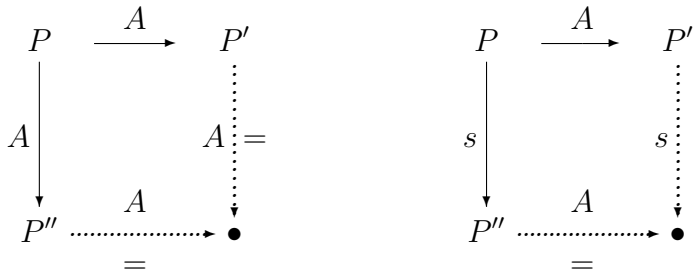
Here $\mathscr{R}$ is a behavioural equivalence finer as possible ($\cong$ often not suffices, e.g. [21]). (i) is often called preservation of execution steps while (ii) is called reflection of executions steps. While preservation is typically easily gained, the reflection of executions is stated as above only for "prompt" encodings [84], i.e. encodings that are immediately ready to execute a move that match an non-translated reduction. Proving reflection in non-prompt encodings often needs workarounds that are typically ad-hoc for the specific translation.

In this section we outline the approach we have used in order to prove the soundness of our encoding (that is non-prompt) and particularly the reflection of executions steps (see Proposition 87). The technique relies on a partitioning of the reduction relation of the transition system of the calculus that is the target of the encoding, in two smaller relations, noted $\xrightarrow{A}$ and $\xrightarrow{s}$, that are partially confluent. Here the reductions $\xrightarrow{A}$ represent those introduced by the encoding in order to prepare the protocol for the commit action and to complete the protocol. In contrast, $\xrightarrow{s}$ represents the commit action that corresponds to the high level execution step.

To ease the readability of the outline, we consider an applied pi calculus with unlabelled semantics and a standard notion of contextuality for behavioral equivalence. We let solid and dotted lines indicate respectively universal and existential quantification. For a relation $\mathscr{R}$ we let $\mathscr{R}^{=}$ be its reflexive closure.

The first step is to define when a reduction is administrative. While in the next section we give a definition based on the classification of the reductions of the encoded processes, for the sake of clearness here we base the definition on the properties sufficient to our aim. Such properties are exploited in the proof of Proposition 65.

Definition 19. *Let P be a closed process. We let administrative reduction, noted $\xrightarrow{A}$, be the subset $\xrightarrow{A}{=}\!\!\rightarrow \setminus \xrightarrow{s}$ induced by the following diagrams.*



We let $\overset{A}{\Longrightarrow}$ be the reflexive and transitive closure of $\xrightarrow{A}$. We update the definition of behavioral equivalence for this simplified applied pi calculus. We say that a relation is contextual if it is closed under all contexts of the language.

Definition 20 (Behavioral equivalence). *Behavioral equivalence, noted $\approx^{A\pi}$, is the largest symmetric relation $\mathscr{R} \subseteq \mathscr{P} \times \mathscr{P}$ that is barb preserving, contextual and such that $P\mathscr{R}Q$ and $P \longrightarrow P'$ implies that there is Q' such that $Q \Longrightarrow Q'$ and $P'\mathscr{R}Q'$.*

Behavioral equivalence relate processes that may need a different number of execution steps to reach related redexes. For this reason, it is often too loose, and smaller relations considering a more finer treatment of weak reductions have been introduced, e.g. [103]. Following this approach, we give a notion of equivalence that ignores only reductions that have the confluence properties defined above.

Definition 21 (Administrative equivalence). *Administrative equivalence, noted $\approx^{A}$, is the largest symmetric relation $\mathscr{R} \subseteq \mathscr{P} \times \mathscr{P}$ that is strong barb preserving, contextual and such that $P\mathscr{R}Q$ implies:*

- *$P \xrightarrow{A} P'$ imply $Q \overset{A}{\Longrightarrow} Q'$ for some Q' such that $P'\mathscr{R}Q'$*
- *$P \xrightarrow{s} P'$ imply $Q \overset{A}{\Longrightarrow}\xrightarrow{s}\overset{A}{\Longrightarrow} Q'$ for some Q' such that $P'\mathscr{R}Q'$.*

We then prove that administrative equivalence is indeed an equivalence relation, and that is included in behavioral equivalence.

Proposition 47. *If* $P \approx^{A} Q$ *then* $P \approx^{A\pi} Q$.

Then we show an important property of administrative equivalence that says that is closed under administrative reductions, in the following sense.

Lemma 48. *If* $P \xrightarrow{A} P'$ *then* $P \approx^{A} P'$.

Roughly, we gain the following result.

Proposition 49 (Operational Correspondence). *Let* P *be an* API@ *calculus process and let* Γ *be a closed type environment such that* $\Gamma \vdash P$.

(i) $P \xrightarrow{\tau} P'$ *implies* $[\![P]\!] \stackrel{A}{\Longrightarrow} \xrightarrow{s} \approx^{A} [\![P']\!]$

(ii) $H \approx^{A} [\![P]\!] \ \wedge \ H \xrightarrow{\tau} H'$ *implies* $\exists P'$ *such that* $P \Longrightarrow P'$ *and* $H' \approx^{A} [\![P']\!]$.

Proof. (i) is proved by induction on the derivation of $P \xrightarrow{\tau} P'$. (ii) is proved in two steps. If $H \xrightarrow{A} H'$ we use the lemma above and infer $H \approx^{A} H'$ and in turn by transitivity of $\approx^{A}$ we obtain $H' \approx^{A} [\![P]\!]$, as requested. Otherwise we have the transition $H \xrightarrow{s} H'$, and by definition there is K, K' such that $[\![P]\!] \stackrel{A}{\Longrightarrow} K \xrightarrow{s} \stackrel{A}{\Longrightarrow} K'$ and $H' \approx^{A} K'$. We then show that there is a *canonical* sequence of administrative reductions, noted $\stackrel{\ulcorner A \urcorner}{\Longrightarrow}$, only including the administrative steps from $[\![P]\!]$ required to enable the synchronization in K, and having the following property: $[\![P]\!] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} \xrightarrow{s} \approx^{A} K'$. We therefore prove that when the reductions above hold, there is P' such that $P \xrightarrow{\tau} P'$ and $[\![P']\!] \approx^{A} K'$. This is done by case analysis on $\xrightarrow{s}$ and by observing that the low-level commit actions arise when reductions occur on a channel queue, and that any of such queue reductions occurs only when at the high level there is a dynamically typed synchronization. Finally we use transitivity of $\approx^{A}$ to infer the expected result, $[\![P']\!] \approx^{A} H'$.

The proposition above and the inclusion of administrative equivalence in behavioral equivalence let us prove the Theorem of soundness of the translation (see [21] and the proof of Theorem 45). We let $\approx^{\pi}$ be the non-indexed version of behavioral equivalence for API@ calculus processes.

Theorem 50. *Let* P, Q *be* API@ *calculus processes and let* Γ, Δ *be closed type environments such that* $\Gamma \vdash P$ *and* $\Delta \vdash Q$. *If* $[\![P]\!] \approx^{A\pi} [\![Q]\!]$ *then* $P \approx^{\pi} Q$.

Discussion The approach we followed draws inspiration from the up to technique used by Fournet in his doctoral dissertation [50] to prove the completeness of the encoding of the pi calculus into the join calculus. Based on a confluence theorem by Van Oostrom [107], the author devised a sound up to technique based on a subset of the reduction relation, namely deterministic reduction, that roughly satisfies our two diagrams plus other diagrams used to "roll back" such reductions. These

further conditions are too strong for our encoding. There is however an important difference among the diagrams we share: we match each commit reduction with exactly one commit reduction, while in [50] it is admitted that a commit reduction it is matched in zero or one commit reductions. This would invalidate our reasonings on the operational corresponding of encodings, that are based on the assumption that administrative equivalence relate processes that have executed exactly the same number of commit while they may differ in administrative steps introduced by the synchronization protocol.

5.2 Administrative equivalences

As introduced in Sections 4.3 and 5.1, our translated processes need to execute several steps before to be ready for the commit synchronization step on the channel queue that corresponds to the high-level synchronization on the channel. As it turns out, however, these steps are not observable and can be factored out in the proof by resorting to a suitable notion of (term-indexed) *administrative* equivalence $\approx^A$. The definition of $\approx^A$ draws on a classification for the reductions of the compiled processes into *commitment* steps, corresponding to synchronizations on the channel queues, and *administrative reductions*, corresponding to the steps that precede and follow the commitment steps (similarly to [55]). Then two processes are equated by $\approx^A$ only if they are behaviorally equivalent and, in addition, they can simulate each other's commitment transitions in a 'strong' way.

Given these intuitions, in the following we present the definition of administrative reduction and equivalence. Then we prove some interesting closure properties of administrative equivalence, and we show that the relation is contained in behavioral equivalence.

Remark The translations we presented in the previous chapter make use of recursion in filtering packets from the network. The recursion mechanism is implemented through the process $\mathsf{rec}\, X.P = (\mathsf{new}\ a)\ (a\langle\rangle \mid !a().P\{a\langle\rangle/X\})$ with $a \notin fn(P)$. Clearly, this make filtering not prompt, i.e. $\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ N\ \mathsf{in}\ P$ needs to execute a deterministic step to be able to receive a packet from the network. As we will show, this step is not visible up to administrative equivalence. Nevertheless, to ease the proofs presented in this chapter it is convenient to consider prompt filters. To this aim we modify recursion as in the following (admitely less elegant) definition:

$$\mathsf{rec}\, X.P \triangleq (\mathsf{new}\ a)\ (P\{a\langle\rangle/X\} \mid !a().P\{a\langle\rangle/X\}) \quad a \notin fn(P)\ .$$

Administrative reductions and equivalence

Based on these intuitions, in the following we introduce the notion of administrative reduction.

A reduction is administrative whenever satisfies one of the clauses of the Definition 22 introduced some lines below; we assume that the processes of interest use a public interface for communications.

The first clause is clear. The second clause describes a protocol that uses a symmetric cryptoscheme and that is formed by a recursive filter process waiting for packets on the public interface and a replicated public emission of the packet waited by the filter; the reduction involving the filter receiving the packet is administrative whenever the content of the packet has the correct arity and there not exists another packet with different content that could be accepted by the filter.

The third clause describes a similar protocol involving a recursive filter and a public output and says that a reduction is administrative whenever is inferred from the filter receiving a wrong packet (i.e., encrypted with a different key or containing terms mismatching the expected arity). The fourth clause describes the behaviour of a filter of type T on a private channel n° which receives a term which represents a type S that is not a sub-type of T. The fifth clause describes the behaviour of a process which receives a set carried on by a private output $n^*\langle M\rangle$ and then tests if a term N belong to the set and if yes continue as Q; the reduction is administrative whenever the existence in the scope of n^* of a testing process on the same term N and continuing as Q' implies that $Q \equiv Q'$.

The sixth clause describes a protocol where a table list carried on a private channel is managed in a way similar to that of the Proxy server (see Tab. 4.2). The managing of the table is done by a meta-process $T_N(Q)$ defined around Q; here Q is a process with free variables that will be closed in $T_N(Q)$. The reduction under analysis is the one where $T_N(Q)$ receives a table M from a private output $t\langle M\rangle$. Next the continuation of $T_N(Q)$ tests if M contains an index N_{ID} (and in case binding the associated entry to the free variables of Q): if yes the process continue as $t\langle M\rangle \,|\, Q$ otherwise it creates a channel for the fresh name n, outputs on t the updated $M :: (N_{ID}; \underline{n}) :: (n_{ID}; \underline{n})$ and executes Q with $\underline{n}$ binding the free variables of Q. The last clause says that that the unblocking reduction by which we implement recursion in $\mathsf{rec}\, X.P$ is administrative.

In the following, for terms M, N we write $M = N$ to indicate that M is syntactically equal to N; we write $|M|$ to refer to the dimension of M.

Definition 22 (Administrative reduction). *Let $net \in fn(P)$ and suppose $P \xrightarrow{\tau} P'$. We say that $P \xrightarrow{\tau} P'$ is an administrative reduction, noted $P \xrightarrow{A} P'$, when $P\downarrow_a \Leftrightarrow P'\downarrow_a$ and one of the following cases holds:*

1. *the reduction is inferred from a synchronization among a replicated input on net and a replicated output on net;*

2. $P \equiv C[\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ P \,|\, !\, net\langle\{M\}_{sk(N)}\rangle]$ *and*
   $P' \equiv C[\mathsf{filter}\ \{M\}_{sk(N)}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ P \,|\, !\, net\langle\{M\}_{sk(N)}\rangle]$ *and* $|M| = |\tilde{y}|$

and $\forall C', M' \;:\; P \equiv C'[net\langle\{M'\}_{sk(N)}\rangle] \wedge |M'| = |\tilde{y}| \;\Rightarrow\; M = M'$ *where* filter $\tilde{M}$ with N in P *is the process* Q *s.t.* filter $\tilde{x}$ with N in $P \xrightarrow{net(\tilde{M})} Q$;

3. $P \equiv C[\text{filter } \tilde{y} \text{ with } sk(N) \text{ in } P \mid net\langle M\rangle]$ *and* $P' \equiv C[\text{filter } M \text{ with } sk(N) \text{ in } P]$ *and* $\neg(M = \{M'\}_{sk(N)} \wedge |M'| = |\tilde{y}|)$;

4. $P \equiv C[(\text{new } n^\circ)Q \mid \text{filter } \tilde{y} \text{ from } n^\circ@T \text{ in } P \mid n^\circ\langle M\rangle]$ *and* $P' \equiv C[(\text{new } n^\circ)Q \mid \text{filter } M \text{ from } n^\circ@T \text{ in } P]$ *and* M *has type* $S \not<: T$ *where* filter M from $c@T$ in P *is the process* Q *s.t.* filter $\tilde{x}$ from $c@T$ in $P \xrightarrow{c(M)} Q$;

5. $P \equiv C[(\text{new } n^*)n^*\langle N\rangle \mid_{i\in I} \text{if } N_i \notin Set_{n^*} \text{ then } Q_i]$ *and* $P' \equiv C[(\text{new } n^*)(\text{if } N_j \notin N \text{ then } n^*\langle N_j :: N\rangle \mid Q_j \text{ else } n^*\langle N\rangle) \mid_{i\in I\setminus j} \text{if } N_i \notin Set_{n^*} \text{ then } Q_i]$ *and* $\forall i, j \in I \,.\, (N_i = N_j) \Rightarrow (Q_i \equiv Q_j)$;

6. $P \equiv C[(\text{new } t)t\langle M\rangle \mid_{i\in I} T_{N_i}(Q_i)]$ *and* $P' \equiv C[(\text{new } t)\text{let } \tilde{y} = ?(N_{jID}, M) \text{ in } t\langle M\rangle \mid Q_j \text{ else } ((\text{new } n)Chan_n \mid t\langle(N_{jID}, \underline{n}) :: (n_{ID}, \underline{n}) :: M\rangle \mid Q_j\{\underline{n}/\tilde{y}\}) \mid_{i\in I\setminus j} T_{N_i}(Q_i)]$ *where* $T_N(Q) = t(z).\text{let } \tilde{y} = ?(N_{ID}, z) \text{ in } t\langle z\rangle \mid Q \text{ else } (\text{new } n)Chan_n \mid t\langle(N_{ID} :: \underline{n}) :: (n_{ID}; \underline{n}) :: z\rangle \mid Q\{\underline{n}/\tilde{y}\}$ *and* $fv(Q) = \tilde{y}$;

7. $P \equiv C[(\text{new } n)n\langle\rangle \mid !n(x).Q\{n\langle\rangle/X\}]$ *and* $P' \equiv C[\text{rec } X.Q]$ *where* $n \notin fn(Q)$;

where $I = 1, \ldots, n$. *We let* $\stackrel{A}{\Longrightarrow}$ *be the reflexive and transitive closure of* $\stackrel{A}{\longrightarrow}$.

We set $\longrightarrow = \xrightarrow{\tau} \setminus \xrightarrow{A}$.

Based on this we have the following notion of equivalence that ignores administrative reductions.

Definition 23 (Administrative equivalence). *Administrative equivalence, noted* $\approx^A$, *is the largest symmetric and contextual term-indexed relation* $\mathscr{R}$ *such that* $\rho \models P\mathscr{R}Q$ *implies:*

- *if* $\rho \models P\downarrow_n$ *then* $\rho \models Q\downarrow_n$
- $P \xrightarrow{A} P'$ *imply* $Q \stackrel{A}{\Longrightarrow} Q'$ *for some* Q' *such that* $\rho \models P'\mathscr{R}Q'$
- $P \longrightarrow P'$ *imply* $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q'$ *for some* Q' *such that* $\rho \models P'\mathscr{R}Q'$.

The first desirable property is that $\approx^A$ is indeed an equivalence. We first need a technical lemma.

Lemma 51. *Assume* $\rho \models P \approx^A_\rho Q$. *The following hold.*

$$(i)\quad (P \stackrel{A}{\Longrightarrow} P') \Rightarrow \exists Q' \,.\, Q \stackrel{A}{\Longrightarrow} Q' \;\wedge\; P' \approx^A_\rho Q'$$
$$(ii)\quad (P \stackrel{A}{\Longrightarrow} \rightarrow \stackrel{A}{\Longrightarrow} P') \Rightarrow \exists Q' \,.\, Q \stackrel{A}{\Longrightarrow} \rightarrow \stackrel{A}{\Longrightarrow} Q' \;\wedge\; P' \approx^A_\rho Q'$$

Proof. The proof of (i) is is analogous to the proof of Proposition 33; we prove (ii). Let $P \approx^A_\rho Q$ and suppose $P \stackrel{A}{\Longrightarrow} P' \longrightarrow P'' \stackrel{A}{\Longrightarrow} P^*$. By definition $\exists Q'$ such that $Q \stackrel{A}{\Longrightarrow} Q'$ and $Q' \approx^A_\rho P'$. By this we infer that there is Q'' such that $Q' \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q''$ and $Q'' \approx^A_\rho P''$ Finally we infer that $\exists Q^*$ such that $Q'' \stackrel{A}{\Longrightarrow} Q^*$ and $P^* \approx^A_\rho Q^*$. We glue the reductions obtaining $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} \approx^A_\rho P^*$, as desired. □

Proposition 52. *$\approx^A$ is an equivalence relation.*

Proof. It's easy to see that $\approx^A$ is reflexive; by definition the relation is symmetric. We therefore prove that $\approx^A$ is transitive. We proceed by chasing-diagram arguments and show that $\approx^A\approx^A$ is an administrative equivalence.

Let $P \approx^A_\rho R \approx^A_\rho Q$. From $fn(P,R) \subseteq fn(\rho)$ and $fn(R,Q) \subseteq fn(\rho)$ we obtain $fn(P,Q) \subseteq fn(\rho)$. To see barb preservation, let $\rho \models P\downarrow_a$; by definition $\rho \models R\downarrow_a$ and in turn $\rho \models Q\downarrow_a$, as needed. To see reduction closure, let $P \stackrel{A}{\longrightarrow} P'$. By hypothesis we have that there is R' such that $R \stackrel{A}{\Longrightarrow} R'$ and $P' \approx^A_\rho R'$. By the hypothesis $R \approx^A_\rho Q$ and (i) we infer that there is Q' such that $Q \stackrel{A}{\Longrightarrow} Q'$ and $R' \approx^A_\rho Q'$, as needed. By gluing $fn(P',R') \subseteq fn(\rho)$ and $fn(R',Q')$ we obtain $fn(P',Q') \subseteq fn(\rho)$. Otherwise assume $P \longrightarrow P'$. By hypothesis we have that there is R' such that $R \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} R'$ and $P' \approx^A_\rho R'$. By (ii) we infer that there is Q' such that $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q'$ and $Q' \approx^A_\rho R'$, as needed.

That $\approx^A\approx^A$ is symmetric can be easily viewed by following the same proof when the hypothesis regards Q. Next we prove that $\approx^A\approx^A$ is contextual. Let $P \approx^A_\rho R \approx^A_\rho Q$. To see closure under composition assume $\rho \vdash S^*$ and let $S = S^*\rho$. By hypothesis both $\rho \models P \,|\, S \approx^A R \,|\, S$ and $\rho \models R \,|\, S \approx^A Q \,|\, S$. Thus $\rho \models P \,|\, S \approx^A\approx^A Q \,|\, S$, as needed. Moreover $fn(P,S,Q) \subseteq fn(\rho)$. Now let $n \notin fn(\rho)$. We have $\rho, n/x \models P \cong^{A\pi} R$ and $\rho, n/x \models R \cong^{A\pi} Q$; thus $\rho, n/x \models P \cong^{A\pi}\cong^{A\pi} Q$, as desired. Trivially we have $fn(P,Q) \subseteq fn(\rho, n/x)$. Finally we know $\rho \setminus n \models (\mathsf{new}\ n)P \cong^{A\pi} (\mathsf{new}\ n)R$ and $\rho \setminus n \models (\mathsf{new}\ n)R \cong^{A\pi} (\mathsf{new}\ n)Q$; this implies $\rho \setminus n \models (\mathsf{new}\ n)P \cong^{A\pi}\cong^{A\pi} (\mathsf{new}\ n)Q$. From the hypotheses above we know $fn((\mathsf{new}\ n)P, (\mathsf{new}\ n)Q) \subseteq fn(\rho \setminus n)$, and we are done.

□

A simple, but important property of $\approx^A$ is that it is contained in $\cong^{A\pi}$. More precisely:

Lemma 53. *If $\rho \models P \approx^A Q$ then $\rho \models P \cong^{A\pi} Q$.*

Proof. Let $\rho \models P\mathscr{R}Q$ whenever $\rho \models P \approx^A Q$. The contextuality of $\mathscr{R}$ follows directly from the definition of $\approx^A$. To see that $\mathscr{R}$ preserve barbs, suppose $\rho \models P\downarrow_a$; by definition of $\approx^A$ we have $\rho \models Q\downarrow_a$ and in turn $\rho \models Q\Downarrow_a$, as requested. To see reduction closure, let $P \stackrel{\tau}{\longrightarrow} P'$. In case $P \stackrel{A}{\longrightarrow} P'$ we infer that $\exists Q'$ s.t. $Q \stackrel{A}{\Longrightarrow} Q'$ and $\rho \models P' \approx^A Q'$. We have thus found Q' s.t. $Q \Longrightarrow Q'$ and $\rho \models P'\mathscr{R}Q'$. In

case $P \longrightarrow P'$ we infer that $\exists Q'$ s.t. $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q'$ and $\rho \models P' \approx^A Q'$. Thus $Q \stackrel{\tau}{\Longrightarrow} Q'$ and $\rho \models P' \mathscr{R} Q'$. □

A hierarchy of administrative equivalences

In carrying out proofs, it is often convenient to consider relations smaller than administrative equivalence. To this aim, we introduce the strong version and an asymmetric version of administrative equivalence (cf. [103]).

We first define the strong version of administrative equivalence.

Definition 24 (Strong administrative equivalence). *Strong administrative equivalence, noted $\sim^A$, is the largest symmetric, contextual term-indexed relation $\mathscr{R}$ such that $\rho \models P \mathscr{R} Q$ implies:*

- $\rho \models P\downarrow_n$ *iff* $\rho \models Q\downarrow_n$
- $P \stackrel{A}{\longrightarrow} P'$ *implies* $Q \stackrel{A}{\longrightarrow} Q'$ *for some* Q' *such that* $\rho \models P' \mathscr{R} Q'$
- $P \longrightarrow P'$ *implies* $Q \longrightarrow Q'$ *for some* Q' *such that* $\rho \models P' \mathscr{R} Q'$.

The proof of the following result follows the same rationale of the respective proofs for $\approx^A$.

Lemma 54. *$\sim^A$ is an equivalence relation.*

The next lemma says that structural congruence is included in strong administrative equivalence. The lemma relies on the commutation of administrative reductions with $\equiv$.

Proposition 55. *Let $P \equiv Q$. The following hold.*

1. *If $P \stackrel{A}{\longrightarrow} P'$ then there is Q' s.t. $Q \stackrel{A}{\longrightarrow} Q'$ and $P' \equiv Q'$;*
2. *If $P \longrightarrow P'$ then there is Q' s.t. $Q \longrightarrow Q'$ and $P' \equiv Q'$.*

Proof. We first prove (1). Suppose $P \stackrel{A}{\longrightarrow} P'$. Let n be the clause of Def. 22 from which $P \stackrel{A}{\longrightarrow} P'$ has been inferred. Let P_n, P'_n be the processes such that in Def. 22(n) we have $P \equiv P_n$ and $P' \equiv P'_n$. For instance when $n = 1$ by Def. 22(1) we infer $P_1 = C[!\, net\langle M\rangle \mid !\, net(x).S]$, for some M and S, and $P'_1 = C[!\, net\langle M\rangle \mid !\, net(x).S \mid S\{M/x\}]$. From transitivity of $\equiv$ we infer $Q \equiv P_n$. From Prop. 37 and $P \equiv Q$ we infer that there is Q' such that $Q \stackrel{\tau}{\longrightarrow} Q'$ and $P' \equiv Q'$. By transitivity of $\equiv$ we have $Q' \equiv P'_n$. Therefore by Definition 22(n) we obtain $Q \stackrel{A}{\longrightarrow} Q'$, as desired. Next we prove (2). Suppose $P \longrightarrow P'$. Therefore for all clause n of Def. 22 we have that $\neg(P \equiv P_n \wedge P' \equiv P'_n)$ where P_n and P'_n are defined as above. From $Q \equiv P$ and Prop. 37 we infer that there is Q' such that $Q \stackrel{\tau}{\longrightarrow} Q'$ and $P' \equiv Q'$. From

transitivity of $\equiv$ and $P \equiv Q$ and $P' \equiv Q'$ and the hypothesis above we infer that for all clause n of Definition 22 we have $\neg(Q \equiv P_n \wedge Q' \equiv P'_n)$, that is $Q \longrightarrow Q'$, as desired.

□

Lemma 56. *If $P \equiv Q$ then $\rho \models P \sim^A Q$.*

Proof. Let $\rho \models P\mathscr{R}Q$ whenever $P \equiv Q$. We show that $\mathscr{R}$ is a strong administrative equivalence. From strong barb preservation of $\equiv$ we infer that $\mathscr{R}$ is strong barb preserving. Since $\equiv$ is preserved by all operators of applied pi calculus and since ρ is arbitrary we easily infer that $\mathscr{R}$ is contextual. We prove that $\mathscr{R}$ is reduction closed. Let $\rho \models P\mathscr{R}Q$. Suppose $P \xrightarrow{A} P'$. By Prop.55(1) we have that there is Q' such that $Q \xrightarrow{A} Q'$ and $P' \equiv Q'$. By definition $\rho \models P'\mathscr{R}Q'$, as desired. Now suppose $P \longrightarrow P'$. We apply Prop.55(2) and find Q' such that $Q \longrightarrow Q'$ and $P' \equiv Q'$. We obtain $\rho \models P'\mathscr{R}Q'$, as desired. □

As one would expect we have that strong administrative equivalence is included in administrative equivalence.

Lemma 57. *If $\rho \models P \sim^A Q$ then $\rho \models P \approx^A Q$.*

Proof. Let $\mathscr{R}$ be the relation containing the couples related by the lemma and suppose $\rho \models P \sim^A Q$. Contextuality and barb preservation of $\mathscr{R}$ are straightforward. To see that R is reduction closed, suppose $P \xrightarrow{A} P'$. By definition of $\sim^A$ we have that there exists Q' such that $Q \xrightarrow{A} Q'$ and $\rho \models P' \sim^A Q'$. We have thus found Q' such that $Q \stackrel{A}{\Longrightarrow} Q'$ and $\rho \models P'\mathscr{R}Q'$. Now suppose $P \longrightarrow P'$. From definition of $\mathscr{R}$ we infer that there exists Q' such that $Q \longrightarrow Q'$ and $\rho \models P' \sim^A Q'$. We have done as this implies $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q'$ and $\rho \models P'\mathscr{R}Q'$, as desired. □

We define an asymmetric variant of administrative equivalence that will be useful in proving that a relation is contained in this equivalence (cf. [103]). We let $\xrightarrow{A^=}$ be the reflexive closure of $\xrightarrow{A}$.

Definition 25 (Administrative expansion). *Administrative expansion, noted $\succeq^A$, is the largest contextual term-indexed relation $\mathscr{R}$ such that $\rho \models P\mathscr{R}Q$ implies:*

- *$\rho \models P\downarrow_n$ iff $\rho \models Q\downarrow_n$*
- *$P \xrightarrow{A} P'$ implies $Q \xrightarrow{A^=} Q'$ for some Q' such that $\rho \models P'\mathscr{R}Q'$*
- *$P \longrightarrow P'$ implies $Q \longrightarrow Q'$ for some Q' such that $\rho \models P'\mathscr{R}Q'$*
- *$Q \longrightarrow Q'$ implies $P \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} P'$ for some P' such that $\rho \models P'\mathscr{R}Q'$*
- *$Q \xrightarrow{A} Q'$ implies $P \stackrel{A}{\Longrightarrow} \xrightarrow{A} \stackrel{A}{\Longrightarrow} P'$ for some P' such that $\rho \models P'\mathscr{R}Q'$.*

When convenient we will write often $\rho \models Q \preceq^A P$ whenever $(\rho, P, Q) \in \succeq^A$.

The following result ensures transitivity of $\succeq^A$. We first need a technical lemma.

Lemma 58. *Let* $P \underbrace{\xrightarrow{A} \ldots \xrightarrow{A}}_{n \text{ times}} P'$ *be noted as* $P \xrightarrow{A^n} P'$ *whenever* $n \geq 0$. *Then*

$$(a) \quad (P \succeq^A_\rho Q \wedge Q \stackrel{A^m}{\Longrightarrow} Q' \wedge m \geq 0) \Rightarrow (\exists P' \,.\, P \stackrel{A^n}{\Longrightarrow} P' \wedge n \geq m \wedge P' \succeq^A_\rho Q')$$

Proof. We proceed by induction on m. To ease the notation we write $P \stackrel{A^+}{\Longrightarrow} P'$ whenever $P \stackrel{A^n}{\Longrightarrow} P'$ and $n \geq 0$. The base case $m = 0$ trivially holds by choosing $P' = P$. For the induction step, let the hypothesis holds for m and assume $Q \stackrel{A^m}{\Longrightarrow} Q' \xrightarrow{A} Q^*$. By I.H. we have $P' \stackrel{A^+}{\Longrightarrow} P^*$ for some P^* s.t. $P^* \succeq^A_\rho Q^*$. We glue the results and infer that $P \stackrel{A^n}{\Longrightarrow} \stackrel{A^+}{\Longrightarrow} \succeq^A_\rho Q^*$ that actually proves the claim since by hypothesis $n \geq m$. □

Proposition 59. $\succeq^A$ *is a preorder.*

Proof. The reflexivity of $\succeq^A$ follows easily by showing that the identity relation is an administrative expansion. To show transitivity, we consider the relation $\succeq^A \succeq^A$ and we show that is an administrative expansion. Let $P \succeq^A_\rho R \succeq^A_\rho Q$. To see that the relation is barb preserving assume $\rho \models P \downarrow_a$. By hypothesis $\rho \models R \downarrow_a$ and again by hypothesis $\rho \models Q \downarrow_a$. The other direction is analogous since the clause for barb preservation is symmetric in $\succeq^A$. To see reduction closure, we first assume $P \xrightarrow{A} P'$. By hypothesis we have that there is R' such that (i) $R \xrightarrow{A} R'$ or (ii) $R' = R$ and $\rho \models P' \succeq^A R'$. In case (i) by hypothesis we infer that there is Q' such that $Q \xrightarrow{A^=} Q'$ with $\rho \models R' \succeq^A Q'$. Thus $\rho \models P' \succeq^A \succeq^A Q'$, as needed. In case (ii) from the hypothesis $\rho \models R \succeq^A Q$ we infer $\rho \models P' \succeq^A \succeq^A Q$, as desired. The case $P \longrightarrow P'$ is trivial. The next case is $Q \longrightarrow Q'$. By hypothesis there is R' such that $R \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} R'$ and $\rho \models R' \succeq^A Q'$. Thus there are processes R_1, R_2 and integers m', m'' such that $R \stackrel{A^{m'}}{\Longrightarrow} R_1 \longrightarrow R_2 \stackrel{A^{m''}}{\Longrightarrow} R'$. By Lemma 58 and the hypothesis $P \succeq^A_\rho R$ we infer that there is P' and $n' \geq m'$ such that $P \stackrel{A^{n'}}{\Longrightarrow} P_1$ and $\rho \models P_1 \succeq^A R_1$. Next by this and $R_1 \longrightarrow R_2$ we infer that $P_1 \longrightarrow P_2$ for some P_2 s.t. $\rho \models P_2 \succeq^A R_2$. Finally from Lemma 58 we infer that for some P' and $n'' \geq m''$ we have $P_2 \xrightarrow{A^{n''}} P'$ and $\rho \models P' \succeq^A R'$. From these results we infer $P \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} P'$ and from $\rho \models P' \succeq^A \succeq^A Q'$ we obtain reduction closure. The case $Q \xrightarrow{A} Q'$ follows directly from Lemma 58. Briefly, from the hypothesis above we infer $R \stackrel{A^m}{\Longrightarrow} R'$ for some $m > 0$ and R' s.t. $\rho \models R' \succeq^A Q'$ and in turn $P \stackrel{A^n}{\Longrightarrow} \succeq^A_\rho R'$ for some $n \geq m$, as requested.

The proof that the relation is contextual is analogous to the demonstration of contextuality of $\approx^A \approx^A$ that we have drawn in Proposition 52. □

The following results relate $\succeq^A$ with $\sim^A$ and $\approx^A$.

Proposition 60. *If $\rho \models P \sim^A Q$ then $\rho \models P \succeq^A Q$ and $\rho \models Q \succeq^A P$.*

Proof. Analogous to the proof of Lemma 57.

Proposition 61. *If $\rho \models P \succeq^A Q$ then $\rho \models P \approx^A Q$.*

Proof. Let $\mathscr{R}$ be the relation such that $\rho \models P \mathscr{R} Q$ whenever $\rho \models P \succeq^A Q$. That $\mathscr{R}$ is barb preserving and contextual is straightforward. To see reduction closure assume $P \xrightarrow{A} P'$. By definition of $\mathscr{R}$ we infer that there is Q' such that $Q \xrightarrow{A^=} Q'$ and $\rho \models P' \succeq^A Q'$. We have thus found Q' such that $Q \stackrel{A}{\Longrightarrow} Q'$ and $\rho \models P' \mathscr{R} Q'$, as desired. Next suppose $P \longrightarrow P'$. By definition of $\mathscr{R}$ we infer that there is Q' such that $Q \longrightarrow Q'$ and $\rho \models P' \succeq^A Q'$. Therefore $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q'$ and $\rho \models P' \mathscr{R} Q'$, as desired. Now suppose $Q \longrightarrow Q'$. By definition of $\mathscr{R}$ we infer that there is P' such that $P \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} P'$ and $\rho \models P' \succeq^A Q'$, as desired since by definition this implies $\rho \models P' \mathscr{R} Q'$. Finally let $Q \xrightarrow{A} Q'$. By definition of $\mathscr{R}$ we infer that there is P' such that $P \stackrel{A}{\Longrightarrow} \xrightarrow{A} \stackrel{A}{\Longrightarrow} P'$ and $\rho \models P' \succeq^A Q'$. We have done as from the previous result we infer $P \stackrel{A}{\Longrightarrow} P'$ and $\rho \models P' \mathscr{R} Q'$, as desired.

□

The following corollary summarizes all previous results about the classification of the relations over applied pi calculus processes that we have introduced.

Corollary 62. *The following inclusions hold.*

$$\equiv \subseteq \sim^A \subseteq \succeq^A \subseteq \approx^A \subseteq \cong^{A\pi}$$

Up to techniques

In this section we setup up to techniques (cf. [101, 94]) for administrative equivalence and for behavioral equivalence. The following technique is used to prove Lemmas 73, 81.

