Static vs Dynamic Typing for Access Control in Pi-Calculus*

Michele Bugliesi, Damiano Macedonio, and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari, Venice
{michele,mace,srossi}@dsi.unive.it

Abstract. Traditional static typing systems for the pi-calculus are built around capability types that control the read/write access rights on channels and describe the type of their payload. While static typing has proved adequate for reasoning on process behavior in typed contexts, dynamic techniques have often been advocated as more effective for access control in distributed/untyped contexts.
We study the relationships between the two approaches – static versus dynamic – by contrasting two versions of the asynchronous pi-calculus. The former, APi, comes with an entirely standard static typing system. The latter, APi@, combines static and dynamic typing: a static type system associates channels with flat types that only express read/write capabilities and disregard the payload type; a dynamically typed synchronization complements the static type system to guarantee type soundness.
We show that APi@ can be encoded into APi in a fully abstract manner, preserving the respective behavioral equivalences of the two calculi. Besides yielding an interesting expressivity result, the encoding also sheds light on the effectiveness of dynamic typing as a mechanism for access control.

1 Introduction

Static typing systems have long been established as an effective device to control the interaction among processes in the pi-calculus and related calculi [7, 8, 12, 13]. In these systems the communication channels are viewed as resources, and their types define the capabilities needed to use them. Thus, for instance $\mathsf{rw}\langle S\,;T\rangle$ is the type of a channel where one can output at type T and input at type S (provided that T is a subtype of S). The nested structure of the types makes it possible to control the way that capabilities are delivered and made available. To illustrate, a process knowing the name (or channel) a at the type $\mathsf{rw}\langle \mathsf{r}\langle S\rangle; \mathsf{rw}\langle S\,;S\rangle\rangle$ may output on a a full-fledged channel (with payload type S) and be guaranteed that any (well-typed) process inputing on a will only be reading on the channel received. This form of type-based control yields powerful techniques to reason on the behavior of processes in typed contexts: in fact, by putting enough structure on the types of the shared channels, one may gain strong control on the interaction of a process with any typed context. Unfortunately, these techniques do not scale well to general, potentially untyped, contexts.

* Work partially supported by M.I.U.R (Italian Ministry of Education, University and Research) under contract n. 2005015785.

To address this shortcoming, [2] introduces a variant of the (asynchronous) pi-calculus, named AΠ@, in which the ability to control the use of channels relies on a combination of static and dynamic typing. The types of channels are still formed around capabilities, but in AΠ@ they only exhibit the "top-level" read/write capabilities, disregarding the types of the values transmitted. Given this simple type structure, the type system is much less effective in providing control on the use of channels. To compensate for that, AΠ@ includes a new form of output construct, noted $\overline{a}\langle v@B\rangle$, that relies on a type coercion to enforce the delivery of v at type B. A static typing system guarantees that v may indeed be assigned the coercion type B, while a mechanism of dynamically typed synchronization guarantees that v is received only at supertypes of B, so as to guarantee the type soundness of each exchange.

As argued in [2], the new typing system succeeds in its goal to provide reasoning methods for typed processes in arbitrary contexts. Indeed, [3] shows that the processes of AΠ@ may be implemented as low-level principals of a cryptographic process calculus based on the applied pi-calculus [1], while preserving their behavioral invariants. The present paper complements the work in [2, 3] by investigating how the combination of static and dynamic typing in AΠ@ impacts on the ability to control the behavior of processes with respect to traditional systems relying solely on static typing, as in AΠ.

In particular, we show that there exists a sub-type preserving encoding of AΠ@ into AΠ which is fully abstract, i.e., preserves the dynamically typed equivalences of AΠ@ in all AΠ contexts. The encoding is interesting in two respects. First, it yields a non-trivial, and in some respects surprising, expressivity result connecting dynamic to static typing. Secondly, it establishes a connection between AΠ and the fully abstract implementation developed in [3]. In particular, it allows us to isolate the fragment of AΠ for which the implementation of [3] is fully abstract.

Plan of the paper. Section 2 reviews the two calculi involved in the encoding. Sections 3 and 4 detail the encodings and outline the proof of full abstraction. Section 5 concludes with final remarks. We include an appendix with more details on the proofs.

2 Static and Dynamic Typing in the Pi-Calculus

We presuppose two countable sets of names and variables, ranged over by $a, b, \ldots, n, m$ and $x, y, \ldots$, respectively; u, v range collectively over names and variables whenever the distinction does not matter; $\tilde{u}$, $\tilde{T}$, $\tilde{A}$ denote (possibly empty) tuples of values, and static and dynamic types, respectively; the notation $\tilde{v}@\tilde{A}$ is a shorthand for the tuple $v_1@A_1, \ldots, v_n@A_n$; a corresponding convention applies to $\tilde{v} : \tilde{A}$. Syntactically, AΠ and AΠ@ differ only in the form of the communication primitives, and in the structure of the types. The productions are as follows:

AΠ – *Static Typing*

$$\begin{array}{rrcl} \textit{Processes } P, Q & ::= & \mathbf{0} \mid P|Q \mid (\nu n{:}S)P \mid\, !\,P \mid [u{=}v]P; Q \mid \overline{u}\langle\tilde{v}\rangle \mid u(\tilde{x}).P \\ \textit{Types } S, T & ::= & \top \mid \mathsf{r}\langle\tilde{T}\rangle \mid \mathsf{w}\langle\tilde{T}\rangle \mid \mathsf{rw}\langle\tilde{S};\tilde{T}\rangle \mid X \mid \mu X.T \end{array}$$

AΠ@ – *Dynamic Typing*

$$\begin{array}{rrcl} \textit{Processes } P, Q & ::= & \mathbf{0} \mid P|Q \mid (\nu n{:}A)P \mid\, !\,P \mid [u{=}v]P; Q \mid \overline{u}\langle\tilde{v}@\tilde{A}\rangle \mid u(\tilde{x}@\tilde{B}).P \\ \textit{Types } A, B & ::= & \top \mid \mathsf{r} \mid \mathsf{w} \mid \mathsf{rw} \end{array}$$

Table 1 Typing and subtyping rules

Subtyping in APi (we write $S <: T$ if $\emptyset \vdash S <: T$).

$$\frac{\Sigma \vdash \tilde{U}_w <: \tilde{T}_w <: \tilde{T}_r}{\Sigma \vdash \mathsf{rw}\langle \tilde{T}_r; \tilde{T}_w\rangle <: \mathsf{w}\langle \tilde{U}_w\rangle} \qquad \frac{\Sigma \vdash \tilde{U}_w <: \tilde{T}_w <: \tilde{T}_r <: \tilde{U}_r}{\Sigma \vdash \mathsf{rw}\langle \tilde{T}_r; \tilde{T}_w\rangle <: \mathsf{rw}\langle \tilde{U}_r; \tilde{U}_w\rangle} \qquad \frac{\Sigma \vdash \tilde{T}_w <: \tilde{T}_r <: \tilde{U}_r}{\Sigma \vdash \mathsf{rw}\langle \tilde{T}_r; \tilde{T}_w\rangle <: \mathsf{r}\langle \tilde{U}_r\rangle}$$

$$\frac{\Sigma, \mu X.T_1 <: T_2 \vdash T_1\{\mu X.T_1/X\} <: T_2}{\Sigma \vdash T_1 <: T_2} \qquad \frac{\Sigma, T_1 <: \mu X.T_2 \vdash T_1 <: T_2\{\mu X.T_2/X\}}{\Sigma \vdash T_1 <: T_2}$$

Typing of communication in APi and APi@

$$\text{(APi-IN)}\quad \frac{\Gamma \vdash u : \mathsf{r}\langle \tilde{T}\rangle \quad \Gamma, \langle \tilde{x} : \tilde{T}\rangle \vdash P}{\Gamma \vdash u(\tilde{x}).P} \qquad \text{(APi-OUT)}\quad \frac{\Gamma \vdash u : \mathsf{w}\langle \tilde{T}\rangle \quad \Gamma \vdash \tilde{v} : \tilde{T}}{\Gamma \vdash \overline{u}\langle \tilde{v}\rangle}$$

$$\text{(APi@-IN)}\quad \frac{\Gamma \vdash u : \mathsf{r} \quad \Gamma, \langle \tilde{x} : \tilde{A}\rangle \vdash P}{\Gamma \vdash u(\tilde{x}@\tilde{A}).P} \qquad \text{(APi@-OUT)}\quad \frac{\Gamma \vdash u : \mathsf{w} \quad \Gamma \vdash \tilde{v} : \tilde{B}}{\Gamma \vdash \overline{u}\langle \tilde{v}@\tilde{B}\rangle}$$

APi is a standard version of the asynchronous pi-calculus with matching, denoted by the construct $[u = v]P; Q$, and recursive capability types á la [7, 8][1]. We use the shorthand $\mathsf{rw}\langle \tilde{T}\rangle$ to mean $\mathsf{rw}\langle \tilde{T}; \tilde{T}\rangle$. APi@ replaces the input and output forms from APi with new constructs that make explicit the types at which values should be exchanged. As anticipated, the types are reduced to the simplest structure that only exhibits the capabilities associated with names.

Typing and Subtyping. In both calculi, the subtype relations $<:$ are partially complete pre-orders with a meet operator $\sqcap$ and top type $\top$. In APi the subtype relation is defined as in [7, 8], with the standard extensions needed to handle the presence of recursive types, cf. Table 1. In APi@ subtyping is generated by the axioms $\mathsf{rw} <: \{\mathsf{r}, \mathsf{w}\} <: \top$. *Type environments*, ranged over by $\Gamma, \Gamma' \ldots$, are finite mappings from names and variables to types. We write $\Gamma \vdash P$ to mean that P is well typed in Γ. The type environment $\Gamma, \langle u : T\rangle$ is $\Gamma, u : T$ if $u \notin dom(\Gamma)$; otherwise it is the type environment Γ' such that $\Gamma'(v) = \Gamma(v)$ for $v \neq u$ and $\Gamma'(u) = \Gamma(u) \sqcap T$ if $\Gamma(u) \sqcap T$ is defined. Subtyping is extended to type environments as expected: $\Gamma <: \Gamma'$ whenever $dom(\Gamma) = dom(\Gamma')$ and for all $v \in dom(\Gamma)$ it holds $\Gamma(v) <: \Gamma'(v)$.

Operational Semantics. The dynamics of APi is expressed by the usual labeled transition system of the asynchronous pi-calculus. On the other hand, in order to express the dynamics for APi@, the input/output labels of pi-calculus are extended with a type capability in order to force the synchronization at the desired type. In Table 2 we report the rules for synchronization: the remaining rules are standard, and common to

[1] In fact, in [7, 8] the types are not recursive; however, as far as we see, the generalization is harmless for the results relevant to our present endeavor.

Table 2 Labeled Transitions

$$\text{(APi-IN)}\quad \frac{}{a(\tilde{x}).P \xrightarrow{a(\tilde{v})} P\{\tilde{v}/\tilde{x}\}} \qquad \text{(APi-OUT)}\quad \frac{}{\overline{a}\langle\tilde{v}\rangle \xrightarrow{\overline{a}\langle\tilde{v}\rangle} \mathbf{0}} \qquad \text{(APi-OPEN)}\quad \frac{P \xrightarrow{(\tilde{c}:\tilde{T})\overline{a}\langle\tilde{v}\rangle} P' \quad b \in \{\tilde{v}\} \setminus \{a, \tilde{c}\}}{(\nu b:S)P \xrightarrow{(b:S\cdot\tilde{c}:\tilde{T})\overline{a}\langle\tilde{v}\rangle} P'}$$

$$\text{(APi-COMM)}\quad \frac{P \xrightarrow{(\tilde{c}:\tilde{T})\overline{a}\langle\tilde{v}\rangle} P' \quad Q \xrightarrow{a(\tilde{v})} Q' \quad \tilde{c} \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{T})(P' \mid Q')}$$

$$\text{(APi@-IN)}\quad \frac{}{a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A})} P\{\tilde{v}/\tilde{x}\}} \qquad \text{(APi@-OUT)}\quad \frac{}{\overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}} \qquad \text{(APi@-OPEN)}\quad \frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{A}@\tilde{v}\rangle} P' \quad b \in \{\tilde{v}\} \setminus \{a, \tilde{c}\}}{(\nu b:B)P \xrightarrow{(b:B\cdot\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'}$$

$$\text{(APi@-COMM)}\quad \frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P' \quad Q \xrightarrow{a(\tilde{v}@\tilde{B})} Q' \quad \tilde{A} <: \tilde{B} \quad \tilde{c} \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')}$$

---

the two calculi, we omit them for the lack of space. Notice how in APi@ the labels carry extra information about the types at which names are exchanged. More interestingly, however, these rules show the fundamentally different nature of the interaction between processes: in APi the receiver acquires the names emitted at the read type of the transmission channel, while in APi@ the type $\tilde{B}$ is decided by the sender. In addition, processes of both calculi may synchronize with a τ transition when exhibiting complementary labels that, in the case of APi@, are required to agree on the type of the values exchanged, as in $a(\tilde{v}@\tilde{A})$ and $\overline{a}\langle\tilde{v}@\tilde{A}\rangle$.

The notion of observational equivalence is based on the usual notion of reduction barbed congruence and is mediated by the type capabilities that the contexts possess on the names shared with the processes. It relies on the notion of *configuration*: a configuration is a pair of the form $\mathcal{I} \triangleright P$, where $\mathcal{I}$ represents what contexts know of the names shared with P: remarkably, P may know those names at more precise types than $\mathcal{I}$, as $\mathcal{I} \triangleright P$ is a well-defined configuration only if there exists $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$.

Definition 1 (Type-indexed Relation). *A type-indexed relation $\mathcal{R}$ is a family of binary relations between processes indexed by type environments. We write $\mathcal{I} \models P\ \mathcal{R}\ Q$ to mean (i) that P and Q are related by $\mathcal{R}$ and (ii) that $\mathcal{I} \triangleright P$ and $\mathcal{I} \triangleright Q$ are configurations.*

Definition 2 (Contextuality). *A type-indexed relation $\mathcal{R}$ is contextual when:*

1. *$\mathcal{I}, a : A \models P\ \mathcal{R}\ Q$ implies $\mathcal{I} \models (\nu a:A)P\ \mathcal{R}\ (\nu a:A)Q$*

2. $\mathcal{I} \models P\ \mathcal{R}\ Q$ *implies* $\mathcal{I}, a : A \models P\ \mathcal{R}\ Q$ *for* $a \notin dom(\mathcal{I})$
3. $\mathcal{I} \models P\ \mathcal{R}\ Q$ *and* $\mathcal{I} \vdash R$ *imply* $\mathcal{I} \models P|R\ \mathcal{R}\ Q|R$

The definition of barbs is based on the actual capability of the context $\mathcal{I}$ to see barbs. The notation $\mathcal{I}^{\mathsf{r}}(a) \downarrow$ indicates that $\mathcal{I}$ (hence the context) has a read capability on the name a, i.e., $\mathcal{I}(a) <: \mathsf{r}$, only in that case the output action performed by the process is observable by the context. Moreover we denote by $\Longrightarrow$ the reflexive and transitive closure of $\xrightarrow{\tau}$.

Definition 3 (Barbs). *Given a configuration* $\mathcal{I} \rhd P$*, we say that*

1. $\mathcal{I} \rhd P{\downarrow_a}$ *if and only if* $\mathcal{I}^{\mathsf{r}}(a) \downarrow$ *and* $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle}$.
2. $\mathcal{I} \rhd P{\Downarrow_a}$ *if and only if* $P \Longrightarrow P'$ *and* $\mathcal{I} \rhd P'{\downarrow_a}$.