Definition 26 (Up to technique). *A term-indexed relation $\mathscr{R}$ is an administrative equivalence up to expansion if it symmetric, strong barb preserving, contextual up to $\succeq^A, \preceq^A$, and s.t. $\rho \models P \mathscr{R} Q$ implies*

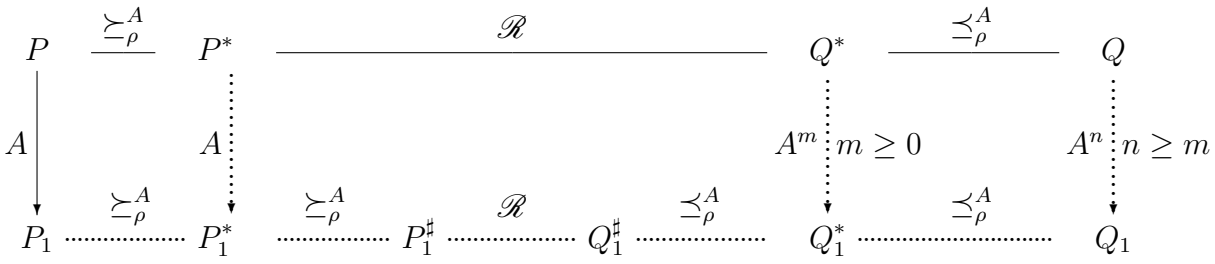
- *$P \xrightarrow{A} P'$ imply $Q \stackrel{A}{\Longrightarrow} Q'$ for some Q' such that $\rho \models P' \succeq^A \ \mathscr{R} \preceq^A Q'$*
- *$P \longrightarrow P'$ imply $Q \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} Q'$ for some Q' such that $\rho \models P' \succeq^A \mathscr{R} \preceq^A Q'$.*

Lemma 63. *If $\mathscr{R}$ is an administrative equivalence up to expansion, then $\mathscr{R} \subseteq \approx^A$.*

Proof. Let $\mathscr{S} = \succeq^A \mathscr{R} \preceq^A$. We need to show that $\mathscr{S}$ is an administrative equivalence. Then by $\mathscr{S} \subseteq \approx^A$ one has $\mathscr{R} \subseteq \approx^A$.

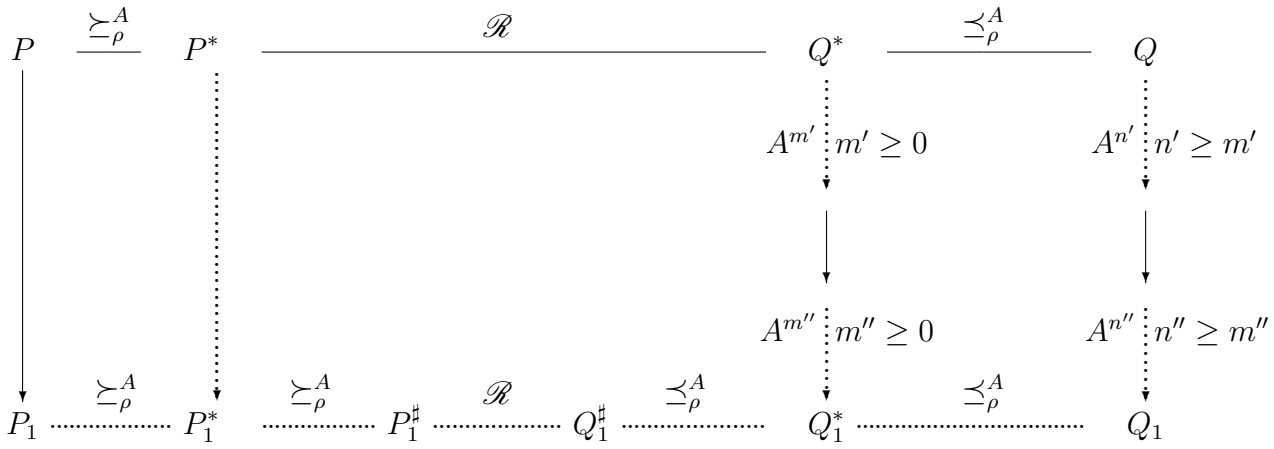
Let $P \succeq^A_\rho P^* \mathscr{R} Q^* \preceq^A_\rho Q$. To see that $\mathscr{S}$ preserve barbs, assume $\rho \models P\downarrow_a$. By definition of $\succeq^A$ we have $\rho \models P^*\downarrow_a$. By definition of $\mathscr{R}$ we have $\rho \models Q^*\downarrow_a$, and by $Q \succeq^A_\rho Q^*$ we infer $\rho \models Q\downarrow_a$. The case $\rho \models Q\downarrow_a$ is analogous.

To see reduction closure assume $P \xrightarrow{A} P_1$. The case whether P^* does not match the move is trivial as in this case $P' \succeq^A_\rho P^* \mathscr{R} Q^* \preceq^A_\rho Q$. Otherwise by the properties of $\succeq^A_\rho$ and $\mathscr{R}$ and by using Lemma 58 we build the following diagram:



From transitivity of $\succeq^A$ (see Prop. 59) we infer that if $\rho \models P \mathscr{S} Q$ and $P \xrightarrow{A} P_1$ there is Q_1 s.t. $Q \xRightarrow{A} Q_1$ and $\rho \models P_1 \mathscr{S} Q_1$. The diagram for $Q \xrightarrow{A} Q_1$ is analogous as $\mathscr{R}$ is symmetric.

It remains to analyze the case $P \longrightarrow P_1$. By the properties of $\succeq^A_\rho$ and $\mathscr{R}$ we build the following diagram; the last inference on the right is obtained exactly by the same steps we did in the proof of transitivity of $\succeq^A$.



From transitivity of $\succeq^A$ and the diagram above we infer that if $\rho \models P \mathscr{S} Q$ and $P \longrightarrow P_1$ there is Q_1 s.t. $Q \xRightarrow{A} \longrightarrow \xRightarrow{A} Q_1$ and $\rho \models P_1 \mathscr{S} Q_1$. The case $Q \longrightarrow Q_1$ is similar.

Last, we prove that $\mathscr{S}$ is contextual. Suppose $\rho \vdash R^*$ and let $R = R^*\rho$. By the contextuality of $\succeq^A$ we infer $\rho \models P \,|\, R \succeq^A P^* \,|\, R$. Similarly $\rho \models Q \,|\, R \succeq^A Q^* \,|\, R$. By the contextuality of $\mathscr{R}$ up to $\succeq^A, \preceq^A$ we have that $\rho \models P^* \,|\, R \succeq^A \mathscr{R} \preceq^A Q^* \,|\, R$. From transitivity of $\succeq^A$ and these results we infer $\rho \models P \,|\, R \,\mathscr{S}\, Q \,|\, R$, as requested.

Now consider a name $n \notin fn(\rho)$. By the contextuality of $\succeq^A$ we have both $\rho, n/x \models P \succeq^A P^*$ and $\rho, n/x \models Q \succeq^A Q^*$. By definition of $\mathscr{R}$ we have $\rho, n/x \models P^* \succeq^A \mathscr{R} \preceq^A Q^*$. By transitivity of $\succeq^A$ and the results above we infer $\rho, n/x \models P \,\mathscr{S}\, Q$, as requested.

Finally from the contextuality of $\succeq^A_\rho$ we infer both $\rho \setminus n \models (\mathsf{new}\ n)P \succeq^A (\mathsf{new}\ n)P^*$ and $\rho \setminus n \models (\mathsf{new}\ n)Q \succeq^A (\mathsf{new}\ n)Q^*$. By definition of $\mathscr{R}$ we have $\rho \setminus n \models (\mathsf{new}\ n)P^* \succeq^A \mathscr{R} \preceq (\mathsf{new}\ n)Q^*$. These results and By transitivity of $\succeq^A$ let us infer $\rho \setminus n \models (\mathsf{new}\ n)P \,\mathscr{S}\, (\mathsf{new}\ n)Q$, as needed. □

The next definition and result will be useful to discard administrative reductions when proving that a relation is included in behavioural equivalence. We use this technique to prove the Theorem of completeness of the translation (see Theorem 95).

Definition 27 (Behavorial equivalence up to administrative equivalence). *A term-indexed relation $\mathscr{R}$ is a behavioral equivalence up to administrative equivalence if it is symmetric, contextual up to $\approx^A$ and such that $\rho \models P \mathscr{R} Q$ implies:*

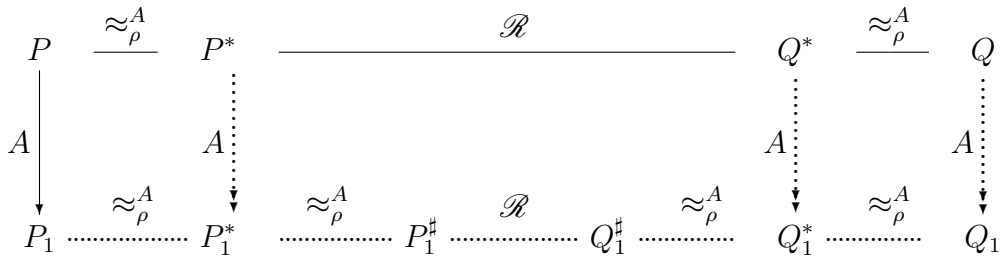
- *if $\rho \models P\downarrow_n$ then $\rho \models Q \Downarrow_n$*
- *$P \stackrel{A}{\Longrightarrow} P'$ implies $Q \stackrel{A}{\Longrightarrow} Q'$ for some Q' such that $\rho \models P' \approx^A \mathscr{R} \approx^A Q'$*
- *$P \stackrel{A}{\Longrightarrow} \longrightarrow \stackrel{A}{\Longrightarrow} P'$ implies $Q \Longrightarrow Q'$ for some Q' such that $\rho \models P' \approx^A \mathscr{R} \approx^A Q'$*

.

Proposition 64. *If $\mathscr{R}$ is a behavioral equivalence up to administrative equivalence then $\mathscr{R} \subseteq \cong^{A\pi}$.*

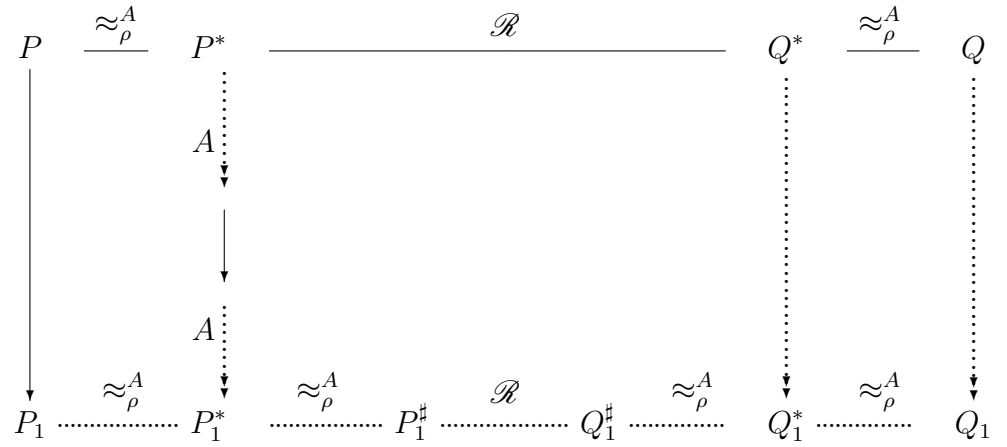
Proof. We prove that $\approx^A \mathscr{R} \approx^A$ is a behavioural equivalence. Let $P \approx^A P^* \mathscr{R} Q^* \approx^A Q$. To see barb preservation assume $\rho \models P\downarrow_a$. By hypothesis $\rho \models P^*\downarrow_a$. By definition of $\mathscr{R}$ we infer $\rho \models Q^* \Downarrow_a$. By applying Lemma 53 to the hypothesis $\rho \models Q^* \approx^A Q$ we obtain $\rho \models Q^* \cong^{A\pi} Q$. We apply Proposition 33 and obtain $\rho \models Q \Downarrow_a$.

Suppose $P \stackrel{\tau}{\longrightarrow} P_1$. We first build the diagram for the case $P \stackrel{A}{\longrightarrow} P_1$; to ease the notation we let $\twoheadrightarrow \triangleq \Longrightarrow$. The last inference on the right is obtained by Lemma 51(i).



From $Q \stackrel{A}{\Longrightarrow} Q'$ we infer $Q \Longrightarrow Q'$ (remember that $\stackrel{A}{\longrightarrow} \in \stackrel{\tau}{\longrightarrow}$).From transitivity of $\approx^A$ stated in Proposition 52 we infer $\rho \models P_1 \approx^A \mathscr{R} \approx^A Q_1$, as desired.

Next we build the following diagram for $P \longrightarrow P_1$. The last inference on the right is inferred by using $\approx^A \subseteq \cong^{A\pi}$, and Proposition 33(ii).



From transitivity of $\approx^A$ we infer $\rho \models P_1 \approx^A \mathscr{R} \approx^A Q_1$, as needed.

To conclude we show contextuality. Assume $\rho \vdash R^*$ and let $R = R^*\rho$. By definition of $\approx^A$ both $\rho \models P \,|\, R \approx^A P^* \,|\, R$ and $\rho \models Q^* \,|\, R \approx^A Q \,|\, R$. By definition of $\mathscr{R}$ we infer $\rho \models P^* \,|\, R \approx^A \mathscr{R} \approx^A Q^* \,|\, R$. By transitivity of $\approx^A$we infer $\rho \models P \,|\, R \approx^A \mathscr{R} \approx^A Q \,|\, R$, as desired. Let $n \notin fn(\rho)$. By the definitions of $\approx^A$ and $\mathscr{R}$ we obtain respectively $\rho, n/x \models P \approx^A P^*$, $\rho, n/x \models Q^* \approx^A Q$ and $\rho, n/x \models P^* \mathscr{R} Q^*$. Thus $\rho, n/x \models P \approx^A \mathscr{R} \approx^A Q$, as needed. In the last case we have $\rho \setminus n \models (\mathsf{new}\ n)P \approx^A (\mathsf{new}\ n)P^*$, $\rho \setminus n \models (\mathsf{new}\ n)Q^* \approx^A (\mathsf{new}\ n)Q$ and $\rho \setminus n \models (\mathsf{new}\ n)P^* \mathscr{R} (\mathsf{new}\ n)Q^*$and we infer $\rho \setminus n \models (\mathsf{new}\ n)P \approx^A \mathscr{R} \approx^A (\mathsf{new}\ n)Q$, as desired.

□

Closure properties of administrative equivalence

The key property of administrative reductions is that they are closed under administrative equivalence, in the following sense.

Proposition 65. *If* $P \xrightarrow{A} P'$, *then* $\rho \models P \approx^{A} P'$.

Proof. Let $\sigma \subseteq \rho$ and let $\tilde{b} \cap fn(\rho) = \emptyset$. We let $\sigma \models C[P] \mathscr{R} C[Q]$ whenever $P \xrightarrow{A} Q$ and $C[-] = (\mathsf{new}\ \tilde{n})R(\rho, \tilde{b}/\tilde{x}) \mid -$ and $\rho, \tilde{b}/\tilde{x} \vdash R$ and $fn(R) = \tilde{n}$. We consider the identity relation Id such that $(P, Q) \in Id$ whenever $P \equiv Q$. We show that $\mathscr{R} \cup Id$ is an administrative equivalence up to administrative expansion. That the identity satisfies Definition 26 follows easily by coinduction and closure of $\xrightarrow{\tau}$ under $\equiv$ (see Proposition 37 in Section 3.2). We show that $\mathscr{R}$ is barb preserving, reduction closed and contextual. To ease the notation we let $S = R(\rho, \tilde{b}/\tilde{x})$. We say that $P \xrightarrow{\tau} Q$ and $P \xrightarrow{\tau} R$ are obtained by non-overlapping redexes if there are C, P', P'' such that $P \equiv C[P' \mid P'']$, and $Q \equiv C[Q' \mid P'']$ and $P' \xrightarrow{\tau} Q'$, and $R \equiv C[P' \mid R'']$ and $P'' \xrightarrow{\tau} R''$.

Barb preservation Let $\rho \models C[P]\downarrow_a$. If $\rho \models C\downarrow_a$ we are done as $\rho \models C[Q]\downarrow_a$. If $\rho \models P\downarrow_a$, by definition of $\xrightarrow{A}$ we have that Q has the same strong barbs: $\rho \models Q\downarrow_a$. We conclude that $\rho \models C[Q]\downarrow_a$.

Reduction closure To see reduction closure, let $C[P] \xrightarrow{\tau} H$. We first tackle the case whenever this reduction is administrative: $C[P] \xrightarrow{A} H$.

We analyze each case of Def. 22 and we show that there is K such that $C[Q] \xRightarrow{A} K$ and (i) $H \xrightarrow{A} K$ or (ii) $H \equiv K$.

1. The first case Def. 22(1) holds whenever $C[P] \xrightarrow{A} H$ is inferred from a replicated input on *net* and a replicated output synchronizing. We have $C[P] \equiv D[!net(\tilde{x}).P' \mid (\mathsf{new}\ \tilde{c})!net\langle\tilde{M}\rangle]$, $H \equiv D[!net(\tilde{x}).P' \mid (\mathsf{new}\ \tilde{c})!net\langle\tilde{M}\rangle \mid P'\{\tilde{M}/\tilde{x}\}]$.

   The administrative reduction $P \xrightarrow{A} Q$ has been inferred from one of the cases outlined in Def. 22. If $P \xrightarrow{A} Q$ belongs to one of the cases Def. 22(4-7) we infer that the reduction occurs on a channel different from *net* and we easily obtain that $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ have been obtained from non-overlapping redexes. Therefore there are D', R such that $D[-] = D'[R \mid -]$ and $C[Q] \equiv D'[R' \mid !net(\tilde{x}).P' \mid (\mathsf{new}\ \tilde{c})!net\langle\tilde{M}\rangle]$ where $R \xrightarrow{A} R'$. We close the diagram by noting that $H \xrightarrow{A} D'[R' \mid !net(\tilde{x}).P' \mid (\mathsf{new}\ \tilde{c})!net\langle\tilde{M}\rangle \mid P'\{\tilde{M}/\tilde{x}\}] \xleftarrow{A} C[Q]$. Indeed we have found $K \triangleq D'[R' \mid !net(\tilde{x}).P' \mid (\mathsf{new}\ \tilde{c})!net\langle\tilde{M}\rangle \mid P'\{\tilde{M}/\tilde{x}\}]$ such that $C[Q] \xrightarrow{A} K$ and $\sigma \models H \mathscr{R} K$.

   If $P \xrightarrow{A} Q$ belongs to the one of the cases Def.22(2-3) we know that this reduction has been inferred from an input redex that is non overlapping with $C[P] \xrightarrow{A} H$; this holds since the inputs involved in case Def.22(2-3) are under replication. On the other hand $P \xrightarrow{A} Q$ may involve the emission $!\,net\langle\tilde{M}\rangle$;

in this case from the semantics of the applied pi calculus, more in detail from the (BANG) rule, we easily infer that there are D', R such that $D[-] \equiv D'[net(\tilde{y}).R \mid -]$ and $C[Q] \equiv (\mathsf{new}\ \tilde{c})D'[R\{\tilde{M}/\tilde{y}\} \mid !net(\tilde{x}).P' \mid !net\langle\tilde{M}\rangle]$. We conclude that $H \xrightarrow{A} (\mathsf{new}\ \tilde{c})D'[R\{\tilde{M}/\tilde{y}\} \mid !net(\tilde{x}).P' \mid !net\langle\tilde{M}\rangle \mid P'\{\tilde{M}/\tilde{x}\}] \xleftarrow{A} C[Q]$, as desired. In case $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ has been obtained by non-overlapping redexes we proceed as above and find K such that $H \xrightarrow{A} K \xleftarrow{A} C[Q]$, as desired.

The remaining case arises whenever $P \xrightarrow{A} Q$ has been inferred from Def. 22(1). The case $H \equiv C[Q]$ is trivial. Assuming $H \not\equiv C[Q]$ then three cases arise: (i) $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ involve the same input prefix or (ii) $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ involve the same output prefix or (iii) $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ are obtained by non-overlapping redexes. Case (ii) is analogous to the case analyzed above where $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ share an input redex. Case (i) is specular; we omit the demonstration that follows the same rationale. Case (iii) is obtained as usual. Thus on both three cases we find K such that $H \xrightarrow{A} K \xleftarrow{A} C[Q]$, as desired.

2. In case $C[P] \xrightarrow{A} H$ has been inferred from Def. 22(2) we have $C[P] \equiv D[\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ R \mid !\, net\langle\{M\}_{sk(N)}\rangle]$ and $H \equiv D[\mathsf{filter}\ \{M\}_N\ \mathsf{with}\ sk(N)\ \mathsf{in}\ R \mid !\, net\langle\{M\}_{sk(N)}\rangle]$ where $|M| = |\tilde{y}|$ and $\forall D', M' : C[P] \equiv D'[net\langle\{M'\}_{sk(N)}\rangle] \wedge |M'| = |\tilde{y}|$ we have $M = M'$. In case $P \xrightarrow{A} Q$ is inferred from one of the cases Def. 22(4-7) we infer that does not have redexes overlapping with $C[P] \xrightarrow{A} H$ as $P \xrightarrow{A} Q$ arises on a channel different from net. We therefore find K such that $H \xrightarrow{A} K \xleftarrow{A} C[Q]$, as desired. In case $P \xrightarrow{A} Q$ has been inferred from Def. 22(1) then or (i) the reduction may involve the redex $!\, net\langle\{M\}_{sk(N)}\rangle$ or (ii) the reduction has not overlapping redexes with $C[P] \xrightarrow{A} H$. This holds since the filter process is not replicated and thus $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ do not share an input redex. In case (i) we infer that there are D', R' such that $D[-] \equiv D'[!net(x).R' \mid -]$ and $C[Q] \equiv D'[R'\{\{M\}_{sk(N)}/x\} \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ R \mid !\, net\langle\{M\}_{sk(N)}\rangle]$. We note that $H \xrightarrow{A} D'[R'\{\{M\}_{sk(N)}/x\} \mid \mathsf{filter}\ \{M\}_N\ \mathsf{with}\ sk(N)\ \mathsf{in}\ R \mid !\, net\langle\{M\}_{sk(N)}\rangle] \xleftarrow{A} C[Q]$, as desired.

In case Def. $P \xrightarrow{A} Q$ is inferred from Def. 22(2) we infer that $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ (i) have an overlapping input redex or (ii) have an overlapping output redex or (iii) do not have overlapping redexes or (iv) have both redexes overlapping. Suppose case (i) holds. In this case we have that there are D', M', N' such that $D[-] \equiv D'[!net\langle\{M'\}_{N'}\rangle \mid -]$ and $C[Q] \equiv D'[\mathsf{filter}\ \{M'\}_{N'}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ R \mid !net\langle\{M'\}_{N'}\rangle \mid !net\langle\{M\}_{sk(N)}\rangle]$. By definition we have that $\mathsf{filter}\ \{M'\}_{N'}\ \mathsf{with}\ sk(N)\ \mathsf{in}\ R \triangleq (\mathsf{new}\ r)\ (\mathsf{decrypt}$

$\{M'\}_{N'}$as $\{\tilde{y}\}_{sk(N)}$in R else $net\langle\{M'\}_{N'}\rangle \,|\, r\langle\rangle)$ $|\, !r.\mathsf{filter}_r\ \tilde{y}$ with $sk(N)$ in R where $\mathsf{filter}_n\ \tilde{y}$ with M in $P \triangleq net(x).\mathsf{decrypt}\ x$ as $\{\tilde{y}\}_M$ in P else $(net\langle x\rangle \,|\, n\langle\rangle)$. Two cases arise corresponding to (a) $N' = sk(N) \wedge |\tilde{M'}| = |\tilde{y}|$ and (b) $N' \neq sk(N) \vee |\tilde{M'}| \neq |\tilde{y}|$. In case (a) by the initial hypothesis of Def. 22(2) we have that $N' = sk(N) \wedge |\tilde{M'}| = |\tilde{y}|$ implies $\tilde{M'} = \tilde{M}$; we easily infer infer that $H \equiv C[Q]$ and we are done. In case (b) we infer that the decryption fails and in turn $\mathsf{filter}\ \{M'\}_{N'}$ with $sk(N)$ in $R \equiv net\langle\{M'\}_{N'}\rangle \,|\, (\mathsf{new}\ r)r\langle\rangle \,|\, !r.\mathsf{filter}_r\ \tilde{y}$ with $sk(N)$ in R. To see that, we note that in case $N' \neq sk(N)$ we obtain the result by inferring $d(\{M'\}_{N'}, sk(N)) \not\rightarrow$ and in turn by the let-else axiom of structural congruence. In case $N' \neq sk(N) \wedge |\tilde{M'}| \neq |\tilde{y}|$ we infer the result by noting that among nested applications of projections there is $i \in 1, \ldots, |\tilde{y}|$ such that $\pi_i(\tilde{M}) \not\rightarrow$; by nested applications of let axiom followed by the let-else axiom we obtain the claim. From the result above we easily obtain that $\mathsf{filter}\ \{M'\}_{N'}$ with $sk(N)$ in $R \xrightarrow{A}$ $\mathsf{filter}\ \tilde{y}$ with $sk(N)$ in $R \,|\, net\langle\{M'\}_{N'}\rangle$ and in turn $C[Q] \xrightarrow{A}\equiv C[P]$; structural congruence is used to let $net\langle\{M'\}_{N'}\rangle$ be absorbed by $!\, net\langle\{M'\}_{N'}\rangle$. We conclude that $C[Q] \stackrel{A}{\Longrightarrow} H$ and we are done as (H, H) is in the identity. In cases (ii),(iii) we proceed as in case (2) and infer that there is K such that $H \xrightarrow{A} K \xleftarrow{A} C[Q]$. In case (iv) we have $H \equiv C[Q]$ and we are done.

In case Def. $P \xrightarrow{A} Q$ is inferred from Def. 22(3) we easily infer that $C[Q] \xrightarrow{A}$ $C[P]$; this move is inferred by executing the reduction that rolls back the reduct to the prompt filter that absorbed a wrong packet. Thus $C[Q] \stackrel{A}{\Longrightarrow} H$ and we are done since (H, H) is in the identity.

3. In case $C[P] \xrightarrow{A} H$ has been inferred from Def. 22(3) we have $C[P] \equiv D[\mathsf{filter}\ \tilde{y}$ with $sk(N)$ in $R \,|\, net\langle M\rangle]$ and $H \equiv D[\mathsf{filter}\ M$ with $sk(N)$ in $R]$ for some term M such that $\neg(M = \{M'\}_N \wedge |M'| = |\tilde{y}|)$. By definition we have $\mathsf{filter}\ M$ with $sk(N)$ in $R \triangleq (\mathsf{new}\ r)(\mathsf{decrypt}\ M$ as $\{\tilde{y}\}_N$ in R else $net\langle M\rangle \,|\, r\langle\rangle) \,|\, !r.\mathsf{filter}_r\ \tilde{y}$ with N in R. As we have seen in the previous case (2) sub-case (i), from the hypotheses on the shape of M we infer $\mathsf{filter}\ M$ with $sk(N)$ in
$R \equiv net\langle M\rangle \,|\, (\mathsf{new}\ r)r\langle\rangle \,|\, !r.\mathsf{filter}_r\ \tilde{y}$ with $sk(N)$ in R. We ignore the administrative reduction that rolls back the filter with administrative expansion:
$\mathsf{filter}\ M$ with $sk(N)$ in $R \succeq^A_\sigma \mathsf{filter}\ \tilde{y}$ with $sk(N)$ in R. From this we infer $H \succeq^A_\sigma C[P]$. We do not need to match this move as $\sigma \models H \succeq^A C[P]\mathscr{R}C[Q]$.

4. In case $C[P] \xrightarrow{A} H$ has been inferred from Def. 22(4) we have $C[P] \equiv D[\mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n@T$ in $R \,|\, n\langle M\rangle]$ and $H \equiv D[\mathsf{filter}\ M\ \mathsf{from}\ n@T$ in $R]$ and M has type $S \not<: T$. By definition $\mathsf{filter}\ M\ \mathsf{from}\ n@T$ in $R \triangleq$ $(\mathsf{new}\ r)\ \mathsf{if}\ WF(M,t)\ \mathsf{then}\ (\mathsf{let}\ \tilde{y} = \Uparrow((\tilde{x},t))\ \mathsf{in}\ P)\ \mathsf{else}\ (n\langle M\rangle \,|\, r\langle\rangle)$ $|\, !r.\mathsf{filter}_r\ \tilde{y}\ \mathsf{from}\ n@T$ in R where $\mathsf{filter}_n\ \tilde{y}\ \mathsf{from}\ c@T$ in $P \triangleq$ $c(\tilde{x}).\mathsf{if}\ WF(\tilde{x},T)\ \mathsf{then}\ (\mathsf{let}\ \tilde{y} = \Uparrow((\tilde{x},T))\ \mathsf{in}\ P)\ \mathsf{else}\ (c\langle\tilde{x}\rangle \,|\, n\langle\rangle)$. As we have

introduced in Section 4.2 the meta-process
$\mathsf{if}\ WF(\tilde{u},T)\ \mathsf{then}\ P\ \mathsf{else}\ Q$ is implemented by nested applications of equality and projection destructors and succeeds whenever $\tilde{u}$ is certified and has type $S <: T$. By the hypothesis $S \not<: T$ we infer that the process is equivalent (due to a let-else axiom of structural congruence) to the else branch; from this we infer $\mathsf{filter}\ M\ \mathsf{from}\ n@T\ \mathsf{in}\ R \equiv n\langle M\rangle \,|\, (\mathsf{new}\ r)r\langle\rangle \,|\, !r.\mathsf{filter}_r\ \tilde{y}\ \mathsf{from}\ n@T\ \mathsf{in}\ R$. We ignore the reduction $\mathsf{filter}\ M\ \mathsf{from}\ n@T\ \mathsf{in}\ R \xrightarrow{A} \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n@T\ \mathsf{in}\ R$ and infer $\mathsf{filter}\ M\ \mathsf{from}\ n@T\ \mathsf{in}\ R \succeq^A_\sigma \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n@T\ \mathsf{in}\ R$ and in turn $H \succeq^A_\sigma C[P]$. We not need to match this move as $\sigma \models H \succeq^A C[P]\mathscr{R}C[Q]$.

5. In case $C[P] \xrightarrow{A} H$ has been inferred from Def. 22(5) we have $C[P] \equiv D[(\mathsf{new}\ n^*)n^*\langle N\rangle \,|_{i\in n}\mathsf{if}\ N_i \in Set_{n^*}\ \mathsf{then}\ R_i]$ and $H \equiv D[(\mathsf{new}\ n^*)(\mathsf{if}\ N_j \notin N\ \mathsf{then}\ n^*\langle N_j :: N\rangle \,|\, R_j\ \mathsf{else}\ n^*\langle N\rangle) \,|_{i\in I\setminus j}\mathsf{if}\ N_i \notin Set_{n^*}\ \mathsf{then}\ R_i]$ and $\forall i,j \in I\ .\ (N_i = N_j) \Rightarrow (R_i \equiv R_j)$. If $P \xrightarrow{A} Q$ has been inferred from one of the cases of Def. 22(1-3) then the reduction occurred on the free channel net and in turn $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ have no overlapping redexes. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(4,6,7) a case analysis shows that the bound channel on which the reduction $P \xrightarrow{A} Q$ occurs is different by n^* and in turn $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ have no overlapping redexes. The remaining case arises whenever $P \xrightarrow{A} Q$ has been inferred from Def. 22(5). We have that there is $k \in 1,\ldots,n$ such that $C[K] \equiv D[(\mathsf{new}\ n^*)(\mathsf{if}\ N_k \notin N\ \mathsf{then}\ n^*\langle N_k :: N\rangle \,|\, R_k\ \mathsf{else}\ n^*\langle N\rangle) \,|_{i\in I\setminus k}\mathsf{if}\ N_k \notin Set_{n^*}\ \mathsf{then}\ R_k]$. There are two cases corresponding to **(a)** $(N_j = N_k)$ or **(b)** $(N_j \neq N_k)$. In case (a) from the hypothesis above we infer $R_j \equiv R_k$. Now two sub-cases arise corresponding to (i) $N_j \in N$ or (ii) $N_j \notin N$. In case (i) by definition of the meta-process $\mathsf{if}\ M \notin N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ there is a nested application of axioms of structural congruence let and let-else followed by a let such that both $\mathsf{if}\ N_j \notin N\ \mathsf{then}\ n^*\langle N_j :: N\rangle \,|\, R_j\ \mathsf{else}\ n^*\langle N\rangle \equiv n^*\langle N\rangle$ and $\mathsf{if}\ N_k \notin N\ \mathsf{then}\ n^*\langle N_k :: N\rangle \,|\, R_k\ \mathsf{else}\ n^*\langle N\rangle \equiv n^*\langle N\rangle$. By using again the same chain of applications of let and let-else axioms we obtain: $H \xrightarrow{A} D[(\mathsf{new}\ n^*)(n^*\langle N\rangle \,|_{i\in I\setminus j,k}\mathsf{if}\ N_i \notin Set_{n^*}\ \mathsf{then}\ R_i] \xleftarrow{A} C[Q]$, as desired. In sub-case (ii) we infer that there is a nested application of axioms of structural congruence let and let-else followed by a let-else such that both $\mathsf{if}\ N_j \notin N\ \mathsf{then}\ n^*\langle N_j :: N\rangle \,|\, R_j\ \mathsf{else}\ n^*\langle N\rangle \equiv n^*\langle N_j :: N\rangle \,|\, R_j$ and $\mathsf{if}\ N_k \notin N\ \mathsf{then}\ n^*\langle N_k :: N\rangle \,|\, R_k\ \mathsf{else}\ n^*\langle N\rangle \equiv n^*\langle N_k :: N\rangle \,|\, R_k$. We then infer that $H \xrightarrow{A} H' \triangleq D[(\mathsf{new}\ n^*)(R_j \,|\, \mathsf{if}\ N_k \in N_j :: N\ \mathsf{then}\ n^*\langle N_j :: N\rangle n^*\langle N_k :: N_j :: N\rangle \,|\, R_k) \,|_{i\in I\setminus j,k}\mathsf{if}\ N_i \notin Set_{n^*}\ \mathsf{then}\ R_i]$. Since $N_j = N_k$ we proceed as in case (a) and obtain $H' \equiv D[(\mathsf{new}\ n^*)(R_j \,|\, n^*\langle N_j :: N\rangle \,|_{i\in I\setminus j,k}\mathsf{if}\ N_i \notin Set_{n^*}\ \mathsf{then}\ R_i]$. Similarly from $N_j = N_k$ we obtain $C[Q] \xrightarrow{A}\equiv D[(\mathsf{new}\ n^*)(R_k \,|\, n^*\langle N_j :: N\rangle \,|_{i\in I\setminus j,k}\mathsf{if}\ N_i \notin Set_{n^*}\ \mathsf{then}\ R_i]$. We glue the results and by using $R_j \equiv R_k$ we infer $H \xrightarrow{A}\xleftarrow{A} C[Q]$, as desired.

Now suppose case (b) arises and let $H \xrightarrow{A} K$ and $C[Q] \xrightarrow{A} K'$ be the reductions inferred respectively from $\mathsf{if}\ N_k \in Set_{n^*}\ \mathsf{then}\ R_k$ receiving the list of nonces carried on by n^* in H (that may be N or $N_j :: N$) and from $\mathsf{if}\ N_j \in Set_{n^*}\ \mathsf{then}\ R_j$ receiving the nonce set carried on by n^* in $C[Q]$ (that may be N or $N_k :: N$). Since $N_j \neq N_k$ we note that the shape of K, K' depends only on N; from this and by proceeding as in case (a) we easily infer $K \equiv K'$ and we are done.

6. In case $C[P] \xrightarrow{A} H$ has been inferred from Def. 22(6) we have $C[P] \equiv D[(\mathsf{new}\ t)t\langle M\rangle \,|_{i\in n} T_{N_i}(R_i)]$ and $H \equiv D[(\mathsf{new}\ t)\mathsf{let}\ \tilde{y} = ?(N_j, M)\ \mathsf{in}\ t\langle M\rangle \,|\, R_j\ \mathsf{else}\ ((\mathsf{new}\ n)t\langle(N_j;\underline{n}) :: (n_h;\underline{n}) :: M\rangle \,|\, R_j\{\underline{n}/\tilde{y}\}) \,|_{i\in n\wedge i\neq j} T_{N_i}(R_i)]$. If $P \xrightarrow{A} Q$ has been inferred from one of the cases of Def. 22(1-5,7) then for reasons that are analogous to those of case (5) we infer that the reduction occurs on channel different from t. From this we deduce that $C[P] \xrightarrow{A} H$ and $P \xrightarrow{A} Q$ have no overlapping redexes. Let $P \xrightarrow{A} Q$ be inferred from Def. 22(6). We have $C[Q] \equiv D[(\mathsf{new}\ t)\mathsf{let}\ \tilde{y} = ?(N_k, M)\ \mathsf{in}\ t\langle M\rangle \,|\, R_k\ \mathsf{else}\ ((\mathsf{new}\ n)t\langle(N_k;\underline{n}) :: (n_h;\underline{n}) :: M\rangle \,|\, R_k\{\underline{n}/\tilde{y}\}) \,|_{i\in n\wedge i\neq k} T_{N_i}(R_i)]$. We use the notation $N \mapsto_M \tilde{N}$ whenever $M = M_1 :: (N, \tilde{N}) :: M_2$ and M_1, M_2 are list of couples (or tables); we write $N \not\mapsto_M$ whenever M does not have the shape above. There are four cases corresponding to **(a)** $N_k \mapsto_M \tilde{O} \wedge N_j \mapsto_M \tilde{O}$ or **(b)** $N_k \mapsto_M \tilde{O} \wedge N_j \not\mapsto_M$ or **(c)** $N_j \mapsto_M \tilde{O}' \wedge N_k \not\mapsto_M$ or **(d)** $N_j \not\mapsto_M \wedge N_k \not\mapsto_M$. Suppose case (a) holds. By nested applications of let and let-else axioms we obtain both $C[Q] \equiv D[(\mathsf{new}\ t)R_k\{\tilde{O}/\tilde{y}\} \,|\, t\langle(M\rangle \,|_{i\in n\wedge i\neq k} T_{N_i}(R_i)]$ and for the same reasons we outlined above we have $H \equiv D[(\mathsf{new}\ t)R_j\{\tilde{O}'/\tilde{y}\} \,|\, t\langle M\rangle \,|_{i\in n\wedge i\neq j} T_{N_i}(R_i)]$. We note that $H \xrightarrow{A} D[(\mathsf{new}\ t)R_j\{\tilde{O}'/\tilde{y}\} \,|\, R_k\{\tilde{O}/\tilde{y}\} \,|\, t\langle M\rangle \,|_{i\in n\wedge i\neq j,k} T_{N_i}(R_i)] \xleftarrow{A} C[Q]$. In case (b) we have $H \equiv D[(\mathsf{new}\ n,t)Chan_n \,|\, R_j\{\underline{n}/\tilde{y}\} \,|\, t\langle(N_j,\underline{n}) :: (n_h,\underline{n}) :: M\rangle \,|_{i\in n\wedge i\neq j} T_{N_i}(R_i)]$ and $C[Q] \equiv D[(\mathsf{new}\ t)R_k\{\tilde{O}/\tilde{y}\} \,|\, t\langle(M\rangle \,|_{i\in n\wedge i\neq k} T_{N_i}(R_i)]$. We conclude by noting that $H \xrightarrow{A} D[(\mathsf{new}\ n,t)Chan_n \,|\, R_k\{\tilde{O}/\tilde{y}\} \,|\, R_j\{\underline{n}/\tilde{y}\} \,|\, t\langle(N_j,\underline{n}) :: (n_h,\underline{n}) :: M\rangle \,|_{i\in n\wedge i\neq j,k} T_{N_i}(R_i)]$. Case (c) is specular to case (b). Suppose case (d) holds. If $N_j \neq N_k$ we proceed similar to the previous cases and find K such that $H \xrightarrow{A} K \xleftarrow{A} C[Q]$. Otherwise assume $N_j = N_k$. We have $C[Q] \equiv D[(\mathsf{new}\ n,t)Chan_n \,|\, R_k\{\underline{n}/\tilde{y}\} \,|\, t\langle(N_j,\underline{n}) :: (n_h,\underline{n}) :: M\rangle \,|_{i\in n\wedge i\neq k} T_{N_i}(R_i)]$ and $H \equiv D[(\mathsf{new}\ m,t)Chan_m \,|\, R_k\{\underline{m}/\tilde{y}\} \,|\, t\langle(N_j,\underline{m}) :: (m_h,\underline{m}) :: M\rangle \,|_{i\in n\wedge i\neq j} T_{N_i}(R_i)]$ We note that $C[Q] \xrightarrow{A} K' \triangleq D[(\mathsf{new}\ n,t)Chan_n \,|\, R_k\{\underline{n}/\tilde{y}\} \,|\, R_j\{\underline{n}/\tilde{y}\} \,|\, t\langle(N_j,\underline{n}) :: (n_h,\underline{n}) :: M\rangle \,|_{i\in n\wedge i\neq j,k} T_{N_i}(R_i)]$ and $H \xrightarrow{A} K \triangleq D[(\mathsf{new}\ m,t)Chan_m \,|\, R_k\{\underline{m}/\tilde{y}\} \,|\, R_j\{\underline{m}/\tilde{y}\} \,|\, t\langle(N_j,\underline{m}) :: (m_h,\underline{m}) :: M\rangle \,|_{i\in n\wedge i\neq j,k} T_{N_i}(R_i)]$.

By alpha-renaming of bound names n, m respectively in K' and K and by

$N_j = N_k$ we obtain $K' \equiv K$, as requested.

7. If $C[P] \xrightarrow{A} H$ has been inferred from Def. 22(7) we have $C[P] \equiv D[(\mathsf{new}\ n)n\langle\rangle \mid !n(x).R\{n\langle\rangle/X\}]$ and $H \equiv D[\mathsf{rec}\, X.R]$ where $n \notin fn(R)$. By the hypotheses above and $P \xrightarrow{A} Q$ we infer that or (a) $C[Q] \equiv H$ or (b) $C[Q] \equiv D'[(\mathsf{new}\ n)n\langle\rangle \mid !n(x).R\{n\langle\rangle/X\}]$ for some D' such that $D \xrightarrow{A} D'$. We note that $H \xrightarrow{A} D'[\mathsf{rec}\, X.R] \xleftarrow{A} C[Q]$; we have thus found $K \triangleq D'[\mathsf{rec}\, X.R]$ such that $C[Q] \xrightarrow{A} K$ and $\rho \models H \mathscr{R} K$.

Next we analyze the case $C[P] \longrightarrow H$.

1. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(1) then $P \equiv D[!net(x).R \mid (\mathsf{new}\ \tilde{c})\ !net\langle M\rangle]$ and $Q \equiv D[!net(x).R \mid (\mathsf{new}\ \tilde{c})!net\langle M\rangle \mid R\{M/x\}]$. Three cases arise corresponding to **(a)** sharing the input redex or **(b)** sharing the output redex or **(c)** have not overlapping redexes. The case whether the two reductions share both redexes does not hold since this would imply $C[P] \xrightarrow{A} H$ that is a contradiction as $\longrightarrow \triangleq \xrightarrow{\tau} \setminus \xrightarrow{A}$. In case (a) we have that there is E and N such that $C[D[-]] \equiv E[net\langle N\rangle \mid -]$ and $H \equiv E[R\{N/x\} \mid !net(x).R \mid (\mathsf{new}\ \tilde{c})!net\langle M\rangle]$. We obtain reduction closure by noting that $H \xrightarrow{A} E[R\{N/x\} \mid R\{M/x\} \mid !net(x).R \mid (\mathsf{new}\ \tilde{c})!net\langle M\rangle] \longleftarrow C[Q]$. In case (b) have that there is E and T such that $C[D[-]] \equiv E[net(x).T \mid -]$ and $H \equiv E[(\mathsf{new}\ \tilde{c})T\{M/x\} \mid !net(x).R \mid !net\langle M\rangle]$. We obtain reduction closure by noting that $H \xrightarrow{A} E[(\mathsf{new}\ \tilde{c})T\{M/x\} \mid R\{M/x\} \mid !net(x).R \mid !net\langle M\rangle] \longleftarrow C[Q]$. In case (c) it must be $C[D] \longrightarrow E$ for some E such that $H \equiv E[!net(x).R \mid (\mathsf{new}\ \tilde{c})!net\langle M\rangle]$ and in turn $H \xrightarrow{A} E[(\mathsf{new}\ \tilde{c})R\{M/x\} \mid !net(x).R \mid !net\langle M\rangle] \longleftarrow C[Q]$, as desired.

2. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(2) we have that $C[P] \equiv D[\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ N\ \mathsf{in}\ R \mid !\, net\langle\{M\}_N\rangle]$ and $C[Q] \equiv D[\mathsf{filter}\ \{M\}_N\ \mathsf{with}\ N\ \mathsf{in}\ R \mid !\, net\langle\{M\}_N\rangle]$ where $|M| = |\tilde{y}|$ and $\forall C', M' : D[-] \equiv C'[net\langle\{M'\}_N\rangle \mid -] \wedge |M'| = |\tilde{y}|$ we have that $M = M'$. Two cases arise corresponding to $C[P] \longrightarrow H$ and $P \xrightarrow{A} Q$ **(a)** sharing the output redex or **(b)** have not overlapping redexes. Notice the two reductions cannot share both redexes as this would imply $C[P] \xrightarrow{A} H$, contradiction. Moreover they cannot share the input redex; we show this by contradiction. Suppose $P \xrightarrow{A} Q$ and $C[P] \xrightarrow{A} H$ share the input redex. Then for some D' and M^* we have that $D[-] \equiv D'[net\langle M^*\rangle \mid -]$ and $H \equiv D'[\mathsf{filter}\ M^*\ \mathsf{with}\ N\ \mathsf{in}\ R \mid !\, net\langle\{M\}_N\rangle]$. Two cases arise corresponding to (i) $(M^* = \{M'\}_N \wedge |\tilde{y}| = |M'|)$ or (ii) not. Suppose (i) holds; by the initial hypothesis we know $M' = M$. By the replication axiom of structural congruence we have $C[P] \equiv D'[\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ N\ \mathsf{in}\ R \mid !\, net\langle\{M\}_N\rangle]$ and $H \equiv D'[\mathsf{filter}\ \{M\}_N\ \mathsf{with}\ N\ \mathsf{in}\ R \mid !\, net\langle\{M\}_N\rangle]$, contradiction, as by definition

Def. 22(1) this implies $C[P] \xrightarrow{A} H$. Now suppose (ii) holds. Therefore the filtering is unsuccessful and by Def. 22(3) we infer $C[P] \xrightarrow{A} H$, contradiction.

Cases (a),(b) are treated similarly to as we did respectively in case (1), subcases (b),(c); in both cases this let us find K such that $H \xrightarrow{A} K \longleftarrow C[Q]$, as desired.

3. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(3) we have that $P \equiv D[\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ N\ \mathsf{in}\ R \,|\, net\langle M^* \rangle]$ and $Q \equiv D[\mathsf{filter}\ M\ \mathsf{with}\ N\ \mathsf{in}\ R]$ and $\neg(M^* = \{M\}_N \wedge |M| = |\tilde{y}|)$. From this we infer that by nested application of let and let-else axioms of structural congruence we have $\mathsf{filter}\ M\ \mathsf{with}\ N\ \mathsf{in}\ R \equiv net\langle M \rangle \,|\, (\mathsf{new}\ r) r\langle\rangle \,|\, !r.\mathsf{filter}_r\ \tilde{y}\ \mathsf{with}\ N\ \mathsf{in}\ R$ and in turn $Q \xrightarrow{A} P$. This let us infer that $C[Q] \xRightarrow{A} H$ and since the couple (H, H) is in the identity we are done.

4. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(4) we have that we have $P \equiv D[(\mathsf{new}\ n^\circ)P' \,|\, \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n@T\ \mathsf{in}\ R \,|\, n\langle M \rangle]$ and $Q \equiv D[(\mathsf{new}\ n^\circ)P' \,|\, \mathsf{filter}\ M\ \mathsf{from}\ n@T\ \mathsf{in}\ R]$ and M has type $S \not<: T$. By this hypothesis and applying let and let-else axioms of structural congruence we obtain $\mathsf{filter}\ M\ \mathsf{from}\ n@T\ \mathsf{in}\ Q \equiv net\langle M \rangle \,|\, (\mathsf{new}\ r) r\langle\rangle \,|\, !r.\mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n@T\ \mathsf{in}\ R$. From $\mathsf{filter}\ M\ \mathsf{from}\ n@T\ \mathsf{in}\ Q \xrightarrow{A} net\langle M \rangle \,|\, \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n@T\ \mathsf{in}\ R$ we infer $Q \xrightarrow{A} P$ and in turn $C[Q] \xRightarrow{A} H$, as desired since the couple (H, H) is in the identity.

5. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(5) then $P \equiv D[(\mathsf{new}\ n^*)n^*\langle N \rangle \,|_{i \in I} \mathsf{if}\ N_i \in Set_{n^*}\ \mathsf{then}\ Q_i]$ and $Q \equiv D[(\mathsf{new}\ n^*)\mathsf{if}\ N_j \notin N\ \mathsf{then}\ n^*\langle N_j :: N \rangle \,|\, Q_j\ \mathsf{else}\ n^*\langle N \rangle) \,|_{i \in I \setminus j} \mathsf{if}\ N_i \notin Set_{n^*}\ \mathsf{then}\ Q_i]$ where $\forall i, j \in I\ .\ (N_i = N_j) \Rightarrow (Q_i \equiv Q_j)$. From this we infer that $P \xrightarrow{A} Q$ and $C[P] \longrightarrow H$ have not overlapping redexes. We show this by contradiction. Suppose $P \xrightarrow{A} Q$ and $C[P] \longrightarrow H$ have an overlapping redex. Since redexes involved by $P \xrightarrow{A} Q$ communicate on channel n^* we infer that the reduction $C[P] \longrightarrow H$ occurred on n^*. By shape analysis we easily see that this implies that there is k such that $H \equiv C[D[(\mathsf{new}\ n^*)n^*\langle N \rangle \mathsf{if}\ N_k \notin N\ \mathsf{then}\ n^*\langle N_k :: N \rangle \,|\, Q_k\ \mathsf{else}\ n^*\langle N \rangle) \,|_{i \in I \setminus k} \mathsf{if}\ N_i \in Set_{n^*}\ \mathsf{then}\ Q_i]$. By hypothesis we have that $\forall i, j \in I\ .\ (N_i = N_j) \Rightarrow (Q_i \equiv Q_j)$ and in turn by Def. 22(5) we obtain $C[P] \xrightarrow{A} H$, contradiction. Since $P \xrightarrow{A} Q$ and $C[P] \longrightarrow H$ have not overlapping redexes we easily find K such that $H \xrightarrow{A} K \longleftarrow C[Q]$, as desired.

6. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(6) we have that $P \equiv D[(\mathsf{new}\ t)t\langle M \rangle \,|_{i \in I} T_{N_i}(Q_i)]$ and $Q \equiv D[(\mathsf{new}\ t)\mathsf{let}\ \tilde{y} = ?(N_{jID}, M)\ \mathsf{in}\ t\langle M \rangle \,|\, Q_j\ \mathsf{else}\ ((\mathsf{new}\ n)Chan_n \,|\, t\langle (N_{jID}, \underline{n}) :: (n_{ID}, \underline{n}) :: M \rangle \,|\, Q_j\{\underline{n}/\tilde{y}\}) \,|_{i \in I \setminus j} T_{N_i}(Q_i)]$. As

in the previous case (5) we infer that $P \xrightarrow{A} Q$ and $C[P] \longrightarrow H$ have not overlapping redexes. This can be shown by contradiction; indeed a shape analysis shows that all reductions on t are administrative. As in previous cases we easily find K such that $H \xrightarrow{A} K \longleftarrow C[Q]$, as desired.

7. If $P \xrightarrow{A} Q$ has been inferred from Def. 22(7) we have that $P \equiv D[(\mathsf{new}\ n)n\langle\rangle \mid !n(x).Q\{n\langle\rangle/X\}]$ and $Q \equiv D[\mathsf{rec}\, X.Q]$ where $n \notin fn(Q)$ By shape analysis we infer that $P \xrightarrow{A} Q$ and $C[P] \longrightarrow H$ have not overlapping redexes and find K such that $H \xrightarrow{A} K \longleftarrow C[Q]$, as desired.

Contextuality Let $\sigma \models C[P]\mathscr{R}C[Q]$ where $C[-] = (\mathsf{new}\ \tilde{n})R(\rho, \tilde{b}/\tilde{x}) \mid -$ and $\rho, \tilde{b}/\tilde{x} \vdash R$ and $fn(R) = \tilde{n}$. The first clause of contextuality requires that whenever $\sigma \vdash T$ then $\sigma \models T\sigma \mid C[P]\mathscr{R}C[Q] \mid T\sigma$. We let $D[-] = (\mathsf{new}\ \tilde{n})(R \mid T)(\rho, \tilde{b}/\tilde{x}) \mid -$ and infer $\sigma \models D[P]\mathscr{R}D[Q]$; by application of scope and composition rules for structural congruence we obtain the desired result. For the second clause, let $n \notin fn(\sigma)$. Since $\sigma \models C[P]\mathscr{R}C[Q]$ and $C[-]$ is built around $R(\rho, \tilde{b}/\tilde{x})$ we infer that $(fn(R(\rho, \tilde{b}/\tilde{x})) \setminus fn(\sigma)) \subseteq \tilde{n}$. Therefore assume without loss of generality that $n \notin fn(\rho)$. From the hypothesis $\sigma \subseteq \rho$ we infer $\sigma, n/x \subseteq \rho, n/x$. We let $D[-] \equiv (\mathsf{new}\ \tilde{n})R(\rho, n/x, \tilde{b}/\tilde{x},) \mid -$ with $\{\tilde{b}, n\} \cap fn(\rho) = \emptyset$ and infer $\rho, n/x \models D[P]\mathscr{R}D[Q]$, as desired. For the last clause, from $\sigma \subseteq \rho$ we infer $\sigma \setminus n \subseteq \rho$ and in turn $\sigma \setminus n \models D[P]\mathscr{R}D[Q]$ where $D[-] \triangleq (\mathsf{new}\ \tilde{n}, n)R(\rho, \tilde{b}/\tilde{x}) \mid -$.

□

The following corollary extends the previous result to an arbitrary number of administrative reductions.

Corollary 66. *If $P \xRightarrow{A} Q$, then $\rho \models P \approx^A Q$.*

Proof. We proceed by induction on the number n of reductions $P \xRightarrow{A} Q$, noted $P \xRightarrow{A^n} Q$. The case $n = 0$ follows straightforwardly by reflexivity of $\approx^A$. Suppose $P \xRightarrow{A^n} P^* \xrightarrow{A} Q$. By I.H. we have that $P \approx^A_\rho P^*$. We apply Prop. 65 to $P^* \xrightarrow{A} Q$ obtaining $P^* \approx^A_\rho Q$. We build the chain:$P \approx^A_\rho P^* \approx^A_\rho Q$, and by transitivity of $\approx^A$ we obtain $P \approx^A_\rho Q$.

5.3 Lemmas on symmetric and asymmetric cryptographic schemes

We first present a lemma saying that a ciphertext encrypted with a symmetric cryptoscheme cannot be distinguished from noise without knowledge of the symmetric key. In the next sections we will use this result to treat leftovers of protocol's private communications as noise. We first need some technical results.

Lemma 67. *Let $\sigma = (\rho, \{M\}_N/x)$, $\varsigma = (\rho, \{M'\}_{N'}/x)$ where $fn(N) \not\subseteq fn(\rho) \wedge fn(N') \not\subseteq fn(\rho)$. We have that there are $\tilde{M}$ closed by $A(\rho), \{M\}_N/x, A(\rho), \{M'\}_{N'}/x$ such that $A(\sigma) = A(\rho), \tilde{N}/\tilde{x}$ and $\tilde{N} = \tilde{M}(A(\rho), \{M\}_N/x)$ and $A(\varsigma) = A(\rho), \tilde{N}'/\tilde{x}$ and $\tilde{N}' = \tilde{M}(A(\rho), \{M\}_{N'}/x)$.*

Proof. First notice that the packets $\{M\}_N, \{M'\}_{N'}$ cannot be destructed, since both N and N' contain some name that is not in the free names of ρ. Formally, we have that the defining equations to open these packets have the form $d(N_1, N_2) \doteq N^*$ in Table 3.1 (3.5),(3.6), (3.7). Since $fn(\rho) = fn(A(\rho))$, we cannot find N_2 such that $N_2 A(\rho) = N$ or $N_2 A(\rho) = N'$, and the claim follows. Therefore each N_i/x_i is obtained by applying σ to some definition in Table 3.1 different from those above, i.e. the message $\{M\}_N$ is used as a normal form. We build each N_i' correspondent to N_i by applying ζ to the correspondent defining equation. □

Lemma 68. *Let $\rho_k = (\rho, \{M\}_{sk(k)}/x)$ and $\rho_n = (\rho, \{N\}_n/x)$ where ρ is a substitution such that $\{n, k\} \cap fn(\rho) = \emptyset$. Let $A(\rho_k) \vdash P$ and $A(\rho_n) \vdash P$. The following hold.*

(a) If $PA(\rho_k) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$ then there are P' and $\tilde{M}'$ closed by $A(\rho_k)$ such that $R \equiv P'A(\rho_k)$ and $\tilde{M} = \tilde{M}'A(\rho_k)$ and $PA(\rho_n) \xrightarrow{(\tilde{b}')a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$; the symmetric result holds for the hypothesis $PA(\rho_n) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$;

(b) If $PA(\rho_k) \xrightarrow{a(\tilde{M})} R$ then there is P' closed by $A(\rho_k, \tilde{M}/\tilde{x})$ such that $R \equiv P'A(\rho_k, \tilde{M}/\tilde{x})$ and $PA(\rho_n) \xrightarrow{a(\tilde{M})} P'A(\rho_n, \tilde{M}/\tilde{x})$; the symmetric result holds for the hypothesis $PA(\rho_n) \xrightarrow{a(\tilde{M})} R$;

(c) If $PA(\rho_k) \xrightarrow{\tau} R$ then there is P' closed by $A(\rho_k)$ such that $R \equiv P'A(\rho_k)$ and $PA(\rho_n) \xrightarrow{\tau} P'A(\rho_n)$; the symmetric result holds for the hypothesis $PA(\rho_n) \xrightarrow{\tau} R$.

Proof. We proceed by induction on the last rule of the inference (see Table 3.2) and prove the left to right direction of each clause. The vice versa direction is analogous. We first prove clause (a).

(OUT) In this case the transition has the form $u\langle\tilde{v}\rangle A(\rho_k) \xrightarrow{u(\tilde{v})A(\rho_k)} \mathbf{0}$. We apply Lemma 68 and we trivially obtain $u\langle\tilde{v}\rangle A(\rho_n) \xrightarrow{u\langle\tilde{v}A(\rho_n)\rangle} \mathbf{0}$.

(OPEN) In this case the transition has the form $(\mathsf{new}\ c)PA(\rho_k) \xrightarrow{(\tilde{b},c)a\langle\tilde{M}\rangle} R$ and has been inferred from $PA(\rho_k) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$. The inductive hypothesis is that there are $P', \tilde{M}'$ closed by $A(\rho_k)$ such that $\tilde{M} = \tilde{M}'A(\rho_k)$ and $R \equiv P'A(\rho_k)$ and

$PA(\rho_n) \xrightarrow{(\tilde{b}')a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$. We apply (OPEN) and infer $(\mathsf{new}\ c)PA(\rho_n) \xrightarrow{(\tilde{b}',c)a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$, as desired.

(LET) In this case the transition has the form $(\mathsf{let}\ y = d(\tilde{N})\ \mathsf{in}\ P\ \mathsf{else}\ Q)A(\rho_k) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$ and has been inferred from $d(\tilde{N}A(\rho_k)) \to N'$ and $P\{N'/y\} \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$. By the definition of analysis we easily infer that there is N closed by $A(\rho_k)$ such that $N' = NA(\rho_k)$. By definition of analysis, we have that $N' \in A(\rho_k)$. We apply Lemma 67 and infer that two cases arise: (a) $N' \in A(\rho)$ or (b) N is closed by $A(\rho), \{M\}_{sk(k)}/x, A(\rho), \{n\}_n/x$ and $N' = NA(\rho), \{M\}_{sk(k)}/x$. In case (a) from $A(\rho) \subseteq A(\rho_n)$ we trivially obtain the desired result, $d(\tilde{N}A(\rho_n)) \to NA(\rho_n)$. In case (b) we apply Lemma 67 and infer that the entry $N'/z \in A(\rho_k)$ needs to be matched in $A(\rho_n)$ by the entry $N(A(\rho), \{n\}_n/x)/z$. Therefore $d(\tilde{N}A(\rho_n)) \to NA(\rho), \{n\}_n/x$, as desired. By induction we have $\tilde{M} = \tilde{M}'A(\rho_k)$ and $R \equiv P'A(\rho_k)$ and $P\{NA(\rho_n)/y\} \xrightarrow{(\tilde{b})a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$. We apply rule (LET) and infer $(\mathsf{let}\ y = d(\tilde{N})\ \mathsf{in}\ P\ \mathsf{else}\ Q)A(\rho_n) \xrightarrow{(\tilde{b})a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$, as desired.

(LET-ELSE) This case is the dual of the previous case. We proceed analogously to that case by exploiting Lemma 67.

The remaining cases for New, Composition and Replication follow directly from the inductive hypothesis. As an example, we draw the composition case.

(PAR) In this case the transition has the form $(P \mid Q)A(\rho_k) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$ and has been inferred from $P(\rho_k) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$. By inductive hypothesis there are $\tilde{M}', P'$ such that $\tilde{M} = \tilde{M}'A(\rho_k)$ and $R \equiv P'A(\rho_k)$ and $P(\rho_n) \xrightarrow{(\tilde{b}')a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$. We apply (PAR) obtaining $(P \mid Q)A(\rho_n) \xrightarrow{(\tilde{b}')a\langle\tilde{M}'A(\rho_n)\rangle} (P' \mid Q)A(\rho_n)$, as desired.

The proof of clause (b) follows the same rationale. The distinguishing case is rule (IN).

(IN) In this case the transition has the form $(u(\tilde{x}).P)A(\rho_k) \xrightarrow{u(\tilde{M})} P\{\tilde{M}/\tilde{x}\}$. From $A(\rho_k) \vdash u(\tilde{x}).P$ we easily obtain $A(\rho_k), \tilde{M}/\tilde{x} \vdash P$. From this we infer $(u(\tilde{x}).P)A(\rho_n) \xrightarrow{u(\tilde{M})} P\{\tilde{M}/\tilde{x}\}A(\rho_n)$, as desired.

Finally we prove clause (c). The distinguishing case is rule (CLOSE); the other cases are analogous to the proof of clause (a).

(CLOSE) In this case the transition has the form $(P \mid Q)A(\rho_k) \xrightarrow{\tau} (\mathsf{new}\ \tilde{b})R' \mid R''$ and has been inferred from $PA(\rho_k) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R'$ and $QA(\rho_k) \xrightarrow{a(\tilde{M})} R''$. By clauses (a) and (b) we know respectively that (a) there are $\tilde{M}'$ and P' such that

$\tilde{M} = \tilde{M}'A(\rho_k)$ and $R' \equiv P'A(\rho_k)$ and $PA(\rho_n) \xrightarrow{(\tilde{b}')a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$ and (b) $R'' \equiv Q'A(\rho_n)$ and $QA(\rho_n) \xrightarrow{a(\tilde{M}'A(\rho_n))} Q'A(\rho_n)$. We apply (CLOSE) and infer $(P \mid Q)A(\rho_n) \xrightarrow{\tau} (\mathsf{new}\ \tilde{b}')(P' \mid Q')A(\rho_n)$, as desired.

□

Lemma 69 (Symmetric cryptography). *For all ρ holds*

$$\rho \models (\mathsf{new}\ k)!\, net\langle\{M\}_{sk(k)}\rangle \sim^A (\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \ .$$

Proof. Let $\rho_k = (\rho, \{M\}_{sk(k)}/x)$ and $\rho_n = (\rho, \{n\}_n/x)$ and $\{n,k\} \cap fn(\rho) = \emptyset$. Let $\mathscr{R}$ be the relation defined as $\rho \models (\mathsf{new}\ k)(C[!\, net\langle x\rangle])A(\rho_k)\mathscr{R}(\mathsf{new}\ n)(C[!\, net\langle x\rangle])A(\rho_n)$ whenever $C[-] \equiv (\mathsf{new}\ \tilde{c})R \mid -$ and $A(\rho_k) \vdash R$ and $A(\rho_n) \vdash R$. We show that $\mathscr{R}$ is included in $\sim^A$. That $\mathscr{R}$ is a term-indexed relation follows from the hypothesis, by Lemma 67 and by $fn(A(\rho)) = fn(\rho)$.

Barb preservation Let $\rho \models (\mathsf{new}\ k)(C[!\, net\langle x\rangle])A(\rho_k)\downarrow_a$. This implies that (i) $a = net$ or (ii) $a \neq net \wedge \rho \models CA(\rho_k)\downarrow_a$. In case (i) we easily match this move with $\rho \models !\, net\langle\{n\}_n\rangle\downarrow_{net}$. In case (ii) we infer that $CA(\rho_k) \xrightarrow{(\tilde{d})a\langle\tilde{M}\rangle}$. By Lemma 68(a) we have $\rho \models CA(\rho_n)\downarrow_a$, as desired. The case $\rho \models (\mathsf{new}\ n)(C[!\, net\langle x\rangle])A(\rho_n)\downarrow_a$ is analogous and exploits the vice versa direction of Lemma 68(a).

Reduction closure Let $(\mathsf{new}\ k)C[!\, net\langle x\rangle]A(\rho_k) \xrightarrow{\tau} H$. From Lemma 68(c) we easily infer that there is R' such that $A(\rho_k) \vdash R'$ and $H \equiv (\mathsf{new}\ k)C'[!\, net\langle x\rangle]A(\rho_k)$ where $C'[-] \equiv (\mathsf{new}\ \tilde{c})R' \mid -$. Here $R'A(\rho_k)$ may be obtained by (a) internal reduction of $CA(\rho_k)$ or (b) by $CA(\rho_k)$ receiving the packet $\{M\}_{sk(k)}$ on net. Again from Lemma 68(c) we infer $(\mathsf{new}\ n)C[!\, net\langle x\rangle]A(\rho_n) \xrightarrow{\tau} K \triangleq (\mathsf{new}\ n)(C'[!\, net\langle x\rangle])A(\rho_n)$. Here $R'A(\rho_n)$ may be obtained (c) by internal reduction of $CA(\rho_n)$ or (d) by $CA(\rho_n)$ receiving the packet $\{n\}_n$ on net. Suppose $(\mathsf{new}\ k)C[!\, net\langle x\rangle]A(\rho_k) \xrightarrow{A} H$. In case (a) we trivially infer $(\mathsf{new}\ n)C[!\, net\langle x\rangle]A(\rho_n) \xrightarrow{A} K$. In case (b) we note that, since both n and k are not in the free names of ρ, no filters are waiting for these packets and in turn $(\mathsf{new}\ n)C[!\, net\langle x\rangle]A(\rho_n) \xrightarrow{A} K$. The case $(\mathsf{new}\ k)C[!\, net\langle x\rangle]A(\rho_k) \longrightarrow H$ is analogous and leads to $(\mathsf{new}\ n)C[!\, net\langle x\rangle]A(\rho_n) \longrightarrow K$. From these results we infer $\rho \models H\mathscr{R}K$, as desired. The case $(\mathsf{new}\ n)C[!\, net\langle x\rangle]A(\rho_n) \xrightarrow{\tau} K$ is done analogously by exploiting the vice versa direction of Lemma 68(c).

Contextuality We have $\rho \models (\mathsf{new}\ k)(C[!\, net\langle x\rangle])A(\rho_k)\mathscr{R}(\mathsf{new}\ n)(C[!\, net\langle x\rangle])A(\rho_n)$.

Let $\rho \vdash S$. Notice that both $\rho \subseteq A(\rho_k)$ and $\rho \subseteq A(\rho_n)$. We let $C'[-] = (\mathsf{new}\ k)R \mid S \mid -$ with $fn(S) \cap (bn(C) \cup \{n,k\}) = \emptyset$ and we obtain $\rho \models (\mathsf{new}\ k)(C'[!\, net\langle x\rangle])A(\rho_k)\mathscr{R}(\mathsf{new}\ n)(C'[!\, net\langle x\rangle])A(\rho_n)$, as desired.

The second clause requires $\rho, n/x \models (\mathsf{new}\ k)(C[!\, net\langle x\rangle])A(\rho_k)\mathscr{R}(\mathsf{new}\ n)(C[!\, net\langle x\rangle])A(\rho_n)$ with $n \notin fn(\rho)$ which actually holds as we don't have any condition on ρ.

Finally $\rho\backslash a \models (\mathsf{new}\ a)(\mathsf{new}\ k)(C[!\, net\langle x\rangle])A(\rho_k)\mathscr{R}(\mathsf{new}\ a)(\mathsf{new}\ n)(C[!\, net\langle x\rangle])A(\rho_n)$ holds by rearranging the context: $C'[-] \equiv (\mathsf{new}\ a)C[-]$. □

The following corollary of the previous Lemma analyze the behaviour of a process that picks up a message from a protocol using a symmetric cryptoscheme.

Corollary 70. *Assume $k, n \notin fn(\rho, P)$ and $\rho, \{M\}_{sk(k)}/x \vdash C$, $\rho, \{n\}_n/x \vdash C$. We have*

$$\begin{aligned}\rho \models\ & (\mathsf{new}\ k)(C(\rho, \{M\}_{sk(k)}/x) \mid !\, net\langle\{M\}_{sk(k)}\rangle \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P) \\ \sim^A\ & (\mathsf{new}\ n)C(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_n\rangle \mid (\mathsf{new}\ k)(!\, net\langle\{M\}_{sk(k)}\rangle \\ & \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)\ .\end{aligned}$$

Proof. (Sketch) We analyze (informally) reduction closure of the relation $\rho \models H\mathscr{R}K$ such that (i) $H \equiv (\mathsf{new}\ k)(C(\rho, \{M\}_{sk(k)}/x) \mid !\, net\langle\{M\}_{sk(k)}\rangle \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)$ and $K \equiv (\mathsf{new}\ n)C(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_n\rangle \mid (\mathsf{new}\ k)(!\, net\langle\{M\}_{sk(k)}\rangle \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)$ or (ii) $H \equiv (\mathsf{new}\ k)(C(\rho, \{M\}_{sk(k)}/x) \mid !\, net\langle\{M\}_{sk(k)}\rangle \mid P\{M/\tilde{y}\})$ and $K \equiv (\mathsf{new}\ n)C(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_n\rangle \mid (\mathsf{new}\ k)(!\, net\langle\{M\}_{sk(k)}\rangle \mid P\{M/\tilde{y}\})$ In the proof of Lemma 69 we have seen that $C(\rho, \{M\}_{sk(k)}/x)$ behaves as $C(\rho, \{n\}_n/x)$ in contexts where k does not occur. Therefore to prove the corollary it is left to analyze interactions of $C(\rho, \{M\}_{sk(k)}/x)$ with $!\, net\langle\{M\}_{sk(k)}\rangle \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)$. Suppose that $C(\rho, \{M\}_{sk(k)}/x)$ receives a packet from $!\, net\langle\{M\}_{sk(k)}\rangle$ and evolves to $C'(\rho, \{M\}_{sk(k)}/x)$: This move is matched by $C(\rho, \{n\}_n/x)$ by receiving a packet from $!\, net\langle\{n\}_n\rangle$ and evolving to $C'(\rho, \{n\}_n/x)$. Now assume that $C(\rho, \{M\}_{sk(k)}/x)$ sends the packet $\{M\}_{sk(k)}$ on the channel net to process $\mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P$ and evolves to $C'(\rho, \{M\}_{sk(k)}/x)$. We have $H \xrightarrow{\tau} H'$ where

$$H' \equiv (\mathsf{new}\ k)(C'(\rho, \{M\}_{sk(k)}/x) \mid !\, net\langle\{M\}_{sk(k)}\rangle \mid \mathsf{filter}\ \{M\}_{sk(k)}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)$$

Therefore $C[-] \equiv C'[net\langle x\rangle \mid -]$ and in turn

$$\begin{aligned} H \ &\equiv\ (\mathsf{new}\ k)(C'(\rho, \{M\}_{sk(k)}/x) \mid !\, net\langle\{M\}_{sk(k)}\rangle \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P) \\ K \ &\equiv\ (\mathsf{new}\ n)(C'(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_{sk(n)}\rangle \mid (\mathsf{new}\ k)!\, net\langle\{M\}_{sk(k)}\rangle \\ &\qquad \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)\end{aligned}$$

This let us infer $H \xrightarrow{A} H'$. We let K match this move with $K \xrightarrow{A} K'$ where

$$\begin{aligned} K' \ &\equiv\ (\mathsf{new}\ n)(C'(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_n\rangle \mid (\mathsf{new}\ k)!\, net\langle\{M\}_{sk(k)}\rangle \\ &\qquad \mid \mathsf{filter}\ \{M\}_{sk(k)}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P)\ .\end{aligned}$$

Indeed it's easy to see that $\mathsf{filter}\ \{M\}_{sk(k)}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P \equiv P\{M/\tilde{y}\}$ and for (ii) we have $\rho \models H'\mathscr{R}K'$. □

The next lemma says that packets encrypted with an asymmetric cryptoscheme containing confounders are viewed as noise by the environment whenever it does not know both the decryption key and the confounder. We introduce technical results similar to the symmetric case that will be useful in the proof of the lemma.