The definition above is given for aPi@; the corresponding definition for aPi is just as expected.

Definition 4 (Typed Behavioral Congruence). *Typed behavioral congruence is the largest symmetric, contextual and type-indexed relation* $\mathcal{R}$ *such that* $\mathcal{I} \models P\ \mathcal{R}\ Q$ *implies*

1. *if* $\mathcal{I} \models P{\downarrow_a}$ *then* $\mathcal{I} \models Q{\Downarrow_a}$
2. *if* $Q \xrightarrow{\tau} Q$ *then there exists* Q' *such that* $Q \Longrightarrow Q'$ *and* $\mathcal{I} \models P'\ \mathcal{R}\ Q'$.

The typed behavioral congruence is denoted as $\approx$ in aPi and as $\approx_@$ in aPi@.

3 A correct encoding

We define our encoding in terms of two related, but independent mappings, for type environments and processes, respectively. The encoding of processes maps typing judgements in aPi@ to processes of aPi: indeed, in our first formulation of the encoding, we define it directly by induction on the structure of processes, as the syntax of aPi@ contains enough information to guide the generation of target code. In the final formulation, however, it is technically more convenient to refer to typing judgements to ease the definition. The encoding of type environments, in turn, maps the capabilities (types) of the observing aPi@ contexts into the corresponding capabilities of aPi contexts. This allows us to establish our full abstraction theorem in terms of the type-indexed relations of behavioral congruences in the two calculi.

In this and the next sections we give the encoding for the monadic fragment of aPi@. In Section 5, we briefly discuss how the approach can be generalized to the polyadic calculus.

First attempt. We start with a relatively simple approach, which almost work, but not quite. The idea is to represent each name n in aPi@ as a 4-tuple of names $\underline{n} = (n_{\mathsf{rw}}, n_{\mathsf{r}}, n_{\mathsf{w}}, n_{\top})$, where each component of the tuple corresponds to one of the four types at which a synchronization may take place on n. Thus, an aPi@ synchronization on n at, say, the type w, will correspond in aPi to a synchronization on n_{w}. Clearly, this idea must be applied systematically, hence exchanging a name in aPi@ will correspond to

Table 3 Encoding functions (with $Q = u(\tilde{x}).P$)

$$\textsc{Read}_l\,\langle Q\rangle \stackrel{def}{=} u(\tilde{x}).\textsc{Test}_l\,\langle Q\rangle$$

$$\textsc{Test}_l\,\langle Q\rangle \stackrel{def}{=} l(z).[z=\mathtt{t}]\textsc{Commit}_l\,\langle Q\rangle \oplus \textsc{Undo}_l\,\langle Q\rangle\,;\,\textsc{Abort}_l\,\langle Q\rangle \qquad z \text{ fresh}$$

$$\textsc{Undo}_l\,\langle Q\rangle \stackrel{def}{=} \overline{l}\langle\mathtt{t}\rangle \mid \overline{u}\langle\tilde{x}\rangle$$

$$\textsc{Commit}_l\,\langle Q\rangle \stackrel{def}{=} \overline{l}\langle\mathtt{f}\rangle \mid P$$

$$\textsc{Abort}_l\,\langle Q\rangle \stackrel{def}{=} \overline{l}\langle\mathtt{f}\rangle \mid \overline{u}\langle\tilde{x}\rangle$$

$$R_1 \oplus R_2 \stackrel{def}{=} (\nu i:\mathsf{rw}\langle\,\rangle)(\overline{i}\langle\,\rangle \mid i().R_1 \mid i().R_2) \qquad i \text{ fresh}$$

exchanging a tuple in aPi. Thus the output $\overline{u}\langle v@A\rangle$ is translated into the output $\overline{u_A}\langle\underline{v}\rangle$ that emits the 4-tuple of names that represent v. The input prefix requires a little more care, as in aPi@ the process $u(x@A).P$ may synchronize with any output on u at a type $B <: A$. In fact, given the translation of the output construct we just outlined, it is easily seen that the behavior of the input form $u(x@A).P$ corresponds precisely to the input guarded choice[2] $\Sigma_{B<:A} u_B(\underline{x}).P$. Combining these intuitions with the encoding of guarded choice from [10] yields the following definition.

$$\begin{aligned}
(\!|\,\overline{u}\langle v@A\rangle\,|\!) &\stackrel{def}{=} \overline{u_A}\langle\underline{v}\rangle\\
(\!|\,u(x@A).P\,|\!) &\stackrel{def}{=} (\nu l:\mathsf{rw}\langle\top\rangle)\left(\overline{l}\langle\mathtt{t}\rangle \mid \textstyle\prod_{B<:A} !\,\textsc{Read}_l\left\langle u_B(\underline{x}).(\!|\,P\,|\!)\right\rangle\right)\\
(\!|\,(\nu\, n)P\,|\!) &\stackrel{def}{=} (\nu\,\underline{n})(\!|\,P\,|\!)\\
(\!|\,[u=v]P;Q\,|\!) &\stackrel{def}{=} [u_\top = v_\top](\!|\,P\,|\!);(\!|\,Q\,|\!)
\end{aligned}$$

Just as in [10], the encoding of input runs a mutual exclusion protocol, installing a local lock on a parallel composition of its branches. The protocol is implemented by the processes in Table 3, which we inherit, essentially unchanged, from [10] (as in that case, $\oplus$ denotes internal choice).The branches $\textsc{Read}_l\,\langle -\rangle$ concurrently try to test the lock after reading messages from the environment. Every branch can 'black out' and return to its initial state after it has taken the lock, just by resending the message. Just one branch will proceed with its continuation and thereby commit the input. Every other branch will then be forced to resend its message and abort its continuation.

Unfortunately, the re-sending of messages is problematic for type preservation. To see why, notice that a aPi@ process may have just the read capability on a channel in order to perform an input, while its encoding must also be granted a write capability in order to run the mutual exclusion protocol. The typing failure would not arise if we dropped the type r, and worked with just three types: indeed, for this fragment of aPi@ the encoding we just illustrated may be shown to be type-preserving and fully abstract. For a more general solution, we need to move the responsibility of running the mutual exclusion protocol from the (encoding of the) input process to some other process possessing the required capabilities. We discuss how that can be accomplished below.

[2] For uniformity with the notation adopted for names, we write $\underline{x}$ to note a quadruple of variables used to store name representations.

Table 4 A correct encoding of processes

Channel Servers

$$\textsc{Chan}(n) \stackrel{def}{=} \prod_{A\in\{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}} !\, n_{\mathsf{r}@A}(h).\textsc{Choose}(n,A,h)$$
$$\textsc{Choose}(n,A,h) \stackrel{def}{=} (\nu l:\mathsf{rw}\langle\top\rangle)\Big(\overline{l}\langle\mathsf{t}\rangle \mid \prod_{B<:A} !\,\textsc{Read}_l\left\langle n_{\mathsf{w}@B}(\underline{z}).\overline{h}\langle\underline{z}\rangle\right\rangle\Big)$$

Clients

$$\{\!|\,\mathbf{0}\,|\!\} \stackrel{def}{=} \mathbf{0}$$
$$\{\!|\,\overline{u}\langle v@A\rangle\,|\!\} \stackrel{def}{=} \overline{u_{\mathsf{w}@A}}\langle\underline{v}\rangle$$
$$\{\!|\,u(x@A).P\,|\!\} \stackrel{def}{=} (\nu h:\mathsf{rw}\langle\mathbb{T}_A\rangle)\left(\overline{u_{\mathsf{r}@A}}\langle h\rangle \mid h(\underline{x}).\{\!|\,P\,|\!\}\right) \quad \text{where the name } h \text{ is fresh}$$
$$\{\!|\,P\mid Q\,|\!\} \stackrel{def}{=} \{\!|\,P\,|\!\}\mid\{\!|\,Q\,|\!\}$$
$$\{\!|\,(\nu a:A)P\,|\!\} \stackrel{def}{=} (\nu\underline{a}:\mathbb{S})\,(\{\!|\,P\,|\!\}\mid\textsc{Chan}(a))$$
$$\{\!|\,[u=v]P;Q\,|\!\} \stackrel{def}{=} [u_{\mathsf{r}@\mathsf{r}}=v_{\mathsf{r}@\mathsf{r}}]\{\!|\,P\,|\!\};\{\!|\,Q\,|\!\}$$
$$\{\!|\,!P\,|\!\} \stackrel{def}{=} !\,\{\!|\,P\,|\!\}$$

Complete Systems

$$\{\!|P|\!\}_{\mathcal{I}} \stackrel{def}{=} \{\!|\,P\,|\!\} \mid \prod_{a\in dom(\mathcal{I})}\textsc{Chan}(a)$$

Fixing the typing problem The solution is inspired by [3] and based on representing channels as processes that serve the input and output requests by a client willing to synchronize: given a name n, we write $\textsc{Chan}(n)$ for the server associated with n. Each exchange on n in the source calculus corresponds to running two separate protocols. For input, a client willing to input on n at type, say A, sends a read request (in the form of a private name) on the name $n_{\mathsf{r}@A}$. In the write protocol, a process willing to output on n and type $B <: A$ sends its output on $n_{\mathsf{w}@B}$. Collectively, each name n from $\textsc{Api}@$ is thus translated into the 8-tuple $\underline{n} = (n_{\mathbb{R}}, n_{\mathbb{W}})$, where the components $n_{\mathbb{R}} = n_{\mathsf{r}@\mathsf{rw}}, n_{\mathsf{r}@\mathsf{r}}, n_{\mathsf{r}@\mathsf{w}}, n_{\mathsf{r}@\top}$ are the names employed in the input protocol to communicate the input requests at the corresponding types, while the components $n_{\mathbb{W}} = n_{\mathsf{w}@\mathsf{rw}}, n_{\mathsf{w}@\mathsf{r}}, n_{\mathsf{w}@\mathsf{w}}, n_{\mathsf{w}@\top}$ serve the same purpose for the output protocol. Thus, for instance, the output $\overline{n}\langle v@\mathsf{rw}\rangle$ corresponds to the output $\overline{n_{\mathsf{w}@\mathsf{rw}}}\langle\underline{v}\rangle$ and synchronizes only with $n_{\mathsf{w}@\mathsf{rw}}(\underline{x})$. The input $n(x@\mathsf{rw}).P$, in turn, sends a request $\overline{n_{\mathsf{r}@\mathsf{rw}}}\langle l\rangle$, where l is a private channel on which the client waits for $\textsc{Chan}(n)$ to reply back a (tuple of) value(s). The server $\textsc{Chan}(n)$ is granted read and write access to all the names $n_{\mathbb{W}}$ and $n_{\mathbb{R}}$, so that it may safely run the mutual exclusion protocol that mimics the synchronizations in the source calculus. Indeed, $\textsc{Chan}(n)$ is the only process with read capabilities on $n_{\mathbb{W}}$ and $n_{\mathbb{R}}$ while clients will, at their best, have write capabilities on these names: as a result, no client may interfere with the protocols that other clients run with the channel server.

The definitions in Table 4 formalize these intuitions: each $\textsc{Api}@$ process corresponds, via the encoding, to a set of clients of the protocols described above: the relevant clauses are those for the input and output forms, while the remaining cases are defined homomorphically. Two remarks are in order, however. For the case of matching, it is enough to compare just one of the components of the tuple representing the source-level names, as long as the components are chosen consistently. For the case of restriction, notice that creating a new name corresponds to allocating a channel server associated to the name, so that the translated processes may synchronize via the name created,

Table 5 Encoding of types.

Client types

$$\mathbb{T}_{\mathsf{rw}} \stackrel{def}{=} (\mathbb{R}, \mathbb{W}) \quad \mathbb{T}_{\mathsf{r}} \stackrel{def}{=} (\mathbb{R}, \mathsf{T}) \quad \mathbb{T}_{\mathsf{w}} \stackrel{def}{=} (\mathsf{T}, \mathbb{W}) \quad \mathbb{T}_{\top} \stackrel{def}{=} (\mathsf{T}, \mathsf{T})$$

$$\mathbb{R} \stackrel{def}{=} (\boldsymbol{T}_{\mathsf{r@rw}}, \boldsymbol{T}_{\mathsf{r@r}}, \boldsymbol{T}_{\mathsf{r@rw}}, \boldsymbol{T}_{\mathsf{r@}\top}) \quad \mathbb{W} \stackrel{def}{=} (\boldsymbol{T}_{\mathsf{w@rw}}, \boldsymbol{T}_{\mathsf{w@r}}, \boldsymbol{T}_{\mathsf{w@rw}}, \boldsymbol{T}_{\mathsf{w@}\top})$$

$$\begin{array}{ll} \boldsymbol{T}_{\mathsf{r@rw}} \stackrel{def}{=} \mathsf{w}\langle \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle\rangle & \boldsymbol{T}_{\mathsf{w@rw}} \stackrel{def}{=} \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle \\ \boldsymbol{T}_{\mathsf{r@r}} \stackrel{def}{=} \mathsf{w}\langle \mathsf{w}\langle \mathbb{R}, \mathsf{T}\rangle\rangle & \boldsymbol{T}_{\mathsf{w@r}} \stackrel{def}{=} \mathsf{w}\langle \mathbb{R}, \mathsf{T}\rangle \\ \boldsymbol{T}_{\mathsf{r@w}} \stackrel{def}{=} \mathsf{w}\langle \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle\rangle & \boldsymbol{T}_{\mathsf{w@w}} \stackrel{def}{=} \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle \\ \boldsymbol{T}_{\mathsf{r@}\top} \stackrel{def}{=} \mathsf{w}\langle \mathsf{w}\langle \mathsf{T}, \mathsf{T}\rangle\rangle & \boldsymbol{T}_{\mathsf{w@}\top} \stackrel{def}{=} \mathsf{w}\langle \mathsf{T}, \mathsf{T}\rangle \end{array}$$

Server types

$$\mathbb{S} \stackrel{def}{=} (\mathbb{R}_{\mathbb{S}}, \mathbb{W}_{\mathbb{S}})$$

$$\mathbb{R}_{\mathbb{S}} \stackrel{def}{=} (\boldsymbol{S}_{\mathsf{r@rw}}, \boldsymbol{S}_{\mathsf{r@r}}, \boldsymbol{S}_{\mathsf{r@w}}, \boldsymbol{S}_{\mathsf{r@}\top}) \quad \mathbb{W}_{\mathbb{S}} \stackrel{def}{=} (\boldsymbol{S}_{\mathsf{w@rw}}, \boldsymbol{S}_{\mathsf{w@r}}, \boldsymbol{S}_{\mathsf{w@w}}, \boldsymbol{S}_{\mathsf{w@}\top})$$

$$\begin{array}{ll} \boldsymbol{S}_{\mathsf{r@rw}} \stackrel{def}{=} \mathsf{rw}\langle \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle\rangle & \boldsymbol{S}_{\mathsf{w@rw}} \stackrel{def}{=} \mathsf{rw}\langle \mathbb{R}, \mathbb{W}\rangle \\ \boldsymbol{S}_{\mathsf{r@r}} \stackrel{def}{=} \mathsf{rw}\langle \mathsf{w}\langle \mathbb{R}, \mathsf{T}\rangle\rangle & \boldsymbol{S}_{\mathsf{w@r}} \stackrel{def}{=} \mathsf{rw}\langle \mathbb{R}, \mathsf{T}\rangle \\ \boldsymbol{S}_{\mathsf{r@w}} \stackrel{def}{=} \mathsf{rw}\langle \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle\rangle & \boldsymbol{S}_{\mathsf{w@w}} \stackrel{def}{=} \mathsf{rw}\langle \mathsf{T}, \mathbb{W}\rangle \\ \boldsymbol{S}_{\mathsf{r@}\top} \stackrel{def}{=} \mathsf{rw}\langle \mathsf{w}\langle \mathsf{T}, \mathsf{T}\rangle\rangle & \boldsymbol{S}_{\mathsf{w@}\top} \stackrel{def}{=} \mathsf{rw}\langle \mathsf{T}, \mathsf{T}\rangle \end{array}$$

Type Environments

$$\{\!|\emptyset|\!\} \stackrel{def}{=} \mathtt{t} : \top, \mathtt{f} : \top \qquad \{\!|\Gamma, v : A|\!\} \stackrel{def}{=} \{\!|\Gamma|\!\}, (\underline{v}) : \mathbb{T}_A$$

using the server. In fact, to mimic all the synchronizations of the source processes, we must allocate channels for all the free names that occur in the source process and that the source process may share with the environment. That is accomplished by the final clause in the definition of the encoding.

Having moved the responsibility of running the mutual exclusion protocol from the input clients to the channel server solves the typing problem of our initial attempt. Table 5, details the encoding of types. Typewise, a read capability on n in aPi@ corresponds in aPi to a write capability on all the names in $n_{\mathbb{R}}$, while a write capability on n corresponds to a write capability on the names $n_{\mathbb{W}}$. With each type A in aPi@ we associate a corresponding tuple of types $\mathbb{T}_A$, as in $\mathbb{T}_{\mathsf{rw}} = (\mathbb{R}, \mathbb{W})$ or $\mathbb{T}_{\mathsf{w}} = (\mathsf{T}, \mathbb{W})$ where $\mathbb{R}$ and $\mathbb{W}$ are the client types associated to the names employed in the read/write protocols (according to the convention that $n_{\mathsf{r}@A} : \boldsymbol{T}_{\mathsf{r}@A}$ and $n_{\mathsf{w}@A} : \boldsymbol{T}_{\mathsf{w}@A}$) and T is the representation of the top type in aPi@. Notice that clients are only granted write capabilities on the channels involved in the protocols, while the channel servers know the same names at the lower types $\mathbb{R}_{\mathbb{S}}$ and $\mathbb{W}_{\mathbb{S}}$ which grant them full access to those names.

Based on these definition, we may now prove that the encoding preserves the expected properties about typing:

Theorem 1 (Typing and subtyping preservation). *For all types A, B in aPi@, $A <: B$ implies $\mathbb{T}_A <: \mathbb{T}_B$ in aPi. Furthermore, whenever $\Gamma \vdash P$ in aPi@, then there exists $\Gamma' <: \{\!|\Gamma|\!\}$ such that $\Gamma' \vdash \{\!|P|\!\}_\Gamma$ in aPi.*

Also, we can show that the encoding is sound, in the sense made precise below.

Theorem 2 (Soundness). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: \mathcal{I}$ be two type environments such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$ in aPi@. Then $\{\!|\mathcal{I}|\!\} \models \{\!|P|\!\}_\Gamma \approx \{\!|Q|\!\}_{\Gamma'}$ implies $\mathcal{I} \models P \approx_@ Q$.*

The converse direction of Theorem 2 does not hold. The problem is that the properties of the communication protocols are based on certain invariants that are verified by the names and the channel servers allocated by the encoding, but may fail for the names created dynamically by the context. Below, we show that this failure breaks full abstraction (i.e., the converse of Theorem 2).

Failure of full abstraction. Take the following two APi@ processes

$$P \stackrel{def}{=} a(x@\mathsf{rw}).x(y@\mathsf{rw}).\overline{x}\langle y@\mathsf{rw}\rangle$$
$$Q \stackrel{def}{=} a(x@\mathsf{rw}).\mathbf{0}$$

As shown in [2], one has $\mathcal{I} \models n(y@\mathsf{rw}).\overline{n}\langle y@\mathsf{rw}\rangle \approx_@ \mathbf{0}$ for all $\mathcal{I}$ such that $n : \mathsf{rw} \in \mathcal{I}$. From this, one easily derives $a : \mathsf{w} \models P \approx_@ Q$. Now take the encoding of the two processes (we omit the type for readability, as they are irrelevant to the present purposes):

$$\{\!|P|\!\} = (\nu h)(\overline{a_{\mathsf{r}@\mathsf{rw}}}\langle h\rangle \,|\, h(\underline{x}).(\nu k)(\overline{x_{\mathsf{r}@\mathsf{rw}}}\langle k\rangle \,|\, k(\underline{y}).\overline{x_{\mathsf{w}@\mathsf{rw}}}\langle \underline{y}\rangle))$$
$$\{\!|Q|\!\} = (\nu h)(\overline{a_{\mathsf{r}@\mathsf{rw}}}\langle h\rangle \,|\, h(\underline{x}).\mathbf{0})$$

We show that the equivalence we just established for P and Q does not carry over to their encodings. In particular, let $\mathcal{I} = a : \mathsf{w}$, so that $\{\!|\mathcal{I}|\!\} = \underline{a} : \mathbb{T}_{\mathsf{w}}$, and assume $\underline{a} : \mathbb{T}_{\mathsf{w}} \models \{\!|P|\!\}_{\mathcal{I}} \approx \{\!|Q|\!\}_{\mathcal{I}}$. Let also $\mathbb{S}$ be the server type defined in Table 5. Then, by contextuality, one would have:

$$\underline{a} : \mathbb{T}_{\mathsf{w}}, \underline{b} : \mathbb{S} \models \{\!|P|\!\}_{\mathcal{I}} \,|\, \overline{a_{\mathsf{w}@\mathsf{rw}}}\langle \underline{b}\rangle \approx \{\!|Q|\!\}_{\mathcal{I}} \,|\, \overline{a_{\mathsf{w}@\mathsf{rw}}}\langle \underline{b}\rangle$$

On the other hand, this equivalence is easily disproved. In fact, on the one hand we have:

$$\{\!|P|\!\}_{\mathcal{I}} \,|\, \overline{a_{\mathsf{w}@\mathsf{rw}}}\langle \underline{b}\rangle \Longrightarrow\approx (\nu k)(\overline{b_{\mathsf{r}@\mathsf{rw}}}\langle k\rangle \,|\, k(\underline{y}).\overline{b_{\mathsf{w}@\mathsf{rw}}}\langle \underline{y}\rangle)$$

with

$$\underline{a} : \mathbb{T}_{\mathsf{w}}, \underline{b} : \mathbb{S} \rhd (\nu k)(\overline{b_{\mathsf{r}@\mathsf{rw}}}\langle k\rangle \,|\, k(\underline{y}).\overline{b_{\mathsf{w}@\mathsf{rw}}}\langle \underline{y}\rangle)\downarrow_{b_{\mathsf{r}@\mathsf{rw}}}$$

because $\underline{b} : \mathbb{S}$ implies that $b_{\mathsf{r}@\mathsf{rw}} : \mathsf{rw}\langle \mathsf{w}\langle \mathbb{T}_{\mathsf{rw}}\rangle\rangle$, hence the context has visibility of the output action by the process. On the other hand, clearly,

$$\underline{a} : \mathbb{T}_{\mathsf{w}}, \underline{b} : \mathbb{S} \rhd \{\!|Q|\!\}_{\mathcal{I}} \,|\, \overline{a_{\mathsf{w}@\mathsf{rw}}}\langle \underline{b}\rangle \Downarrow_{b_{\mathsf{r}@\mathsf{rw}}}$$

as Q never attempts to output on b, and correspondingly its encoding makes no requests on any of the components in $\underline{b}$. Thus, it follows that $\underline{a} : \mathbb{T}_{\mathsf{w}} \not\models \{\!|P|\!\}_{\mathcal{I}} \approx \{\!|Q|\!\}_{\mathcal{I}}$ as we anticipated.

4 A fully abstract encoding

To recover full abstraction, we need to protect the clients generated by the encoding from direct interactions on context-generated names such as the one illustrated above. To accomplish that, we adopt a solution inspired by [3], which relies on a *proxy* service

Table 6 Fully Abstract Encoding of aPi@ into aPi.

Proxy

$$\textsc{Proxy} \stackrel{def}{=} (\nu t:\mathtt{TBL}[\top,\mathbb{S}])\,(\textsc{Server}(t) \mid \mathtt{TABLE}(t,[\,]))$$
$$\textsc{Server}(t) \stackrel{def}{=} \prod_A\, !\,p_A(h,z).(\nu r:\mathsf{rw}\langle\top,\mathbb{S}\rangle)\left(\mathtt{LOOKUP}(z,t,r) \mid r(x,\underline{y})[x=\mathtt{t}]\overline{h}\langle\underline{y}\rangle;(\overline{h}\langle\underline{y}\rangle \mid \textsc{Chan}(y))\right)$$

Clients

$$(\![\,\mathbf{0}\,]\!)_\Gamma \stackrel{def}{=} \mathbf{0}$$
$$(\![\,\overline{u}\langle v@A\rangle\,]\!)_\Gamma \stackrel{def}{=} \textsc{Link}_\Gamma\ (u,\underline{x})\ \textsc{in}\ \overline{x_{\mathsf{w}@A}}\langle\underline{v}\rangle$$
$$(\![\,u(y@A).P\,]\!)_\Gamma \stackrel{def}{=} \textsc{Link}_\Gamma\ (u,\underline{x})\ \textsc{in}\ (\nu h:\mathsf{rw}\langle\mathbb{T}_A\rangle)\left(\overline{x_{\mathsf{r}@A}}\langle h\rangle \mid h(\underline{y}).(\![\,P\,]\!)_{\Gamma,y:A}\right) \qquad \text{with } h \text{ fresh}$$
$$(\![\,P \mid Q\,]\!)_\Gamma \stackrel{def}{=} (\![\,P\,]\!)_\Gamma \mid (\![\,Q\,]\!)_\Gamma$$
$$(\![\,(\nu a:A)P\,]\!)_\Gamma \stackrel{def}{=} (\nu \underline{a}:\mathbb{T}_A)(\![\,P\,]\!)_{\Gamma,a:A}$$
$$(\![\,[u=v]P;Q\,]\!)_\Gamma \stackrel{def}{=} [u_{\mathsf{r}@\mathsf{r}}=v_{\mathsf{r}@\mathsf{r}}](\![\,P\,]\!)_{\Gamma,\langle u:\Gamma(u)\rangle,\langle v:\Gamma(v)\rangle};(\![\,Q\,]\!)_\Gamma$$
$$(\![\,!\,P\,]\!)_\Gamma \stackrel{def}{=} !\,(\![\,P\,]\!)_\Gamma$$

Complete Systems

$$(\![\,P\,]\!)^*_\Gamma \stackrel{def}{=} (\![\,P\,]\!)_\Gamma \mid \textsc{Proxy}$$

to filter the interactions between channel servers, clients and the context. The *proxy* introduces a separation between *client names*, used by the context and the translated processes to communicate, and the corresponding *proxy names*, generated within the system and associated with system generated channels which are employed in the actual protocols for communication.

The proxy server is a process, noted Proxy, that maintains an association map between client and server names in order to preserve the expected interactions among clients. The read and write protocols follow the same rationale as in the previous encoding, with the difference that in the new version of the encoding a client must obtain the access to the system channel with a request to Proxy before being able to start the input/output protocols. The interaction between clients and Proxy is now as follows: the client presents a name to the proxy, and the proxy replies with the corresponding server name. When the proxy server receives a client name for the first time, it maps it to a fresh proxy name, and allocates a channel server for the new proxy name.

The definition of the proxy server is reported in Table 6. Server is a process always ready to serve client requests along the four channels p_A, one for each $A \in \{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}$. These channels are known to the clients and to the context at the type $p_A : \mathsf{w}\langle\mathsf{w}\langle\mathbb{T}_A\rangle,\mathbb{T}_A\rangle$, while the Server is granted full access rights on them. After receiving an input on p_A, the Server starts a search on the association table and replies with the requested system name. If the client name was not known to the proxy, a fresh proxy name is created, the association table extended with the new pair, and a channel server for the newly created proxy name allocated. We omit most of the largely obvious details of the implementation of the association table, and describe it in terms of the following macros.

- $\mathtt{TABLE}(t,\mathcal{T})$ is a process parameterized over a structure $\mathcal{T}$ representing the table, and $t : \mathtt{TBL}[\top,\mathbb{S}]$ is the reference to the table, i.e., a name providing access to the table's entries. We organize $\mathcal{T}$ as a list of pairs that maps channel names to server

names: we write $(n, \underline{k}) \in \mathcal{T}$ when n is associated to the tuple $\underline{k}$, and $n \in dom(\mathcal{T})$ to say that there exists $\underline{k}$ with $(n, \underline{k}) \in \mathcal{T}$. Initially, the list $\mathcal{T}$ is empty (cf. Table 6).

– $\texttt{LOOKUP}(n, t, r)$ is a process employed by the proxy to access the table referenced to by $t : \texttt{TBL}[\top, \mathbb{S}]$. Here, $n : \top$ is the name (the client name) to be looked up in the table, and $r : \texttt{w}\langle\top, \mathbb{S}\rangle$ is the reply channel where to report back the result of the search. The result, in turn, comes as a pair that comprises the proxy names associated together with a boolean flag that says whether a new entry was created in the table as a result of the search. Thus, $\texttt{LOOKUP}(n, t, r)$ replies on r the pair $(x, \underline{k})$ where $\underline{k}$ is the tuple associated with n, and x is $\texttt{t}$ iff $\underline{k}$ was created freshly for n. This information is used by the remaining component of the proxy to allocate a new channel server for newly created proxy names, and to forward the proxy names to the requesting clients.

Operationally, we define the behavior of the macro processes by means of the following two ad-hoc internal reductions:

(Table-Lookup-Found)
$$\frac{(n, \underline{k}) \in \mathcal{T}}{\texttt{LOOKUP}(n, t, r) \mid \texttt{TABLE}(t, \mathcal{T}) \xrightarrow{\tau} \overline{r}\langle \texttt{t}, \underline{k}\rangle \mid \texttt{TABLE}(t, \mathcal{T})}$$

(Table-Lookup-NotFound)
$$\frac{n \notin dom(\mathcal{T}), \quad \underline{k} \text{ fresh in } \mathcal{T}}{\texttt{LOOKUP}(n, t, r) \mid \texttt{TABLE}(t, \mathcal{T}) \xrightarrow{\tau} (\nu \underline{k} : \mathbb{S}) \left(\overline{r}\langle \texttt{f}, \underline{k}\rangle \mid \texttt{TABLE}(t, [(n, \underline{k}) :: \mathcal{T}]) \right)}$$

On the client side, the interaction with the proxy server requires a new initialization step to link the name available to the client with the corresponding proxy name associated with it. This init step is accomplished as follows:

$$\textsc{Link}_{\Gamma}\ (u, \underline{x})\ \textsc{in}\ P \stackrel{def}{=} (\nu h : \texttt{rw}\langle \mathbb{T}_A \rangle)(\overline{p_A}\langle h, u_{\texttt{r}@\texttt{r}} \rangle \mid h(\underline{x}).P) \quad \text{with } A = \Gamma(u)$$

To link the client name u (in fact, the component names in $\underline{u}$) with a corresponding proxy name, the client selects the first component in $\underline{u}$ on the proxy channel dedicated to serve the link requests, and waits for the proxy to reply on the channel h. Notice that the definition of the link process is parameterized on the context Γ, which helps recover the type that then guides the selection of the appropriate channel p_A used in the interaction with the proxy. To make this parametrization meaningful, the new encoding is given by induction on typing judgements. The other important difference with respect to the original definition in Table 4 is that the case of restriction is now defined purely homomorphically, as the allocation of the channel server is delegated to the proxy. For the same reason, the encoding of complete system does not need to allocate any channel for the free names shared with the context. It does, instead, require the presence of the proxy server to filter the synchronizations between clients.