Lemma 71. *Let* $\sigma = (\rho, \{M'\}_{ek(M)}/x)$, $\varsigma = (\rho, \{N'\}_N/x)$ *where* ρ *is a substitution such that* $M, dk(M) \notin A(\rho)$ *and* $fn(M') \not\subseteq fn(\rho) \wedge fn(N) \not\subseteq fn(\rho)$. *We have that there are* $\tilde{M}$ *closed by* $A(\rho), \{M'\}_{ek(M)}/x, A(\rho), \{N'\}_N/x$ *such that* $A(\sigma) = A(\rho), \tilde{N}/\tilde{x}$ *and* $\tilde{N} = \tilde{M}(A(\rho), \{M'\}_{ek(M)}/x)$ *and* $A(\varsigma) = A(\rho), \tilde{N}'/\tilde{x}$ *and* $\tilde{N}' = \tilde{M}(A(\rho), \{N'\}_N/x)$.

Lemma 72. *Let* $\rho_M = (\rho, \{N, c\}_{ek(M)}/x)$ *and* $\rho_n = (\rho, \{N'\}_n/x)$ *where* ρ *is a substitution such that* $\{n, c\} \cap fn(\rho) = \emptyset$ *and* $\{M, dk(M)\} \not\subseteq A(\rho)$. *Let* $A(\rho_M) \vdash P$ *and* $A(\rho_n) \vdash P$. *The following hold.*

(a) If $PA(\rho_M) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$ *then there are* P' *and* $\tilde{M}'$ *closed by* $A(\rho_M)$ *such that* $R \equiv P'A(\rho_M)$ *and* $\tilde{M} = \tilde{M}'A(\rho_M)$ *and* $PA(\rho_n) \xrightarrow{(\tilde{b}')a\langle\tilde{M}'A(\rho_n)\rangle} P'A(\rho_n)$; *the symmetric result holds for the hypothesis* $PA(\rho_n) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$;

(b) If $PA(\rho_M) \xrightarrow{a(\tilde{M})} R$ *then there is* P' *closed by* $A(\rho_M, \tilde{M}/\tilde{x})$ *such that* $R \equiv P'A(\rho_M, \tilde{M}/\tilde{x})$ *and* $PA(\rho_n) \xrightarrow{a(\tilde{M})} P'A(\rho_n, \tilde{M}/\tilde{x})$; *the symmetric result holds for the hypothesis* $PA(\rho_n) \xrightarrow{a(\tilde{M})} R$;

(c) If $PA(\rho_M) \xrightarrow{\tau} R$ *then there is* P' *closed by* $A(\rho_M)$ *such that* $R \equiv P'A(\rho_M)$ *and* $PA(\rho_n) \xrightarrow{\tau} P'A(\rho_n)$; *the symmetric result holds for the hypothesis* $PA(\rho_n) \xrightarrow{\tau} R$.

Proof. The proof follows the same rationale of Lemma 68, in this case using Lemma 72. As in Lemma 68 the decryption of the packet $\{N, c\}_{ek(M)}$ bound to x does not apply here by the hypothesis $\{M, dk(M)\} \not\subseteq A(\rho)$. What is peculiar to asymmetric cryptoscheme is the use of a fresh confounder to prevent cryptanalysis attacks. Indeed the environment may try to comparing bit a bit a message built by itself (notice ρ may know both N and $ek(M)$) with the asymmetric packet; the presence of c prevents this attack. To prove the Lemma we analyze the application of *equals* in (3.2). Indeed in such case the result does not follows directly from Lemma 72, because the analysis is insensible to the adding of packet still in its range.

We draw the case let $equals(N_1, N_2)$ in P else $QA(\rho_M) \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$; input and silent actions cases are analogous. From the hypothesis above we know $equals(N_1, N_2)A(\rho_M) \to N'$ and $P\{N'/y\} \xrightarrow{(\tilde{b})a\langle\tilde{M}\rangle} R$. By the definition of analysis we easily infer that there is N closed by $A(\rho_M)$ such that $N' = NA(\rho_M)$. From Lemma71 we know $A(\rho_M) = A(\rho), \tilde{M}\sigma/\tilde{x}$ where $\sigma = A(\rho), \{N, c\}_{ek(M)}/x$ and

$A(\rho_n) = A(\rho), \tilde{M}\varsigma/\tilde{x}$ where $\varsigma = A(\rho), \{n\}_n/x$. If both N_1, N_2 are closed by $A(\rho)$ we trivially infer $equals(N_1, N_2)A(\rho_n) \rightarrow NA(\rho_n)$. Similarly if both N_1, N_2 contain some variable z in $\tilde{x}$ such that $M_z\sigma$ is associated to z and $c \in fn(M_z\sigma)$, we match the evaluation with $M_z\varsigma$ associated to z. Suppose N_1 contain such z and N_2 does not. From this we easily infer $c \in fn(N_1A(\rho_M))$ and $c \notin fn(N_2A(\rho_M))$, contradiction, because we assumed $equals(N_1, N_2)A(\rho_M) \rightarrow$. The latter sentence follows from $c \notin fn(\rho)$ an by the hypothesis To conclude, by inductive hypothesis we have $\tilde{M} = \tilde{M'}A(\rho_M)$ and $R \equiv P'A(\rho_M)$ and $P\{NA(\rho_n)/y\} \xrightarrow{(\tilde{b})a\langle \tilde{M'}A(\rho_n)\rangle} P'A(\rho_n)$. We apply rule (LET) and infer $(\mathsf{let}\ y = d(\tilde{N})\ \mathsf{in}\ P\ \mathsf{else}\ Q)A(\rho_n) \xrightarrow{(\tilde{b})a\langle \tilde{M'}A(\rho_n)\rangle} P'A(\rho_n)$, as desired.

□

We introduce the following notation to represent the process reached by a channel server that interact with the environment and execute administrative reductions.

Notation 1. *Let Q be a process such that for some $M_1, \ldots, M_n$ we have $Chan_n \overset{A}{\Longrightarrow} \xrightarrow{net(M_1)} \cdots \xrightarrow{net(M_n)} \overset{A}{\Longrightarrow} Q \equiv (\mathsf{new}\ n^\circ, n^*)n^*\langle N\rangle \mid P \mid WS_n \mid RS_n$ and $P = P_1 \mid \cdots \mid P_n$ and $fn(P_i) \cap \{n^\circ, n^*\} \neq \varnothing$. We indicate Q with $Chan_n\{N\}[P]$.*

The clause saying that threads of P cannot be rearranged outside the scope of n^*, n° let us discard messages wrongly picked by the server.

Lemma 73 (Asymmetric Cryptography). *Let ρ be a substitution s.t. $a \in fn(\rho) \wedge a, g(a), dk(g(a)) \notin A(\rho)$ where $g \in \{rd, wr\}$. Let $N = N' :: c :: N''$ and let $\tilde{d} \subseteq fn(M)$. We have*

$$\rho \models W \mid (\mathsf{new}\ c, \tilde{d})Chan_a\{N\}[] \mid !\, net\langle\{M, c\}_{ek(g(a))}\rangle \approx^A W \mid (\mathsf{new}\ c)Chan_a\{N\}[]\ .$$

Proof. Let $\rho_a = \rho, \{M, c\}_{f(a)}/x$ and $\rho_n = \rho, \{n\}_n/x$, where where $f = ek \circ g$ and ρ satisfies the conditions of the statement.

We consider the relation $\mathscr{R}$ containing couples formed by the context representation of $A(\rho_a)$ on the left and $A(\rho_n)$ on the right respectively containing the channel containing the nonce c and nonces and threads obtained by $A(\rho_a)$ on the left and the channel with the nonce c and nonces and threads obtained by $A(\rho_n)$ on the right. The definition of $\mathscr{R}$ is complex. We let

$$\varphi \models C'[W_{\tilde{n}} \mid Chan_a\{N'\}[P']]\mathscr{R}(\mathsf{new}\ n)C''[W_{\tilde{n}} \mid Chan_a\{N''\}[P'']]$$

whenever

$$\begin{array}{rcl}
\varphi & \subseteq & \rho, \tilde{b}/\tilde{x} \ \wedge \ \tilde{b} \cap fn(\rho) = \emptyset \\
\varsigma' & = & A(\rho_a), \tilde{b}/\tilde{x}, \sigma_{\tilde{n}} \quad \varsigma'' = A(\rho_n), \tilde{b}/\tilde{x}, \sigma_{\tilde{n}} \\
C'[-] & = & (\mathsf{new}\ c, \tilde{d}, \tilde{a}, \tilde{n})(!\, net\langle x\rangle \mid R)\varsigma' \mid - \\
C''[-] & = & (\mathsf{new}\ \tilde{a}, \tilde{n})(!\, net\langle x\rangle \mid R)\varsigma'' \mid - \\
W_{\tilde{n}} & = & W \mid !\, net\langle\{n_1\}_{n_1}\rangle \mid \cdots \mid !\, net\langle\{n_m\}_{n_m}\rangle \\
N' & = & (N :: c :: N^*)\varsigma' \quad N'' = (N :: c :: N^*)\varsigma'' \\
P' & = & P\varsigma' \quad P'' = P\varsigma''
\end{array}$$

where $\tilde{d} \subseteq fn(M)$ and $\sigma_{\tilde{n}} = \{n_1\}_{n_1}/y_1, \cdots, \{n_m\}_{n_m}/y_m$ and R, N, N^*, P are closed both by ς' and ς'' where $fn(R, N, N^*) \subseteq \tilde{a}$ and $fn(P) \subseteq \{\tilde{a}, a^\circ, a^*\}$.

We give the intuitions of this definition. We illustrate the left side, the right side is specular. That the index is a subset of $\rho, \tilde{b}/\tilde{x}$ is useful to prove contextuality; $\tilde{b}$ typically are bindings introduced by weakening. The bindings $\tilde{n}$ and the related substitution $\sigma_{\tilde{n}}$ are obtained by $W \equiv (\mathsf{new}\ \tilde{n})W_{\tilde{n}}$: indeed packets coming from the noise process may be absorbed by the context. Therefore the context is built around $A(\rho_a), \tilde{b}/\tilde{x}$ and $\sigma_{\tilde{n}}$; since the context can send packets to the channel, also nonces and messages in its queue are built around these substitutions plus names $\tilde{a}$ generated by the context. Since c is not known to the context we explicit its presence in the nonce list.

We prove that $\mathscr{R}$ is an administrative equivalence up to expansion. Lemma 63 ensures that in this case $\mathscr{R} \subseteq \approx^A$. The expansion will be used in proving reduction closure, particularly in the case of the channel receiving the packet $\{M, c\}_{f(a)}$. To ease the notation we will write often H and K for the term respectively on the left and on the right of $\mathscr{R}$ as in $\varphi \models H \mathscr{R} K$.

Barb preservation Let $\varphi \models H \downarrow_b$. Then by shape analysis of H it's easy to see that or $n = net$ or the barb is inferred from $\varphi \models CA(\rho_a) \downarrow_b$. The case $b = net$ is trivial. Otherwise $C' \xrightarrow{(\tilde{a})b\langle\tilde{M}\rangle}$. By Lemma 72(a) we infer $\varphi \models C'' \downarrow_b$ and we are done.

Reduction closure Let $H \xrightarrow{\tau} H'$. The following cases arise.

(*Context Reduction*) The move has been inferred from $R\varsigma' \xrightarrow{\tau} R'\varsigma'$ and may be administrative or not. We match this move with the move inferred from $R\varsigma'' \longrightarrow R'\varsigma''$. From Lemmas 71 and by definition of administrative reduction (Def. 22) we easily infer that the last reduction is administrative if and only if $R\varsigma' \xrightarrow{A} R'\varsigma'$. Most cases of Def. 22 can be easily tackled by shape analysis on the hypothesis $R\varsigma' \xrightarrow{A} R'\varsigma'$. Clauses Def. 22(3,4) do not apply as neither $\{M, c\}_{f(a)}$ or $\{n\}_n$ are packet encrypted with an symmetric crypto-scheme. (5) does not apply since there is not a type $\underline{T}$

"for" $\{M,c\}_{f(a)}$ or$\{n\}_n$. In Def. 22(6) the variable x may occur in a nonce list or in a nonce to be checked; in this case $\{M,c\}_{f(a)}$ and $\{n\}_n$ are used as nonces respectively in $R\varsigma'$ and $R\varsigma''$ and from $c,n \notin fn(\rho)$ we infer that both reductions are administrative. Similar in case Def. 22(7) the packets bound to x may be used as seed indexes of a table and from $c,n \notin fn(\rho)$ we infer that both reductions are administrative.

(*Channel Reduction*) The move has been inferred from $Chan_a\{N'\}[P'] \xrightarrow{\tau} Q$. If $Chan_a\{N'\}[P'] \xrightarrow{A} Q$ then we easily find N_1', P_1' closed by ς' with such that $Q \equiv Chan_a\{N_1'\}[P_1']$ where (i) $N_1' = N'$ and $P_1' \xrightarrow{d} P$ or (ii) $N_1' = (N_1 :: N :: c :: N^*)\varsigma'$ and $P_1' = (P_1 \mid P)\varsigma'$ with $fn(N_1) \subseteq \tilde{a}$ and $fn(P_1) \subseteq \{\tilde{a}, a^\circ, a^*\}$ This follows by noting that $Chan_a\{N'\}[P'] \xrightarrow{A} Q$ has been inferred or (i) from an unsuccessful synchronization on a° (Def. 22(5)) or (ii) from a synchronization on a^* (Def. 22(6)). We match this move by building N_1'', P_1'' specularly: $Chan\{N''\}[P''] \xrightarrow{A} Chan\{N_1''\}[P_1'']$. Otherwise $Chan\{N'\}[P'] \xrightarrow{n@T} Chan\{N'\}[P_1'] \mid !\, net\langle\{\underline{n_1}\Uparrow\underline{T}\}_{M_1'}\rangle$ for some n_1, M_1, T and P_1' obtained by removing the synchronizing threads from P'. Since P is built around ς' we infer that there exists R' closed by ς' and with $fn(R') \subseteq \tilde{a}$ such that $R'\varsigma' = !\, net\langle\{\underline{n_1}\Uparrow\underline{T}\}_{M_1}\rangle$. We match this move with $Chan\{N''\}[P''] \xrightarrow{n@T} Chan\{N''\}[P_1''] \mid !\, net\langle\{\underline{n_2}\Uparrow\underline{T}M_2\}_{\rangle}$ where P_1'' is obtained by removing the synchronizing threads from P''. We obtain that $R'\varsigma'' = !\, net\langle\{\underline{n_2}\Uparrow\underline{T}\}_{M_2}\rangle$ and we let $C_1'[-] = C'[R' \mid -]$ and $C_1''[-] = C''[R' \mid -]$. We conclude that $\varphi \models C_1'[W_{\tilde{n}} \mid Chan_a\{N'\}[P_1']\mathscr{R}C_1''[W_{\tilde{n}} \mid Chan_a\{N''\}[P_1'']]$.

(*Communication*) The case whether the context receives a packet from $W_{\tilde{n}}$ $\mid Chan_a\{N'\}[P']$ is trivial as $Chan_a\{N'\}[P']$ does not have free outputs syntactically occurring in its definition. Thus the packet has been received from $W_{\tilde{n}}$: $R\varsigma' \mid W_{\tilde{n}} \xrightarrow{\tau} R_1$. We easily infer that (i) $\exists R'$ closed by ς' with $fn(R') \subseteq \tilde{a}$ such that $R_1 = R'\varsigma'$ or (ii) $\exists R'$ closed by ς', σ_m with $fn(R') \subseteq \tilde{a}$ such that $R_1 = (\mathsf{new}\ m)R'\varsigma', \sigma_m$. We match this move with $R\varsigma'' \mid W_{\tilde{n}} \xrightarrow{\tau} R_2$; a case analysis on administrative reductions show that this move is administrative if and only if $R\varsigma' \mid W_{\tilde{n}} \xrightarrow{A} R_1$. We easily infer that

$$\begin{aligned}
\text{(i)}\quad \varphi \models\ & (\mathsf{new}\ c, \tilde{d}, \tilde{a}, \tilde{n})!\, net\langle\{M,c\}_{f(a)}\rangle \mid R_1 \mid W_{\tilde{n}} \mid Chan_a\{N'\}[P']\mathscr{R} \\
& (\mathsf{new}\ n, \tilde{a}, \tilde{n})!\, net\langle\{n\}_n\rangle R_2 \mid W_{\tilde{n}} \mid Chan_a\{N''\}[P'']] \\
\text{(ii)}\quad \varphi \models\ & (\mathsf{new}\ c, \tilde{d}, \tilde{a}, \tilde{n}, m)R_1 \mid W_{\tilde{n},m} \mid Chan_a\{N'\}[P']\mathscr{R} \\
& (\mathsf{new}\ n, \tilde{a}, \tilde{n}, m)R_2 \mid W_{\tilde{n},m} \mid Chan_a\{N''\}[P'']]\ .
\end{aligned}$$

The remaining case arises when the reduction is inferred from the context sending a packet to the channel:

$$R\varsigma' \mid Chan_a\{N'\}[P'] \xrightarrow{\tau} R'\varsigma' \mid Chan_a\{N'\}[P' \mid P^*]$$

where (i) $P^* = \mathsf{decrypt}\ M\ \mathsf{as}\ (\{\tilde{x}, z\}_{a_w^-}\ \mathsf{in}\ \mathsf{if}\ z \in Set_{a^*}\ \mathsf{then}\ a^\circ\langle\tilde{x}\rangle)\ \mathsf{else}\ net\langle M\rangle$ or (ii) $P^* = \mathsf{decrypt}\ M\ \mathsf{as}\ \{k,t,z\}_{a_r^-}\ \mathsf{in}\ (\mathsf{if}\ z \in$

Set_{a^*} then filter $\tilde{x}$ from $a^{\circ}@t$ in $!\,net\langle\{\tilde{x}\}_k\rangle$) else $net\langle M\rangle$. If the decryption is unsuccessful both in (i) and (ii) we have that $H' \equiv H$ and we have done. Otherwise (i) $M = \{\tilde{M}, N\}_{a_w^+}$ and $P \equiv$ if $N \in Set_{a^*}$ then $a^{\circ}\langle\tilde{M}\rangle$ or (ii) $M = \{M_1, T, N\}_{a_r^+}$ and $P \equiv$ if $N \in Set_{a^*}$ then filter $\tilde{x}$ from $a^{\circ}@T$ in $!\,net\langle\{\tilde{x}\}_M\rangle$. We easily infer that there are $\tilde{M}^*, M_1^*, N^*$ closed by ς' and with free names in $\tilde{a}, \tilde{d}$ such that $\tilde{M} = \tilde{M}^*\varsigma'$, $M_1 = M_1^*\varsigma', N = N^*\varsigma'$. There are two cases corresponding to (a) $N = c$ or (b) not. In case (a) by hypothesis $c \notin fn(\rho)$ we know that $M = \{M, c\}_{f(a)}$, i.e. the inferred move is administrative. Indeed the presence of $!\,net\langle M\rangle$ let us rearrange the output $net\langle M\rangle$: $!net\langle M\rangle \mid net\langle M\rangle \equiv !net\langle M\rangle$.Finally the recursion on net of the receiving filter let us apply Def. 22(2). From $Chan_a\{N'\}[P' \mid P^*] \xrightarrow{A} Chan_a\{N'\}[P']$ we infer $\varphi \models Chan_a\{N'\}[P' \mid P^*] \succeq^A Chan_a\{N'\}[P']$. In this case we do not match this move as $H \xrightarrow{A} H'$ and $\varphi \models H' \succeq^A H\mathscr{R}K$. If case (b) holds we match this move with

$$R\varsigma'' \mid Chan_a\{N''\}[P''] \xrightarrow{\tau} R'\varsigma'' \mid Chan_a\{N''\}[P'' \mid P_1^*]$$

where the move is administrative if and only if $H \xrightarrow{A} H'$ and P_1^* is such that both in case (i) and (ii) we may infer $\rho \models H'\mathscr{R}K'$ where $K' = (\text{new } n, \tilde{a}, \tilde{n})R'\varsigma'' \mid W_{\tilde{n}} \mid Chan_a\{N''\}[P'' \mid P_1^*]$. We omit the details of the last inferences that are analogous to those introduced above.

Contextuality Let $\varphi \models H\mathscr{R}K$.

Let $\varphi \vdash R'$. We know $\varphi \subseteq \rho$ and $\rho \subseteq A(\rho_a)$ and $\rho \subseteq A(\rho_n)$. We define $C_1'[-] = (\text{new } \tilde{n})(!\,net\langle x\rangle \mid R \mid R')\varsigma' \mid -$ and $C_1''[-] = (\text{new } \tilde{n})(!\,net\langle x\rangle \mid R \mid R')\varsigma'' \mid -$ with $\tilde{n}, c, n$ not clashing with the bound names of R'. Finally we obtain $\varphi \models H_1\mathscr{R}K_1$ where H_1, K_1 are respectively H with $C_1'[-]$ in place of $C'[-]$ and K with $C_1''[-]$ in place of $C''[-]$.

That $\varphi\backslash n \models (\text{new } n)H\mathscr{R}(\text{new } n)K$ follows from $\varphi \subseteq \rho$ and shape of $C'[-], C''[-]$. For the last clause suppose $n \notin fn(\varphi)$. By $\varphi, n/x \subseteq \rho, \tilde{b}/\tilde{x}, n/x$ and $n \notin fn(\rho, \tilde{b}/\tilde{x})$ we infer $\varphi, n/x \models H\mathscr{R}K$. That $n \notin fn(\rho, \tilde{b}/\tilde{x})$ follows by $\mathscr{R}$ term indexed relation: $fn(C'; C'') \subseteq fn(\rho)$. Since the free names of C' are those of ς' that are not bound by C', and the free names o of C'' are those of ς'' not bound by C'', the result follows by alpha-renaming n in case $n \in bn(C', C'')$.

□

The following corollary of the previous Lemma analyze the behaviour of a context that picks up a message from a protocol using an asymmetric cryptoscheme; it's proof is analogous to that of Corollary 70. The hypotheses on the thread P inside the scope of the channel are necessary since the context can actually read from the channel queue.

Corollary 74. *Let ρ be a substitution such that $a \in fn(\rho) \wedge a, g(a), dk(g(a)) \notin A(\rho)$ where $g \in \{rd, wr\}$. Assume $\rho, \{M, c\}_{f(g(a))}/x \vdash C$, $\rho, \{n\}_n/x \vdash C$. For*

all processes P and terms M^* such that $c \notin fn(P)$ and $P \equiv P^* \mid a^\circ\langle M^*\rangle$ implies $a, g(a), dk(g(a)) \notin A(\rho, M^*/x)$ we have

$$\begin{aligned}\rho \models (\mathsf{new}\ c)(C(\rho, \{M, c\}_{N^*}/x) \mid !\, net\langle\{M, c\}_{N^*}\rangle \mid Chan_a\{N\}[P]) \\ \approx^A (\mathsf{new}\ n)C(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_n\rangle \mid !\, net\langle\{M, c\}_{N^*}\rangle \mid Chan_a\{N\}[P])\ .\end{aligned}$$

with $N^* = ek(g(a))$ for some $g \in \{rd, wr\}$.

The next Lemma says that we can remove the nonces from channels provided they do not appear in the environment.

Lemma 75. $\rho \models (\mathsf{new}\ c)Chan_a\{N :: c :: N'\}[] \approx^A Chan_a\{N :: N'\}[]$.

Proof. Let ρ be a substitution such that $c \notin fn(\rho)$. Let $\sigma = A(\rho), \tilde{b}/\tilde{x}$ where $fn(\tilde{b}) \cap \{fn(\rho), c\} = \emptyset$, and let $\varphi \subseteq \rho, \tilde{b}/\tilde{x}$. We define

$$\varphi \models C[(\mathsf{new}\ c)Chan_a\{N_1\}[P]]\mathscr{R}C[(\mathsf{new}\ c)Chan_a\{N_2\}[P]]$$

where $C[-] = (\mathsf{new}\ \tilde{n})Q\sigma \mid -$ and $N_1 = N^*\sigma :: N' :: c :: N''$ and $N_2 = N^*\sigma :: N' :: N''$ and $P = P^*\sigma$ where $\sigma \vdash N^*, P^* \wedge fn(N^*) \subseteq \tilde{n} \wedge fn(P^*) \subseteq \{\tilde{n}, a^\circ, a^*\}$.

We show that $\mathscr{R}$ is an administrative equivalence. Barb preservation of $\mathscr{R}$ is straightforward. To see reduction closure, let $C[(\mathsf{new}\ c)Chan_a\{N_1\}[P]] \xrightarrow{\tau} H$. The following cases arise.

(*Context Reduction*) The move has been inferred from $Q\sigma \xrightarrow{\tau} Q'\sigma$. We trivially obtain $C[(\mathsf{new}\ c)Chan_a\{N_2\}[P]] \xrightarrow{\tau} K$ with $\varphi \models H\mathscr{R}K$ and the reduction administrative if and only $Q\sigma \xrightarrow{A} Q'\sigma$.

(*Communication*) The move has been inferred from $Q\sigma \mid (\mathsf{new}\ c)Chan_a\{N_1\}[P] \xrightarrow{\tau} Q'\sigma \mid (\mathsf{new}\ c)Chan_a\{N_1\}[P \mid P']$ where P' is closed by σ and $fn(P') \subseteq \{\tilde{n}, a^\circ, a^*\}$. We match this move with inferred from $Q\sigma \mid (\mathsf{new}\ c)Chan_a\{N_2\}[P] \xrightarrow{\tau} Q'\sigma \mid (\mathsf{new}\ c)Chan_a\{N_2\}[P \mid P']$.

(*Channel reduction*) The move has been inferred from $Chan_a\{N_1\}[P] \xrightarrow{\tau} H^*$. Two cases arises. If the reduction is not administrative (i) then $Chan_a\{N_1\}[P] \xrightarrow{a@T} H^*$ and $H^* \equiv Chan_a\{N_1\}[P'] \mid !\, net\langle\{\tilde{M}\}_M\rangle$ where $P = P' \mid P''$. We easily find Q' closed by σ and with $fn(Q') \subseteq \tilde{n}$ such that $Q'\sigma = !\, net\langle\{\tilde{M}\}_M\rangle$. We match this move with the move inferred from $Chan_a\{N_2\}[P] \xrightarrow{a@T} Chan_a\{N_2\}[P'] \mid !\, net\langle\{\tilde{M}\}_M\rangle$. Otherwise (ii) suppose $Chan_a\{N_1\}[P] \xrightarrow{\tau} H^*$. This reduction may be inferred from a deterministic reduction, from an unsuccessful synchronization on a°, or from a synchronization on a^*. The first two cases are trivial; the interesting one is the latter. Indeed N_1 contains the nonce c while N_2 does not. However, from $c \notin fn(\rho)$ and the

conditions on $\tilde{b}$ we infer $c \notin fn(\sigma)$. Thus $c \notin fn(P)$, and the same reduction can be matched on the right side.

Finally that $\mathscr{R}$ is contextual follows as usual directly by the definition of the base and context.

□

The next lemmas state results for the proxy analogous to that introduced above for the channels.

Notation 2. *Let Q be a process such that for some $M_1, \ldots, M_n$ we have that $Proxy \overset{A}{\Longrightarrow} \xrightarrow{net(M_1)} \cdots \xrightarrow{net(M_n)} \overset{A}{\Longrightarrow} (\mathsf{new}\ \tilde{n})(Chan_{\tilde{n}}\{\tilde{N}\}[R] \mid Q)$ and $Q \equiv (\mathsf{new}\ t, t^*)\ P \mid P_t \mid t\langle N\rangle \mid t^*\langle M\rangle \mid$ and $P = P_1 \mid \cdots \mid P_n$ and $fn(P_i) \cap \{t, t^*\} = \varnothing$. We indicate Q with $Proxy_N\{M\}[P]$ and write $\mathsf{CE}^*[-]$ to indicate the context $C[-]$ such that for terms $\tilde{N}, N, M$ and processes P, R holds $C[-] = - \mid W \mid (\mathsf{new}\ \tilde{n})(Chan_{\tilde{n}}\{\tilde{N}\}[R] \mid Proxy_N\{M\}[P])$.*

Lemma 76. *Let ρ be a substitution s.t. $k \in fn(\rho) \wedge k, dk(k) \notin A(\rho)$. Let $N = N' :: c :: N''$. For all M we have*

$$\rho \models W \mid (\mathsf{new}\ c)Proxy_M\{N\} \mid !\, net\langle\{\tilde{M}, c\}_{ek(k)}\rangle \approx^A W \mid (\mathsf{new}\ c)Proxy_M\{N\}\ .$$

In the following corollary, unlike Corollary 74, we do not set conditions on the thread P inside the scope of the Proxy; this is because the terms released by the proxy to the environment are system generated encryption keys.

Corollary 77. *Let ρ be a substitution such that $k \in fn(\rho) \wedge k, dk(k) \notin A(\rho)$. Assume $\rho, \{M, c\}_{ek(k)}/x \vdash C$, $\rho, \{n\}_n/x \vdash C$. For all P we have*

$$\begin{aligned}\rho \models\ & (\mathsf{new}\ c)(C(\rho, \{M, c\}_{ek(k)}/x) \mid !\, net\langle\{M, c\}_{ek(k)}\rangle \mid Proxy_M\{N\}[P]) \\ \approx^A\ & (\mathsf{new}\ n)C(\rho, \{n\}_n/x) \mid !\, net\langle\{n\}_n\rangle \mid !\, net\langle\{M, c\}_{ek(k)}\rangle \mid \mid Proxy_M\{N\}[P])\ .\end{aligned}$$

Lemma 78. *Let $c \notin fn(M, N, N')$. We have that for all ρ holds $\rho \models (\mathsf{new}\ c)Proxy_M\{N :: c :: N'\} \approx^A Proxy_M\{N :: N'\}$.*

5.4 Operational correspondence

In the proofs presented in this section, we will often abbreviate $\langle\, \Gamma \triangleright P \,\rangle$ with $\langle\, P \,\rangle$ whenever the type environment Γ is unnecessary. To ease the notations we redefine the definition of $\{\!|\, \mathrm{I} \,|\!\}$ by letting the base index include the proxy key k_P^+:

$$\{\!|\, \emptyset \,|\!\} = \{net/x_o, k_P^+/x_k\}, \quad \{\!|\, \mathrm{I}, a : A \,|\!\} = \{\!|\, \mathrm{I} \,|\!\}, \underline{a} \Uparrow \underline{A}/\underline{x}$$

where $\underline{x} \notin \mathrm{dom}(\{\!|\, \mathrm{I} \,|\!\})$.

We first show that the translation is closed by substitution, in the sense made precise below.

We let $\equiv_{\{\Gamma\}}$ denotes a congruence relation that relates translated pi processes that are structurally congruent up to type consistent substitutions of the form: $\{\underline{v_i}{\Uparrow}T_i/\underline{x_i} \mid \Gamma(v_i) <: T_i <: \Gamma(x_i)\}$.

Below we define the main axioms for $\equiv_{\{\Gamma\}}$; the remaining homeomorphic cases arise as expected. We let $\mathscr{C} \equiv_{\{\Gamma\}} \mathscr{D}$ whenever $\mathscr{C}[\mathbf{0}] \equiv_{\{\Gamma\}} \mathscr{D}[\mathbf{0}]$ where $\mathscr{C}, \mathscr{D}$ are full contexts built around the grammar for processes plus the hole $-$.

$$
\begin{array}{lll}
\langle\, \Gamma \triangleright u\langle v@T\rangle \,\rangle_{\,\mu} & \equiv_{\{\Gamma\}} & \langle\, \Gamma \triangleright x\langle y@T\rangle \,\rangle_{\,\mu}\{\underline{u}{\Uparrow}\underline{A}/\underline{x}, \underline{v}{\Uparrow}\underline{S}/\underline{y}\} \\
 & & \qquad \Gamma(u) <: A <: \mathsf{w}, \Gamma(v) <: S <: T \\
\langle\, \Gamma \triangleright u(x@T).P \,\rangle_{\,\mu} & \equiv_{\{\Gamma\}} & (\mathsf{let}\ \underline{\hat{w}} = [\![\, w \,]\!]_{\,\mu}\ \mathsf{in}\ \mathsf{link}\ (\underline{\hat{z}}{\Uparrow}\mathsf{r}, \underline{y})\ \mathsf{in} \\
 & & \ (\mathsf{new}\ k)\, \mathsf{emit}(\{sk(k), \underline{T}\}_{\underline{y}^+_{\mathsf{r}}}))\{\underline{u}{\Uparrow}\underline{A}/\underline{w}\} \\
 & & \ \mid \mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ H \\
 & & \qquad \Gamma(u) <: A <: \mathsf{r}\ H \equiv_{\{\Gamma\}} \langle\, \Gamma, x : T \triangleright P \,\rangle \\
\langle\, \Gamma \triangleright [u = v]\, P; Q \,\rangle_{\,\mu} & \equiv_{\{\Gamma\}} & \mathsf{if}\ \underline{x}_{ID} = \underline{y}_{ID}\{\underline{u}{\Uparrow}\underline{A}/\underline{x}, \underline{v}{\Uparrow}\underline{B}/\underline{y}\}\ \mathsf{then}\ H\ \mathsf{else}\ K \\
 & & \qquad \Gamma(u) <: A, \Gamma(v) <: B, \\
 & & \qquad H \equiv_{\{\Gamma'\}} \langle\, \Gamma' \triangleright P \,\rangle_{\,\mu[u \leftrightarrow v]}\ \Gamma' = \Gamma{\sqcap}v{:}\Gamma(u){\sqcap}u{:}\Gamma(v) \\
 & & \qquad K \equiv_{\{\Gamma\}} \langle\, \Gamma \triangleright Q \,\rangle_{\,\mu}
\end{array}
$$

Lemma 79. *Let Γ be a closed type environment such that $\Gamma \vdash v : A$. Then:*

$$\langle\, \Gamma, x{:}A \triangleright P \,\rangle\, \{(\underline{v}{\Uparrow}\underline{A})/\underline{x}\} \equiv_{\{\Gamma\}} \langle\, \Gamma \triangleright P\{v/x\} \,\rangle$$

Proof. By induction on the structure of the compiled process. We show that for a full applied pi calculus context $\mathscr{C}$ that closes its hole with variables $\tilde{y}$ and an open type environment Γ such that $\Gamma = \Delta, \tilde{y} : \tilde{T}$ for some closed environment Δ and some types $\tilde{T}$, we have that

$$\mathscr{C}[\,\langle\, \Gamma, x{:}A \triangleright P \,\rangle\, \{(\underline{v}{\Uparrow}\underline{A})/\underline{x}\}] \equiv_{\{\Delta\}} \mathscr{C}'[\,\langle\, \Gamma \triangleright P\{v/x\} \,\rangle\,]$$

whenever $\mathscr{C}' \equiv_{\{\Delta\}} \mathscr{C}$.