Typewise, there are only few differences with respect to the original definitions in Table 5. Indeed, the exact same types work with the new translation of clients,

while a two remarks are in order for the types employed in the definition of the proxy. First, the channels p_A are made available to the clients (and the context) at the types $\mathsf{w}\langle\mathsf{w}\langle\mathbb{T}_A\rangle, \mathbb{T}_A\rangle$, with $A \in \{\mathsf{rw}, \mathsf{r}, \mathsf{w}, \top\}$, that only grant write access. Correspondingly, we have a new encoding of type environments:

$$(\![\, \Gamma \,]\!) \stackrel{def}{=} \{\!|\, \Gamma \,|\!\} \cup \{p_A : \mathsf{w}\langle\mathsf{w}\langle\mathbb{T}_A\rangle, \mathbb{T}_A\rangle\}_{A \in \{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}}$$

with $\{\!|\, \Gamma \,|\!\}$ as in Table 5. Lower types, preciselythe types $\mathsf{rw}\langle\mathsf{w}\langle\mathbb{T}_A\rangle, \mathbb{T}_A\rangle$, are instead available for type checking the proxy definition.

As for the type $\mathtt{TBL}[\top, \mathbb{S}]$ employed in the definition of the `TABLE` macro, $\top$ and $\mathbb{S}$ are the types of the entries in the table, while we assume the following ad-hoc typing rules for the `LOOKUP` and `TABLE` macro processes:

$$\frac{\Gamma \vdash t : \mathtt{TBL}[\top, \mathbb{S}], n : \top, r : \mathsf{rw}\langle\top, \mathbb{S}\rangle}{\Gamma \vdash \mathtt{LOOKUP}(n, t, r)}\ (\text{T-Lookup}) \qquad \frac{\Gamma \vdash t : \mathtt{TBL}[\top, \mathbb{S}], n_1 : \top, \underline{k}_1 : \mathbb{S}, \ldots n_l : \top, \underline{k}_l : \mathbb{S}}{\Gamma \vdash \mathtt{TABLE}(t, [(n_1, \underline{k}_1) :: \cdots (n_l, \underline{k}_l)])}\ (\text{T-Table})$$

Based on these definitions, we can show that the encoding has the desired properties of type and subtype preservation.

Theorem 3 (Typing and subtyping preservation). *For all types A, B in* aPi@*, $A <: B$ implies $\mathbb{T}_A <: \mathbb{T}_B$ in* aPi*. Furthermore, whenever $\Gamma \vdash P$ in* aPi@*, then there exists $\Gamma' <: (\![\, \Gamma \,]\!)$ such that $\Gamma' \vdash (\![\, P \,]\!)^*_\Gamma$ in* aPi*.*

Furthermore, the presence of the proxy now makes the encoding fully abstract.

Theorem 4 (Full Abstraction). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: I$ be two type environments such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$. Then $\mathcal{I} \models P \approx_@ Q$ if and only if $(\![\, \mathcal{I} \,]\!) \models (\![\, P \,]\!)^*_\Gamma \approx (\![\, Q \,]\!)^*_{\Gamma'}$.*

Below we outline the full abstraction proof. However, before doing that, it is instructive to look at how the new encoding solves the problem with the example discussed in Section 3. In that case, the encodings of the two processes

$$P \stackrel{def}{=} a(x@\mathsf{rw}).x(y@\mathsf{rw}).\overline{x}\langle y@\mathsf{rw}\rangle$$
$$Q \stackrel{def}{=} a(x@\mathsf{rw}).\mathbf{0}$$

can be distinguished by any context that sends a fresh name on a and retains full access to the components of that name, because any such context may observe the read request made by the encoding of P.

The presence of the proxy solves the problem as now the encoding of P makes its request not on the name received by the context, but rather on the proxy name that is associated with the context name. Thus, if b is the name send over the channel a, we have:

$$(\![\, P \,]\!)_\Gamma \,|\, \overline{a_{\mathsf{w}@\mathsf{rw}}}\langle\underline{b}\rangle \Longrightarrow\approx \textsc{Link}_\Gamma\ (b, x)\ \textsc{in}\ (\nu k)(\overline{x_{\mathsf{r}@\mathsf{rw}}}\langle k\rangle \,|\, k(\underline{y}).\overline{x_{\mathsf{w}@\mathsf{rw}}}\langle\underline{y}\rangle)$$

where now

$$\underline{a} : \mathbb{T}_\mathsf{w}, \underline{b} : \mathbb{S} \rhd \textsc{Link}_\Gamma\ (b, x)\ \textsc{in}\ (\nu k)(\overline{x_{\mathsf{r}@\mathsf{rw}}}\langle k\rangle \,|\, k(\underline{y}).\overline{x_{\mathsf{w}@\mathsf{rw}}}\langle\underline{y}\rangle)\ \not\Downarrow_{b_{\mathsf{r}@\mathsf{rw}}}$$

as the context has no read access on the components of the proxy name $\underline{x}$ associated with b.

Proof of Theorem 4 (outline). The proof is difficult and rather elaborate, especially in the "if" direction (*soundness*), which as usual requires the following properties of operational correspondence:

- If $P \Longrightarrow P'$ then $(\!|\, P \,|\!)^*_\Gamma \Longrightarrow K$ with $(\!|\, \mathcal{I} \,|\!) \models K \approx (\!|\, P' \,|\!)^*_\Gamma$.
- If $(\!|\, P \,|\!)^*_\Gamma \Longrightarrow K$ then there exists P' such that $P \Longrightarrow P'$ and $(\!|\, \mathcal{I} \,|\!) \models K \approx (\!|\, P' \,|\!)^*_\Gamma$.

The "reflection" direction, stated by the second item above, is subtle, because our encoding is not "prompt" [10]. Note, in fact, that it takes several steps for the encoding of a process to be ready for the commit action that corresponds to the aPi@ 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, noted $\approx_A$ and included in $\approx$. The definition of $\approx_A$ draws on a classification for the reductions of the translated processes into *commitment* steps, corresponding to synchronizations in the aPi@ processes, and *administrative reductions*, corresponding to the steps that precede and follow the commitment steps. 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. The relation $\approx_A$ can be used to prove the following variant of operational correspondence:

Lemma 1 (Operational Correspondence). *Let $\mathcal{I} \triangleright P$ be a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. Then:*

1. *If $P \xrightarrow{\tau} P'$ then $(\!|\, P \,|\!)^*_\Gamma \Longrightarrow H$ with $(\!|\, \mathcal{I} \,|\!) \models H \approx_A (\!|\, P' \,|\!)^*_\Gamma$.*
2. *If $(\!|\, \mathcal{I} \,|\!) \models H \approx_A (\!|\, P \,|\!)^*_\Gamma$ and $H \xrightarrow{\tau} K$, then either $(\!|\, \mathcal{I} \,|\!) \models H \approx_A K$ or there exists P' such that $P \xrightarrow{\tau} P'$ and $(\!|\, \mathcal{I} \,|\!) \models K \approx_A (\!|\, P' \,|\!)^*_\Gamma$.*

Proof. (Sketch). The first item (i.e., the "preservation" direction) follows routinely. For the second item, the first case occurs when the move from H is an administrative step, while the second corresponds to the case when H is finally prompt to commit on a synchronization reduction that reflects a source-level synchronization. □

The proof of soundness requires a further preliminary lemma:

Lemma 2 (Barb Correspondence). *Let $\mathcal{I} \triangleright P$ a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. Then there exists $\mathcal{I}'$, $h \in dom(\mathcal{I}')$ and $C[\cdot]$ such that*

1. *If $\mathcal{I} \triangleright P\!\downarrow_a$ then $(\!|\, \mathcal{I} \,|\!), \mathcal{I}' \triangleright C[(\!|\, P \,|\!)^*_\Gamma]\!\Downarrow_h$*
2. *If $(\!|\, \mathcal{I} \,|\!), \mathcal{I}' \triangleright C[(\!|\, P \,|\!)^*_\Gamma]\!\Downarrow_h$ then $\mathcal{I} \triangleright P\!\downarrow_a$.*

Proof. Choose $\mathcal{I}' = h : \mathsf{rw}\langle \mathbb{T}_\top \rangle$ and $C[-] = \textsc{Link}_\Gamma\ (a, \underline{x})$ in $\overline{x_{\mathsf{r}@A}}\langle h \rangle \mid -$ □

Relying on Lemmas 1 and 2 have:

Theorem 5 (Soundness). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: \mathcal{I}$ be such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$. Then $(\!|\, \mathcal{I} \,|\!) \models (\!|\, P \,|\!)^*_\Gamma \approx (\!|\, Q \,|\!)^*_{\Gamma'}$ implies $\mathcal{I} \models P \approx_@ Q$.*

Proof. Let $\mathcal{R}$ be the type indexed relation defined as follows: $\mathcal{I} \models P\ \mathcal{R}\ Q$ whenever $(\!|\, \mathcal{I} \,|\!) \models (\!|\, P \,|\!)^*_\Gamma \approx_@ (\!|\, Q \,|\!)^*_{\Gamma'}$. Show that $\mathcal{R}$ is barb preserving, reduction closed and contextual. □

In the "only if" direction (*completeness*), the proof follows, more directly, by coinduction. However, the definition of the candidate relation requires some care, as we need to keep track of the states reached by the Proxy components as a result of the interactions with its clients. Let then $\mathsf{E}_{\mathcal{T}}[-]$ be a context representing that state, for an arbitrary list $\mathcal{T}$. We have:

$$\mathsf{E}_{\mathcal{T}}[-] \stackrel{def}{=} \; - \; \mid \prod_{(n,\underline{k}) \in \mathcal{T}} \textsc{Chan}(k) \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}])\, (\,\textsc{Server}(t) \mid \mathtt{TABLE}(t, \mathcal{T}))$$

so that $\mathsf{E}_{\mathcal{T}}[(\!| P |\!)_{\Gamma}] = (\!| P |\!)^{*}_{\Gamma}$ at the initial state in when $\mathcal{T} = [\,]$. Then we have:

Theorem 6 (Completeness). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: \mathcal{I}$ be two type environments such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$. Then $\mathcal{I} \models P \approx_{@} Q$ implies $(\!| \mathcal{I} |\!) \models (\!| P |\!)^{*}_{\Gamma} \approx (\!| Q |\!)^{*}_{\Gamma'}$.*

Proof. (Sketch). Let $\mathcal{R}$ be the type indexed relation such that

$$(\!| \mathcal{I} |\!), a_1 : T_1, \ldots a_n : T_n \models C[\,\mathsf{E}_{\mathcal{T}}[(\!| P |\!)_{\Gamma}]\,] \;\; \mathcal{R} \;\; C[\,\mathsf{E}_{\mathcal{T}}[(\!| Q |\!)_{\Gamma'}]\,]$$

whenever (i) $\mathcal{I} \models P \approx_{@} Q$, (ii) $\Gamma <: \mathcal{I}$ and $\Gamma \vdash P$, (iii) $\Gamma' <: \mathcal{I}$ and $\Gamma' \vdash Q$ (iv) $a_i \notin dom((\!| \mathcal{I} |\!))$, and (v) $C[-]$ is an evaluation context that binds the $\underline{k}$ in $\mathcal{T}$ and such that $(\!| \mathcal{I} |\!), a_1 : T_1, \ldots a_n : T_n \vdash C[-]$. The proof follows by showing that $\mathcal{R}$ satisfies the properties of the administrative equivalence. The most difficult part is the proof that $\mathcal{R}$ is reduction closed, as the processes reached by $C[\,\mathsf{E}_{\mathcal{T}}[(\!| P |\!)_{\Gamma}]\,]$ after an arbitrary number of reductions will, in general, have the form $C'[\,\mathsf{E}_{\mathcal{T}'}[K]\,]$ with K a derivative of $(\!| P' |\!)_{\Gamma}$, for some P', rather than $(\!| P' |\!)_{\Gamma}$ as required by $\mathcal{R}$. We rely on an up-to technique to factor out the administrative steps required to close the relation on two terms with the required format. □

5 Conclusions

We have given a fully abstract encoding of aPi@ into aPi. In its present form, the encoding only applies to the monadic fragment of aPi@, and requires the presence of recursive types in aPi. In fact, the same technique would work for the polyadic calculus, as long as we can count on a finite bound on the maximal arity. In that case, every aPi@ channel may be associated to different tuples of names, one for each possible arity. Similarly, we could do without recursive types in aPi, as in the original formulation of [7, 8] by assuming a finite bound on the number of cascading re-transmission via other names. In fact, the dynamic types of aPi@ allow any channel to communicate its own name: in the general case, this requires (or at least it appears to require) types with arbitrarily deep nesting, viz, recursive types.

The encoding is interesting as it shows how the dynamically typed synchronization of aPi@ may be simulated by a combination of untyped synchronizations on suitably designed channels, and it allows us to identify precisely the subclass of the static types of aPi that correspond to the dynamic types of aPi@. On the one hand, the recursive structure of the static types that emerges from the encoding shows that the dynamic types of aPi@ offer limited access control mechanisms, as they only provide ways to control the use of the top-level capabilities associated with names. On the other hand,

it is precisely because of its limited expressive power, that the dynamic typing system may be used effectively in arbitrary, untyped contexts, as shown by their secure implementation described in [3].

Types and advanced techniques for behavioral observation are emerging as powerful tools for the analysis of distributed computations and open systems. Types have been employed to describe resources and their usage [4–6, 9, 11] and typed equational theories have been studied to characterize the observational properties of processes [2, 7, 8, 13]. In particular, the type systems introduced in [4, 5], for Ambient calculus, and in [6], for KLAIM calculus, guarantee the delivery of resources at the expected type by resorting to type coercion on outputs. As in API@, the soundness of these systems is given by a combination of static and dynamic typing. Thus, as future work we are considering Ambient calculus and KLAIM calculus in order to extend the results we obtained here.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 104–115. ACM press, 2001.
2. M. Bugliesi and M. Giunti. Typed processes in untyped contexts. In *Proc. of the International Symposium on Trustworthy Global Computing (TGC)*, volume 3705 of *LNCS*, pages 19–32. Springer, 2005.
3. M. Bugliesi and M. Giunti. Secure implementations of typed channel abstractions. In *Proc. of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 251–262. ACM press, 2007.
4. M. Coppo, F. Cozzi, M. Dezani-Ciancaglini, E. Giovannetti, and R. Pugliese. A mobility calculus with local and dependent types. In *Processes, Terms and Cycles*, volume 3838 of *LNCS*. Springer, 2005.
5. M. Coppo, M. Dezani-Ciancaglini, E. Giovannetti, and R. Pugliese. Dynamic and local typing for mobile ambients. In *Proc. of International Conference on Theoretical Computer Science (IFIP TCS)*, pages 577–590. Kluwer, 2004.
6. D. Gorla and R. Pugliese. Resource access and mobility control with dynamic privileges acquisition. In *Proc. of International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2719 of *LNCS*, pages 119–132. Springer, 2003.
7. M. Hennessy. *A Distributed Pi-Calculus*. Cambridge University Press, 2007.
8. M. Hennessy and J. Rathke. Typed behavioural equivalences for processes in the presence of subtyping. *Mathematical Structures in Computer Science*, 14(5):651–684, 2004.
9. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173(1):82–120, 2002.
10. U. Nestmann and B C. Pierce. Decoding choice encodings. *Information and Computation*, 163(1):1–59, 2000.
11. R. De Nicola, G. L. Ferrari, R. Pugliese, and B. Venneri. Types for access control. *Theoretical Computer Science*, 240(1):215–254, 2000.
12. B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.
13. B. C. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *Journal of the ACM*, 47(3):531–584, 2000.