(T-In@) The judgment has the form $\Gamma, x : A \vdash u(y@T).P$ with $\Gamma(u) <: \mathsf{r}$ and $\Gamma, x : A, y : T \vdash P$. The case $x \neq u$ follows directly by induction. Otherwise let $x = u$; from this we infer $A <: \mathsf{r}$. We have that

$$
\begin{array}{l}
\langle\, x(y@T).P \,\rangle_{\,\mu} \triangleq (\mathsf{let}\ \underline{\hat{x}} = \mathsf{Meet}\{\underline{w} \mid \mu \vdash x \leftrightarrow w\}\ \mathsf{in}\ \mathsf{link}\ (\underline{\hat{x}}{\Uparrow}\mathsf{r}, \underline{y})\ \mathsf{in} \\
\quad (\mathsf{new}\ k)\, \mathsf{emit}(\{sk(k), \underline{T}\}_{\underline{y}^+_{\mathsf{r}}}))\, \mid \mathsf{filter}\ \underline{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle\, \Gamma, y : T \triangleright P \,\rangle_{\,\mu}
\end{array}
$$

By induction on $\Gamma, y : T, x : A \vdash P$ we obtain that if $\mathscr{D}$ is a full context that closes its hole with variables $\tilde{y}, y$ and $\mathscr{D}'$ is a full context such that $\mathscr{D}' \equiv_{\{\Gamma\}} \mathscr{D}$, and $\Gamma = \Delta, \tilde{y} : \tilde{T}$ with Δ closed, we have that $\mathscr{D}[\,\langle\, \Gamma, y : T \triangleright P \,\rangle_{\,\mu}\{\underline{v}{\Uparrow}\underline{A}/\underline{x}\}] \equiv_{\{\Gamma\}}$

$\mathscr{D}'[\,\langle\,\Gamma, y : T \triangleright P\{v/x\}\,\rangle_{\,\mu}]$. From $A <: \mathsf{r}$ and $(\Gamma, x : A)(x) = A$ and induction we infer the desired result,

$$\mathscr{C}[\Gamma, x : A \triangleright \langle\, x(y@T).P \,\rangle_{\,\mu}\{\underline{v}{\Uparrow}\underline{A}/\underline{x}\}] \equiv_{\{\!|\,\Delta\,|\!\}} \mathscr{C}[\Gamma \triangleright \langle\, v(y@T).P\{v/x\}\,\rangle_{\,\mu}]$$

(T-Out@) The judgment has the form $\Gamma, x : A \vdash u\langle w@T\rangle$ with $\Gamma(u) <: \mathsf{w}$ and $\Gamma(w) <: T$. If $x = u$ we infer $A <: \mathsf{w}$; if $x = w$ we infer $A <: T$. From these results, assuming $\Gamma = \Delta, \tilde{y} : \tilde{T}$ with Δ closed, we infer $\mathscr{C}[[\![\,\Gamma, x : A \triangleright u\langle w@T\rangle\,]\!]\,\{(\underline{v}{\Uparrow}\underline{A})/\underline{x}\}] \equiv_{\{\!|\,\Delta\,|\!\}} \mathscr{C}'[[\![\,\Gamma \triangleright u\{v/x\}\langle w\{v/x\}@T\rangle\,]\!]\,]$, as desired.

(T-Match@) The judgment has the form $\Gamma, x : A \vdash [u = w]\,P; Q$ and has been inferred from $\Gamma, x : A \vdash u : S$ and from $\Gamma, x : A \vdash w : T$, and from $(\Gamma, x : A) \sqcap u : T \sqcap w : S \vdash P$, and from $\Gamma, x : A \vdash Q$. Let $\Gamma = \Delta, \tilde{y} : \tilde{T}$ with Δ closed.
Let $\Delta' = \Delta \sqcap u : T \sqcap w : S$ if u, w are names else $\Delta' = \Delta \sqcap u : T$ if u is a name and w is a variable else $\Delta' = \Delta \sqcap w : S$ if w is a name and u is a variable else $\Delta' = \Delta$. Assume that $\mathscr{D}$ is a full context that closes its hole with variables $\tilde{y}$ and $\mathscr{D}'$ is a full context such that $\mathscr{D}' \equiv_{\{\!|\,\Delta'\,|\!\}} \mathscr{D}$.
By induction we have that $\mathscr{D}[\,\langle\,(\Gamma, x : A) \sqcap u : T \sqcap w : S \triangleright P\,\rangle_{\,\mu}\{\underline{v}{\Uparrow}\underline{A}/\underline{x}\}] \equiv_{\{\!|\,\Delta'\,|\!\}} \mathscr{D}'[\,\langle\,((\Gamma, x : A) \sqcap u : T \sqcap v : S) \setminus x \triangleright P\{v/x\}\,\rangle_{\,\mu}]$ and that $\mathscr{D}[\,\langle\,\Gamma, x : A \triangleright Q\,\rangle_{\,\mu}\{\underline{v}{\Uparrow}\underline{A}/\underline{x}\}] \equiv_{\{\!|\,\Delta\,|\!\}} \mathscr{D}'[\,\langle\,\Gamma \triangleright Q\{v/x\}\,\rangle_{\,\mu}]$. Particularly, the equation above holds for types S, T such that $S = (\Gamma, x : A)(u)$ and $T = (\Gamma, x : A)(w)$; this follows from the hypotheses $\Gamma, x : A \vdash u : S$ and $\Gamma, x : A \vdash w : T$, and subsumption. From this we infer the desired result,

$$\begin{aligned}&\mathscr{C}[\,\langle\,\Gamma, x{:}A \triangleright [u = w]\,P; Q\,\rangle\,\{(\underline{v}{\Uparrow}\underline{A})/\underline{x}\}]\\ &\quad\equiv_{\{\!|\,\Delta\,|\!\}} \mathscr{C}'[\,\langle\,\Gamma \triangleright [u\{v/x\} = w\{v/x\}]\,P\{v/x\}; Q\{v/x\}\,\rangle\,]\ .\end{aligned}$$

Lemma 80 (Substitution Closure). *Let Γ be a closed type environment such that $\Gamma \vdash v : A$. Then:*

$$\langle\,\Gamma, x{:}A \triangleright P\,\rangle\,\{(\underline{v}{\Uparrow}\underline{A})/\underline{x}\} \approx^{A}_{\{\!|\,\Gamma\,|\!\}} \langle\,\Gamma \triangleright P\{v/x\}\,\rangle$$

Proof. It follows from Lemma 79 and by observing that $\equiv_{\{\!|\,\Gamma\,|\!\}}$ is finer than $\approx^{A}$, i.e. $\langle\,\Gamma \triangleright P\,\rangle \equiv_{\{\!|\,\Gamma\,|\!\}} \langle\,\Gamma \triangleright Q\,\rangle$ implies $\langle\,\Gamma \triangleright P\,\rangle \approx^{A}_{\{\!|\,\Gamma\,|\!\}} \langle\,\Gamma \triangleright Q\,\rangle$. The proof of this result is by coinduction and relies on the observation that processes equated by $\equiv_{\{\!|\,\Gamma\,|\!\}}$ exhibit the same labels. We omit the rather obvious, but long, details. □

Next, we introduce a lemma saying that secret channels are not visible up to strong administrative equivalence.

Lemma 81. *Let $N = (N_1, \underline{s}) :: (s_{ID}, \underline{s}) :: N^*$. We have that $\rho \models (\mathsf{new}\ s)Chan_s \mid Proxy_N\{C\} \approx^{A} Proxy_{N^*}\{C\}$.*

Proof. Let $\sigma \models C[P]\mathscr{R}C[Q]$ whenever there is S such that $s \notin fn(S)$ and such that $P = (\mathsf{new}\ s)Chan_s \mid Proxy_M\{D\}[S]$ and $Q = Proxy_{M^*}\{C\}\{D\}[S]$ and $M = M_1 :: N$ and $M^* = M_1 :: N^*$ and whenever $\sigma = \rho \setminus \tilde{c}$ and $C[-] = (\mathsf{new}\ \tilde{c})R\rho \mid - \wedge \rho \vdash R$. We prove that $\mathscr{R}^=$ is a weak administrative equivalence up to expansion: Lemma 63 ensures that this implies $R \subseteq \approx^A$. We prove reduction closure, barb preservation and contextuality are straightforward.
Let $C[P] \xrightarrow{\tau} H$. The following cases arise.

(Context out - Process in). We have $C \xrightarrow{(\tilde{e})net\langle M\rangle} C'$ and $P \xrightarrow{net(M)} P$, $H \equiv (\mathsf{new}\ \tilde{e})C'[P']$. A case analysis shows that the free inputs of P are inputs on the free name net corresponding to (a) the proxy filter on k_P^- and (b) corresponding to the s channel filter on s_w^-, s_r^-. Suppose that $C[P] \xrightarrow{\tau} H$ has been inferred from a synchronization due to case (b); the channel rejects the packet M since it is not encrypted under the key s_w^+ or s_r^+. Therefore by Def.22(3) we infer that $C[P] \xrightarrow{A} H$. By application of let and let-else axioms of structural congruence we have that $P' \equiv (\mathsf{new}\ s)Chan'_s \mid net\langle M\rangle \mid Proxy_M\{D\}[S]$ where $Chan'_s = (\mathsf{new}\ s^*, s^\circ)\ s^*\langle\emptyset\rangle \mid RS_s \mid WS_s \mid (\mathsf{new}\ r)r\langle\rangle \mid !r().R$ where R is $\mathsf{filter}_r\ (\underline{x}, z)\ \mathsf{with}\ s_w^-\ \mathsf{in}\ \mathsf{if}\ z \notin Set_{s^*}\ \mathsf{then}\ s^\circ\langle\underline{x}\rangle\ \mathsf{or}\ \mathsf{filter}\ (y, t, z)\ \mathsf{with}\ s_r^-\ \mathsf{in}$ $\mathsf{if}\ z \notin Set_{s^*}\ \mathsf{then}\ \mathsf{filter}\ \underline{x}\ \mathsf{from}\ s^\circ@t\ \mathsf{in}\ !\, net\langle\{\underline{x}\}_y\rangle$. It easy to see that for any ζ we have $\zeta \models Chan'_s \succeq^A Chan_s$. From this we infer $\sigma \models H \succeq^A C[P]$. We do not match this move as $C[P] \xrightarrow{A} \succeq^A_\sigma C[P]$ and $\sigma \models C[P]\mathscr{R}C[Q]$. Suppose $C[P] \xrightarrow{\tau} H$ has been inferred from a synchronization due to case (a). There are two sub-cases corresponding to (i) $M = \{\tilde{M}\}_{k_P^+}$ and $\tilde{M}$ has arity consistent with the proxy filter or (ii) not. Sub-case (ii) is analogous to case (a): we infer that $C[P] \xrightarrow{A} H$ and $H \succeq^A_\sigma C[P]$. Suppose sub-case (i) holds. The packet is a valid proxy request. By application of let and let-else axioms of structural congruence we easily find T in the scope of t^* such that $P' \equiv Proxy_M\{D\}[S \mid T]$. We match this move by letting C send packet M to the Proxy inside Q:

$$C[Q] \xrightarrow{\tau} K \triangleq C'[Proxy_{M^*}\{D\}[S \mid T]]$$

Clearly $C[P] \xrightarrow{A} H$ iff $C[Q] \xrightarrow{A} K$; we are done since $\sigma \models H\mathscr{R}K$.

(Process reduction). This is the case whether $C[P] \xrightarrow{\tau} H$ is inferred from $P \xrightarrow{\tau} P'$ and $H \equiv C[P']$. This case is trivial as a case analysis shows that there is P_1 such that $Proxy_N\{C\} \xrightarrow{\tau} P_1$ and $P' \equiv (\mathsf{new}\ s)Chan_s \mid P_1$; this is due to the fact that $Chan_s$ does not reduce and does not interact with the Proxy. A case analysis shows that the following cases arise for P_1: (a) $P_1 \equiv Proxy_{N_1::N}\{C\}[S_1]$ or (b) $Proxy_N\{C_1 :: C\}[S_1]$ or (c) $Proxy_N\{C\}[S_1]$ or (d) $Proxy_N\{C\}[S_1] \mid !net\langle M\rangle$. We match this move with $C[Q] \xrightarrow{\tau} K$ where in case (a) $K \equiv Proxy_{N_1::N^*}\{C\}[S_1]$ or (b) $K \equiv C[Q] \xrightarrow{\tau} Proxy_{N^*}\{C_1 :: C\}[S_1]$ or (c) $K \equiv C[Q] \xrightarrow{\tau} Proxy_{N^*}\{C\}[S_1]$ or (d) $K \equiv C[Q] \xrightarrow{\tau} Proxy_{N^*}\{C\}[S_1] \mid !net\langle M\rangle$. A case analysis shows that in

cases (a,b,c) we have $C[P] \xrightarrow{A} H \wedge C[Q] \xrightarrow{A} K$ while in case (d) $C[P] \longrightarrow H \wedge C[Q] \longrightarrow K$. Since for all cases (a,b,c,d) we have $\sigma \models H \mathscr{R} K$, we are done.

(Context reduction). Trivial.

Now let $C[Q] \xrightarrow{\tau} K$. The case whether the reduction is inferred from Q receiving a packet from the context is analogous to case (Context out - Process in) in the sub case whether $C[P] \xrightarrow{\tau} H$ has been inferred from a synchronization due to case (a). We omit the details that are very similar to that case. The remaining cases whether $K \equiv C[Q']$ and $Q \xrightarrow{\tau} Q'$ or whether $K \equiv C'[Q]$ and $C \xrightarrow{\tau} C'$ are specular to their respective cases above.

□

The next lemma says that typed processes are implementable with the low-level capabilities corresponding to the high level type environment.

Lemma 82. *Let $\Gamma \vdash P$. There exists R closed by $\{\!|\,\Gamma\,|\!\}$ such that $\langle\, P \,\rangle \equiv_{\{\Gamma\}} R\{\!|\,\Gamma\,|\!\}$.*

Proof. Suppose $\Gamma \vdash P$. We let $\Delta = \{x_a : \Gamma(a) \;:\; a \in \Gamma\}$ be the environment formed by variables indexed by names in Γ having the type of the names of Γ. By the substitution Lemma we have that $\Delta \vdash P\{\tilde{x}_{fn(P)}/fn(P)\}$ where $P\{\tilde{x}_{n^1,\ldots,n^m}/n^1,\ldots,n^m\}$ is a shorthand for $P\{x_{n^1}/n^1,\ldots,x_{n^m}/n^m\}$. We apply Lemma 80 and infer $\langle\, (P\{\tilde{x}_{fn(P)}/fn(P)\})\{\mathrm{dom}(\Gamma)/\tilde{x}_\Gamma\}\,\rangle \equiv_{\{\Gamma\}} \langle\, (P\{\tilde{x}_{fn(P)}/fn(P)\})\,\rangle\{\!|\,\Gamma\,|\!\}$. It's easy to see that $P\{\tilde{x}_{fn(P)}/fn(P)\})\{\mathrm{dom}(\Gamma)/\tilde{x}_\Gamma\} = P$. We have thus found $R \triangleq \langle\, (P\{\tilde{x}_{fn(P)}/fn(P)\})\,\rangle$ that proves the claim, and we are done. □

The following notation will be useful to indicate a successful synchronization on a channel queue; this case is the dual of Def.22(4).

Notation 3. *Let $P \xrightarrow{\tau} P'$. Whenever $P \equiv C[(\mathsf{new}\ n^\circ)Q \,|\, \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ n^\circ@T\ \mathsf{in}\ P \,|\, n^\circ\langle M\rangle]$ and $P' \equiv C[(\mathsf{new}\ n^\circ)Q \,|\, \mathsf{filter}\ M\ \mathsf{from}\ n^\circ@T\ \mathsf{in}\ P]$ and M has type $S <: T$ we say that P successfully synchronizes to P' on $n^\circ@T$, written $P \xrightarrow{n^\circ@T} P'$.*

We are ready to prove that the encoding $\langle\cdot\rangle$ preserve the high level executions steps.

Proposition 83 (Preservation of execution steps). *Let Γ be a closed type environment such that $\Gamma \vdash P$. If $P \xrightarrow{\tau} P'$ then $\mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \overset{A}{\Longrightarrow} \longrightarrow \approx^A_{\{\Gamma\}} \mathsf{CE}[\,\langle\, \Gamma \triangleright P' \,\rangle\,]$.*

Proof. We proceed by induction on the derivation of $P \xrightarrow{\tau} P'$.

(PI-CLOSE@) The process of interest is $P \,|\, Q$ where $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle b@B\rangle} P'$ and $Q \xrightarrow{a(b@B')} Q'$ and $B <: B'$ and $\tilde{c} \cap fn(Q) = \emptyset$ and $P \,|\, Q \xrightarrow{\tau} (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \,|\, Q')$.

Let $\Gamma \vdash P$. From $P \xrightarrow{(\tilde{c}:\tilde{C})a\langle b@B\rangle} P'$ we infer $P \equiv (\mathsf{new}\ \tilde{c} : \tilde{C})P' \,|\, a\langle b@B\rangle$; from $Q \xrightarrow{a(b@B')} Q'$ we infer $Q \equiv (\mathsf{new}\ \tilde{a} : \tilde{A})a(x@B').Q_1 \,|\, Q''$ and $Q' \equiv (\mathsf{new}\ \tilde{a} : \tilde{A})Q_1\{b/x\} \,|\, Q''$. By definition of $\langle\cdot\rangle$ we have $\langle P \rangle = (\mathsf{new}\ \tilde{c} : \tilde{C})\langle P' \rangle \mid \langle a\langle b@B\rangle \rangle$ and $\langle Q \rangle = (\mathsf{new}\ \tilde{a})\langle a(x@B').Q_1 \rangle \mid \langle Q'' \rangle$.

From the semantics of the applied calculus and from the definition of administrative reduction we infer the following reductions:

$$\mathsf{CE}[\langle P \rangle \mid \langle Q \rangle] \overset{A}{\Longrightarrow} H \triangleq W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, c, d, e, f, k, h_1, h_2)\mathit{Chan}_{a_1}\{c :: d\}[G] \,|\, \mathit{Proxy}_N\{e :: f\} \,|\, (\mathsf{filter}\ \underline{x}\ \mathsf{from}\ sk(k)\ \mathsf{in}\ \langle \Gamma, \tilde{a} : \tilde{A}, x : B' \triangleright Q_1 \rangle) \,|\, |\, \langle Q'' \rangle \mid \langle P' \rangle \,|\, L(c, d, e, f, h_1, h_2)$$

where $G = a_1^{\circ}\langle \underline{b@B} \rangle \,|\, \mathsf{filter}\ \underline{y}\ \mathsf{from}\ a_1^{\circ}@B'\ \mathsf{in}\ !\, net\langle \{\underline{y}\}_{sk(k)} \rangle$ and

$$L(c, d, e, f, h_1, h_2) = !net\langle \{\underline{a}{\Uparrow}\mathsf{r}, sk(h_1), e\}_{k_P^+} \rangle \,|\, !net\langle \{\underline{a}{\Uparrow}\mathsf{w}, sk(h_2), f\}_{k_P^+} \rangle \,|\, !net\langle \{\underline{a_1}{\Uparrow}\mathsf{r}\}_{sk(h_1)} \rangle \,|\, !net\langle \{\underline{a_1}{\Uparrow}\mathsf{w}\}_{sk(h_2)} \rangle \,|\, !net\langle \{B', sk(k), d\}_{a_{1r}^+} \,|\, !net\langle \{\underline{b}{\Uparrow}\underline{B}, c\}_{a_{1w}^+} \rangle \rangle$$

We illustrate the process $L(c, d, e, f, h_1, h_2)$. The first emission on the left of the definition is the request to the proxy for the read capability linked to a; the second is the analogous request for the write capability linked to a. The third and the fourth emission are the proxy's answer respectively to the read and write proxy request. The fifth emission is the read request to the channel server linked to a while the sixth is the write request to the same channel server.

We let $H \xrightarrow{a@B'}$ and we complete the protocol with the administrative reductions by which the filter receives the answer $!\, net\langle \{\underline{b}{\Uparrow}\underline{B'}\}_{sk(k)} \rangle$:

$$H \xrightarrow{a_1^{\circ}@B'} K_0 \overset{A}{\Longrightarrow} K \triangleq W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, c, d, e, f, k, h_1, h_2)\mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \,|\, \langle \Gamma, \tilde{a} : \tilde{A}, x : B' \triangleright Q_1 \rangle\{\underline{b}{\Uparrow}\underline{B'}/\underline{x}\} \,|\, |\, \langle Q'' \rangle \mid \langle P' \rangle \,|\, R \,|\, !net\langle \{\underline{b}{\Uparrow}\underline{B'}\}_{sk(k)} \rangle \,|\, L(c, d, e, f, h_1, h_2)$$

where $R = (\mathsf{new}\ r)!r.\mathsf{filter}_r\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle \Gamma, \tilde{a} : \tilde{A}, x : B' \triangleright Q_1 \rangle$ and $\mathsf{filter}_n\ \tilde{y}\ \mathsf{with}\ M\ \mathsf{in}\ P$
$\triangleq net(x).\mathsf{decrypt}\ x\ \mathsf{as}\ \{\tilde{y}\}_M\ \mathsf{in}\ P\ \mathsf{else}\ (net\langle x \rangle \,|\, n\langle\rangle)$.

First note that from closure of $\approx^A$ under administrative reductions (Corollary 66) we have $\{\!|\, \Gamma \,|\!\} \models K_0 \approx^A K$. Next we discard R by noting that for any ρ holds $\rho \models R \sim^A \mathbf{0}$. This is indeed a variant of the well known pi calculus equation $(\mathsf{new}\ a)a.P \sim \mathbf{0}$ and holds for strong administrative equivalence as well; we omit the proof details that are rather obvious. Let ρ be a substitution such that $\{\!|\, \Gamma \,|\!\} \subseteq \rho$ and such that ρ is sufficient to build K from a process closed by ρ. By contextuality of $\sim^A$ we infer

$$\rho \models K \sim^A W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, c, d, e, f, k, h_1, h_2)\mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \,|\, \langle \Gamma, \tilde{a} : \tilde{A}, x : B' \triangleright Q_1 \rangle\{\underline{b}{\Uparrow}\underline{B'}/\underline{x}\} \,|\, |\, \langle Q'' \rangle \mid \langle P' \rangle \,|\, \mathbf{0} \,|\, !net\langle \{\underline{b}{\Uparrow}\underline{B'}\}_{sk(k)} \rangle \,|\, L(c, d, e, f, h_1, h_2)$$

We use Lemma 80 to bring the floating substitution $\{\underline{b}{\Uparrow}\underline{B'}/\underline{x}\}$ inside the translation. Indeed from the hypotheses $\Gamma \vdash P \,|\, Q$ and $P \equiv (\mathsf{new}\ \tilde{c} : \tilde{C})P' \,|\, a\langle b@B\rangle$ we infer $\Gamma, \tilde{c} : \tilde{C} \vdash b : B$. From the hypothesis $B <: B'$ and subsumption we deduce $\Gamma, \tilde{c} : \tilde{C} \vdash b : B'$. We use weakening of Prop. 4 and Proposition 7 and we infer respectively $\Gamma, \tilde{c} : \tilde{C}, \tilde{a} : \tilde{A} \vdash b : B'$ and $\Gamma, \tilde{c} : \tilde{C}, \tilde{a} : \tilde{A}, x : B' \vdash Q_1$. We apply Lemma 80 and obtain:

$$\rho \models \langle\, \Gamma, \tilde{c} : \tilde{C}, \tilde{a} : \tilde{A}, x : B' \triangleright Q_1 \,\rangle\, \{\underline{b}{\Uparrow}\underline{B'}/\underline{x}\} \approx^A \langle\, \Gamma, \tilde{c} : \tilde{C}, \tilde{a} : \tilde{A} \triangleright Q_1\{b/x\} \,\rangle$$

Next, we apply Proposition 41 to erase bindings in $\rho \setminus \{\!|\, \Gamma \,|\!\}$ and infer

$$\begin{aligned}\{\!|\, \Gamma \,|\!\} \models K \sim^A W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, c, d, e, f, k, h_1, h_2)\mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \\ |\, \langle\, \Gamma, \tilde{c} : \tilde{C}, \tilde{a} : \tilde{A} \triangleright Q_1\{b/x\} \,\rangle \,|\, |\, \langle\, Q'' \,\rangle \,|\, \langle\, P' \,\rangle \,|\, !\mathit{net}\langle\{\underline{b}{\Uparrow}\underline{B'}\}_{sk(k)}\rangle \,|\, L(c, d, e, f, h_1, h_2)\end{aligned}$$

where we used the axiom of structural congruence for the null process and $\equiv \subseteq \sim^A$.

Next we use Lemma 69 to remove emissions encrypted under a session key not known to the environment:

$$\rho \models W \,|\, (\mathsf{new}\ h_1, h_2, k)!\mathit{net}\langle\{\underline{a_1}{\Uparrow}\mathsf{r}\}_{sk(h_1)}\rangle \,|\, !\mathit{net}\langle\{\underline{a_1}{\Uparrow}\mathsf{w}\}_{sk(h_2)}\rangle \,|\, !\mathit{net}\langle\{\underline{b}{\Uparrow}\underline{B'}\}_{sk(k)}\rangle \sim^A W$$

As above the procedure is to consider a base ρ larger than $\{\!|\, \Gamma \,|\!\}$ that can build K, to exploit contextuality of $\sim^A$, and then to erase unuseful terms. We omit the details and directly provide the following the following equation where we use the previous equation and transitivity of $\sim^A$:

$$\begin{aligned}\{\!|\, \Gamma \,|\!\} \models K \sim^A W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, c, d, e, f, k, h_1, h_2)\mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \\ |\, \langle\, Q_1\{b/x\} \,\rangle \,|\, |\, \langle\, Q'' \,\rangle \,|\, \langle\, P' \,\rangle \,|\, L(c, d, e, f)\end{aligned}$$

where $L(c, d, e, f)$ is $L(c, d, e, f, h_1, h_2)$ without the emissions encrypted with symmetric keys.

Next we apply Lemma 73 and Lemma 76 to remove emissions that have been accepted respectively by $\mathit{Chan}_{a_1}\{c :: d\}$ and $\mathit{Proxy}_N\{e :: f\}$. We assume that the ρ above satisfies the requirements of both Lemma and that is sufficient to build $\langle\, Q_1\{b/x \,\rangle \,|\, \langle\, Q'' \,\rangle \,|\, \langle\, P' \,\rangle$.

$$\begin{aligned}\rho \models W \,|\, (\mathsf{new}\ , c, d, e, f, h_1, h_2) \,|\, \mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \approx^A \\ (\mathsf{new}\ c, d, e, f, h_1, h_2)L(c, d, e, f) \,|\, \mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\}\end{aligned}$$

We use $\sim^A \subseteq \approx^A$ and proceed as in previous steps to obtain:

$$\begin{aligned}\{\!|\, \Gamma \,|\!\} \models K \ \approx^A\ & W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, c, d, e, f, k, h_1, h_2)\mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \\ & |\, \langle\, Q_1\{b/x\} \,\rangle \,|\, |\, \langle\, Q'' \,\rangle \,|\, \langle\, P' \,\rangle\end{aligned}$$

Next we remove nonces both from the channel and from the proxy by using respectively Lemma 75 and Lemma 78:

$$\rho \models (\mathsf{new}\ a_1, c, d, e, f)\mathit{Chan}_{a_1}\{c :: d\} \,|\, \mathit{Proxy}_N\{e :: f\} \approx^A (\mathsf{new}\ a_1)\mathit{Chan}_{a_1} \,|\, \mathit{Proxy}_N$$

We remove the channel for a' and the entries in the proxy by using Lemma 81:

$$\rho \models (\mathsf{new}\ a_1)Chan_{a_1} \,|\, Proxy_N \sim^A (\mathsf{new}\ a_1)Proxy$$

We obtain that

$$\{\!|\, \Gamma \,|\!\} \models K \approx^A W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}, a_1, k, h_1, h_2)Proxy \,|\, \langle\, Q_1\{b/x\} \,\rangle \,|\, \langle\, Q'' \,\rangle \,|\, \langle\, P' \,\rangle$$

Summing up, by removing restriction on names a_1, k, h_1, h_2 not occurring elsewhere by using the axiom of structural congruence for new, we obtain :

$$\{\!|\, \Gamma \,|\!\} \models K \approx^A \quad W \,|\, (\mathsf{new}\ \tilde{a}, \tilde{c}) \,|\, \langle\, Q_1\{b/x\} \,\rangle \,|\, \langle\, Q'' \,\rangle \,|\, \langle\, P' \,\rangle \,|\, Proxy$$

With minor structural rearrangements we obtain

$$\{\!|\, \Gamma \,|\!\} \models K \approx^A \quad W \,|\, Proxy \,|\, \langle\, (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \,|\, (\mathsf{new}\ \tilde{a} : \tilde{A})(Q'' \,|\, Q_1\{b/x\})) \,\rangle \ .$$

Summing up we have that

$$\mathsf{CE}[\,\langle\, P \,|\, Q \,\rangle\,] \stackrel{A}{\Longrightarrow}\longrightarrow\approx^A_{\{\!|\Gamma|\!\}} K \approx^A_{\{\!|\Gamma|\!\}} \quad \mathsf{CE}[\,\langle\, (\mathsf{new}\ \tilde{c} : \tilde{C})(P' \,|\, Q') \,\rangle\,]$$

and by transitivity of $\approx^A$ we have done.

(PI-MATCH) The process of interest is $[u = v]\, P; Q$ and $[u = v]\, P; Q \xrightarrow{\tau} R$ may be inferred from (a) $P \xrightarrow{\tau} R$ or (b) $Q \xrightarrow{\tau} R$. Suppose case (a) holds. Since $[u = v]\, P; Q$ is closed, it must be that $u = a = v$ for some name a in I (remember that if P is compatible with I then $fn(P) \subseteq \mathrm{dom(I)}$). The I.H. is that $\exists K$ such that $\mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \stackrel{A}{\Longrightarrow}\longrightarrow K$ and $K \approx^A_{\{\!|\Gamma|\!\}} \mathsf{CE}[\,\langle\, \Gamma \triangleright R \,\rangle\,]$. By definition $\langle\, \Gamma \triangleright [a = a]\, P; Q \,\rangle_\mu \triangleq \mathsf{if}\ \underline{a}_{ID} = \underline{a}_{ID}\ \mathsf{then}\ \langle\, \Gamma \sqcap a{:}\Gamma(a) \sqcap a{:}\Gamma(a) \triangleright P \,\rangle_\mu \mathsf{else}\ \langle\, \Gamma \triangleright Q \,\rangle_\mu$; notice that we do not add to μ the unuseful binding $a \leftrightarrow a$. By using the let axiom of structural congruence and $\Gamma \sqcap a{:}\Gamma(a) \sqcap a{:}\Gamma(a) = \Gamma$ we infer $\langle\, \Gamma \triangleright [a = a]\, P; Q \,\rangle_\mu \equiv \langle\, \Gamma \triangleright P \,\rangle_\mu$. We use the induction hypothesis and infer $\mathsf{CE}[\,\langle\, \Gamma \triangleright [a = a]\, P; Q \,\rangle\,]_\mu \stackrel{A}{\Longrightarrow}\longrightarrow K$, as desired. In case (b) we proceed similarly and by exploiting the let else axiom of structural congruence and induction we infer the desired result.

(PI-PAR) The process of interest is $P \,|\, Q$; the transition under analysis is $P \,|\, Q \xrightarrow{\tau} S$. By the (PI-PAR) rule in Tab. 2.2 we infer that or (a) $\exists P'.P \xrightarrow{\tau} P'$ and $S \equiv P' \,|\, Q$ or (b) $\exists Q'.Q \xrightarrow{\tau} Q'$ and $S \equiv P \,|\, Q'$. We draw the case (a); the case (b) is specular. Suppose case (a) holds. By the inductive hypothesis we have that there exists K such that $\mathsf{CE}[\,\langle\, P \,\rangle\,] \stackrel{A}{\Longrightarrow}\longrightarrow\stackrel{A}{\Longrightarrow} K$ and $K \approx^A_{\{\!|\Gamma|\!\}} \mathsf{CE}[\,\langle\, P' \,\rangle\,]$. By the operational semantics of the applied pi calculus, particularly from rule (PAR), and by definition of administrative equivalence, we easily infer that $\langle\, Q \,\rangle \,|\, \mathsf{CE}[\,\langle\, P \,\rangle\,] \stackrel{A}{\Longrightarrow}\longrightarrow\stackrel{A}{\Longrightarrow} \langle\, Q \,\rangle \,|\, K$. From $\Gamma \vdash P \,|\, Q$ and (T-PAR) we infer $\Gamma \vdash Q$.

From Lemma 82 we infer that there is R closed by $\{\!|\,\Gamma\,|\!\}$ such that $\langle\, Q \,\rangle \equiv_{\{\!|\mathrm{I}|\!\}} R\{\!|\,\Gamma\,|\!\}$. From the contextuality of $\approx^A$ we infer $\{\!|\,\Gamma\,|\!\} \models R\{\!|\,\Gamma\,|\!\} \mid K \approx^A \mathsf{CE}[\,\langle\, P' \,\rangle\,] \mid R\{\!|\,\Gamma\,|\!\}$ and in turn $\{\!|\,\Gamma\,|\!\} \models \langle\, Q \,\rangle \mid K \approx^A \langle\, Q \,\rangle \mid \mathsf{CE}[\,\langle\, P' \,\rangle\,]$, because $\equiv_{\{\!|\mathrm{I}|\!\}}$ is finer than $\approx^A$ (see Lemma 80). By using axioms of structural congruence for parallel reordering we obtain $\{\!|\,\Gamma\,|\!\} \models \langle\, Q \,\rangle \mid K \approx^A \mathsf{CE}[\,\langle\, P' \mid Q \,\rangle\,]$, as desired.

(PI-RES) The process of interest is $(\mathsf{new}\ a)P$; the transition under analysis is $(\mathsf{new}\ a)P \xrightarrow{\tau} (\mathsf{new}\ a : A)P'$. From the typing system we infer $\Gamma, a : A \vdash P$. By the (PI-RES) rule in Tab. 2.2 we infer that $P \xrightarrow{\tau} P'$; the inductive hypothesis is that there exists K such that $\mathsf{CE}[\,\langle\, P \,\rangle\,] \xRightarrow{A} K$ and $\{\!|\,\Gamma, a : A\,|\!\} \models K \approx^A \mathsf{CE}[\,\langle\, P' \,\rangle\,]$. By the (RES) rule of the applied pi calculus semantics and by definition of administrative equivalence we infer that $(\mathsf{new}\ a)\mathsf{CE}[\,\langle\, P \,\rangle\,] \xRightarrow{A} (\mathsf{new}\ a)K$. From contextuality of $\approx^A$ and the I.H. we infer that $\{\!|\,\Gamma, a : A\,|\!\} \setminus a \models (\mathsf{new}\ a)K \approx^A (\mathsf{new}\ a)\mathsf{CE}[\,\langle\, P' \,\rangle\,]$. It's easy to see that $\{\!|\,\Gamma, a : A\,|\!\} \setminus a = \{\!|\,\Gamma\,|\!\}$; by using the axiom of structural congruence for New with $a \notin fn(\mathsf{CE}[\,\langle\, P' \,\rangle\,])$ we infer $\{\!|\,\Gamma\,|\!\} \models (\mathsf{new}\ a)K \approx^A \mathsf{CE}[\,\langle\, (\mathsf{new}\ a)P' \,\rangle\,]$, as desired.

(PI-REPL) Analogous to case (PI-PAR).

□

The reverse direction of operational correspondence is subtler, as the encoding is not *prompt*. We therefore need a generalization of the standard reflection result, based on the relation $\approx^A$. First we introduce useful notation and terminology and then state a few preliminary lemmas. Let $P \xRightarrow{A} P' \longrightarrow Q$. We call the reduction sequence $\xRightarrow{A}$ *canonical*, and write it as in $\xRightarrow{\ulcorner A \urcorner}$, if it only includes the administrative steps from P required to enable the synchronization in P' (as stated, this is loose, but can be made precise, as we know exactly which are those steps).

Lemma 84. *The following hold.*

1. $\rho \models Proxy_{N::M}\{C\}[P] \sim^A Proxy_{M::N}\{C\}[P]$
2. $\rho \models Proxy_N\{C :: D\}[P] \sim^A Proxy_N\{D :: C\}[P]$
3. $\rho \models Chan_a\{N :: M\}[P] \sim^A Chan_a\{M :: N\}[P]$.

Lemma 85. *Let* $\mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \xRightarrow{A} H \longrightarrow K$. *Then there exists* H' *such that* $\mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \xRightarrow{\ulcorner A \urcorner} H' \longrightarrow \approx^A_\rho K$.