A Typing and transition Rules

Table 7 Typing rules for aPi.

Environments

(ST-EMPTY)
$$\frac{}{\vdash \diamond}$$

(ST-EXT)
$$\frac{\Gamma \vdash \diamond \quad \Gamma \downarrow u : T}{\Gamma, u : T \vdash \diamond}$$

Capabilities

(ST-SUB)
$$\frac{\Gamma, u : T, \Gamma' \vdash \diamond \quad T <: T'}{\Gamma, u : T, \Gamma' \vdash u : T'}$$

(ST-MEET)
$$\frac{\Gamma \vdash u : T \quad \Gamma \vdash u : T'}{\Gamma \vdash u : (T \sqcap T')}$$

(ST-TUP)
$$\frac{\Gamma \vdash u_i : T_i \quad (i = 1, \ldots, n)}{\Gamma \vdash (u_1, \ldots, u_n) : (T_1, \ldots, T_n)}$$

Procesees

(ST-STOP)
$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}}$$

(ST-PAR)
$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q}$$

(ST-IN)
$$\frac{\Gamma \vdash u : \mathsf{r}\langle \tilde{T} \rangle \quad \Gamma, \langle \tilde{x} : \tilde{T} \rangle \vdash P}{\Gamma \vdash u(\tilde{x}).P}$$

(ST-REPL)
$$\frac{\Gamma \vdash P}{\Gamma \vdash !P}$$

(ST-NEW)
$$\frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a : T)P}$$

(ST-OUT)
$$\frac{\Gamma \vdash u : \mathsf{w}\langle \tilde{T} \rangle \quad \Gamma \vdash \tilde{v} : \tilde{T}}{\Gamma \vdash \overline{u}\langle \tilde{v} \rangle}$$

(ST-MATCH)
$$\frac{\Gamma \vdash u_1 : T_1,\ u_2 : T_2 \quad \Gamma \vdash Q \quad \Gamma, \langle u_1 : T_1 \rangle, \langle u_2 : T_2 \rangle \vdash \diamond \quad \Gamma, \langle u_1 : T_1 \rangle, \langle u_2 : T_2 \rangle \vdash P}{\Gamma \vdash [u_1 = u_2]P; Q}$$

Table 8 Subtyping Rules for aPi.

$$\frac{}{\Sigma \vdash T <: T} \qquad \frac{\Sigma \vdash T_1 <: T_1' \ \cdots \ T_n' <: T_n'}{\Sigma \vdash (T_1, \ldots, T_n) <: (T_1', \ldots, T_n')} \qquad \frac{}{\Sigma \vdash T <: \top}$$

$$\frac{\Sigma \vdash \tilde{U}_w <: \tilde{T}_w}{\Sigma \vdash \mathsf{w}\langle \tilde{T}_w \rangle <: \mathsf{w}\langle \tilde{U}_w \rangle} \qquad \frac{\Sigma \vdash \tilde{U}_w <: \tilde{T}_w \quad \tilde{T}_w <: \tilde{T}_r \quad \tilde{T}_r <: \tilde{U}_r}{\Sigma \vdash \mathsf{rw}\langle \tilde{T}_r; \tilde{T}_w \rangle <: \mathsf{rw}\langle \tilde{U}_r; \tilde{U}_w \rangle} \qquad \frac{\Sigma \vdash \tilde{T}_r <: \tilde{U}_r}{\Sigma \vdash \mathsf{r}\langle \tilde{T}_r \rangle <: \mathsf{r}\langle \tilde{U}_r \rangle}$$

$$\frac{\Sigma \vdash \tilde{T}_w <: \tilde{T}_r}{\Sigma \vdash \mathsf{rw}\langle \tilde{T}_r; \tilde{T}_w \rangle <: \mathsf{w}\langle \tilde{T}_w \rangle} \qquad \frac{\Sigma \vdash \tilde{T}_w <: \tilde{T}_r}{\Sigma \vdash \mathsf{rw}\langle \tilde{T}_r; \tilde{T}_w \rangle <: \mathsf{r}\langle \tilde{T}_r \rangle}$$

$$\frac{\Sigma, \mu X.T_1 <: T_2 \vdash T_1\{\mu X.T_1/X\} <: T_2}{\Sigma \vdash T_1 <: T_2} \qquad \frac{\Sigma, T_1 <: \mu X.T_2 \vdash T_1 <: T_2\{\mu X.T_2/X\}}{\Sigma \vdash T_1 <: T_2}$$

Judgments are enriched by a subtyping environment of the form $\Sigma = \{S <: S', \ldots\}$.
We say that $S <: T$ if $\emptyset \vdash S <: T$

Table 9 Operational Semantics for aPi.

(ST-IN)
$$\frac{}{a(\tilde{x}).P \xrightarrow{a(\tilde{v})} P\{\tilde{v}/\tilde{x}\}}$$

(ST-OUT)
$$\frac{}{\overline{a}\langle\tilde{v}\rangle \xrightarrow{\overline{a}\langle\tilde{v}\rangle} \mathbf{0}}$$

(ST-OPEN)
$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}\rangle} P' \quad b \in \{\tilde{v}\} \setminus \{a, \tilde{c}\}}{(\nu b:B)P \xrightarrow{(b:B\cdot\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}\rangle} P'}$$

(ST-CTXT)
$$\frac{P \xrightarrow{\mu} P' \quad bn(\mu) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\mu} P' \mid Q}$$

(ST-COMM)
$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}\rangle} P' \quad Q \xrightarrow{a(\tilde{v})} Q' \quad \tilde{c} \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')}$$

(ST-COMM)
$$\frac{P \xrightarrow{a(\tilde{v})} P' \quad Q \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}\rangle} Q' \quad \tilde{c} \cap fn(P) = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')}$$

(ST-RES)
$$\frac{P \xrightarrow{\mu} P' \quad b \notin n(\mu)}{(\nu b:T)P \xrightarrow{\mu} (\nu b:T)P'}$$

(ST-REPL)
$$\frac{P \xrightarrow{\mu} P'}{!P \xrightarrow{\mu} P' \mid !P}$$

(ST-EQ)
$$\frac{P \xrightarrow{\mu} P'}{[u=u]P; Q \xrightarrow{\mu} P'}$$

(ST-NEQ)
$$\frac{Q \xrightarrow{\mu} Q' \quad u \neq v}{[u=v]P; Q \xrightarrow{\mu} Q'}$$

Table 10 Typing rules for aPi@.

Environments

(DYN-EMPTY)
$$\frac{}{\vdash \diamond}$$

(DYN-EXT)
$$\frac{\Gamma \vdash \diamond \quad \Gamma \downarrow u : A}{\Gamma, u : A \vdash \diamond}$$

Capabilities

(DYN-SUB)
$$\frac{\Gamma, u : A, \Gamma' \vdash \diamond \quad A <: A'}{\Gamma, u : A, \Gamma' \vdash u : A'}$$

(DYN-MEET)
$$\frac{\Gamma \vdash u : A \quad \Gamma \vdash u : A'}{\Gamma \vdash u : (A \sqcap A')}$$

(DYN-TUP)
$$\frac{\Gamma \vdash u_i : A_i \quad (i = 1, \ldots, n)}{\Gamma \vdash (u_1, \ldots, u_n) : (A_1, \ldots, A_n)}$$

Procesees

(DYN-STOP)
$$\frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathbf{0}}$$

(DYN-PAR)
$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q}$$

(DYN-IN)
$$\frac{\Gamma \vdash u : \mathsf{r} \quad \Gamma, \langle\tilde{x} : \tilde{A}\rangle \vdash P}{\Gamma \vdash u(\tilde{x}@\tilde{A}).P}$$

(DYN-REPL)
$$\frac{\Gamma \vdash P}{\Gamma \vdash !P}$$

(DYN-NEW)
$$\frac{\Gamma, a : A \vdash P}{\Gamma \vdash (\nu a:A)P}$$

(DYN-OUT)
$$\frac{\Gamma \vdash u : \mathsf{w} \quad \Gamma \vdash \tilde{v} : \tilde{B}}{\Gamma \vdash \overline{u}\langle\tilde{v}@\tilde{B}\rangle}$$

(DYN-MATCH)
$$\frac{\Gamma \vdash u_1 : A_1,\ u_2 : A_2 \quad \Gamma \vdash Q \quad \Gamma, \langle u_1 : A_1\rangle, \langle u_2 : A_2\rangle \vdash \diamond \quad \Gamma, \langle u_1 : A_1\rangle, \langle u_2 : A_2\rangle \vdash P}{\Gamma \vdash [u_1 = u_2]P; Q}$$

Table 11 Subtyping Rules for aPi@.

$$\frac{}{A <: A} \qquad \frac{}{\mathsf{rw} <: \mathsf{r}} \qquad \frac{}{\mathsf{rw} <: \mathsf{w}} \qquad \frac{}{A <: \top} \qquad \frac{A_1 <: A'_1 \ \cdots \ A'_n <: A'_n}{(A_1, \ldots, A_n) <: (A'_1, \ldots, A'_n)}$$

Table 12 Operational Semantics for aPi@.

(DYN-IN)
$$\frac{}{a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A})} P\{\tilde{v}/\tilde{x}\}}$$

(DYN-OUT)
$$\frac{}{\overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}}$$

(DYN-OPEN)
$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{A}@\tilde{v}\rangle} P' \quad b \in \{\tilde{v}\} \setminus \{a, \tilde{c}\}}{(\nu b:B)P \xrightarrow{(b:B\cdot\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'}$$

(ST-CTXT)
$$\frac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$$

(ST-COMM)
$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P' \quad Q \xrightarrow{a(\tilde{v}@\tilde{B})} Q' \quad \tilde{A} <: \tilde{B} \quad \tilde{c} \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')}$$

(ST-COMM)
$$\frac{P \xrightarrow{a(\tilde{v}@\tilde{A})} P' \quad Q \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} Q' \quad \tilde{B} <: \tilde{A} \quad \tilde{c} \cap fn(P) = \emptyset}{P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')}$$

(ST-RES)
$$\frac{P \xrightarrow{\alpha} P' \quad b \notin n(\alpha)}{(\nu b:T)P \xrightarrow{\alpha} (\nu b:T)P'}$$

(ST-REPL)
$$\frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}$$

(ST-EQ)
$$\frac{P \xrightarrow{\alpha} P'}{[u=u]P; Q \xrightarrow{\alpha} P'}$$

(ST-NEQ)
$$\frac{Q \xrightarrow{\alpha} Q' \quad u \neq v}{[u=v]P; Q \xrightarrow{\alpha} Q'}$$

B Typing Preservation

The aim of this section is to prove that the encodings of Sections 3 and 4 preserve aPi@ typing and subtyping. To do that, we need some auxiliary results. Firs of all we specify the type T as expected: $\mathsf{T} \stackrel{def}{=} (\top, \top, \top, \top)$. Then a direct consequence of the definition is that the subtyping lattice of aPi@ is preserved by the recursive types of Table 5, as stated by the following lemma.

Lemma 3 (Subtyping Preservation). $\mathbb{T}_{\mathsf{rw}} <: \{\mathbb{T}_{\mathsf{r}}, \mathbb{T}_{\mathsf{w}}\} <: \mathbb{T}_{\top}$.

Proof. Apply the definition. Note in particular that $\mathsf{w}\langle\mathsf{w}\langle T\rangle\rangle$ is covariant in T. □

Corollary 1. *If* $\Gamma <: \Gamma'$ *in* aPi@, *then* $\{\!|\, \Gamma \,|\!\} <: \{\!|\, \Gamma' \,|\!\}$ *in* aPi.

Table 5 introduces also the server types $\boldsymbol{S}_{\mathsf{r}@A}$ and $\boldsymbol{S}_{\mathsf{w}@s}$, which are still based on the tuples $\mathbb{R}$ and $\mathbb{W}$ but provide both read and write capabilities on $n_{\mathsf{r}@A}$ and $n_{\mathsf{w}@A}$. A first consequence of the definition is listed in the following lemma.

Lemma 4. $\boldsymbol{S}_{r@A} <: \boldsymbol{T}_{r@A}$ *and* $\boldsymbol{S}_{w@A} <: \boldsymbol{T}_{w@A}$ *for every type* A *in* aPi@.

Server types provide an auxiliary encoding for type environments, which is useful for type preservation. The auxiliary encoding is based on the type $\mathbb{S} \stackrel{def}{=} (\mathbb{R}_{\mathbb{S}}, \mathbb{W}_{\mathbb{S}})$, where

$$\mathbb{R}_{\mathbb{S}} \stackrel{def}{=} (\boldsymbol{S}_{\mathsf{r}@\mathsf{rw}}, \boldsymbol{S}_{\mathsf{r}@\mathsf{r}}, \boldsymbol{S}_{\mathsf{r}@\mathsf{w}}, \boldsymbol{S}_{\mathsf{r}@\top}) \qquad \mathbb{W}_{\mathbb{S}} \stackrel{def}{=} (\boldsymbol{S}_{\mathsf{w}@\mathsf{rw}}, \boldsymbol{S}_{\mathsf{w}@\mathsf{r}}, \boldsymbol{S}_{\mathsf{w}@\mathsf{w}}, \boldsymbol{S}_{\mathsf{w}@\top}).$$

Clearly $\mathbb{R}_{\mathbb{S}} <: \mathbb{R}$ and $\mathbb{W}_{\mathbb{S}} <: \mathbb{W}$. Then $\mathbb{S} <: \mathbb{T}_{\mathsf{rw}}$ and thus by Lemma 3:

$$\mathbb{S} <: T \text{ for every } T \in \{\mathbb{T}_{\mathsf{rw}}, \mathbb{T}_{\mathsf{r}}, \mathbb{T}_{\mathsf{w}}, \mathbb{T}_{\top}\}. \tag{1}$$

B.1 Encoding $\{\!|\, - \,|\!\}_\Gamma$

Here we prove the type preservation for the encoding $\{\!|\, - \,|\!\}_\Gamma$ defined in Table 4. We need the auxiliary encoding $\{\!|\, \,|\!\}_{\mathbb{S}}$ defined as

$$\begin{aligned} \{\!|\, \emptyset \,|\!\}_{\mathbb{S}} &\stackrel{def}{=} \mathtt{t} : \top, \mathtt{f} : \top \\ \{\!|\, \Gamma, v : A \,|\!\}_{\mathbb{S}} &\stackrel{def}{=} \{\!|\, \Gamma \,|\!\}, \underline{v} : \mathbb{S} \end{aligned}$$

The observation (1) implies the following fact.

Fact 7 *If* Γ *is a typing environment in* aPi@, *then* $\{\!|\, \Gamma \,|\!\}_{\mathbb{S}} <: \{\!|\, \Gamma \,|\!\}$.

As we anticipated, $\{\!|\, \Gamma \,|\!\}_{\mathbb{S}}$ is a good typing environment to type-check channel managers. This will be clear from Corollary 2.

Lemma 5. *If* Γ *is a typing environment in* aPi@, *then* $\{\!|\, \Gamma \,|\!\}, \langle \underline{a} : \mathbb{S} \rangle \vdash \textsc{Chan}(a)$ *in* aPi *for every* $a \in dom(\Gamma)$.

Proof. Let $a \in dom(\Gamma)$ and consider $\textsc{Chan}(a) \stackrel{def}{=} \prod_A\, !\, u_{\mathsf{r}@A}(h).\textsc{Choose}(u, A, h)$. In order to prove that $\{\!|\, \Gamma \,|\!\}, \langle \underline{a} : \mathbb{S} \rangle \vdash \textsc{Chan}(a)$, we show that

$$\{\!|\, \Gamma \,|\!\}, \langle \underline{a} : \mathbb{S} \rangle \vdash u_{\mathsf{r}@A}(h).\textsc{Choose}(u, A, h)$$

for every type A in aPi@. The proof follows a common pattern for every type.

For instance, let $A = \mathsf{w}$ and check that $\{\!|\, \Gamma \,|\!\}, \langle \underline{a} : \mathbb{S} \rangle \vdash u_{\mathsf{r}@\mathsf{w}}(h).\textsc{Choose}(u, \mathsf{w}, h)$. Essentially this amounts to derive

1. $\{\!|\, \Gamma \,|\!\}, h : \mathsf{w}\langle \mathsf{T}, \mathbb{W} \rangle, l : \mathsf{rw}\langle \top \rangle, \langle \underline{a} : \mathbb{S} \rangle \vdash \textsc{Read}_l \left\langle u_{\mathsf{w}@\mathsf{w}}(\underline{z}).\overline{h}\langle \underline{z} \rangle \right\rangle$,
2. $\{\!|\, \Gamma \,|\!\}, h : \mathsf{w}\langle \mathsf{T}, \mathbb{W} \rangle, l : \mathsf{rw}\langle \top \rangle, \langle \underline{a} : \mathbb{S} \rangle \vdash \textsc{Read}_l \left\langle u_{\mathsf{w}@\mathsf{rw}}(\underline{z}).\overline{h}\langle \underline{z} \rangle \right\rangle$.

Consider item 1. Let $\Delta_1 = \{\!|\Gamma|\!\}_{\mathbb{S}}, h : \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle, l : \mathsf{rw}\langle \top\rangle, (z_{\mathbb{R}}, z_{\mathbb{W}}) : (\mathsf{T}, \mathbb{W}), \langle \underline{a} : \mathbb{S}\rangle$. Since $\Delta_1 \vdash u_{\mathsf{w@w}} : \mathsf{rw}\langle \mathsf{T}, \mathbb{W}\rangle <: \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle$ and $\Delta_1 \vdash h : \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle$, the rules of Table 7 derive $\Delta_1 \vdash \textsc{Test}_l \left\langle u_{\mathsf{w@w}}(\underline{z}).\overline{h}\langle \underline{z}\rangle \right\rangle$. Moreover $\Delta_1 \vdash u_{\mathsf{w@w}} : \mathsf{rw}\langle \mathsf{T}, \mathbb{W}\rangle <: \mathsf{r}\langle \mathsf{T}, \mathbb{W}\rangle$, hence item 1 follows by (ST-INP). Item 2 is analogous. Let $\Delta_2 = \{\!|\Gamma|\!\}_{\mathbb{S}}, h : \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle, l : \mathsf{rw}\langle \top\rangle, (z_{\mathbb{R}}, z_{\mathbb{W}}) : (\mathbb{R}, \mathbb{W}), \langle \underline{a} : \mathbb{S}\rangle$. Since $\Delta_2 \vdash u_{\mathsf{w@rw}} : \mathsf{rw}\langle \mathbb{R}, \mathbb{W}\rangle <: \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle$ and $\Delta_2 \vdash h : \mathsf{w}\langle \mathsf{T}, \mathbb{W}\rangle <: \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle$, the rules of Table 7 derive $\Delta_2 \vdash \textsc{Test}_l \left\langle u_{\mathsf{w@rw}}(\underline{z}).\overline{h}\langle \underline{z}\rangle \right\rangle$. Moreover $\Delta_2 \vdash u_{\mathsf{w@rw}} : \mathsf{rw}\langle \mathbb{R}, \mathbb{W}\rangle <: \mathsf{r}\langle \mathbb{R}, \mathbb{W}\rangle$, hence item 2 follows by (ST-INP). Now, items 1 and 2 imply $\{\!|\Gamma|\!\}, h : \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle, \langle \underline{a} : \mathbb{S}\rangle \vdash \textsc{Choose}(u, \mathsf{w}, h)$ by (ST-REPL), (ST-PAR) and (ST-NEW). Then, since $\{\!|\Gamma|\!\}, h : \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle, \langle \underline{a} : \mathbb{S}\rangle \vdash u_{\mathsf{r@w}} : \mathsf{rw}\langle \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle\rangle <: \mathsf{r}\langle \mathsf{w}\langle \mathbb{R}, \mathbb{W}\rangle\rangle$, an application of the rule (ST-INP) concludes that $\{\!|\Gamma|\!\}, \langle \underline{a} : \mathbb{S}\rangle \vdash u_{\mathsf{r@w}}(h).\textsc{Choose}(u, \mathsf{w}, h)$.

As already said, the typing judgement $\{\!|\Gamma|\!\}, \langle \underline{a} : \mathbb{S}\rangle \vdash u_{\mathsf{r}@A}(h).\textsc{Choose}(u, A, h)$ can be derived in a similar way also when A is rw, r, or $\top$. Thus $\{\!|\Gamma|\!\}, \langle \underline{a} : \mathbb{S}\rangle \vdash \textsc{Chan}(a)$ follows by (ST-REPL) and (ST-PAR). □

Corollary 2. *For every Γ in* APi@ *it holds* $\{\!|\Gamma|\!\}_{\mathbb{S}} \vdash \textsc{Manager}_\Gamma$ *in* APi.

On the other hand, client types are enough to type the encoding of APi@ processes..

Lemma 6. *If $\Gamma \vdash P$ in* APi@, *then $\{\!|\Gamma|\!\} \vdash \{\!|P|\!\}$ in* APi.

Proof. First check that $\Gamma \vdash \diamond$ implies $\{\!|\Gamma|\!\} \vdash \diamond$. Then, thanks to Lemma 3, show that $\Gamma \vdash v : A$ implies $\{\!|\Gamma|\!\} \vdash \underline{v} : \mathbb{T}_A$. Finally prove the lemma by induction on the derivation of $\Gamma \vdash P$ in APi@. The interesting cases are the inductive steps for (DYN-INP), (DYN-OUT) and (DYN-NEW).

When (DYN-IN) is the last rule of the type derivation then

$$\frac{\Gamma \vdash u : \mathsf{r} \quad \Gamma, \langle x : A\rangle \vdash P}{\Gamma \vdash u(x@A).P}$$

Assume for instance that $A = \mathsf{w}$. The induction hypotheses say that (i) $\{\!|\Gamma|\!\} \vdash \underline{u} : \mathbb{T}_{\mathsf{r}}$, hence $\{\!|\Gamma|\!\} \vdash u_{\mathsf{r@w}} : \mathsf{w}\langle \mathsf{w}\langle \mathbb{T}_{\mathsf{w}}\rangle\rangle$; and (ii) $\{\!|\Gamma|\!\}, \langle \{\!| x : \mathsf{w} |\!\}\rangle \vdash \{\!|P|\!\}$, namely $\{\!|\Gamma|\!\}, \langle \underline{x} : \mathbb{T}_{\mathsf{w}}\rangle \vdash \{\!|P|\!\}$. Now the typing rules for APi (in Table 7), (ST-IN) and (ST-IN) in particular, allow the derivation of the typing judgment $\{\!|\Gamma|\!\} \vdash (\nu h : \mathsf{rw}\langle \mathbb{T}_{\mathsf{w}}\rangle)\left(\overline{u_{\mathsf{r@w}}}\langle h\rangle \,|\, h(\underline{x}).\{\!|P|\!\}\right)$, that is $\{\!|\Gamma|\!\} \vdash \{\!|P|\!\}$. The cases $A = \mathsf{rw}, \mathsf{r}, \top$ are similar.

When (DYN-OUT) is the last rule of the type derivation

$$\frac{\Gamma \vdash u : \mathsf{w} \quad \Gamma \vdash v : A}{\Gamma \vdash \overline{u}\langle v@A\rangle}$$

then (i) $\{\!|\Gamma|\!\} \vdash \underline{u} : \mathbb{T}_A$, hence $\{\!|\Gamma|\!\} \vdash u_{\mathsf{w}@A} : \mathsf{w}\langle \mathbb{T}_A\rangle$; and (ii) $\{\!|\Gamma|\!\} \vdash (v_{\mathbb{R}}, v_{\mathbb{W}}) : \mathbb{T}_A$. Hence $\{\!|\Gamma|\!\} \vdash \overline{u_{\mathsf{w}@A}}\langle \underline{v}\rangle$, by (DYN-OUT), and so $\{\!|\Gamma|\!\} \vdash \{\!|\overline{u}\langle v@A\rangle|\!\}$.

When (DYN-NEW) is the last rule of the type derivation then

$$\frac{\Gamma, a : A \vdash P}{\Gamma \vdash (\nu a : A)P}$$

The induction hypothesis says that $\{\!|\Gamma, a : A|\!\} \vdash \{\!|P|\!\}$ in APi, that is $\{\!|\Gamma|\!\}, \underline{a} : \boldsymbol{T}_A \vdash \{\!|P|\!\}$, hence $\{\!|\Gamma|\!\}, \underline{a} : \mathbb{S} \vdash \{\!|P|\!\}$. Moreover $\{\!|\Gamma|\!\}, \underline{a} : \mathbb{S} \vdash \textsc{Chan}(a)$ by Lemma 5, then $\{\!|\Gamma|\!\}, \underline{a} : \mathbb{S} \vdash \{\!|P|\!\} \,|\, \textsc{Chan}(a)$ by (ST-PAR). Finally $\{\!|\Gamma|\!\} \vdash \{\!|(\nu a : A)P|\!\}$ by (ST-NEW). □

Theorem 1 (Typing and subtyping preservation) *For all types A, B in* aPi@*, $A <: B$ implies $\mathbb{T}_A <: \mathbb{T}_B$ in* aPi*. Furthermore, whenever $\Gamma \vdash P$ in* aPi@*, then there exists $\Gamma' <: \{\!|\Gamma|\!\}$ such that $\Gamma' \vdash \{\!|P|\!\}_\Gamma$ in* aPi*.*

Proof. The first part of the theorem is given by Lemma 3. For the second part, let $\Gamma \vdash P$ in aPi@. Lemma 6 says that $\{\!|\Gamma|\!\} \vdash \{\!|P|\!\}$ in aPi, hence $\{\!|\Gamma|\!\}_{\mathbb{S}} \vdash \{\!|P|\!\}$ by Fact 7. Moreover $\{\!|\Gamma|\!\}_{\mathbb{S}} \vdash \textsc{Manager}_\Gamma$ by Corollary 2. Thus $\{\!|\Gamma|\!\}_{\mathbb{S}} \vdash \{\!|P|\!\}_\Gamma$ by (ST-PAR). The thesis of the theorem is obtained with $\Gamma' = \{\!|\Gamma|\!\}_{\mathbb{S}}$. □

Finally, the encoding is well defined as it actually maps configurations into configurations.

Corollary 3. *If $\mathcal{I} \rhd P$ is a configuration in* aPi@*, with $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$, then $\{\!|\mathcal{I}|\!\} \rhd \{\!|P|\!\}_\Gamma$ is a configuration in* aPi*.*

Proof. Let $\mathcal{I} \rhd P$ be a configuration, with $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$ in aPi@. Theorem 1 says that there exists $\Gamma' <: \{\!|\Gamma|\!\}$ such that $\Gamma' \vdash \{\!|P|\!\}_\Gamma$ in aPi. Lemma 3 says that $\{\!|\Gamma|\!\} <: \{\!|\mathcal{I}|\!\}$, hence $\Gamma' <: \{\!|\mathcal{I}|\!\}$. Thus $\{\!|\mathcal{I}|\!\} \rhd \{\!|P|\!\}_\Gamma$ is a configuration in aPi. □

B.2 Encoding $(\!|-|\!)^*_\Gamma$

Here we prove the type preservation for the encoding $(\!|-|\!)^*_\Gamma$ defined in Table 6. Essentially, the proof is a consequence of the type preservation of the encoding $\{\!|-|\!\}_\Gamma$ given in Section B.1. As explained in Section 4, the encoding of typing contexts $(\!|-|\!)$ extends $\{\!|-|\!\}$ with the four types that describe the behavior of the proxy channels p_A, with $A = \mathsf{rw}, \mathsf{r}, \mathsf{w}, \top$:

$$(\!|\Gamma|\!) \stackrel{def}{=} \{\!|\Gamma|\!\} \cup \{p_A : \mathsf{w}\langle\mathsf{w}\langle\mathbb{T}_A\rangle, \mathbb{T}_A\rangle\}_{A\in\{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}}.$$

Again, this definition preserve subtyping.

Lemma 7. *If $\Gamma <: \Gamma'$ in* aPi@*, then $(\!|\Gamma|\!) <: (\!|\Gamma'|\!)$ in* aPi*.*

Moreover, the definition ensures that the encoding of aPi@ processes as clients (see Table 6) is well typed.

Lemma 8. *If $\Gamma \vdash P$ in* aPi@*, then $(\!|\Gamma|\!) \vdash (\!|P|\!)^*_\Gamma$ in* aPi*.*

Proof. First observe that the definition $\textsc{Link}_\Gamma\ (a, \underline{x})$ IN $P \stackrel{def}{=} (\nu h : \mathsf{rw}\langle\mathbb{T}_A\rangle)(\overline{p_A}\langle h, a_{\mathsf{r}@\mathsf{r}}\rangle \mid h(\underline{x}).P)$, with $A = \Gamma(a)$, ensures that the type of the tuple received on the private channel h is exactly the one which $\{\!|\Gamma|\!\}$ assigns to $\underline{a}$. Then follow the proof of Lemma 6. □

Corollary 4. *If $\Gamma_2 <: \Gamma_1$ and $\Gamma_1 \vdash P$ in* aPi@*, then $(\!|\Gamma_2|\!) \vdash (\!|P|\!)^*_{\Gamma_1}$ in* aPi*.*

As done in Section B.1, in order to show that the proxy manager is well typed, we need to resort to auxiliary types. Contrarily to Section B.1, we do not have to provide full capabilities on the tuples $\underline{m}, \underline{n}$ that correspond to the channels m, n in aPi@. It is sufficient to provide the server with read capability on the channels p_A used by clients for their requests. The management of the private association table will then guarantee the server read capabilities on the proxy channels. The auxiliary encoding is defined as follows:

$$(\!|\Gamma|\!)_{\mathsf{rw}} \stackrel{def}{=} \{\!|\Gamma|\!\} \cup \{p_A : \mathsf{rw}\langle\mathsf{w}\langle\mathbb{T}_A\rangle, \mathbb{T}_A\rangle\}_{A\in\{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}}.$$

Since we omit the implementation of the association table, we assume the two ad-hoc typing rules (T-Lookup) and (T-Table) for LOOKUP and TABLE that are GIVEN in Section 4. With these assumptions it is straight to prove the following lemma.

Lemma 9. *For every Γ in* aPi@ *it holds* $(\![\Gamma]\!)_{rw} \vdash$ Proxy *in* aPi.

Theorem 3 (Typing and subtyping preservation) *For all types A, B in* aPi@*, $A <: B$ implies $\mathbb{T}_A <: \mathbb{T}_B$ in* aPi*. Furthermore, whenever $\Gamma \vdash P$ in* aPi@*, then there exists $\Gamma' <: (\![\Gamma]\!)$ such that $\Gamma' \vdash (\![P]\!)^*_\Gamma$ in* aPi.

Proof. The first part of the theorem is given by Lemma 3. For the second part, take $\Gamma' \stackrel{def}{=} (\![\Gamma]\!)_{\mathsf{rw}}$. Clearly $\Gamma' <: (\![\Gamma]\!)$. Moreover $\Gamma' \vdash (\![P]\!)^*_\Gamma$ by Lemma 8 and $\Gamma' \vdash$ Proxy by Lemma 9. Then conclude $\Gamma' \vdash (\![P]\!)^*_\Gamma$ by (st-par). □

Again, the encoding is well defined.