Proof. Let $\mathsf{CE}[\,\langle\, P \,\rangle\,] \xRightarrow{A} H \longrightarrow K$. A case analysis shows that $H \xrightarrow{n@t} K$. To see that $H \longrightarrow K$ implies $H \xrightarrow{n@t} K$, first consider that the names occurring in inputs and outputs of H are *net*, $\tilde{n}^\circ, \tilde{n}^*$ and "recursion" names $\tilde{r}$ used by processes $\mathsf{rec}\,X.P$. A case analysis shows that: (i) H's outputs on net are under replication while its inputs on net are under replication or under recursion: reductions involving replicated input and replicated output satisfy Def. 22(1) while reductions involving

recursive input and replicated output satisfy Def. 22(2) or Def. 22(3) since such inputs are filters waiting for a term M encrypted under a session key $sk(N)$ and no other terms $M' \neq M$ are encrypted under $sk(N)$ in H; (ii) reductions on n° or satisfy Def. 22(4) or are (n,t) synchronizations for some t; (iii) reductions on n^* satisfy Def. 22(5) since translated processes never associate the same nonce to different packets; (iv) reductions on t^* satisfy Def. 22(6); (v) reductions on recursion channels satisfy Def. 22(7).

As canonical sequence we choose the one composed by the minimal completion of the write protocol (which is asynchronous) followed by the minimal sequence to complete the asynchronous part of the read protocol. The read protocol actually completes when, after a (n,t)-synchronization, the reply is received by the input continuation. The completion of the write protocol arises in two steps. The first step is the reduction inferred from $!\mathsf{filter}\ (\underline{x}, z)\ \mathsf{with}\ n_w^-\ \mathsf{in\ if}\ z \notin Set_{n^*}\ \mathsf{then}\ n^\circ\langle \underline{x}\rangle$ synchronizing with $\mathsf{emit}(\{M\}_{n_w^+})$; the second step is inferred from the continuation of the filter synchronizing with $n^\circ\langle M\rangle$. Whenever $Q \xrightarrow{\tau} Q' \xrightarrow{\tau} Q''$ we write $Q \xrightarrow{W1} Q' \xrightarrow{W2} Q''$ if $Q \xrightarrow{\tau} Q'$ is inferred from the first step of the write protocol and $Q' \xrightarrow{\tau} Q''$ is inferred from the second part of the write protocol. Specularly, the completion of the asynchronous part of the read protocol arises in two steps. The first step is inferred from $!\,\mathsf{filter}\ (y,t,z)\ \mathsf{with}\ n_r^-\ \mathsf{in\ if}\ z \notin Set_{n^*}\ \mathsf{then\ filter}\ \underline{x}\ \mathsf{from}\ n^\circ@t\ \mathsf{in}\ !\,net\langle\{\underline{x}\}_y\rangle$ synchronizing with $\mathsf{emit}(\{M\}_{n_r^+})$; the second step is inferred from the continuation of the filter synchronizing with $n^\circ\langle M\rangle$. former. Whenever $Q \xrightarrow{\tau} Q' \xrightarrow{\tau} Q''$ we write $Q \xrightarrow{R1} Q' \xrightarrow{R2} Q''$ to indicate that $Q \xrightarrow{\tau} Q'$ is inferred from the first step of the read protocol and $Q' \xrightarrow{\tau} Q''$ is inferred from the second part of the read protocol.

We are ready to find H' such that $\mathsf{CE}[\,\langle\, P\,\rangle\,] \overset{\ulcorner A \urcorner}{\Longrightarrow} H' \longrightarrow \approx_\rho^A K$; we proceed as follows. We first move the reduction inferred from the first step of the write protocol backwards; next we roll back the reduction inferred from the second step of the write protocol. Then we do the same procedure for the asynchronous part of the read protocol. Finally we roll back the (n,t) synchronization, and we show that the process reached with the (n,t) synchronization is administrative equivalent to K, as desired.

(Write protocol - First step) Let $\mathsf{CE}[\,\langle\, P\,\rangle\,] \overset{A}{\Longrightarrow} H \xrightarrow{n@t} K$ we know that there exists Q_1, Q_2 such that $\mathsf{CE}[\,\langle\, P\,\rangle\,] \overset{A}{\Longrightarrow} Q_1 \xrightarrow{W1} Q_2 \overset{A}{\Longrightarrow} H$; this is due to the fact that the write protocol and the asynchronous part of the read protocol must be completed for H be ready to commit. If $\mathsf{CE}[\,\langle\, P\,\rangle\,] = Q_1$ then we go can jump to the paragraph above describing the proof for the second step of the write protocol. Otherwise suppose $\mathsf{CE}[\,\langle\, P\,\rangle\,] \overset{A}{\Longrightarrow} Q_0 \xrightarrow{A} Q_1 \xrightarrow{W1} Q_2 \overset{A}{\Longrightarrow} H$. Since $Q_1 \xrightarrow{W1} Q_2$ has been inferred from a replicated input and a replicated output that are present in $\mathsf{CE}[\,\langle\, P\,\rangle\,]$, we infer that these two threads are ready to synchronize even in Q_0: $\exists H\,.\,\mathsf{CE}[\,\langle\, P\,\rangle\,] \overset{A}{\Longrightarrow} Q_0 \xrightarrow{W1} H \xrightarrow{A} Q_2$. By repeating the procedure for all reductions $\mathsf{CE}[\,\langle\, P\,\rangle\,] \overset{A}{\Longrightarrow} Q_0$ we infer that there is J such that $\mathsf{CE}[\,\langle\, P\,\rangle\,] \xrightarrow{W1} J \overset{A}{\Longrightarrow} H$.

(Write protocol - Second step) In the previous step we have obtained $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} H$. Now there must exists Q_1, Q_2 such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} Q_1 \xrightarrow{W2} Q_2 \stackrel{A}{\Longrightarrow} H$. If $J = Q_1$ we jump to the paragraph above involving the first step of the read protocol. Otherwise there is Q_0 such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} Q_0 \xrightarrow{A} Q_1 \xrightarrow{W2} Q_2 \stackrel{A}{\Longrightarrow} H$. We know that $Q_0 \xrightarrow{W2} Q_1$ has been inferred from the continuation of the filter on n_w^- receiving a nonce list on the private channel n^*. We analyze possible cases for the reduction $Q_0 \xrightarrow{A} Q_1$. The following cases arise: the reduction is inferred from (a) a recursion reduction which unblocks the filter on n_w^- or (b) a reduction on n^* or (c) else. In case (a) we know that there is $Q \equiv Q_1$ such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} Q \xrightarrow{A} Q_0 \xrightarrow{A} Q_1 \xrightarrow{W2} Q_2 \stackrel{A}{\Longrightarrow} H$. From Proposition 55 we infer $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} Q \xrightarrow{W2} Q_2 \stackrel{A}{\Longrightarrow} H$. In case (b) we infer that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} Q_0 \xrightarrow{W2} \xrightarrow{A} \sim_\rho^A Q_2 \wedge Q_2 \stackrel{A}{\Longrightarrow} H$. Indeed we may execute action (b) after the second step of write protocol and what we obtain is a reordering of the list of nonces; Lemma 84 ensures that such reordering is not visible up to strong administrative equivalence. Notice indeed that the nonces exchanged in these reductions are different; if not we have a contradiction since by hypothesis the write protocol is completed due to $Q_1 \xrightarrow{W2} Q_1$. In case (c) we easily infer that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \stackrel{A}{\Longrightarrow} Q_0 \xrightarrow{W2} \xrightarrow{A} Q_2$. By repeating the procedure for all reductions (a) $J \stackrel{A}{\Longrightarrow} Q$ or (b,c) $J \stackrel{A}{\Longrightarrow} Q_0$ and by transitivity of $\sim^A$ we find Y such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \xrightarrow{W2} Y \stackrel{A}{\Longrightarrow} \sim_\rho^A H$.

(Read protocol - First step) In the previous step we obtained that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \xrightarrow{W2} Y \stackrel{A}{\Longrightarrow} X$ for some $X \sim_\rho^A H$. We proceed analogously to the first step of write protocol and find G such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \xrightarrow{W2} Y \xrightarrow{R1} G \stackrel{A}{\Longrightarrow} X$.

(Read protocol - Second step) $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} J \xrightarrow{W2} Y \stackrel{A}{\Longrightarrow} X \wedge X \sim_\rho^A H$. We proceed analogously to the second step of write protocol and fin Z such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} \xrightarrow{W2} \xrightarrow{R1} \xrightarrow{R_2} \stackrel{A}{\Longrightarrow} Z$ and $Z \sim_\rho^A X$. By $Z \sim_\rho^A H$ and chasing diagrams argument we easily have that there is K^* such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} \xrightarrow{W2} \xrightarrow{R1} \xrightarrow{R_2} \xrightarrow{A} \stackrel{}{\Longrightarrow} Z \longrightarrow K^* \wedge K^* \sim_\rho^A K$. A case analysis shows that $Z \xrightarrow{n@t} K^*$.

((n,t) step) In previous steps we obtained that there is K' such that $\mathsf{CE}[\langle P \rangle] \xrightarrow{W1} \xrightarrow{W2} \xrightarrow{R1} \xrightarrow{R_2} K' \stackrel{A}{\Longrightarrow} Z \xrightarrow{n@t} K^*$ and $K^* \sim_\rho^A K$. In case $K' = Z$ we are done as $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} K' \xrightarrow{n@t} K^*$ and $K^* \sim_\rho^A K$. From $\sim^A \subseteq \approx^A$ we obtain the claim. Otherwise we have that there is Q_1 such that $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \stackrel{A}{\Longrightarrow} Q_1 \xrightarrow{A} Z \xrightarrow{n@t} K^*$. Three cases arise for $Q_1 \xrightarrow{A} Z$ corresponding to (a) the reduction involves the output on n° or (b) the reduction occurs on a recursion channel and unblocks the input on n° or (c) else. In case (a) we infer that there is G such that $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \stackrel{A}{\Longrightarrow} Q_1 \xrightarrow{n@t} G \approx_\rho^A K^*$. Indeed in G the output on n° is not more available; since in $Q_1 \xrightarrow{A} Z$ such packet was picked up from a recur-

sive filter, we have that in Z there is a non-prompt filter on n°, hence the use of $\approx^A_\rho$ to make the filter prompt as it was in Q_1. In case (b) we infer that there is $Q \equiv Z$ such that $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \stackrel{A}{\Longrightarrow} Q \stackrel{A}{\longrightarrow} Q_1 \stackrel{A}{\longrightarrow} Z \stackrel{n@t}{\longrightarrow} K^*$. From Proposition 55 we infer $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \stackrel{A}{\Longrightarrow} Q \stackrel{n@t}{\longrightarrow} K^*$. In case (c) we easily infer $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \stackrel{A}{\Longrightarrow} Q_1 \stackrel{n@t}{\longrightarrow} \stackrel{A}{\longrightarrow} K^*$. By repeating the procedure for all reductions (a,c) $H' \stackrel{A}{\Longrightarrow} Q_1$ or (b) $H' \stackrel{A}{\Longrightarrow} Q$ and by transitivity of $\approx^A$ we find T such that $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \stackrel{n@t}{\longrightarrow} T \approx^A_\rho K^*$. From $K^* \sim^A_\rho K$ and $\sim^A \subseteq \approx^A$ we infer that $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H' \longrightarrow \approx^A_\rho K$, as requested. □

The following Lemma is the key to prove that the encoding reflects high level execution steps.

Lemma 86. *Suppose* $\mathsf{CE}[\langle \Gamma \triangleright P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H \longrightarrow K$. *Then there exists* P' *such that* $P \stackrel{\tau}{\longrightarrow} P'$ *and* $\{\!|\, \Gamma \,|\!\} \models K \approx^A \mathsf{CE}[\langle \Gamma \triangleright P' \rangle]$.

Proof. From the hypothesis of the Lemma we infer the shape of P and in turn the shape of H. First from $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H \longrightarrow K$ we infer $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H \stackrel{a@B'}{\longrightarrow} K$, for some name a and type B' (see the proof of Lemma 85). Next a case analysis shows that $P \equiv C[a\langle b@B \rangle \mid a(x@B').Q]$ where $B <: B'$. Similarly to the (Pi-Close) case of the proof of preservation of execution steps (Prop. 83) we infer the shape of H. The presence of the context is easily tackled by noting that $\langle C[P] \rangle = \langle C \rangle [\langle P \rangle]$. As in that case we let K to complete the read protocol by inferring the administrative reductions that let the continuation $\langle Q \rangle$ receive the terms $\underline{b} \Uparrow B'$. Finally we discard emissions encrypted under symmetric keys by using Lemma 69, we ignore emissions encrypted under asymmetric keys by using Lemmas 73,76, we remove nonces from channels and proxy by using Lemmas 75, 78, and we remove both the entries in the proxy for a and the channel linked to those entries by using Lemma 81. Finally we use Lemma 80 to bring $\underline{b} \Uparrow B'$ inside $\langle Q \rangle$, and by $P' \equiv C[Q\{b/x\}]$ and minor structural rearrangements we obtain that there is K^* such that:

$$\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H \stackrel{a@B'}{\longrightarrow} K \stackrel{A}{\Longrightarrow} K^* \;\wedge\; \{\!|\, \Gamma \,|\!\} \models K^* \approx^A \mathsf{CE}[\langle P' \rangle]$$

From closure of $\approx^A$ under administrative reductions (Corollary 66) we infer $\{\!|\, \Gamma \,|\!\} \models K \approx^A K^*$; by transitivity of $\approx^A$ we infer $\{\!|\, \Gamma \,|\!\} \models K \approx^A \mathsf{CE}[\langle P' \rangle]$, as desired. □

Proposition 87 (Reflection of execution steps). *Assume* $\{\!|\, \Gamma \,|\!\} \models H \approx^A \mathsf{CE}[\langle \Gamma \triangleright P \rangle]$ *and* $H \stackrel{\tau}{\longrightarrow} K$. *Then either* $\{\!|\, \Gamma \,|\!\} \models H \approx^A K$ *or there exists* P' *such that* $P \stackrel{\tau}{\longrightarrow} P'$ *and* $\{\!|\, \Gamma \,|\!\} \models K \approx^A \mathsf{CE}[\langle \Gamma \triangleright P' \rangle]]$.

Proof. Let $H \stackrel{\tau}{\longrightarrow} K$. If $H \stackrel{A}{\longrightarrow} K$, we apply Proposition 65 and we obtain $\{\!|\, \Gamma \,|\!\} \models H \approx^A K$, as desired. Otherwise we have that $H \longrightarrow K$, and by $\{\!|\, \Gamma \,|\!\} \models H \approx^A \mathsf{CE}[\langle P \rangle]$ we infer that there are J, Z s.t. $\mathsf{CE}[\langle P \rangle] \stackrel{A}{\Longrightarrow} \longrightarrow J \stackrel{A}{\Longrightarrow} Z$ and $\{\!|\, \mathrm{I} \,|\!\} \models K \approx^A Z$. We apply Lemma 85 and we infer that there is Q s.t. $\mathsf{CE}[\langle P \rangle] \stackrel{\ulcorner A \urcorner}{\Longrightarrow} \longrightarrow Q$

and $\{\!|\,\Gamma\,|\!\} \models Q \approx^A J$. We apply Lemma 86 ad we obtain that there is P' s.t. $P \xrightarrow{\tau} P'$ and $\{\!|\,\Gamma\,|\!\} \models Q \approx^A \mathsf{CE}[\,\langle\, P' \,\rangle\,]$. From $J \overset{A}{\Longrightarrow} Z$ and Corollary 66 we deduce $\{\!|\,\Gamma\,|\!\} \models J \approx^A Z$. Summing up we have the chain: $\{\!|\,\Gamma\,|\!\} \models K \approx^A Z \approx^A J \approx^A Q \approx^A \mathsf{CE}[\,\langle\, P' \,\rangle\,]$. We use the transitivity of $\approx^A$ and we infer $\{\!|\,\Gamma\,|\!\} \models K \approx^A \mathsf{CE}[\,\langle\, P' \,\rangle\,]$, as desired.

To prove the operational correspondence of the encoding we need a technical lemma.

Lemma 88. *If* $\{\!|\,\Gamma\,|\!\} \models P \approx^A Q$ *and* $\Gamma <: \mathrm{I}$ *then* $\{\!|\,\mathrm{I}\,|\!\} \models P \approx^A Q$.

Proof. Let $\{\!|\,\Gamma\,|\!\} \models P \approx^A Q$ and $\Gamma <: \mathrm{I}$. Assume $\Gamma = \tilde{a} : \tilde{A}$, $\mathrm{I} = \tilde{a} : \tilde{B}$. We use the notation $\underline{\tilde{n}}{\Uparrow}\underline{\tilde{T}}$ to indicate the tuples $\underline{n_1}{\Uparrow}\underline{T_1}, \ldots, \underline{n_m}{\Uparrow}\underline{T_m}$. We have that $\{\!|\,\Gamma\,|\!\} = \underline{\tilde{a}}{\Uparrow}\underline{\tilde{A}}$ and $\{\!|\,\mathrm{I}\,|\!\} = \underline{\tilde{a}}{\Uparrow}\underline{\tilde{B}}$. Consider a tuple $\underline{a}{\Uparrow}\underline{A} \in \{\!|\,\Gamma\,|\!\}$ and assume $\underline{a}{\Uparrow}A = M_1, M_2, M_3, M_4$. Let $\underline{a}{\Uparrow}\underline{B}$ be the correspondent tuple in $\underline{\tilde{a}}{\Uparrow}\underline{\tilde{B}}$ and assume $\underline{a}{\Uparrow}\underline{B} = N_1, N_2, N_3, N_4$. From $\Gamma <: \mathrm{I}$ we easily infer $A <: B$. A case analysis on the encoding of values shows that $M_1 = N_1$ and $M_4 = N_4$ and $M_2 \neq N_2$ implies $N_2 = \top$ and $M_3 \neq N_3$ implies $N_3 = \top$.
Let $\tilde{M}/\tilde{x}$ be the set of substitutions in $\{\!|\,\Gamma\,|\!\}$ such that for each $M/x \in \tilde{M}/\tilde{x}$ with $M \neq \top$ holds that $\top/x \in \{\!|\,\mathrm{I}\,|\!\}$. We apply Proposition 41 to $\{\!|\,\Gamma\,|\!\} \models P \approx^A Q$ and remove bindings $\tilde{M}/\tilde{x}$ obtaining $\{\!|\,\Gamma\,|\!\} \setminus \tilde{M}/\tilde{x} \models P \approx^A Q$. Then we build $\{\!|\,\mathrm{I}\,|\!\}$ by adding bindings $\tilde{\top}/\tilde{x}$ to $\{\!|\,\Gamma\,|\!\}\setminus\tilde{M}$ as in: $(\{\!|\,\Gamma\,|\!\}\setminus\tilde{M}/\tilde{x}), \tilde{\top}/\tilde{x} = \{\!|\,\mathrm{I}\,|\!\}$. Since $fn(\top) = \varnothing$, we apply Proposition 40 to $\{\!|\,\Gamma\,|\!\} \setminus \tilde{M}/\tilde{x} \models P \approx^A Q$ and infer $\{\!|\,\mathrm{I}\,|\!\} \models P \approx^A Q$, as desired. □

We have all the ingredients to prove the operation correspondence of the encoding $\langle\cdot\rangle$.

Theorem 89 (Operational Correspondence). *Let* Γ, I *be closed type environment such that* $\Gamma \vdash P$ *and* $\Gamma <: \mathrm{I}$. *The following hold.*

1. *If* $P \xrightarrow{\tau} P'$ *then* $\mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \overset{A}{\Longrightarrow} \longrightarrow \approx^A_{\{\mathrm{I}\}} \mathsf{CE}[\,\langle\, \Gamma \triangleright P' \,\rangle\,]$;
2. *If* $\{\!|\,\mathrm{I}\,|\!\} \models H \approx^A \mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,]$ *and* $H \longrightarrow K$ *then there exists* P' *such that* $P \xrightarrow{\tau} P'$ *and* $\{\!|\,\mathrm{I}\,|\!\} \models K \approx^A \mathsf{CE}[\,\langle\, \Gamma \triangleright P' \,\rangle\,]$.

Proof. Apply Propositions 83,87 and Lemma 88. □

The following corollaries establish a weak version of operational correspondence.

Corollary 90. *Let* Γ, I *be closed type environment such that* $\Gamma \vdash P$ *and* $\Gamma <: \mathrm{I}$. *If* $\mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \Longrightarrow H$ *then there exists* P' *such that* $P \Longrightarrow P'$ *and* $\{\!|\,\mathrm{I}\,|\!\} \models H \approx^A \mathsf{CE}[\,\langle\, \Gamma \triangleright P' \,\rangle\,]$.

Proof. By hypothesis there is $n \geq 0$ such that $\mathsf{CE}[\langle P \rangle](\overset{A}{\Longrightarrow}\longrightarrow)^n \overset{A}{\Longrightarrow} H$. We proceed by induction on n. If $n = 0$ then by Corollary 66 we have that $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{CE}[\langle P \rangle] \approx^A H$. We have thus found P' such that $P \Longrightarrow P'$ satisfying the equation above. Otherwise assume $\mathsf{CE}[\langle P \rangle](\overset{A}{\Longrightarrow}\longrightarrow)^n \overset{A}{\Longrightarrow} K \overset{A}{\Longrightarrow}\longrightarrow\overset{A}{\Longrightarrow} H$. By inductive hypothesis we have that there is Q such that $P \Longrightarrow Q$ and $\{\!|\, \Gamma \,|\!\} \models K \approx^A \mathsf{CE}[\langle Q \rangle]$. Let X, Y be such that $K \overset{A}{\Longrightarrow} X \longrightarrow Y \overset{A}{\Longrightarrow} H$. From Corollary 66 we infer $\{\!|\, \Gamma \,|\!\} \models K \approx^A X$ and by transitivity of $\approx^A$ we obtain $\{\!|\, \Gamma \,|\!\} \models X \approx^A \mathsf{CE}[\langle Q \rangle]$. We apply Theorem 89(2) to $\{\!|\, \Gamma \,|\!\} \models X \approx^A \mathsf{CE}[\langle Q \rangle] \wedge X \longrightarrow Y$ and we infer that there exists P' such that $Q \overset{\tau}{\longrightarrow} P'$ and $\{\!|\, \Gamma \,|\!\} \models Y \approx^A \mathsf{CE}[\langle P' \rangle]$. By closure of $\approx^A$ under administrative reductions we infer $\{\!|\, \Gamma \,|\!\} \models Y \approx^A H$. Therefore we have the chain: $\{\!|\, \Gamma \,|\!\} \models H \approx^A Y \approx^A \mathsf{CE}[\langle P' \rangle]$. From this equation we infer $\{\!|\, \Gamma \,|\!\} \models H \approx^A \mathsf{CE}[\langle P' \rangle]$. From the hypothesis $\mathsf{CE}[\langle P \rangle](\overset{A}{\Longrightarrow}\longrightarrow)^n \overset{A}{\Longrightarrow} K \overset{A}{\Longrightarrow}\longrightarrow\overset{A}{\Longrightarrow} H$ we infer the weaker $\mathsf{CE}[\langle P \rangle] \Longrightarrow H$. From $P \Longrightarrow P'$ and $\{\!|\, \Gamma \,|\!\} \models H \approx^A \mathsf{CE}[\langle P' \rangle]$ we obtain the proof of the corollary. □

Corollary 91. *Let* Γ, I *be closed type environment such that* $\Gamma \vdash P$ *and* $\Gamma <: \mathrm{I}$. *If* $P \Longrightarrow P'$ *then* $\mathsf{CE}[\langle \Gamma \triangleright P \rangle] \Longrightarrow \approx^A_{\{\mathrm{I}\}} \mathsf{CE}[\langle \Gamma \triangleright P' \rangle]$.

Proof. Standard, by induction on the length n of the derivation $P \Longrightarrow P'$. □

The last result we need before proving the soundness of the translation is preservation and reflection of high level barbs. Given a type environment I, we define testing processes of the form

$$T^n \;=\; \mathsf{link}\ (\underline{n}\Uparrow\mathsf{r}, \underline{y})\ \mathsf{in}\ (\mathsf{new}\ k)\mathsf{emit}(\{sk(k), \top\}_{y_r^+}) \,|\, \mathsf{filter}\ \tilde{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \omega\langle\rangle$$

We assume $\omega \notin fn(\{\!|\, \mathrm{I} \,|\!\})$. The following barb predicate $\mathrm{I} \models P \Downarrow_{n_w^+} \triangleq (n_r^+ \in A(\{\!|\, \mathrm{I} \,|\!\}) \wedge \{\!|\, \mathrm{I} \,|\!\}, \omega/x \models (T^n \,|\, P) \Downarrow_\omega$ correspond to pi calculus barb $\mathrm{I} \models P \Downarrow_n$, in the following sense.

Proposition 92. *The following hold.*

1. $\mathrm{I} \models P \downarrow_a$ *implies* $\mathrm{I} \models \mathsf{CE}[\langle \Gamma \triangleright P \rangle] \Downarrow_{a_w^+}$
2. $\mathrm{I} \models \mathsf{CE}[\langle \Gamma \triangleright P \rangle] \Downarrow_{a_w^+}$ *implies* $\mathrm{I} \models P \Downarrow_a$.

Proof. We first prove (1). By $\mathrm{I} \models P \downarrow_a$ we infer that (a) $\mathrm{I} \vdash a : \mathsf{r}$ and (b) $P \equiv C[a\langle b@B \rangle]$ for some name b, type B and context C that does not binds a. From (a) we deduce $a_r^+ \in A(\{\!|\, \mathrm{I} \,|\!\})$. From (b), similarly to the proof of of Proposition 83 (case (Pi-Close@)) we infer that there are H, D such that $T^a \,|\, \mathsf{CE}[\langle P \rangle] \Longrightarrow H$ and $H \equiv D[!\, net\langle\{\underline{b}\Uparrow\top\}_{sk(k)}\rangle \,|\, \mathsf{filter}\ \underline{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \omega\langle\rangle]$ where D does not bind ω. We therefore obtain $H \overset{\omega\langle\rangle}{\Longrightarrow}$ and in turn $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \models \mathsf{CE}[\langle P \rangle] \Downarrow_\omega$, as desired.
To see (2), suppose $\mathrm{I} \models \mathsf{CE}[\langle P \rangle] \Downarrow_{a_w^+}$. By definition we have that (a) $a_r^+ \in A(\{\!|\, \mathrm{I} \,|\!\}$ and (b) $\{\!|\, \mathrm{I} \,|\!\}, \omega/x \models (T^a \,|\, \mathsf{CE}[\langle P \rangle]) \Downarrow_\omega$. From $fn(P) \subseteq \mathrm{dom}(\mathrm{I})$ we infer

$fn(P) \subseteq \mathrm{dom}(\{\!|\,\mathrm{I}\,|\!\})$. From $\omega \notin fn(\{\!|\,\mathrm{I}\,|\!\})$ and the result above we conclude that $\omega \notin fn(\mathsf{CE}[\,\langle\, P \,\rangle\,])$. Therefore the barb on ω comes from T^a and it is easy to see that the barb is unblocked iff the filter receives a packet $!\,net\langle\{M\}_{sk(k)}\rangle$ with M having consistent arity and k being the fresh name generated by T^a itself. A case analysis shows that such packet must be generated by the channel server a' linked to a upon a $(a', \top)$ synchronization. Since such synchronization may occur iff a proxy write request on a is followed by a write request on a', we infer that $P \Longrightarrow C[a\langle b@B\rangle]$ for some name b, type B and context C that does not binds a. From (a) we infer $\mathrm{I} \vdash a : \mathsf{r}$. We glue the results and infer $\mathrm{I} \models P \Downarrow_a$, as desired. □

The next Theorem ensures that the translation equates only high level equivalent processes.

Theorem 93 (Soundness). *Let* Γ, Δ *and* I *be closed type environments s.t.* $\Gamma, \Delta <: \mathrm{I}$. *We have that* $\{\!|\,\mathrm{I}\,|\!\} \models \mathsf{CE}[\,\langle\, \Gamma \triangleright P \,\rangle\,] \cong^{A\pi} \mathsf{CE}[\,\langle\, \Delta \triangleright Q \,\rangle\,]$ *implies* $\mathrm{I} \models P \cong^{\pi} Q$.

Proof. We consider the relation defined as $\mathrm{I} \vDash P \mathscr{R} Q$ whenever $\{\!|\,\mathrm{I}\,|\!\} \models \mathsf{CE}[\,\langle\, P \,\rangle\,]] \cong^{A\pi} \mathsf{CE}[\,\langle\, Q \,\rangle\,]$, and we prove that $\mathscr{R}$ is a typed behavioural equivalence. Suppose $\mathrm{I} \vDash P \mathscr{R} Q$. We first prove that $\mathscr{R}$ is reduction closed by using the operational correspondence of the encoding (see Theorem 89). Let $P \xrightarrow{\tau} P'$. We apply Theorem 89(1) and find K s.t. $\mathsf{CE}[\,\langle\,\langle\, P \,\rangle\,\rangle\,] \overset{A}{\Longrightarrow\!\!\longrightarrow} K$ and $\{\!|\,\mathrm{I}\,|\!\} \models K \approx^{A} \mathsf{CE}[\,\langle\, P' \,\rangle\,]$. From $\approx^{A} \subseteq \cong^{A\pi}$ (see Cor. 62) we have $\{\!|\,\mathrm{I}\,|\!\} \models K \cong^{A\pi} \mathsf{CE}[\,\langle\, P' \,\rangle\,]$. By definition of $\mathscr{R}$ we deduce that there exists H s.t. $\mathsf{CE}[\,\langle\, Q \,\rangle\,] \Longrightarrow H$ and $\{\!|\,\mathrm{I}\,|\!\} \models K \cong^{A\pi} H$. We apply Corollary 90 to the last weak transition and we obtain that there exists Q' s.t. $Q \Longrightarrow Q'$ and $\{\!|\,\mathrm{I}\,|\!\} \models H \approx^{A} \mathsf{CE}[\,\langle\, Q' \,\rangle\,]$. By $\approx^{A} \subseteq \cong^{A\pi}$ we have $\{\!|\,\mathrm{I}\,|\!\} \models H \cong^{A\pi} \mathsf{CE}[\,\langle\, Q' \,\rangle\,]$.
From the results above we build the following chain: $\{\!|\,\mathrm{I}\,|\!\} \models \mathsf{CE}[\,\langle\, P' \,\rangle\,] \cong^{A\pi} K \cong^{A\pi} H \cong^{A\pi} \mathsf{CE}[\,\langle\, Q' \,\rangle\,]$. From transitivity of $\cong^{A\pi}_{\{\mathrm{I}\}}$ (Prop. 34) we conclude that $\{\!|\,\mathrm{I}\,|\!\} \models \mathsf{CE}[\,\langle\, P' \,\rangle\,] \cong^{A\pi} \mathsf{CE}[\,\langle\, Q' \,\rangle\,]$, which in turn implies $\mathrm{I} \models P' \mathscr{R} Q'$, as needed.

Barb preservation is obtained directly by using Proposition 92. Let $\mathrm{I} \models P \downarrow_a$; thus $\mathrm{I}(a) <: \mathsf{r}$ and in turn $a_r^+ \in \mathrm{Range}(\{\!|\,\mathrm{I}\,|\!\})$. By the contextuality of $\cong^{A\pi}$, we find $\omega \notin fn(\{\!|\,\mathrm{I}\,|\!\})$ s.t. $\{\!|\,\mathrm{I}\,|\!\}, \omega/x \models \mathsf{CE}[\,\langle\, P \,\rangle\,] \cong^{A\pi} \mathsf{CE}[\,\langle\, Q \,\rangle\,]$. Now we easily find T s.t. $\{\!|\,\mathrm{I}\,|\!\}, \omega/x \vdash T$ and $T^a \equiv T(\{\!|\,\mathrm{I}\,|\!\}, \omega/x)$. By closure under parallel composition of $\cong^{A\pi}$ we infer $\{\!|\,\mathrm{I}\,|\!\}, \omega/x \models \mathsf{CE}[\,\langle\, P \,\rangle\,] \,|\, T^a \cong^{A\pi} \mathsf{CE}[\,\langle\, Q \,\rangle\,] \,|\, T^a$. By Proposition 92(1) we have $\mathrm{I} \models \mathsf{CE}[\,\langle\, P \,\rangle\,] \Downarrow_{a_w^+}$, that is $\{\!|\,\mathrm{I}\,|\!\}, \omega/x \models (\mathsf{CE}[\,\langle\, P \,\rangle\,] \,|\, T^a) \Downarrow_\omega$. By definition of $\mathscr{R}$ we have that $\{\!|\,\mathrm{I}\,|\!\}, \omega/x \models (\mathsf{CE}[\,\langle\, Q \,\rangle\,] \,|\, T^a) \Downarrow_\omega$. This together with $a_r^+ \in \mathrm{Range}(\{\!|\,\mathrm{I}\,|\!\})$ let us infer $\mathrm{I} \models \mathsf{CE}[\,\langle\, Q \,\rangle\,] \Downarrow_{a_w^+}$. We apply Prop. 92(2) and we obtain $\mathrm{I} \models Q \Downarrow_a$, as needed.

Finally we show that $\mathscr{R}$ is contextual. Let $\mathrm{I}, a : A \models P \mathscr{R} Q$ and let $\Gamma \vdash P, \Delta \vdash Q$, where $\Gamma <: \mathrm{I}, a : A$ and $\Delta <: \mathrm{I}, a : A$. We need to prove that $\mathrm{I} \models (\mathsf{new}\ a : A)P \mathscr{R} (\mathsf{new}\ a : A)Q$. By definition of $\mathscr{R}$ we have that $\{\!|\,\mathrm{I}, a : A\,|\!\} \setminus a \models (\mathsf{new}\ a)\mathsf{CE}[\,\langle\, P \,\rangle\,]] \cong^{A\pi} (\mathsf{new}\ a)\mathsf{CE}[\,\langle\, Q \,\rangle\,]$. It is easy to see that

$\{\!|\, \mathrm{I}, a : A \,|\!\} \setminus a = \{\!|\, \mathrm{I} \,|\!\}$. By using new axiom of structural congruence and the definition $\langle\, (\mathsf{new}\ a : T)P \,\rangle = (\mathsf{new}\ a)\langle\, P \,\rangle$ we infer $\{\!|\, \mathrm{I} \,|\!\} \setminus a \models \mathsf{CE}[\langle\, (\mathsf{new}\ a : A)P \,\rangle] \cong^{A\pi} \mathsf{CE}[\langle\, (\mathsf{new}\ a : A)Q \,\rangle]$. We conclude that $\mathrm{I} \models (\mathsf{new}\ a : A)P \mathscr{R} (\mathsf{new}\ a : A)Q$, as desired. For the second clause, let $\mathrm{I} \models P \mathscr{R} Q$ and let $\mathrm{I} \vdash R$. We need to show that $\mathrm{I} \models P \,|\, R \mathscr{R} Q \,|\, R$. From Lemma 82 we infer that there exists R^* closed by $\{\!|\, \mathrm{I} \,|\!\}$ such that $\langle\, R \,\rangle \equiv_{\{\!|\mathrm{I}|\!\}} R^*\{\!|\, \mathrm{I} \,|\!\}$ and in turn $\rho \models \langle\, R \,\rangle \cong^{A\pi} R^*\{\!|\, \mathrm{I} \,|\!\}$ (from $\equiv_{\{\!|\mathrm{I}|\!\}} \subseteq \approx^A \subseteq \cong^{A\pi}$). From contextuality of $\cong^{A\pi}$ we infer $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{CE}[\langle\, P \,\rangle] \,|\, R^*\{\!|\, \mathrm{I} \,|\!\} \cong^{A\pi} \mathsf{CE}[\langle\, Q \,\rangle] \,|\, R^*\{\!|\, \mathrm{I} \,|\!\}$ and with minor structural rearrangements we obtain $\{\!|\, \mathrm{I} \,|\!\} \models \mathsf{CE}[\langle\, P \,|\, R \,\rangle] \cong^{A\pi} \mathsf{CE}[\langle\, Q \,|\, R \,\rangle]$. Therefore $\mathrm{I} \models P \,|\, R \mathscr{R} Q \,|\, R$, as desired.
The last clause requires that $\mathrm{I} \models P \mathscr{R} Q$ and $a \notin fn(\mathrm{I})$ implies $\mathrm{I}, a : A \models P \mathscr{R} Q$. By definition of $\mathscr{R}$ and contextuality of $\cong^{A\pi}$ we have that $\{\!|\, \mathrm{I} \,|\!\}, a/x \models \mathsf{CE}[\langle\, P \,\rangle] \cong^{A\pi} \mathsf{CE}[\langle\, Q \,\rangle]$. We add entries $\underline{a} \Uparrow \underline{A}/\underline{x}$ to the environment by using Proposition 40. We obtain: $\{\!|\, \mathrm{I} \,|\!\}, a/x, \underline{a} \Uparrow \underline{A}/\underline{x} \models \mathsf{CE}[\langle\, P \,\rangle] \cong^{A\pi} \mathsf{CE}[\langle\, Q \,\rangle]$. We remove the entry a/x by using Proposition 41: $\{\!|\, \mathrm{I} \,|\!\}, \underline{a} \Uparrow \underline{A}/\underline{x} \models \mathsf{CE}[\langle\, P \,\rangle] \cong^{A\pi} \mathsf{CE}[\langle\, Q \,\rangle]$. These results let us deduce $\{\!|\, \mathrm{I}, a : A \,|\!\} \models \mathsf{CE}[\langle\, P \,\rangle] \cong^{A\pi} \mathsf{CE}[\langle\, Q \,\rangle]$ and in turn $\mathrm{I}, a : A \models P \mathscr{R} Q$, as needed. □

5.5 Full abstraction theorem

We introduce a encoding which extends $\langle\, \cdot \,\rangle$ by considering arbitrary representations of term capabilities of a given type. To motivate, notice that the environment can legitimately create terms capabilities that are not encryption keys and send it to system processes. However, the implementation uses these terms only as a token to receive system generated encryption keys. At the high level we represent term capabilities with typings and we keep the association among them with an injective function $\beta = \{(a : A, N), \dots\}$ from typings to terms such that the type of N is $\underline{A}$ and such that the domain of β is a type environment. The encoding of values $\underline{u}_\beta$ maps values u such that $u : A$ is in the domain of β to the image of $u : A$ in β:

$$\underline{u}_\beta = \begin{cases} \beta(u : A) & u : A \in \mathrm{dom}(\beta) \\ \underline{u} & \text{else} \end{cases}$$

The definition of $\langle\, \Gamma \triangleright P \,\rangle_\beta$ is obtained by substituting the encoding of values $\underline{v}$ arising in the definition $\langle\, \Gamma \triangleright P \,\rangle$ with $\underline{v}_\beta$. In the following we will abbreviate $\langle\, \Gamma \triangleright P \,\rangle_\beta$ with $\langle\, P \,\rangle_\beta$ whenever the environment Γ is unnecessary.

We first introduce a lemma useful to prove the theorem of completeness.

Lemma 94. *Let ρ be an environment such that $k_p, k_p^- \notin A(\rho)$ and assume $\rho \vdash C$. Suppose $\mathsf{CE}[\langle\, P \,\rangle_\beta] \stackrel{A}{\Longrightarrow} H$ and $C[H] \stackrel{\tau}{\longrightarrow} K$ is inferred from $C \stackrel{a(M)}{\longrightarrow} C'\{M/x\}$ and $H \stackrel{(\tilde{c})a\langle M \rangle}{\longrightarrow} H'$ and $K \equiv (\mathsf{new}\ \tilde{c})C'\{M/x\}[H']$. Then we have:*

$$\rho \models K \approx^A (\mathsf{new}\ n)!\, net\langle \{n\}_n \rangle \,|\, C'\{\{n\}_n/x\}[H] \ .$$

Proof. By case analysis on $H \xrightarrow{(\tilde{c})a\langle M\rangle} H'$. By syntactic analysis of the encoding $\langle\cdot\rangle^{\beta}$, we infer that this interaction has occurred on the channel *net*, which is the only free name used as channel in the encoding $\langle\cdot\rangle^{\beta}$. A case analysis on $\langle\cdot\rangle^{\beta}$ shows that there is an environment configuration $\mathsf{CE}^*[-]$ and a process P^* such that such that

$$H \equiv (\mathsf{new}\ \Delta, \tilde{c})\mathsf{CE}^*[(P^* \mid !\, net\langle M\rangle)] \quad H' \equiv (\mathsf{new}\ \Delta, \tilde{c})\mathsf{CE}^*[P^* \mid !\, net\langle M\rangle]$$

Here Δ are the system-generated names used to build trusted channel servers and CE^*, P^* are obtained by interactions of $\langle P\rangle^{\beta}$ with CE. Moreover, a case analysis shows that all emissions on *net* occurring in the encoding are encrypted by using symmetric or asymmetric schemes with the following properties: symmetric keys are generated around fresh names, asymmetric packets contain a fresh nonce. This let us infer that two cases arise for M:

1. $M = \{\tilde{N}\}_{sk(k)}$ and $k \in \tilde{c}$
2. $M = \{\tilde{N}, c\}_{ek(N)}$ and $c \in \tilde{c}$ and $\{dk(N), N\} \cap A(\rho) = \varnothing$

Case (1) says that the session key used for symmetric encryption is fresh. Case (2) says that the nonce is fresh and that the environment does not knows the terms $dk(N), N$ that could be used to open the crypto-packet. To see that, notice that $ek(N)$ or is built around the proxy public key as in $N = k_P$, or is built around a system-generated seed as in $wr(n), rd(n)$ with $n \in \Delta$.

In case (1) we use Corollary 70 and we infer the stronger $\rho \models K \sim^{A} (\mathsf{new}\ n)\, C'\{\{n\}_n/x\}[H]$. In case (2), we infer that there are a process Q and a term T such that (a) exists $n \in \Delta$ such that $N = rd(n), wr(n)$ and $\mathsf{CE}^*[-]$ contains $Chan_n\{T\}[Q]$ and Q satisfies the conditions of Corollary 74 or (b) $N = k_p$ and $\mathsf{CE}^*[-]$ contains $Proxy_N\{T\}[Q]$. In sub-case (a) we use Corollary 74 and infer the desired result, $\rho \models K \approx^{A} (\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid C'\{\{n\}_n/x\}[H]$. In sub-case (b) we use Corollary 77 and we infer the desired result. □

The following theorem ensures that the forward direction of Theorem 46 actually holds for the particular $\beta = \emptyset$.

Theorem 95 (Completeness). *Let* Γ, Δ *and* I *be closed type environment such that* $\Gamma, \Delta <: \mathrm{I}$. *Assume* $\mathrm{dom}(\beta) \subseteq \mathrm{I}$. *Then* $\mathrm{I} \vDash P \cong Q$ *implies* $\{\!|\,\mathrm{I}\,|\!\} \models \mathsf{CE}[\langle\, \Gamma \triangleright P\,\rangle_{\beta}] \cong^{A\pi} \mathsf{CE}[\langle\, \Delta \triangleright Q\,\rangle_{\beta}]$.

Proof. To build our candidate relation, we consider computing environments of the form

$$\mathsf{CE}^*[-] \triangleq - \mid W \mid Chan_{\Delta}\{\tilde{N}\}[S] \mid Proxy_{\tilde{M};\Delta}\{M\}[T]$$

The proxy configuration $Proxy_{\tilde{M};\Delta}\{M\}[T]$ associates client terms in $\tilde{M}$ to their server counter-part Δ and contains proxy requests T of the context. Formally the

proxy map contains entries of the form $(M_{ID}, \underline{n})$ where $n \in \Delta$ and $M \in \tilde{M}$. Notice that the scope of Δ is outside this definition: the context may have indeed received capabilities built around Δ due to a linking request. The channel servers $Chan_\Delta$ created by the proxy contain processes S generated by read/write requests submitted by the context and their administrative redexes.

In the candidate relation the computing environment is surrounded by a context representing the low-level knowledge of I and containing the capabilities generated apart Δ sended by the proxy. As usual, we augment $\{\!|\, \mathrm{I} \,|\!\}$ with fresh names $\tilde{b}$ and we let the index of the relation to be a subset of this base to ensure closure under weakening. We require that names $\tilde{b}$ do not clash with the free names of P, Q.

Finally encoded processes up to β are immersed in the computing environment. We will use the β clause in the sub-case whether the context send a packet to a process (see the intrusion paragraph).

Given these intuitions, we let the candidate relation $\mathscr{R}$ be defined as:

$$\rho \models C[\mathsf{CE}^*[\,\langle\, P \,\rangle_\beta]] \mathscr{R} C[\mathsf{CE}^*[\,\langle\, Q \,\rangle_\beta]])$$

whenever

1. $\mathrm{I} \models P \cong Q$
2. $(a : A) \in \mathrm{dom}(\beta)$ implies $\mathrm{I}(a) <: A$
3. $\rho \subseteq (\{\!|\, \mathrm{I} \,|\!\}, \tilde{b}/\tilde{x}) \wedge \tilde{b} \cap fn(\{\!|\, \Gamma \,|\!\}) = \emptyset$ where $\Gamma = \mathrm{I} \cap fn(P, Q)$
4. $C[-] = (\mathsf{new}\ \Delta, \tilde{c})(R(\{\!|\, \mathrm{I}, \Delta \,|\!\}, \tilde{b}/\tilde{x}) \,|\, -)$ and $(\{\!|\, \mathrm{I}, \Delta \,|\!\}, \tilde{b}/\tilde{x}) \vdash R$
5. $\mathsf{CE}^*[-] = - \,|\, W \,|\, Chan_\Delta\{\tilde{N}\}[S] \,|\, Proxy_{\tilde{M}}\{M\}[T]$

We prove that $\mathscr{R}$ is a weak equivalence up to administrative equivalence. We use the following naming conventions: $C[-]$ is the context, $\mathsf{CE}^*[-]$ is the environment, what fills the hole of the environment is the process. We let $H = C[\mathsf{CE}^*[\,\langle\, P \,\rangle^\beta]]$ and $K = C[\mathsf{CE}^*[\,\langle\, Q \,\rangle^\beta]]$.

Barb preservation. Let $\rho \models H \downarrow_a$. If this barb is inferred from $\rho \models \mathsf{CE}^*[\,\langle\, P \,\rangle_\beta] \downarrow_a$, a case analysis shows that $a = net$. We match this barb with $\rho \models K \Downarrow_a$ inferred from $\rho \models W \downarrow_{net}$. Otherwise $\rho \models C \downarrow_a$ and we trivially have $\rho \models K \Downarrow_a$.

Reduction Closure We need to check the cases $H \overset{A}{\Longrightarrow} H'$ and $H \overset{A}{\Longrightarrow} \longrightarrow \overset{A}{\Longrightarrow} H'$ of the definition of equivalence up to administrative equivalence.

Let $H \overset{A}{\Longrightarrow} H'$. By Proposition 65 we have $\{\!|\, \mathrm{I} \,|\!\} \models H \approx^A H'$ and we have done since $\rho \models H' \approx^A H \mathscr{R} K$.

Otherwise let $H \stackrel{A}{\Longrightarrow} Y \longrightarrow Y' \stackrel{A}{\Longrightarrow} H'$. We find K' s.t. $K \Longrightarrow K'$ and $\rho \models Y' \approx^A \mathscr{R} \approx^A K'$. The desired result, $\rho \models H' \approx^A \mathscr{R} \approx^A K'$, will follow since $\approx^A$ is closed under administrative reductions and is transitive.
Consider $H \stackrel{A}{\Longrightarrow} Y \longrightarrow Y'$. Then there is a canonical sequence to Y^* such that $H \stackrel{\ulcorner A \urcorner}{\Longrightarrow} Y^* \longrightarrow\!\!\stackrel{A}{\Longrightarrow} Y'$. We let

$$H \stackrel{\ulcorner A* \urcorner}{\Longrightarrow} J$$

be obtained by replacing in $H \stackrel{\ulcorner A \urcorner}{\Longrightarrow} Y^*$ any reduction inferred from the environment receiving a packet from the process, with a reduction inferred by the environment receiving a packet from the noise. To illustrate, assume $H \stackrel{\ulcorner A \urcorner}{\Longrightarrow} H^* \stackrel{A}{\longrightarrow} H_1^* \stackrel{A}{\Longrightarrow} Y^*$ and $H^* \stackrel{A}{\longrightarrow} H_1^*$ is obtained by the hypothesis above. We have that for some processes E, R and term M holds $H^* \equiv E[W \,|\, !net\langle M\rangle \,|\, net(x).R]$ and $H_1^* \equiv E[W \,|\, !net\langle M\rangle \,|\, R\{M/x\}]$. In such case we find H_2^* such that $H_1^* \stackrel{A}{\longrightarrow} E[W \,|\, !net\langle M\rangle \,|\, (\mathsf{new}\ n)\mathbf{!}\, net\langle\{n\}_n\rangle \,|\, R\{\{n\}_n/x\}]$. Notice that the process's output is inaffected by this operation, because is replicated. By repeated applications of Lemma 94 we obtain that $\rho \models J \approx^A Y^*$. From this we infer that there is J' such that $J \longrightarrow J' \stackrel{A}{\Longrightarrow}\!\sim^A Y'$. Also notice that no interactions among the context and the process arise in $H \stackrel{\ulcorner A* \urcorner}{\Longrightarrow} J$. Indeed the canonical sequence does not contain reductions inferred from the context sending terms to the process, i.e. this reduction is not canonical for any J' such that $J \longrightarrow J'$. This holds since the free outputs occurring in the translation are non-blocking and since the free inputs are those of filter processes that wait packets encrypted under a session key not known to the context.

We proceed by analysis of possible cases of interaction of $J \longrightarrow J'$.

(Context - Process). This is the case whether $J \longrightarrow J'$ is inferred from (a) the context receiving a term from the process or (b) the process receiving a term from the context. By syntactic analysis of the encoding $\langle\cdot\rangle^\beta$, we infer that this interaction has occurred on the channel *net*, which is the only free name used as channel in the encoding.

We first analyze case (a). We have that there is CE_1^* such that $J \equiv C'[net(x).R \,|\, (\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[P^* \,|\, \mathbf{!}\, net\langle M\rangle]]$ and $J' \equiv C'[(\mathsf{new}\ \tilde{b})R\{M/x\} \,|\, \mathsf{CE}_1^*[P^* \,|\, \mathbf{!}\, net\langle M\rangle]]$. As shown in Lemma 94, all outputs coming out from the translation are emissions of packets (a) encrypted under fresh symmetric keys or (b) encrypted with asymmetric keys and containing a fresh nonce; therefore these packets are seen as noise by the context. We apply the Lemma and obtain $\rho \models J' \approx^A C'[(\mathsf{new}\ n)\mathbf{!}\, net\langle\{n\}_n\rangle \,|\, R\{\{n\}_n/x\} \,|\, (\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[P^* \,|\, \mathbf{!}\, net\langle M\rangle]]$. Next, we unroll canonical administrative interactions among $\langle P\rangle$ and CE. There is CE_0^* obtained by interactions of CE with C such that

$$H \stackrel{\ulcorner A \urcorner}{\Longrightarrow} C'[net(x).R \,|\, \mathsf{CE}_0^*[\langle P\rangle^\beta]] \longrightarrow$$
$$C'[(\mathsf{new}\ n)\mathbf{!}\, net\langle\{n\}_n\rangle \,|\, R\{\{n\}_n/x\} \,|\, \mathsf{CE}_0^*[\langle P\rangle^\beta]] \stackrel{A}{\Longrightarrow} J'$$

We match this move with

$$K \Longrightarrow C'[net(x).R \mid \mathsf{CE}_0^*[\,\langle\, Q \,\rangle^{\,\beta}]] \xrightarrow{\tau} C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, Q \,\rangle^{\,\beta}]]$$

By using closure of administrative equivalence under administrative reductions we glue the results obtaining:

$$\begin{array}{c}\rho \models Y' \approx^A J' \approx^A C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, P \,\rangle^{\,\beta}]] \\ \mathscr{R}\ C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, Q \,\rangle^{\,\beta}]]\end{array}$$

Now suppose case (b) holds. A case analysis shows that all input on *net* of the translation of P occur in processes filtering packet under a session key not known to the context. We have that there is CE_1^* such that $J \equiv C'[net\langle M\rangle \mid \mathsf{CE}_1^*[(\mathsf{new}\ k)P^* \mid \mathsf{filter}\ \tilde{x}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P]]$ and $J' \equiv C'[\mathsf{CE}_1^*[(\mathsf{new}\ k)P^* \mid \mathsf{filter}\ M\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P]$. We easily obtain $J' \xrightarrow{A} J$ by rolling back the filter to its initial state. Next, we unroll canonical administrative interactions among $\langle\, P \,\rangle$ and CE: there is CE_0^* obtained by interactions of CE with C such that:

$$H \overset{\ulcorner A \urcorner}{\Longrightarrow} C'[net\langle M\rangle \mid \mathsf{CE}_0^*[\,\langle\, P \,\rangle^{\,\beta}]] \overset{A}{\Longrightarrow} J$$

We match this move with

$$K \Longrightarrow C'[net\langle M\rangle \mid \mathsf{CE}_0^*[\,\langle\, Q \,\rangle^{\,\beta}]]$$

We obtain :

$$\rho \models Y' \approx^A J' \approx^A J \approx^A C'[net\langle M\rangle \mid \mathsf{CE}_0^*[\,\langle\, P \,\rangle^{\,\beta}]] \mathscr{R} C'[net\langle M\rangle \mid \mathsf{CE}_0^*[\,\langle\, Q \,\rangle^{\,\beta}]]\ .$$

(Context - Environment). This is the case whether $J \longrightarrow J'$ is inferred from (i) the context sending a packet to the environment on the channel *net* or (ii) the context receiving a packet from the noise generator W. To see that no other cases are possible, notice that, as we have seen in the proof for barb preservation, the environment does not contain free outputs but the noise. We stress that an emission containing capabilities generated by the proxy is considered part of the context when the request is due to the context, and part of the process when the answer is generated upon a process's request.

We first analyze case (i). We have that there is CE_1^* such that $J \equiv C'[net\langle M\rangle \mid \mathsf{CE}_1^*[P^*]]$ and $J' \equiv C'[\mathsf{CE}_{1,\mathsf{M}}^*[P^*]]$ where $\mathsf{CE}_1^* \xrightarrow{net(M)} \mathsf{CE}_{1,\mathsf{M}}^*$. Next, we unroll canonical administrative interactions among $\langle\, P \,\rangle$ and CE. There is CE_0^* obtained by interactions of CE with C such that

$$H \overset{\ulcorner A \urcorner}{\Longrightarrow} C'[net\langle M\rangle \mid \mathsf{CE}_0^*[\,\langle\, P \,\rangle^{\,\beta}]] \longrightarrow C'[\mathsf{CE}_{0,\mathsf{M}}^*[\,\langle\, P \,\rangle^{\,\beta}]] \overset{A}{\Longrightarrow} J'$$

where $\mathsf{CE}_0^* \xrightarrow{net(M)} \mathsf{CE}_{0,\mathsf{M}}^*$.

We match this move with

$$K \Longrightarrow C'[net\langle M\rangle \mid \mathsf{CE}_0^*[\,\langle\, P\,\rangle^{\,\beta}]] \xrightarrow{\tau} C'[\mathsf{CE}_{0,\mathsf{M}}^*[\,\langle\, Q\,\rangle^{\,\beta}]]$$

By using closure of administrative equivalence under administrative reductions we glue the results obtaining:

$$\rho \models Y' \approx^A J' \approx^A C'[\mathsf{CE}_{0,\mathsf{M}}^*[\,\langle\, P\,\rangle^{\,\beta}]] \mathscr{R} C'[\mathsf{CE}_{0,\mathsf{M}}^*[\,\langle\, Q\,\rangle^{\,\beta}]] \ .$$

In case (ii) we infer that there is CE_0^* obtained by interactions of CE with C such that $J \equiv C'[net(x).R \mid \mathsf{CE}_0^*[\,\langle\, P\,\rangle^{\,\beta}]]$ and $J' \equiv C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, P\,\rangle^{\,\beta}]]$. We match this reduction with

$$K \Longrightarrow C'[net(x).R \mid \mathsf{CE}_0^*[\,\langle\, Q\,\rangle^{\,\beta}]] \xrightarrow{\tau} C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, Q\,\rangle^{\,\beta}]]]$$

We obtain:

$$\begin{array}{c}\rho \models Y' \approx^A J' \equiv C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, P\,\rangle^{\,\beta}]] \\ \mathscr{R} C'[(\mathsf{new}\ n)!\, net\langle\{n\}_n\rangle \mid R\{\{n\}_n/x\} \mid \mathsf{CE}_0^*[\,\langle\, Q\,\rangle^{\,\beta}]]\end{array}$$

Process reduction. This is the case whether we have $J \xrightarrow{n@t} J'$ and the reduction is inferred from a read request and a write request of the process. In such case no previous interaction with the context is needed. We have that there is CE_1^* such that $J \equiv C[(\mathsf{new}\ \tilde{c})\mathsf{CE}_1^*[P^*] \xrightarrow{n@t} J'$. We apply reflection of execution steps (in the canonical form of Lemma 86) to $H \xRightarrow{\ulcorner A \urcorner} J \xrightarrow{n@t} J'$ and we infer that there is a pi calculus process P' such $P \xrightarrow{\tau} P'$ and $\rho \models J' \approx^A C[\mathsf{CE}^*[\,\langle\, P'\,\rangle^{\,\beta}]]$. We exploit the hypothesis $\mathrm{I} \models P \cong Q$ to infer that there is Q' such that $Q \Longrightarrow Q'$ with $\mathrm{I} \models P' \cong Q'$. We use preservation of execution steps (Corollary 91) and infer that there is K' such that $K \Longrightarrow K'$ and $\rho \models K' \approx^A C[\mathsf{CE}^*[\,\langle\, Q'\,\rangle^{\,\beta}]]$. We have done as

$$\rho \models Y' \approx^A J' \approx^A C[\mathsf{CE}^*[\,\langle\, P'\,\rangle^{\,\beta}]] \mathscr{R} C[\mathsf{CE}^*[\,\langle\, Q'\,\rangle^{\,\beta}]] \approx^A K' \ .$$

Intrusion. This is the case whereas the encoding of P receives a term from the environment and this term was inserted in the queue of a channel by the context. We have $P \equiv (\mathsf{new}\ \tilde{b})P_2 \mid a(y@T).P_1$ and $\mathrm{I} \vdash a : \mathsf{w}$. Moreover, the intrusion hypothesis let us infer that there is CE_1^* such that

$$\begin{array}{rcl} J & \equiv & C'[(\mathsf{new}\ \tilde{b}, \tilde{c})\mathsf{CE}_1^*[\,\langle\, P_2\,\rangle \mid P^* \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle\, P_1\,\rangle^{\,\beta} \\ \mathsf{CE}_1^*[-] & \equiv & -\mid W \mid Chan_{\Delta'}\{\tilde{N}\}[R \mid \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ a_1^\circ@T\ \mathsf{in}\ !\, net\langle\{\tilde{y}\}_{sk(k)}\rangle \\ & & \mid a_1^\circ\langle N\rangle] \mid Proxy_{\tilde{M};\Delta}\{M\}[R] \end{array}$$

Here a_1 is the server counterpart of a, and the "type " of N is some $\underline{S}$ such that $\underline{S} <: \underline{T}$. The bindings $\tilde{c}$ are formed by the seed k of the shared key $sk(k)$ and the

nonce c contained in the input request of the encoding of $a(x@T)$; P^* is the leftover of this request.

Process J' is obtained by the (successful) synchronization of the two threads in the environment; this is because of the hypothesis $\underline{S} <: \underline{T}$. We let the environment $\mathsf{CE}^*_{0,\mathsf{c}}$ differ from CE^* in that the nonce c is stored and the name a is registered (it may be registered as well in CE^*).

$$J \xrightarrow{a_1@T} J' \equiv C'[(\mathsf{new}\ \tilde{b},\tilde{c})\mathsf{CE}^*_{0\mathsf{c}}[\,\langle\, P_2 \,\rangle \mid P^* \mid !\, net\langle\{N{\Uparrow}\underline{T}\}_{sk(k)}\rangle \mid \mathsf{filter}\ \tilde{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ \langle\, P_1 \,\rangle^{\beta}$$

By letting the continuation of P to receive the emission received from the channel and by removing the leftover of the communication with Lemmas 69,73,75,78 we obtain:

$$J' \overset{A}{\Longrightarrow}\approx^A_\rho C'[\mathsf{CE}^*_0[(\mathsf{new}\ \tilde{b})\,\langle\, P_2 \,\rangle^{\beta} \mid \langle\, P_1 \,\rangle^{\beta}\{N{\Uparrow}\underline{T}/\tilde{y}\}$$

The environment CE^*_0 differs from $\mathsf{CE}^*_{0\mathsf{c}}$ in that the nonce c is not saved.

Now two cases arise corresponding to (i) $N = \underline{a}{\Uparrow}\underline{T'}$ for some a in I or (ii) not. We let $\beta' = \beta$ in case (i), else $\beta, = \beta, (n : S, N)$ with $n \notin \mathrm{dom(I)}$. We use Lemma 80 and closure of administrative equivalence under administrative reductions and infer

$$\rho \models J' \approx^A C'[\mathsf{CE}^*_0[\,\langle\, (\mathsf{new}\ \tilde{b})P_2 \mid P_1\{n/y\} \,\rangle^{\beta'}]]$$

By the hypothesis $\mathrm{I} \models P \cong Q$ we infer that (i) if $n \in \mathrm{dom(I)}$ then $\mathrm{I} \models a\langle n@S\rangle \mid P \cong a\langle n@S\rangle \mid Q$ else (ii) $\mathrm{I}, n : \mathsf{rw} \models a\langle n@S\rangle \mid P \cong a\langle n@S\rangle \mid Q$. Since $a\langle n@S\rangle \mid P \xrightarrow{\tau} P' \triangleq (\mathsf{new}\ \tilde{b})P_2 \mid P_1\{n/x\}$, by hypothesis there is Q' s.t. $a\langle n@S\rangle \mid Q \Longrightarrow Q'$ and (i) $\mathrm{I} \models P' \cong Q'$ or (ii) $\mathrm{I}, n : \mathsf{rw} \models P' \cong Q'$. We apply preservation of execution steps (Corollary 91) to the process $\langle\, a\langle n@S\rangle \mid Q \,\rangle^{\beta'}$ immersed in $C'[\mathsf{CE}^*_0[-]]$ and obtain that there is K^* such that

$$C'[\mathsf{CE}^*_0[\,\langle\, a\langle n@S\rangle \mid Q \,\rangle^{\beta'}]] \Longrightarrow K^* \tag{5.1}$$

and $\rho \models K^* \approx^A C'[\mathsf{CE}^*_0[\,\langle\, Q' \,\rangle^{\beta'}]]$.

Given $H \overset{\ulcorner A^* \urcorner}{\Longrightarrow} J$, and that J contains the message $a\langle \underline{n_{\beta'}}{\Uparrow}\underline{S}\rangle$ that by hypothesis has not been inserted by the encoding of P, we infer that the same message $a\langle \underline{n_{\beta'}}{\Uparrow}\underline{S}\rangle$ can be produced on the right side. With some calculations we infer that:

$$K \equiv C[\mathsf{CE}^*[\,\langle\, Q \,\rangle^{\beta}]] \overset{A}{\Longrightarrow}\approx^A_\rho \overset{A}{\Longleftarrow} C'[\mathsf{CE}^*_0[\,\langle\, a\langle n@S\rangle \mid Q \,\rangle^{\beta'}]]$$

Administrative equivalence is used to remove the leftover of the process write request, as usual. Therefore by closure of $\approx^A$ under administrative reductions we obtain

$$C[\mathsf{CE}^*[\,\langle\, Q \,\rangle^{\beta}]] \approx^A_\rho C'[\mathsf{CE}^*_0[\,\langle\, a\langle n@S\rangle \mid Q \,\rangle^{\beta'}]]$$

This together with equation (5.1) let us infer that there is K' such that

$$K \Longrightarrow K' \approx_{\rho}^{A} K^{*}$$

We finally obtain:

$$\rho \models Y' \approx^{A} J' \approx^{A} C'[\mathsf{CE}_0^*[\langle P' \rangle^{\beta'}]] \mathscr{R} C'[\mathsf{CE}_0^*[\langle Q' \rangle^{\beta'}]] \approx^{A} K^{*} \approx^{A} K'$$

Extrusion. This is the case whether the process sends capabilities to the environment and the contexts pick up such capabilities. We have $P \equiv (\mathsf{new}\ \tilde{b} : \tilde{B})P' \mid a\langle n@S \rangle$ and $\mathrm{I} \vdash a : \mathsf{r}$.

The extrusion hypothesis let us infer that there is CE_1^* such that :

$$\begin{array}{rcl} J & \equiv & C'[(\mathsf{new}\ \tilde{b}, \tilde{c})\mathsf{CE}_1^*[\langle P' \rangle^{\beta} \mid P^*]] \\ \mathsf{CE}_1^*[-] & \equiv & W \mid Chan_{\Delta'}\{\tilde{N}\}[S \mid \mathsf{filter}\ \tilde{y}\ \mathsf{from}\ a_1^{\circ}@T\ \mathsf{in}\ !\, net\langle\{\tilde{y}\}_{sk(N)}\rangle \\ & & \mid a_1^{\circ}\langle \underline{n}_{\beta} \Uparrow \underline{S} \rangle] \mid Proxy_{\tilde{M};\Delta}\{M\}[R] \end{array}$$

Here a_1 is the server counterpart of a and $S <: T$.

Process J' is obtained by letting the threads synchronizing:

$$J \xrightarrow{a_1@T} J' \equiv C'[(\mathsf{new}\ \tilde{b}, \tilde{c})!\, net\langle\{\underline{n}_{\beta} \Uparrow \underline{T}\}_{sk(N)}\rangle \mid \mathsf{CE}_{0,\mathsf{c}}^*[\langle P' \rangle^{\beta} \mid P^*]$$

The environment $\mathsf{CE}_{0,\mathsf{c}}^*$ differs from CE^* in that the nonce c is stored and the name a is registered (it may be registered as well in CE^*). Here we also used the trivial result $(M \Uparrow \underline{A}) \Uparrow \underline{B} = M \Uparrow \underline{B}$ whenever $\underline{A} <: \underline{B}$.

By removing the leftover of the communication with Lemmas 69,73,75,78 we obtain:

$$J' \approx_{\rho}^{A} C'[(\mathsf{new}\ \tilde{b})!\, net\langle\{\underline{n}_{\beta} \Uparrow \underline{T}\}_{sk(N)}\rangle \mid \mathsf{CE}_0^*[\langle P' \rangle^{\beta}]]$$

The environment CE_0^* differs from $\mathsf{CE}_{0,\mathsf{c}}^*$ in that the nonce c is not saved.

We exploit the characterization of $\cong^{A\pi}$ in terms of typed bisimulation presented in Chapter 2 and we infer that: $\mathrm{I} \vDash P \cong Q$ and $\mathrm{I} \triangleright P \xrightarrow{(\tilde{b})a\langle n@T \rangle} \mathrm{I} \sqcap n : T \triangleright P'$ implies that there is Q' such that

$$\mathrm{I} \triangleright Q \xRightarrow{(\tilde{b})a\langle n@T \rangle} \mathrm{I} \sqcap n : T \triangleright Q'$$

and $\mathrm{I} \sqcap n : T \models P' \cong Q'$. From this we infer that there exists Q_0, s.t.

$$\begin{array}{l} Q \Longrightarrow Q_0 \\ \mathrm{I} \triangleright Q_0 \xrightarrow{(\tilde{b})a\langle n@T \rangle} \mathrm{I} \sqcap n : T \triangleright Q_0' \\ Q_0' \Longrightarrow Q' \end{array}$$

From these results we infer that Q_0 has the following shape: $Q_0 \equiv (\mathsf{new}\ \tilde{b})Q_0' \mid a\langle n@T\rangle$.

We apply preservation of execution steps (Corollary 91) to $Q \Longrightarrow Q_0$ and infer

$$K \equiv C[\mathsf{CE}^*[\,\langle\, Q \,\rangle^{\,\beta}]] \Longrightarrow\!\approx^A_\rho C[\mathsf{CE}^*[\,\langle\, Q_0 \,\rangle_{\,\beta}]]$$

From $H \overset{\ulcorner A^* \urcorner}{\Longrightarrow} J$, and that J contains the read request $\mathsf{filter}\ \tilde{y}\ \mathsf{from}\ a_1^\circ@T\ \mathsf{in}\ !\,net\langle\{\tilde{y}\}_{sk(N)}\rangle$ that by hypothesis has not been inserted by the encoding of P, we infer that the request can be produced on the right side. From J containing the write request $a_1^\circ\langle \underline{n}_\beta \Uparrow \underline{S}\rangle$ and from shape of Q_0, we infer that the message in the context $\mathsf{CE}_1^*[-]$ can be produced administratively from Q_0. These results let us infer:

$$C[\mathsf{CE}^*[\,\langle\, Q_0 \,\rangle_{\,\beta}]] \overset{A}{\Longrightarrow}\!\approx^A C'[(\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]]\ .$$

We conclude that

$$C[\mathsf{CE}^*[\,\langle\, Q_0 \,\rangle_{\,\beta}]] \approx^A C'[(\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]]\ .$$

Now we let the synchronization occur inside the environment $\mathsf{CE}_1^*[-]$:

$$C'[(\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]] \xrightarrow{a_1@T} C'[(\mathsf{new}\ \tilde{b})!\,net\langle\{\underline{n}_\beta \Uparrow \underline{T}\}_{sk(N)}\rangle \mid \mathsf{CE}_0^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]]\ .$$

Last we apply preservation of execution steps to $Q_0' \Longrightarrow Q'$ and we obtain that there is K^* such that

$$C'[(\mathsf{new}\ \tilde{b})!\,net\langle\{\underline{n}_\beta \Uparrow \underline{T}\}_{sk(N)}\rangle \mid \mathsf{CE}_0^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]] \Longrightarrow K^*$$

and $\rho \models K^* \approx^A C'[(\mathsf{new}\ \tilde{b})!\,net\langle\{\underline{n}_\beta \Uparrow \underline{T}\}_{sk(N)}\rangle \mid \mathsf{CE}_0^*[\,\langle\, Q' \,\rangle_{\,\beta}]]$.

Summing up, the last two results says that:

$$C'[(\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]] \Longrightarrow K^* \tag{5.2}$$

We glue the results $K \Longrightarrow C[\mathsf{CE}^*[\,\langle\, Q_0 \,\rangle_{\,\beta}]]$, $C[\mathsf{CE}^*[\,\langle\, Q_0 \,\rangle_{\,\beta}]] \approx^A C'[(\mathsf{new}\ \tilde{b})\mathsf{CE}_1^*[\,\langle\, Q_0' \,\rangle_{\,\beta}]]$, and of equation (5.2), and we infer that there is K' such that

$$K \Longrightarrow K' \ \wedge\ K' \approx^A_\rho K^*$$

Finally from $\mathrm{I} \sqcap n : T \models P' \cong^\pi Q'$, and from the facts that the context $C'[(\mathsf{new}\ \tilde{b})!\,net\langle\{\underline{n}_\beta \Uparrow \underline{T}\}_{sk(N)}\rangle \mid -]$ (i) both can be built around $\{\!|\, \mathrm{I} \sqcap n : T, \Delta \,|\!\}, \tilde{b}/\tilde{y}$ (the term N actually was received by the context), and (ii) satisfies the definition required by $\mathscr{R}$, we obtain the desired result,

$$\begin{aligned}\rho \models Y' \approx^A J' \approx^A\ & C'[(\mathsf{new}\ \tilde{b})!\,net\langle\{\underline{n}_\beta \Uparrow \underline{T}\}_{skN}\rangle \mid \mathsf{CE}_0^*[\,\langle\, P' \,\rangle^{\,\beta}]] \\ \mathscr{R}\ & C'[(\mathsf{new}\ \tilde{b})!\,net\langle\{\underline{n}_\beta \Uparrow \underline{T}\}_{sk(N)}\rangle \mid \mathsf{CE}_0^*[\,\langle\, Q' \,\rangle_{\,\beta}]] \approx^A K^* \approx^A K'\ .\end{aligned}$$

Contextuality Follows straightforwardly from the definition of $\mathscr{R}$. Let $\rho \models H\mathscr{R}K$. For the first clause, let $\rho \vdash S$. By $\rho \subseteq (\{\!|\, \mathrm{I} \,|\!\}, \tilde{b}/\tilde{x})$ we have $\rho \models C[S\rho \mid \mathsf{CE}^*[\, \langle\, P \,\rangle^{\,\beta}]]\mathscr{R}C[S\rho \mid \mathsf{CE}^*[\, \langle\, Q \,\rangle^{\,\beta}]])$ where $fn(S\rho) \cap bn(C) = \emptyset$. For the second clause, let $n \notin fn(\rho)$. Since $\rho \subseteq \{\!|\, \mathrm{I} \,|\!\}, \tilde{b}/\tilde{x}$, we have $\rho, n/x \subseteq \{\!|\, \mathrm{I} \,|\!\}, \tilde{b}/\tilde{x}, n/x$. From $\mathscr{R}$ term indexed relation we infer that $fn(\, \langle\, P \,\rangle\, , \, \langle\, Q \,\rangle\,) \subseteq fn(\rho)$ and thus $n \notin fn(\{\!|\, fn(P, Q) \,|\!\})$. From this we infer that $\rho, n/x$ satisfies the conditions for the base of $\mathscr{R}$ and in turn since $C = C\{n/x\}$ (notice C is closed) we obtain $\rho \models H\mathscr{R}K$, as desired. Last, we analyze the third clause. From $\rho \backslash n \subseteq (\{\!|\, \mathrm{I} \,|\!\}, \tilde{b}/\tilde{x})$ we infer $\rho \backslash n \models (\mathsf{new}\ n)H\mathscr{R}(\mathsf{new}\ n)K$ as $(\mathsf{new}\ n)C[-]$ satisfies the definition required in $\mathscr{R}$ by the hypothesis on $C[-]$. $\square$

6

A distributed implementation

The main drawback of the implementation devised in Chapter 4 is that all clients forward requests towards a centralized proxy manager. Clearly, this solution is hardly realistic and may compromise the scalability of the system, because the increase of clients' requests can quickly turn the proxy manager into a bottleneck. In this chapter, we discuss a new implementation that distributes the proxy services among different servers. The new solution is based on the idea of partitioning the network in domains each of which administrated by a proxy server.