Corollary 5. *If $\mathcal{I} \rhd P$ is a configuration in* aPi@*, with $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$, then $(\![\mathcal{I}]\!) \rhd (\![P]\!)^*_\Gamma$ is a configuration in* aPi.

Proof. Let $\mathcal{I} \rhd P$ be a configuration, with $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$ in aPi@. Theorem 3 says that there exists $\Gamma' <: (\![\Gamma]\!)$ such that $\Gamma' \vdash (\![P]\!)^*_\Gamma$ in aPi. Lemma 3 says that $(\![\Gamma]\!) <: (\![\mathcal{I}]\!)$, hence $\Gamma' <: (\![\mathcal{I}]\!)$. Thus $(\![\mathcal{I}]\!) \rhd (\![P]\!)^*_\Gamma$ is a configuration in aPi. □

C Soundness and Completeness

The aim of this section is to present the main results that prove soundness and completeness of the encoding defined in Section 4. We do not consider the encoding given in Section 3 as it can be proved to be sound just by following the reasoning that leads to Theorem 5. In particular Theorem 2 is proved just by considering the administrative reductions which appear in Table 14 and which do not involve the proxy server.

The guideline of the whole line of argument is the proof of full abstraction in [3][3] The proof is based on a new type of behavioral equivalence: the administrative equivalence. This equivalence is defined on particular kind of reductions, called *administrative*. Essentially, an administrative reduction represents the synchronization between the client process and the proxy server that happens in the encodings.

First we need some auxiliary notations. We let the functions ρ, σ range over renaming functions. With $P\rho$ we mean the name substitution applied to the names of the process P. Then we can introduce the idea of derivative processes.

Definition 5 (Derivative process). *We say that an* aPi *process H is a $(\![\ ]\!)_\Gamma$-derivative if there exist an* aPi *process K, an* aPi@ *process P such that $\Gamma \vdash P$ and two renaming functions ρ and σ such that $H = K\sigma$ and $(\![P]\!)^*_\Gamma \Longrightarrow K\rho$.*

[3] The proofs of the results presented in [3] are fully reported in: "M. Giunti. *Secure Implementations of Typed Channel Abstractions*. PhD Thesis TD-2007-1, Informatics Department, University Ca Foscari of Venice, 2007".

Just for the formal Definition 6, in order to identify the administrative reductions, we resort to a two sorted set of names: 'standard' noted by $m, n \ldots$ and 'signed' noted by $\mathfrak{h}, \mathfrak{m}, \ldots$ In the encoding, signed names are those sent on the channels $u_{\mathfrak{r}@A}$, i.e., the channels used for the private communication between a process willing for an input on (the encoding of) channel u and the channel manager. Formally, the encoding of an input becomes:

$$(\![\, u(y@A).P \,]\!)_\Gamma \stackrel{def}{=} \textsc{Link}_\Gamma\ (u, \underline{x})\ \textsc{in}\ (\nu \mathfrak{h} : \mathsf{rw}\langle \mathbb{T}_A \rangle) \left(\overline{x_{\mathfrak{r}@A}}\langle \mathfrak{h} \rangle \,|\, \mathfrak{h}(\underline{y}).(\![\, P \,]\!)_{\Gamma, y:A} \right)$$

We call administrative every internal communication of the encodings that *is not* a synchronization along a signed channel.

Definition 6 (Administrative Reduction). *We say that $P \xrightarrow{\tau} P'$ is an $\mathcal{I}$-administrative reduction, noted $P \xrightarrow{A_\mathcal{I}} P'$, when $\mathcal{I} \rhd P\downarrow_a$ iff $\mathcal{I} \rhd P'\downarrow_a$ and $P = C[H]$, $P' = C[H']$ with H, H' $(\![\ ]\!)_\Gamma$-derivatives such that $H \xrightarrow{\tau} H'$ is a synchronization on a standard channel.*

Thanks to the characterization provided by Lemma 15, we will use a single sorted set of names and abandon the idea of signed channels.

We define $\stackrel{A_\mathcal{I}}{\Longrightarrow}$ to be the reflexive and transitive closure of $\xrightarrow{A_\mathcal{I}}$. Moreover we mark as $\xrightarrow{\tau_\mathcal{I}}$ a τ-transition which is not $\mathcal{I}$-administrative.

Definition 7 (Administrative Equivalence). *The administrative equivalence $\approx_A$ is the largest symmetric, contextual and term indexed relation $\mathcal{R}$ such that $\mathcal{I} \models P \mathcal{R} Q$ implies*

1. *if $\mathcal{I} \rhd P\downarrow_a$, then $\mathcal{I} \rhd Q\downarrow_a$*
2. *if $P \xrightarrow{A_\mathcal{I}} P'$, then $Q \stackrel{A_\mathcal{I}}{\Longrightarrow} Q'$ for some Q' such that $\mathcal{I} \models P' \mathcal{R} Q'$*
3. *if $P \xrightarrow{\tau_\mathcal{I}} P'$, then $Q \stackrel{A_\mathcal{I}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{I}} \stackrel{A_\mathcal{I}}{\Longrightarrow} Q'$ for some Q' such that $\mathcal{I} \models P' \mathcal{R} Q'$*

It is easy to see that the above definition implies the following property, useful to prove that the administrative equivalence is indeed an equivalence relation.

Lemma 10. *Given $\mathcal{I} \models P \approx_A Q$ then:*

- *if $P \stackrel{A_\mathcal{I}}{\Longrightarrow} P'$ then $Q \stackrel{A_\mathcal{I}}{\Longrightarrow} Q'$ for some Q' such that $\mathcal{I} \models P' \approx_A Q'$*
- *if $P \stackrel{A_\mathcal{I}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{I}} \stackrel{A_\mathcal{I}}{\Longrightarrow} P'$ then $Q \stackrel{A_\mathcal{I}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{I}} \stackrel{A_\mathcal{I}}{\Longrightarrow} Q'$ for some Q' such that $\mathcal{I} \models P' \approx_A Q'$*

Lemma 11. *The relation $\approx_A$ is an equivalence over* aPi *processes.*

Proof. Reflexivity can be checked by showing that the identity relation *Id* satisfies the properties required by Definition 7. Symmetry holds by definition. Finally, Lemma 11 is useful to show that $\approx_A \approx_A$ satisfies Definition 7 and thus to prove transitivity. □

An important observation is that $\approx_A$ is finer than $\approx$.

Lemma 12. *If $\mathcal{I} \models P \approx_A Q$ then $\mathcal{I} \models P \approx Q$.*

Proof. We show that the relation $\approx_A$ satisfies the properties of Definition 4. The contextuality of $\approx_A$ is a consequence of Definition 7. To see that $\approx_A$ preserves barbs, assume that $\mathcal{I} \models P \approx_A Q$ and $\mathcal{I} \rhd P \downarrow_a$, then $\mathcal{I} \rhd Q \downarrow_a$ by Definition 7, and thus $\mathcal{I} \rhd Q \Downarrow_a$. To see that $\approx_A$ is reduction closed, assume that $\mathcal{I} \models P \approx_A Q$ and $P \xrightarrow{\tau} P'$. In case $P \xrightarrow{A_{\mathcal{I}}} P'$, then there exists Q' such that $\mathcal{I} \models P' \approx_A Q'$ and $Q \overset{A_{\mathcal{I}}}{\Longrightarrow} Q'$, hence $Q \Longrightarrow Q'$. In case $P \xrightarrow{\tau_{\mathcal{I}}} P'$, then there exists Q' such that $\mathcal{I} \models P' \approx_A Q'$ and $Q \overset{A_{\mathcal{I}}}{\Longrightarrow} \xrightarrow{\tau_{\mathcal{I}}} \overset{A_{\mathcal{I}}}{\Longrightarrow} Q'$, hence $Q \Longrightarrow Q'$. □

The next definition will be useful to discard administrative reductions when proving that a relation is included in the administrative equivalence. We will use this technique in the proof of Theorem 6 (completeness).

Definition 8 (Behavioral Equivalence up to Administrative Equivalence). *A type-indexed relation $\mathcal{R}$ is a behavioral equivalence up to administrative equivalence if it is symmetric, contextual up to $\approx_A$ and such that $\mathcal{I} \models P \mathcal{R} Q$ implies*

1. *if $\mathcal{I} \rhd P \downarrow_a$, then $\mathcal{I} \rhd P \Downarrow_a$*
2. *if $P \overset{A_{\mathcal{I}}}{\Longrightarrow} P'$, then $Q \overset{A_{\mathcal{I}}}{\Longrightarrow} Q'$ for some Q' such that $\mathcal{I} \models P' \approx_A \mathcal{R} \approx_A Q'$*
3. *if $P \overset{A_{\mathcal{I}}}{\Longrightarrow} \xrightarrow{\tau_{\mathcal{I}}} \overset{A_{\mathcal{I}}}{\Longrightarrow} P'$, then $Q \Longrightarrow Q'$ for some Q' such that $\mathcal{I} \models P' \approx_A \mathcal{R} \approx_A Q'$*

Next lemma says that a behavioral equivalence up to administrative equivalence is finer than the behavioral equivalence $\approx$. For its proof we refer to [3].

Lemma 13. *If $\mathcal{R}$ is a behavioral equivalence up to administrative equivalence, then $\mathcal{I} \models P \mathcal{R} Q$ then $\mathcal{I} \models P \approx_A Q$.*

The usual congruence of pi-calculus, denoted by $\equiv$, will simplify the proofs in the following of the paper. Basically, it identifies processes with the same internal structure, and it is defined as the smallest congruence that is closed under α-conversion and that satisfies the following axioms:

$$\begin{array}{rrr}
P|Q \equiv Q|P & P|(\nu a:T)Q \equiv (\nu a:T)(P|Q) & !(P|Q) \equiv !P|!Q \\
P|\mathbf{0} \equiv P & (\nu a:T)\mathbf{0} \equiv \mathbf{0} & !P \equiv !P|P \\
[a=a]P;Q \equiv P & [a=b]P;Q \equiv P \quad (\text{if } a \neq b) & !!P \equiv !P \\
\multicolumn{3}{c}{(\nu a_1:T_1)(\nu a_2:T_2)P \equiv (\nu a_2:T_2)(\nu a_1:T_1)P}
\end{array}$$

It is easy to see that congruence is sound with respect the operational semantics given in Table 9. In fact, if $P \equiv Q$ and $P \xrightarrow{\alpha} P'$ then there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \equiv Q'$. As expected, congruence is finer that $\approx_A$, as stated by the following lemma, whose proof is fairly standard.

Lemma 14. *If $P \equiv Q$ then $\mathcal{I} \models P \approx_A Q$ for every $\mathcal{I}$.*

It is straightforward to see how the congruence relation provides a characterization of administrative reductions. The details are outlined in tables 13 and 14. In particular, Table 13 depicts the administrative reductions due to the interaction with the proxy server, and Table 14 depicts the administrative reductions due to the interaction with the channel manager. We can use a *single sorted* set of names, as all the reduction that are not listed in the tables are not administrative. Thus we will not refer to standard or signed names.

Table 13 Administrative Reductions – Interaction Client/Proxy Server.

Case 1:
$H \equiv \textsc{Link}_{\Gamma}\ (a, \underline{x})\ \textsc{in}\ P \mid \textsc{Server}(t)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\,(\, h(\underline{x}).P \mid \textsc{Reply}_t(a_{\mathsf{r}@\mathsf{r}}, h)\,) \mid \textsc{Server}(t)$

Case 2(a):
$H \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid \textsc{Reply}_t(a, h)\,\big) \mid \mathtt{TABLE}(t, \mathcal{T})$
$H' \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)$
$\quad \big(\, h(\underline{x}).P \mid (\nu r{:}\mathsf{rw}\langle \top, \mathbb{S} \rangle)\big(\, \overline{r}\langle \mathtt{t}, \underline{k} \rangle \mid r(x, \underline{y})[x = \mathtt{t}]\overline{h}\langle \underline{y} \rangle; (\overline{h}\langle \underline{y} \rangle \mid \textsc{Chan}(y))\,\big)\,\big) \mid \mathtt{TABLE}(t, \mathcal{T})$
Case 2(b):
$H \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid \textsc{Reply}_t(a, h)\,\big) \mid \mathtt{TABLE}(t, \mathcal{T})$
$H' \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\,(\nu \underline{k}{:}\mathbb{S})$
$\quad \big(\, h(\underline{x}).P \mid (\nu r{:}\mathsf{rw}\langle \top, \mathbb{S} \rangle)\big(\, \overline{r}\langle \mathtt{f}, \underline{k} \rangle \mid r(x, \underline{y})[x = \mathtt{t}]\overline{h}\langle \underline{y} \rangle; (\overline{h}\langle \underline{y} \rangle \mid \textsc{Chan}(y))\,\big)\ \mid \mathtt{TABLE}(t, (a, \underline{k}) :: \mathcal{T})\,\big)$

Case 3(a):
$H \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid (\nu r{:}\mathsf{rw}\langle \top, \mathbb{S} \rangle)\big(\, \overline{r}\langle \mathtt{t}, \underline{k} \rangle \mid r(x, \underline{y})[x = \mathtt{t}]\overline{h}\langle \underline{y} \rangle; (\overline{h}\langle \underline{y} \rangle \mid \textsc{Chan}(y))\,\big)\,\big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid [\mathtt{t} = \mathtt{t}]\overline{h}\langle \underline{k} \rangle; (\overline{h}\langle \underline{k} \rangle \mid \textsc{Chan}(k))\,\big)$
Case 3(b):
$H \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid (\nu r{:}\mathsf{rw}\langle \top, \mathbb{S} \rangle)\big(\, \overline{r}\langle \mathtt{f}, \underline{k} \rangle \mid r(x, \underline{y})[x = \mathtt{t}]\overline{h}\langle \underline{y} \rangle; (\overline{h}\langle \underline{y} \rangle \mid \textsc{Chan}(y))\,\big)\,\big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid [\mathtt{f} = \mathtt{t}]\overline{h}\langle \underline{k} \rangle; (\overline{h}\langle \underline{k} \rangle \mid \textsc{Chan}(k))\,\big)$

Case 4(a):
$H \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid [\mathtt{t} = \mathtt{t}]\overline{h}\langle \underline{k} \rangle; (\overline{h}\langle \underline{k} \rangle \mid \textsc{Chan}(k))\,\big)$
$H' \equiv P\left\{\underline{k}/\underline{x}\right\}$
Case 4(b):
$H \equiv (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A \rangle)\ \big(\, h(\underline{x}).P \mid [\mathtt{f} = \mathtt{t}]\overline{h}\langle \underline{k} \rangle; (\overline{h}\langle \underline{k} \rangle \mid \textsc{Chan}(k))\,\big)$
$H' \equiv P\left\{\underline{k}/\underline{x}\right\} \mid \textsc{Chan}(k)$

With $\textsc{Reply}_t(a, h) \stackrel{def}{=} (\nu r{:}\mathsf{rw}\langle \top, \mathbb{S} \rangle)\ \big(\, \mathtt{LOOKUP}(a, t, r) \mid r(x, \underline{y})[x = \mathtt{t}]\overline{h}\langle \underline{y} \rangle; (\,\overline{h}\langle \underline{y} \rangle \mid \textsc{Chan}(y)\,)\,\big)$

Lemma 15 (Administrative Reduction – Characterization). *The reduction $P \xrightarrow{\tau} P'$ is $\mathcal{I}$-administrative when $\mathcal{I} \rhd P\downarrow_a$ iff $\mathcal{I} \rhd P'\downarrow_a$ and $P = C[H]$, $P' = C[H']$ with H, H' satisfying one of the eight cases depicted in tables 13 and 14.*

C.1 Closure Properties of Administrative Equivalence

The key property of administrative reductions is that they are closed under administrative equivalence, as stated by Lemma 17. The notion of non-overlapping transitions and Lemma 16 are useful to prove Lemma 17.