6.1 An API@ calculus with domains

To model the partitioning of the network in domains, we extend our high level calculus with the syntactic category of nets, representing compositions of processes labelled by domains labels, and defined by the following productions:

$$S, T \ ::= \ \delta\{P\} \ \mid \ S \,|\, T \ \mid \ (\mathsf{new}\ n : A)S \ \mid \ \mathbf{stop}$$

Domain labels, ranged over by δ are drawn from a denumerable set disjoint from the set of names and the set of variables. We let $fd(S)$ be the set of domain labels in S. We emphasize that domain labels are not names, and are never exchanged over channels, nor are they created dynamically by a restriction. The typing and dynamics of nets arise in the simplest possible way from the corresponding notions defined for processes. The two core rules are as follows:

$$\frac{\Gamma \vdash P}{\Gamma \vdash \delta\{P\}}\ (\textsc{Typing}) \qquad \frac{P \xrightarrow{\alpha} Q}{\delta\{P\} \xrightarrow{\alpha} \delta\{Q\}}\ (\textsc{Dynamics})$$

As a result, domains have no impact on the dynamics and/or the typing of the high-level calculus: indeed, they serve a different purpose, namely to help devise the association of processes to proxies in the implementation. Notice, in particular, that the same (channel) name may be known at different domains: in the implementation, this will correspond to the name being represented by different channels, located at the different domains at which the name is known.

Observational Equivalence

The definition of observational equivalence for nets is inherited from that of processes. There is an important difference, however, in the notion of contextuality, in that a context may not include new domains, but only processes belonging to existing domains. This is ensured by the side condition $fd(U) \subseteq fd(S,T)$ in the definition below, and constitutes the key assumption for our distributed implementation: namely, we do not trust domains and proxy servers generated by the environment. While this is a somewhat strong assumption, on the other hand it appears to be realistic: notice, in fact, that the procedure of adding a domain to a network in real-world scenarios requires physical authentication, rather than network protocols. More in detail, we are modeling subsystems that represent physical locations with the need for asynchronous communications in-between, rather than subsystems representing units of trust or principals. Indeed, the latter notion of locality in distributed systems requires mutual distrust among principals and defensive implementations, while we assume that all the proxies participate and fully trust each other.

Definition 28 (Contextuality for Nets). *A type-indexed relation $\mathscr{R}$ over nets is contextual whenever*

- $\mathrm{I} \vDash S\mathscr{R}T$ *and* $\mathrm{I} \vdash U$ *and* $fd(U) \subseteq fd(S,T)$ *implies* $\mathrm{I} \vDash (S \mid U)\mathscr{R}(T \mid U)$
- $\mathrm{I} \vDash S\mathscr{R}T$ *implies* $\mathrm{I}, a : A \vDash S\mathscr{R}T$
- $\mathrm{I}, a : A \vDash S\mathscr{R}T$ *implies* $\mathrm{I} \vDash ((\mathsf{new}\ a : A)S)\mathscr{R}((\mathsf{new}\ a : A)T)$

The definition of behavioral equivalence for nets, noted again $\cong^{\pi}$, arises now as expected.

Definition 29 (Typed behavioral equivalence). *Typed behavioral equivalence, noted $\cong^{\pi}$, is the largest symmetric and contextual type-indexed relation $\mathscr{R}$ over nets such* $\mathrm{I} \models S\mathscr{R}T$ *implies*

(i) *if* $\mathrm{I} \vDash S\downarrow_n$ *then* $\mathrm{I} \vDash T\Downarrow_n$

(ii) *if* $S \xrightarrow{\tau} S'$ *then* $T \Longrightarrow T'$ *and* $\mathrm{I} \models S'\mathscr{R}T'$ *for some* T' .

6.2 A compositional implementation using domain authorities

Each domain of a high-level net corresponds to a domain manager, which manages the channels it has created and acts as a proxy for the processes of the domain; the processes of a domain, in turn, are instructed to send their requests to the proxy associated with their domain. Since different proxies may have different entries for the same client name (remember that a name is possibly known in more domains),

more channels servers may correspond to the same pi calculus name. The domain managers must therefore mask the presence of multiple queues located at the distributed channels associated to the same client name. This, in turn, is based on a further service to gain access to fellow proxies.

We represent public keys used to support the proxy service in each domain manager by means of a one-way constructor p, and define $\delta_p^+ \triangleq ek(p(\delta))$ and $\delta_p^- \triangleq dk(p(\delta))$; these keys corresponds to the keys k_P^+, k_P^- utilized in the centralized translation. Further, we use an one-way constructor q to represent the public keys used by the queue service for a domain δ: $\delta_q^+ \triangleq ek(q(\delta))$ and $\delta_q^- \triangleq dk(q(\delta))$. We assume that domain managers are connected via secure links, represented by a shared key $sk(k_D)$ generated from the private name k_D. Finally, we introduce the following two bits of new notation:

- $Chan_n[P]$ indicates the process $(\mathsf{new}\ n^*, n^\circ)\ n^*\langle\emptyset\rangle \mid RS_n \mid WS_n \mid P$, that is $Chan_n$ augmented with a further process P
- the notation for links is extended as expected, with a parameter corresponding to the assigned proxy:

$$\mathsf{link}^\delta\ (M, \underline{y})\ \mathsf{in}\ P \triangleq \\ (\mathsf{new}\ k)\mathsf{emit}(\{sk(k), M\}_{\delta_p^+}) \mid \mathsf{filter}\ \underline{y}\ \mathsf{with}\ sk(k)\ \mathsf{in}\ P$$

The new implementation is given in Table 6.1. For each domain δ, the domain manager consists of three threads $P_{q,t}^\delta$, $Q_{q,t}^\delta$, $D_{q,l}^\delta$ responsible for the proxy, queue and domain services, respectively. The three threads share a channel q collecting the public keys that grant access to fellow proxies. The domain service $D_{q,l}^\delta$ is responsible for updating this queue with newly acquired domain identities, and for publishing the public queue key of the proxy associate to the domain δ.

The proxy and the queue service also share the table of binding client and server names stored on the private channel t. The new definition of the proxy service extends the one given in Table 4.2, by including a collection of forwarders. Each of the forwarders, tries to extract a message from the channel queue n° and sends it off to one of the domain managers known at the time the channel n was created.

The queue service $Q_{q,t}^\delta$ waits for the packets sent by the forwarders and by other domain managers. The server retrieves the name index and checks if the entry is associated to a channel. If it is, the message is sent to the queue of the associated channel. Otherwise both the name index and the message are non-deterministically sent to some of the known domain managers. The non-deterministic choice $\mathsf{Choose}_{k\in X}\ \mathsf{in}\ P$ may be implemented in the standard way, namely in terms of a non-deterministic synchronization on a private name: $(\mathsf{new}\ n)n\langle\rangle \mid \Pi_{k\in X} n().P$.

The translation of nets is compositional, and does not rely on any pre-existing infrastructure for communications, as now the domain managers are dynamically generated within the translation; as in previous implementations we assume the

Table 6.1 Distributed Translation

Proxy Service

$$P^{\delta}_{q,t} \triangleq \; !\,\mathsf{filter}\ (k, \underline{x}, y)\ \mathsf{with}\ \delta^{-}_{p}\ \mathsf{in\ if}\ y \notin Set_{t^*}\ \mathsf{then}$$
$$\mathsf{let}\ s = Typeof(\underline{x})\ \mathsf{in}$$
$$t(z).(\mathsf{let}\ \tilde{y} = ?(\underline{x}_{ID}, y)\ \mathsf{in}\ t\langle z\rangle \mid !\, net\langle\{\underline{y}\Uparrow s\}_k\rangle$$
$$\mathsf{else}\ (\mathsf{new}\ n)\ t\langle z.(\underline{x}_{ID}; \underline{n}).(n_{ID}; \underline{n})\rangle \mid !\, net\langle\{\underline{n}\Uparrow s\}_k\rangle$$
$$\mid Chan_n[q(X).(q\langle X\rangle \mid \Pi_{k_\delta \in X}\, !\, n^{\circ}(\underline{w}).\mathsf{emit}(\{\underline{x}_{ID}, \underline{w}\}_{k_\delta}))])$$

Queue Service

$$Q^{\delta}_{q,t} \triangleq \; !\,\mathsf{filter}\ (x, \underline{s}, y)\ \mathsf{with}\ \delta^{-}_{q}\ \mathsf{in\ if}\ y \notin Set_{q^*}\ \mathsf{then}$$
$$t(z).(\, t\langle z\rangle \mid \mathsf{let}\ \tilde{y} = ?(x, z)\ \mathsf{in}\ \mathsf{emit}(\{\underline{s}\}_{\underline{y}^{+}_{w}})\ \mathsf{else}$$
$$q(X).(q\langle X\rangle \mid \mathsf{Choose}_{k_\delta \in X}\ \mathsf{in}\ \mathsf{emit}(\{x, \underline{s}\}_{k_\delta}))$$

Domain Service

$$D^{\delta}_{q,l} \triangleq \; !\,\mathsf{filter}\ (x, c)\ \mathsf{with}\ sk(k_D)\ \mathsf{in\ if}\ c \notin Set_{l^*}\ \mathsf{then}\ q(y).q\langle y :: x\rangle$$
$$\mid \mathsf{emit}(\{\delta^{+}_{q}\}_{sk(k_D)})$$

Domain Manager

$$M^{\delta} \triangleq \; (\mathsf{new}\ t, t^*, q, q^*, l^*)$$
$$P^{\delta}_{q,t} \mid Q^{\delta}_{q,t} \mid D^{\delta}_{q,l} \mid t\langle\emptyset\rangle \mid t^*\langle\emptyset\rangle \mid q\langle\emptyset\rangle \mid q^*\langle\emptyset\rangle \mid l^*\langle\emptyset\rangle$$

Translation of nets

$$(\,\Gamma \triangleright \delta\{P\}\,)_{\mu} \triangleq M^{\delta} \mid \langle\,\Gamma \triangleright P\,\rangle^{\delta}_{\mu}$$
$$(\,\Gamma \triangleright S \mid T\,)_{\mu} \triangleq (\,\Gamma \triangleright S\,)_{\mu} \mid (\,\Gamma \triangleright T\,)_{\mu}$$
$$(\,\Gamma \triangleright (\mathsf{new}\ n : A)S\,)_{\mu} \triangleq (\mathsf{new}\ n)\,(\,\Gamma, n : A \triangleright S\,)_{\mu}$$
$$(\,\Gamma \triangleright \mathbf{stop}\,)_{\mu} \triangleq \mathbf{0}$$

Channels – as in Table 4.2

Clients – as in Table 4.2, with all definitions now parametrized on δ.

Top–level translation $(\,\Gamma \triangleright S\,) \triangleq (\,\Gamma \triangleright S\,)_{\emptyset}$

presence of noise on the communication interface. In the top-level translation of nets we tacitly assume that each domain label occurs at most once. This represents no loss of generality, as the dynamics of the net $\delta\{P_1\} \mid \delta\{P_2\}$ is just the same as that of the net $\delta\{P_1 \mid P_2\}$. Similarly, $(\mathsf{new}\ n : A)\delta\{P\}$ is the same as $\delta\{(\mathsf{new}\ n : A)P\}$.

We extend the low-level term environment with the public encryption keys of the proxy managers corresponding to the high-level free domains:

$$\{\!|\, \delta_1, \ldots, \delta_n \,|\!\} \triangleq \{\delta_{1_p}^{+}/x, \ldots, \delta_{n_p}^{+}/z\}$$

We finally obtain:

Theorem 96. *Let* Γ, Δ *and* I *be closed type environment such that* $\Gamma, \Delta <: \mathrm{I}$. *Then:* $\mathrm{I} \models S \cong^{\pi} T$ *if and only if* $\{\!|\, \mathrm{I} \,|\!\}, \{\!|\, fd(S,T) \,|\!\} \models W \mid (\mathsf{new}\ k_D)\,(\,\Gamma \triangleright S\,) \cong^{A\pi} W \mid (\mathsf{new}\ k_D)\,(\,\Delta \triangleright T\,)$.

Proof. The soundness result is based on the fact that the interactions introduced by the distributed protocol w.r.t. the centralized implementation are "administrative". More in detail, once extended the definition of administrative reduction (Definition 22) so to include the reductions introduced by the distributed implementation, the proof follows the schema outlined in Section 5.1, crucially relying on the fact that Proposition 65 still holds when the definition of administrative reduction is extended. The proof of completeness follows the same schema of Theorem 95, namely by considering a relation containing computing environments that include the context, and by proving that the relation is a weak equivalence up to administrative equivalence. As in Theorem 95 the most delicate clause to analyze is reduction closure and follows by a case analysis of the possible interactions. The interaction among the context and the computing environment containing the encoded processes is treated as in Theorem 95, particularly by analyzing which outputs and inputs are available both for the context and the process. In the case whether the context receives an output, we apply Lemma 94 to the fact that all outputs coming out from the translation are emissions of packets (a) encrypted under fresh symmetric keys or (b) encrypted with asymmetric keys and containing a fresh nonce or (c) encrypted under the symmetric key k_D that is known only to proxies and containing a fresh nonce, in order to infer that these packets are seen as noise by the context. Particularly encoded processes now encrypt linking requests by using the keys δ_{ip}^{+} (rather than by using k_p^{+}), and the low-level term-environment does not contain neither the related decryption keys δ_{ip}^{-}, nor the seeds δ_i (see Lemma 94). In the case whether the context sends an output, we use the fact that distributed proxies use the filtering mechanism introduced by the first translation; therefore in case a proxy receives a wrong message, it rolls back to the initial state and releases the packet absorbed on the net, which are sufficient conditions for reduction closure. The intrusion, extrusion and process reduction cases are analogous, since the commit reduction occurs in the queue of a channel manager, as in Theorem 95.

There is a new case that arises when a message in a queue of a channel is moved to a queue of another channel; this gives raise to a configuration evolving to

another configuration. More in detail, the first reduction on the channel's queue is treated similarly to the process reduction case, as this move needs to be strongly matched by the related process; however the same message is available on the related process since we consider the same configuration on both sides. After the internal synchronization, the packet released by the channel is absorbed administratively by the queue manager, that in turn forwards the packet to some channel manager (after an administrative reduction on the queue channel) that receives it administratively. As in Theorem 95 the use of the up to technique permits to ignore all such subsequent administrative reductions, and to consider the updated configuration in order to gain reduction closure.

□

Conclusions

We have developed a secure implementation of a typed pi calculus, in which the access to communication channels is regulated by capability types. The implementation is effective even when the low-level compiled principals are deployed in open contexts, for which no assumption on trust and behavior may be made. The implementation draws on a representation of the typed capabilities in the high-level calculus as term capabilities protected by encryption keys only known to the intended receivers. Our full abstraction results rely on a proxy service to protect against malformed messages from the environment. This is achieved by generating certified names (and associated channels) to represent the context-generated names within the system.

Our implementation results show that the dynamic management of capabilities enforced at the high level by using a global type system, can be effectively enforced in untyped, open, distributed environments by using cryptography without any assumption on the behavior or trust of the low-level contexts. To the best of our knowledge, this is the first result of this kind for typed process calculi.

As a by-product, since our high-level process calculus is a conservative extension of the untyped pi calculus, we provided a direct secure implementation of the pi calculus with matching. This resolves the problem of preserving the forward secrecy of communications in implementations of calculi which support the dynamic exchange of write *and/or* read access rights among processes.

We have also developed a distributed implementation in which the certification service is implemented by a set of distributed proxies. Being fully compositional, the distributed implementation appears to be adequate for open-ended networks. The only limitation in this respect is represented by our current assumption that all proxies that participate in the synchronization protocols be fully trusted; basically we concern with distributed subsystems that represent physical locations and communicate asynchronously trusting each other. While a certain degree of trust appears necessary to achieve a secure implementation, it would be desirable to have some form of guarantees also in the presence of malicious proxies. This would model distributed subsystems representing unit of trusts or principals, with the need for mutual distrust and defensive implementations. Achieving that seems feasible with our implementation by strengthening the protocols that govern the interactions among proxies; for instance we may admit self-organized generation and distribution of keys, and key-authentication based on chains of certificates (c.f. [34]). We leave this to our plans of future work.

Our full abstraction results rely on the presence of a generator of noise in the network interface that of course is hardly realistic. Moreover, pragmatically the

presence of some background traffic rarely prevents information leaks trough traffic analysis (due to message sizes, addresses). The use of noise could be avoided by considering semantic equations that hold with a certain degree of probability, e.g. [63, 77]. More realistic features as timeouts, expiration of keys should be considered. The resulting implementation, interestingly, would be very close to the actual distributed scenarios; this represents a challenging area of research that we plan to investigate.

Another interesting area of research consists in devise cryptographically sound implementations for the type-based management of capabilities we have presented in this thesis, in the spirit of [14, 6]. The approach followed in that papers maps high level processes into a computational setting, rather than in an algebraic framework; high-level functionalities of pi calculi are implemented by using concrete cryptography based on probabilistic algorithms that operate on bitstrings (although [6] considers an intermediate language inspired by the spi calculus that reflects low-level implementation constraints of the cryptographic library). This is in contrast with the algebraic view of cryptographic protocols followed in this thesis and in other recent works, e.g. [10, 8]. This would bring the implementation more close to the reality, and represents a perfect complementation of the results of this thesis.

Bibliography

[1] Martín Abadi. Protection in programming-language translations. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 868–883, 1998. 1, 1, 1.1, 2, 3.3, 4, 4.1, 4.5

[2] Martín Abadi. Secrecy by typing in security protocols. *J. ACM*, 46(5):749–786, 1999. 2.6

[3] Martín. Abadi. Logic in access control. In *Proc. of the Eighteenth Annual IEEE Symposium on Logic in Computer Science (LICS '03)*, pages 228–233. IEEE Computer Society, 2003. 2.6

[4] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *J. ACM*, 52(1):102–146, 2005. 3.3, 4.2

[5] Martín. Abadi, Michael Burrows, Butler W. Lampson, and Gordon D. Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, Semptember 1993. 2.6

[6] Martín Abadi, Ricardo Corin, and Cédric Fournet. Computational secrecy by typing for the pi calculus. In *APLAS*, pages 253–269, 2006. 6.2

[7] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Proc. of the 28th ACM Symposium on Principles of Programming Languages (POPL '01)*, pages 104–115. ACM Press, 2001. 1.2, 3, 3.1, 3.3

[8] Martín Abadi, Cédric Fournet, and George Gonthier. Secure implementation of channel abstractions. *Information and Computation*, 174(1):37–83, April 2002. (document), 1, 1.1, 2, 4.1, 4.2, 4.2, 4.3, 4.3, 4.5, 5, 6.2

[9] Martín Abadi, Cédric Fournet, and Georges Gonthier. Secure communications processing for distributed languages. In *IEEE Symposium on Security and Privacy*, pages 74–88, 1999. 4.5

[10] Martín Abadi, Cédric Fournet, and Georges Gonthier. Authentication primitives and their compilation. In *POPL*, pages 302–315, 2000. 4.5, 6.2

[11] Martín Abadi and Andrew D. Gordon. A bisimulation method for cryptographic protocols. *Nord. J. Comput.*, 5(4):267–, 1998. 3.3

[12] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, Jan 1999. 3.3, 4.5

[13] Luca Aceto and Andrew D. Gordon, editors. *Algebraic Process Calculi: The First Twenty Five Years and Beyond (PA'05)*, 2005.

[14] Pedro Adão and Cédric Fournet. Cryptographically sound implementations for communicating processes. In *ICALP (2)*, pages 83–94, 2006. 4.5, 6.2

[15] Roberto M. Amadio, Ilaria Castellani, and Davide Sangiorgi. On bisimulations for the asynchronous pi-calculus. *Theor. Comput. Sci.*, 195(2):291–324, 1998. 2.4, 2.6

[16] Michael Baldamus, Joachim Parrow, and Björn Victor. Spi calculus translated to pi-calculus preserving may-testing. In *Proceedings of the 19th IEEE Symposium on Logic in Computer Science (LICS '04)*, pages 22–31, 2004. 4.5

[17] Michael Baldamus, Joachim Parrow, and Björn Victor. A fully abstract encoding of the π-calculus with data terms. In *Proc. of ICALP '05*, pages 1202–1213, 2005. 3.1, 3.2, 3.3, 4.5

[18] David E. Bell and Len La Padula. Secure computer system: Unified exposition and multics interpretation,. Technical Report MTR-2997, MITRE Corporation, Bedford, MA 01730, March 1976. 1

[19] Bruno Blanchet. From Secrecy to Authenticity in Security Protocols. In Manuel Hermenegildo and Germán Puebla, editors, *9th International Static Analysis Symposium (SAS'02)*, volume 2477 of *Lecture Notes on Computer Science*, pages 342–359, Madrid, Spain, September 2002. Springer Verlag. 3.1, 3.1, 3.3, 4.2

[20] Bruno Blanchet. Automatic Proof of Strong Secrecy for Security Protocols. In *IEEE Symposium on Security and Privacy*, pages 86–100, Oakland, California, May 2004. 3.1, 3.1, 3.2, 3.3

[21] Michele Boreale. On the expressiveness of internal mobility in name-passing calculi. *Theor. Comput. Sci.*, 195(2):205–226, 1998. 4, 4.3, 4.5, 5.1, 5.1

[22] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. *SIAM Journal on Computing*, 31(3):947–986, 2001. 3, 3.2, 3.3

[23] Michele Boreale and Davide Sangiorgi. Bisimulation in name-passing calculi without matching. In *Proc. of 13th IEEE Symposium on Logic in Computer Science (LICS '98)*. IEEE Computer Society Press, 1998. 1, 2.3, 2.6, 3.3

[24] Johannes Borgström, Sébastien Briais, and Uwe Nestmann. Symbolic bisimulation in the spi calculus. In *CONCUR*, pages 161–176, 2004. 3.3

[25] Johannes Borgström and Uwe Nestmann. On bisimulations for the spi calculus. *Mathematical Structures in Computer Science*, 15(3):487–552, 2005. 3.3

[26] Gérard Boudol. Asynchrony and the π-calculus. Research Report 1702, INRIA, http://www.inria.fr/rrrt/rr-1702.html. Also available from http://www-sop.inria.fr/mimosa/personnel/Gerard.Boudol.html, 1992. 1, 2.6

[27] Gérard Boudol. The pi-calculus in direct style. In *POPL*, pages 228–241, 1997. 4.5

[28] Michele Bugliesi, Dario Colazzo, and Silvia Crafa. Type based discretionary access control. In Philippa Gardner and Nobuko Yoshida, editors, *Proc. of CONCUR '04*, volume 3170 of *LNCS*, pages 225–239, 2004. 2.6

[29] Michele Bugliesi and Marco Giunti. Typed processes in untyped contexts. In Rocco De Nicola and Davide Sangiorgi, editors, *Proc. of TGC 2005, Symposium on Trustworthy Global Computing*, volume 3705 of *Lecture Notes on Computer Science*, pages 19–32. Springer-Verlag, 2005. 1.1, 1.2, 2

[30] Michele Bugliesi and Marco Giunti. Secure implementations of typed channel abstractions. In *Proc. of the 34th ACM Symposium on Principles of Programming Languages (POPL'07)*. ACM Press, 2007. 1.1, 1.2

[31] Michele Bugliesi and Sabina Rossi. Non-interference proof techniques for the analysis of cryptographic protocols. *Journal of Computer Security*, 13(1):87–113, 2005. 2.6

[32] Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. Replication vs. recursive definitions in channel based calculi. In *ICALP*, pages 133–144, 2003. 4.5

[33] Diletta Cacciagrano and Flavio Corradini. On synchronous and asynchronous communication paradigms. In *Proc. of the 7th Italian Conference of Theoretical Computer Science (ICTCS 2001)*, volume 2202. Springer-Verlag, 2001. 4.5

[34] Srdjan Capkun, Levente Buttyán, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Trans. Mob. Comput.*, 2(1):52–64, 2003. 6.2

[35] Marco Carbone and Sergio Maffeis. On the expressive power of polyadic synchronisation in pi-calculus. *Nordic Journal Of Computing*, 10(2):70–98, 2003. 4.5

[36] Luca Cardelli. Type systems. *ACM Comput. Surv.*, 28(1):263–264, 1996. 1

[37] Luca Cardelli, Giorgio Ghelli, and Andrew D. Gordon. Secrecy and group creation. *Inf. Comput.*, 196(2):127–155, 2005. 2.6

[38] Ilaria Castellani and Matthew Hennessy. Testing theories for asynchronous languages. In Vikraman Arvind and R. Ramanujam, editors, *Proc. of Foundations of Software Technology and Theoretical Computer Science 1998 (FSTTCS '98)*, volume 1530 of *Lecture Notes in Computer Science*, pages 90–101. Springer, 1998. 2.6

[39] Konstantinos Chatzikokolakis and Catuscia Palamidessi. A framework for analyzing probabilistic protocols and its application to the partial secrets exchange. In *Proc. of TGC'05*, pages 146–162, 2005. 4.1

[40] Mario Coppo, Federico Cozzi, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Rosario Pugliese. A mobility calculus with local and dependent types. In *Processes, Terms and Cycles*, pages 404–444, 2005. 2.6

[41] Mario Coppo, Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Rosario Pugliese. Dynamic and local typing for mobile ambients. In *IFIP TCS*, pages 577–590, 2004. 2.6

[42] Silvia Crafa and Sabina Rossi. A theory of noninterference for the pi-calculus. In *Proc. of TGC'05*, pages 2–18, 2005. 2.6

[43] Rocco De Nicola, Gianluigi Ferrari, and Rosario Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998. 2.6

[44] Rocco De Nicola, Gianluigi Ferrari, Rosario Pugliese, and Betti Venneri. Types for access control. *Theoretical Computer Science*, 240(1):215–254, 2000. 2.6

[45] Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984. 1

[46] Yuxin Deng and Davide Sangiorgi. Towards an algebraic theory of typed mobile processes. *Theor. Comput. Sci.*, 350(2-3):188–212, 2006. 2.3, 2.6

[47] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983. 3.3

[48] Uffe .H. Engberg and Morgen Nielsen. A calculus of communicating systems with labels passing - Ten years after. In *Proof, language, and interaction: essays in honour of Robin Milner*, Foundations of Computing, pages 599–622. MIT Press, 2000.

[49] Riccardo Focardi and Roberto Gorrieri. Classification of Security Properties (Part I: Information Flow). In R. Focardi and R. Gorrieri, editors, *Proc. of Foundations of Security Analysis and Design (FOSAD'01)*, volume 2171 of *Lecture Notes in Computer Science*, pages 331–396. Springer-Verlag, 2001. 2.6

[50] Cédric Fournet. *The Join-Calculus: a Calculus for Distributed Mobile Programming*. PhD thesis, Ecole Polytechnique, Palaiseau., November 1998. Also published by INRIA, TU-0556. 4.1, 5.1

[51] Cédric Fournet and Georges Gonthier. The reflexive cham and the join-calculus. In *POPL'96*, pages 372–385, 1996. 1.1, 4.5

[52] Cédric Fournet and Georges Gonthier. A hierarchy of equivalences for asynchronous calculi. *J. Log. Algebr. Program.*, 63(1), 2005. 2.6

[53] Cédric Fournet and Cosimo Laneve. Bisimulations in the join-calculus. *Theor. Comput. Sci.*, 266:569–603, 2001. 4.5

[54] Philippa Gardner, Cosimo Laneve, and Lucian Wischik. Linear forwarders. In *Proc. of CONCUR'03*, pages 408–422, 2003. 4.5

[55] Daniele Gorla. *Semantic Approaches to Global Computing Systems*. PhD thesis, Dip. Sistemi ed Informatica, Univ. di Firenze, 2004. 5.2

[56] Daniele Gorla and Rosario Pugliese. Resource access and mobility control with dynamic privileges acquisition. In *ICALP*, pages 119–132, 2003. 2.6

[57] Joseph Y. Halpern and Vicky Weissman. Using first-order logic to reason about policies. In *Proc. of the 16th IEEE Computer Security Foundations Workshop (CSFW '03)*, pages 187–201, 2003. 2.6

[58] Matthew Hennessy. The security pi-calculus and non-interference. *J. Log. Algebr. Program.*, 63(1):3–34, 2005. 2.6

[59] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J. ACM*, 32(1):137–161, 1985.

[60] Matthew Hennessy and James Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Mathematical Structures in Computer Science*, 14(5):651–684, 2003. 1, 1, 1.2, 2, 2, 2, 2.1, 2.3, 2.3, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.6

[61] Matthew Hennessy and James Riely. Information flow vs resource access in the asynchronous π-calculus. *ACM Transactions on Programming Languages and Systems*, 24(5):566–591, 2002. 2.6

[62] Matthew Hennessy and James Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173:82–120, 2002. 2.6

[63] Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous pi-calculus. In *FoSSaCS*, pages 146–160, 2000. 4.1, 6.2

[64] Kohei Honda and Mario Tokoro. An object calculus for asynchronous communication. In *ECOOP*, pages 133–147, 1991. 2.6

[65] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. *Theor. Comput. Sci.*, 151(2):437–486, 1995. 1, 2.6, 4.5

[66] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. *ACM Trans. Program. Lang. Syst.*, 21(5):914–947, 1999. 2.6

[67] Butler W. Lampson. Protection. *ACM Operating Systems Rev.*, 8(1):18–24, Jan. 1974. 1

[68] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. 4.1

[69] Massimo Merro. *Locality in the π-calculus and applications to distributed objects*. PhD thesis, École de Mines de Paris, October 2000. 4.5

[70] Massimo Merro and Davide Sangiorgi. On asynchrony in name-passing calculi. *Mathematical Structures in Computer Science*, 14(5):715–767, 2004. 2.6

[71] Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989. 4.1

[72] Robin Milner. The polyadic π-calculus: a tutorial. Technical Report ECS–LFCS–91–180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK, October 1991. Appeared in *Proceedings of the International Summer School on Logic and Algebra of Specification*, Marktoberdorf, August 1991. Reprinted in *Logic and Algebra of Specification*, ed. F. L. Bauer, W. Brauer, and H. Schwichtenberg, Springer-Verlag, 1993. 2

[73] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge, 1999. 2

[74] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Parts I and II. *Information and Computation*, 100:1–77, September 1992. 1, 1, 2

[75] Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theor. Comput. Sci.*, 114(1):149–171, 1993.

[76] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 685–695, 1992. 1

[77] John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theor. Comput. Sci.*, 353(1-3):118–164, 2006. 4.1, 6.2

[78] Andrew C. Myers. *Mostly Static Decentralized Information Flow Control*. PhD thesis, MIT, Jan 1999. 2.6

[79] Andrew C. Myers and Barbara Liskov. Complete, safe information flow with decentralized labels. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, 1998. 2.6

[80] Andrew C. Myers and Barbara Liskov. Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.*, 9(4):410–442, 2000. 2.6

[81] Andrew C. Myers and Andrei Sabelfeld. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications, special issue on Formal Methods for Security*, 21(1):5–19, January 2003. 2.6

[82] George C. Necula. Proof carrying code. In *24th Ann. ACM Symp. on Principles of Programming Languages*. ACM Press, 1997. 1

[83] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978. 4.2

[84] Uwe Nestmann and Benjamin C. Pierce. Decoding choice encodings. *Inf. Comput.*, 163(1):1–59, 2000. 1.2, 2.6, 4, 4.3, 4.5, 5.1

[85] Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003. 2.6, 4.5

[86] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *LICS*, pages 176–185, 1998. 4.5

[87] Anna Philippou and David Walker. On confluence in the pi-calculus. In *ICALP*, pages 314–324, 1997.

[88] Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5), 1996. 1, 1, 2, 2, 2, 2, 2.3, 2.6

[89] Benjamin C. Pierce. *Foundational Calculi for Programming Languages*, chapter 139. CRC Press, 1996.

[90] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002. 1

[91] Benjamin C. Pierce and Davide Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000. 1, 2.3, 2.6

[92] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. In Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 455–494. MIT Press, 2000. 2.6

[93] Benjamin C. Pierce and David N. Turner. The Pict programming language, 2001. http://www.cis.upenn.edu/ bcpierce/papers/pict/Html/Pict.html.

[94] Damien Pous. Up-to techniques for weak bisimulation. In *ICALP*, pages 730–741, 2005. 5.2

[95] Rosario Pugliese. *Semantic Theories for Asynchronous Languages*. PhD thesis, Dipartimento di Scienze dell'Informazione - Universita' di Roma "La Sapienza", 1996. 2.6

[96] Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In *FoSSaCS*, pages 468–483, 2004. 4.1

[97] James Riely and Matthew Hennessy. A typed language for distributed mobile processes. In *Proceedings of POPL'98*, pages 378–390. ACM Press, 1998. 2.6

[98] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, LNCS 2171. Springer-Verlag, 2001.

[99] Davide 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. 2.3, 4.5

[100] Davide Sangiorgi. An interpretation of typed objects into typed pi-calculus. *Inf. Comput.*, 143(1):34–73, 1998.

[101] Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–480, 1998. 5.2

[102] Davide Sangiorgi. The name discipline of uniform receptiveness. *Theoretical Computer Science*, 221(1–2):457–493, 1999.

[103] Davide Sangiorgi and Robin Milner. The problem of "weak bisimulation up to". In *Proceedings of CONCUR 1992*, pages 32–46, 1992. 2.4, 5.1, 5.2, 5.2

[104] Davide Sangiorgi and David Walker. *The π-calculus A theory of mobile processes*. Cambridge, 2001. 2, 2.1, 2.1, 18, 3.2, 4.5

[105] Peter Selinger. First-order axioms for asynchrony. In *Proc. of CONCUR 1997*, pages 376–390, 1997. 2.6

[106] Peter Sewell. On implementations and semantics of a concurrent programming language. In *CONCUR*, pages 391–405, 1997.

[107] Vincent van Oostrom. Confluence by decreasing diagrams. *Theor. Comput. Sci.*, 126(2):259–280, 1994. 5.1

[108] Lucian Wischik and Philippa Gardner. Explicit fusions. *Theor. Comput. Sci.*, 340(3):606–630, 2005. 4.5

List of PhD Thesis

TD-2004-1 Moreno Marzolla
"Simulation-Based Performance Modeling of UML Software Architectures"

TD-2004-2 Paolo Palmerini
"On performance of data mining: from algorithms to management systems for data exploration"

TD-2005-1 Chiara Braghin
"Static Analysis of Security Properties in Mobile Ambients"

TD-2006-1 Fabrizio Furano
"Large scale data access: architectures and performance"

TD-2006-2 Damiano Macedonio
"Logics for Distributed Resources"

TD-2006-3 Matteo Maffei
"Dynamic Typing for Security Protocols"

TD-2006-4 Claudio Silvestri
"Distributed and Stream Data Mining Algorithms for Frequent Pattern Discovery"

TD-2007-1 Marco Giunti
"Secure Implementations of Typed Channel Abstractions"

TD-2007-2 Francesco Lelli
"Bringing Instruments to a Service-Oriented Interactive Grid"

TD-2007-3 Matteo Mordacchini
"Grid and Peer-to-Peer Resource Discovery Systems"