Definition 9 (Non-overlapping reductions). *The reductions $P \xrightarrow{\tau} Q$ and $P \xrightarrow{\tau} R$ are non-overlapping if there are $C[]$, P' and P'' such that $P \equiv C[P' \mid P'']$, $Q \equiv C[Q' \mid P'']$ with $P' \xrightarrow{\tau} Q'$, and $R \equiv C[P' \mid R'']$ with $P'' \xrightarrow{\tau} R''$.*

Table 14 Administrative Reductions – Interaction Client/Channel Manager.

Case 5:
$H \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(\,\overline{u_{\mathsf{r}@A}}\langle h\rangle \mid h(\underline{x}).Q\,) \mid !\,u_{\mathsf{r}@A}(h).\textsc{Choose}(\underline{u}, A, h)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(\,h(\underline{x}).Q \mid \textsc{Choose}(\underline{u}, A, h)\,) \mid !\,u_{\mathsf{r}@A}(h).\textsc{Choose}(\underline{u}, A, h)$

Case 6:
$H \equiv \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle \mid (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\Big(\,h(\underline{x}).Q \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\Big(\,\bar{l}\langle\mathtt{t}\rangle \mid R_A\langle h, l\rangle\,\Big)\Big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(\,h(\underline{x}).Q \mid Q' \mid$
$\quad \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\,\Big(\,\bar{l}\langle n\rangle \mid R_A\langle h, l\rangle \mid l(z).[z{=}\mathtt{t}](\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle) \oplus (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle); (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\,\Big)\Big)$

Case 7:
$H \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(\,h(\underline{x}).Q \mid$
$\quad \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\,\Big(\,\bar{l}\langle n\rangle \mid Q' \mid R_A\langle h, l\rangle \mid l(z).[z{=}\mathtt{t}](\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle) \oplus (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle); (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\,\Big)\Big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(\,h(\underline{x}).Q \mid$
$\quad \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\,\Big(\,Q' \mid R_A\langle h, l\rangle \mid [n{=}\mathtt{t}](\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle) \oplus (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle); (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\,\Big)\Big)$

Case 8(a):
$H \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(\,h(\underline{x}).Q \mid$
$\quad \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\,\Big(\,Q' \mid R_A\langle h, l\rangle \mid [\mathtt{t}{=}\mathtt{t}](\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle) \oplus (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle); (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\,\Big)\Big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(h(\underline{x}).Q \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\Big(\,Q' \mid R_A\langle h, l\rangle \mid \bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle \mid (\nu i{:}\mathsf{rw}\langle\,\rangle)i(\,).(\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\Big)\Big)$

Case 8(b):
$H \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(\,h(\underline{x}).Q \mid$
$\quad \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\,\Big(\,Q' \mid R_A\langle h, l\rangle \mid [\mathtt{t}{=}\mathtt{t}](\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle) \oplus (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle); (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\,\Big)\Big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(h(\underline{x}).Q \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\Big(\,Q' \mid R_A\langle h, l\rangle \mid \bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle \mid (\nu i{:}\mathsf{rw}\langle\,\rangle)i(\,).(\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle)\Big)\Big)$

Case 8(c):
$H \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(\,h(\underline{x}).Q \mid$
$\quad \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\,\Big(\,Q' \mid R_A\langle h, l\rangle \mid [\mathtt{f}{=}\mathtt{t}](\bar{l}\langle\mathtt{f}\rangle \mid \overline{h}\langle\underline{v}\rangle) \oplus (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle); (\bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle)\,\Big)\Big)$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,\Big(h(\underline{x}).Q \mid (\nu l{:}\mathsf{rw}\langle\top\rangle)\Big(\,Q' \mid R_A\langle h, l\rangle \mid \bar{l}\langle\mathtt{t}\rangle \mid \overline{u_{\mathsf{w}@B}}\langle\underline{v}\rangle\Big)\Big)$

With $R_A\langle h, l\rangle \stackrel{def}{=} \prod_{C<:A} !\,u_{\mathsf{w}@C}(\underline{z}).\textsc{Test}_l\langle u_{\mathsf{w}@C}(\underline{z}).\overline{h}\langle\underline{z}\rangle\rangle$.

Lemma 16. *If $P \xrightarrow{\tau} Q$ and $P \xrightarrow{\tau} R$ are two non-overlapping transitions, then there exists a process H such that $Q \xrightarrow{\tau} H$ and $R \xrightarrow{\tau} H$.*

Proof. The definition says that there are $C[]$, P' and P'' such that $P \equiv C[P' \mid P'']$, $Q \equiv C[Q' \mid P'']$ with $P' \xrightarrow{\tau} Q'$, and $R \equiv C[P' \mid R'']$ with $P'' \xrightarrow{\tau} R''$. Then let H be $C[Q' \mid R'']$ and check that $Q \xrightarrow{\tau} H$ and $R \xrightarrow{\tau} H$. □

Lemma 17. *If $P \xrightarrow{A_{\mathcal{I}}} P'$ then $\mathcal{I} \models P \approx_A P'$.*

Proof. We take the type indexed relation $\mathcal{R} \stackrel{def}{=} (\mathcal{R}_1 \cup \mathcal{R}_2 \cup \equiv)$, where $\mathcal{I} \models C[P]\mathcal{R}_1 C[Q]$ and $\mathcal{I} \models C[Q]\mathcal{R}_2 C[P]$ whenever $P \xrightarrow{A_{\mathcal{I}}} Q$ and $C[-]$ is an evaluating context defined as

$C[-] \stackrel{def}{=} (\nu \tilde{b}{:}\tilde{T})(\,- \mid R)$, with $\mathcal{I} \vdash R$. Then we show that $\mathcal{R}$ is symmetric, barb preserving, reduction closed and contextual.

Symmetry. It holds by definition

Barb Preservation. Assume that $\mathcal{I} \models C[P]\ \mathcal{R}_1\ C[Q]$. If $\mathcal{I} \rhd C[]\downarrow_a$ then $\mathcal{I} \rhd C[P]\downarrow_a$ if and only if $\mathcal{I} \rhd C[Q]\downarrow_a$. Otherwise $\mathcal{I} \rhd P\downarrow_a$ if and only if $\mathcal{I} \rhd Q\downarrow_a$ by Definition 6, and again $\mathcal{I} \rhd C[P]\downarrow_a$ if and only if $\mathcal{I} \rhd C[Q]\downarrow_a$. The case for $\mathcal{R}_2$ is analogous and the case for $\equiv$ follows from Lemma 14.

Reduction Closure. The relation $\mathcal{R}_1$ is the only significative one. Then we assume that $\mathcal{I} \models C[P]\ \mathcal{R}_1\ C[Q]$, meaning that $P \xrightarrow{A_I} Q$. We need to prove that *(a)* if $C[P] \xrightarrow{A_I} H$ then $C[Q] \stackrel{A_I}{\Longrightarrow} K$ with $H\ \mathcal{R}\ K$, and that *(b)* if $C[P] \xrightarrow{\tau_I} H$ then $C[Q] \stackrel{A_I}{\Longrightarrow}\xrightarrow{\tau_I}\stackrel{A_I}{\Longrightarrow} K$ with $H\ \mathcal{R}\ K$. The two cases are essentially treated in a similar way, and they rely on Lemma 16 as administrative reductions are in general non-overlapping. In the following we discuss two significative cases

In case $C[P] \xrightarrow{A_I} H$ and this reduction has been generated by item 5 in Table 14. If the administrative reduction $P \xrightarrow{A_I} Q$ has been inferred from the items 1–4 of Table 13 or 6–8 of Table 14, then the reductions $C[P] \xrightarrow{A_I} C[Q]$ and $C[P] \xrightarrow{A_I} H$ are non overlapping, hence we conclude that there exists K such that $C[Q] \xrightarrow{A_I}\equiv K$ and $H \xrightarrow{A_I}\equiv K$ by Lemma 16. On the other hand, if the administrative reduction $P \xrightarrow{A_I} Q$ has been inferred from the item 5 of Table 14, then it must be a synchronization on a channel $u_{\mathsf{r}@A}$. Also $C[P] \xrightarrow{A_I} H$ is a synchronization on a channel $u'_{\mathsf{r}@A'}$. Now, if $u \neq u'$ and $A \neq A'$, then the two transitions are non-overlapping. We conclude as above. If $u = u'$ and $A = A'$ then we have two possibilities. On the one hand, if $H = C[Q]$ then we conclude that $C[Q] \stackrel{A_I}{\Longrightarrow} H$. On the other hand, if there are two synchronizations on two different outputs on $u_{\mathsf{r}@A}$ it must be the case that

$$\begin{aligned} C[P] \equiv C'[\ & (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(\overline{u_{\mathsf{r}@A}}\langle h\rangle \mid h(\underline{x}).Q_1) \\ & (\nu k{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(\overline{u_{\mathsf{r}@A}}\langle k\rangle \mid k(\underline{x}).Q_2) \mid !\, u_{\mathsf{r}@A}(h).\textsc{Choose}(\underline{u}, A, h)\ ]. \end{aligned}$$

Then it is easy to see that $C[P] \xrightarrow{A_I} H \xrightarrow{A_I} K_1$ and $C[Q] \xrightarrow{A_I} K_2$ with

$$\begin{aligned} C[Q] \equiv C'[\ & (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(h(\underline{x}).Q_1 \mid \textsc{Choose}(\underline{u}, A, h)) \mid \\ & (\nu k{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(\overline{u_{\mathsf{r}@A}}\langle k\rangle \mid k(\underline{x}).Q_2) \mid !\, u_{\mathsf{r}@A}(h).\textsc{Choose}(\underline{u}, A, h)\ ] \\ H \equiv C'[\ & (\nu k{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(k(\underline{x}).Q_2 \mid \textsc{Choose}(\underline{u}, A, k)) \mid \\ & (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(\overline{u_{\mathsf{r}@A}}\langle h\rangle \mid h(\underline{x}).Q_1) \mid !\, u_{\mathsf{r}@A}(h).\textsc{Choose}(\underline{u}, A, h)\ ] \\ K_1 \equiv K_2 \equiv C'[\ & (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(h(\underline{x}).Q_1 \mid \textsc{Choose}(\underline{u}, A, h)) \mid \\ & (\nu k{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)(k(\underline{x}).Q_2 \mid \textsc{Choose}(\underline{u}, A, k)) \mid !\, u_{\mathsf{r}@A}(h).\textsc{Choose}(\underline{u}, A, h)\ ]. \end{aligned}$$

In case $C[P] \xrightarrow{A_I} H$ according to item 6 in Table 14. If the administrative reduction $P \xrightarrow{A_I} Q$ has been inferred from the items 1–4 of Table 13 and 5, 7, 8 of Table 14 then we conclude as done above thanks to Lemma 16. On the other hand if the administrative reduction $P \xrightarrow{A_I} Q$ has been inferred from the item 6 of Table 14, then it must be a synchronization on a channel $u_{\mathsf{w}@A}$. Also $C[P] \xrightarrow{A_I} H$ is a synchronization on a channel

$u'_{\mathsf{w}@A}$. Again, if $H = C[Q]$ then we conclude that $C[Q] \overset{A_I}{\Longrightarrow} H$. Moreover if they are non-overlapping synchronizations then we conclude as above. The only overlapping case is when $u = u'$ and

$$\begin{aligned} C[P] \equiv C'[\ & (\nu\tilde{n}{:}\tilde{T})(\overline{u_{\mathsf{w}@B_1}}\langle \underline{v}_1\rangle \mid Q'_1) \mid \\ & (\nu\tilde{n}{:}\tilde{T})(\overline{u_{\mathsf{w}@B_2}}\langle \underline{v}_2\rangle \mid Q'_2) \mid \\ & (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A\rangle)(h(\underline{x}).Q \mid (\nu l{:}\mathsf{rw}\langle \top\rangle)(\bar{l}\langle \mathtt{t}\rangle \mid R_A\langle h, l\rangle))\ ] \\ C[Q] \equiv C'[\ & (\nu\tilde{n}{:}\tilde{T})(\overline{u_{\mathsf{w}@B_1}}\langle \underline{v}_1\rangle \mid Q'_1) \mid \\ & (\nu\tilde{n}{:}\tilde{T})(\overline{u_{\mathsf{w}@B_2}}\langle \underline{v}_2\rangle \mid Q'_2) \mid \\ & (\nu h{:}\mathsf{rw}\langle \mathbb{T}_A\rangle)(h(\underline{x}).Q \mid (\nu l{:}\mathsf{rw}\langle \top\rangle)(\bar{l}\langle \mathtt{t}\rangle \mid R_A\langle h, l\rangle))\ ] \end{aligned}$$

Now we can conclude as done above for the previous case thanks to the presence of the replicated input channel $\prod_{C<:A} !\, u_{\mathsf{w}@C}(\underline{z}).\textsc{Test}_l\langle u_{\mathsf{w}@C}(\underline{z}).\overline{h}\langle \underline{z}\rangle\rangle$ in $R_A\langle h, l\rangle$.
Contextuality. It is a direct consequence of the definition of $\mathcal{R}_1$ and $\mathcal{R}_2$, as we have introduced the evaluating context $C[-]$. □

The result of Lemma 17 can be extended to an arbitrary number of administrative reductions.

Corollary 6. *If $P \overset{A_I}{\Longrightarrow} Q$ then $\mathcal{I} \models P \approx_A Q$.*

Proof. Extend Lemma 17 by induction on the number of reductions $P \overset{A_I}{\Longrightarrow} Q$. □

C.2 Operational Correspondence

Soundness relies on a operational correspondence between the source process P and its encoding $(\![P]\!)^*_\Gamma$. The 'preservation' direction (Lemma 18) is fairly standard, whereas the 'reflection' (Lemma 21) direction needs more care as the encoding is not 'prompt' (see [10]): in fact, the encoding takes several steps to commit a synchronization of the source process. In this section we show that those steps have the following properties: each reduction leading to a commit *(i)* does not preclude any other reduction and *(ii)* it is administrative, thus not visible to the context.

Lemma 18 (Operational Preservation). *Let $\mathcal{I} \rhd P$ a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. If $P \xrightarrow{\tau} P'$ then $(\![P]\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow}\xrightarrow{\tau_\mathcal{J}} K$, with $\mathcal{J} = (\![\mathcal{I}]\!)$ and $\mathcal{J} \models K \approx_A (\![P']\!)^*_\Gamma$.*

Proof. By induction on the derivation of $P \xrightarrow{\tau} P'$ in aPi@, we see that there exists a reduction $(\![P]\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow} H$ which follows the steps in (the same order of) Table 13 and Table 14 with $H \xrightarrow{\tau_\mathcal{J}} K \equiv (\nu\underline{k}{:}\mathbb{S})\left((\![P']\!)_\Gamma \mid (\nu t{:}\mathtt{TBL}[\top, \mathbb{S}])(\textsc{Server}(t) \mid \mathtt{TABLE}(t, [(n, \underline{k})]))\right)$ and $\mathcal{J} \models K \approx_A (\![P']\!)^*_\Gamma$ as the association table does not influence the behavior of the encoded process. □

For the reverse direction of the operational correspondence we introduce an auxiliary notion. Consider the reductions $P \overset{A_I}{\Longrightarrow} P' \xrightarrow{\tau_I} Q$, we say that the sequence $\overset{A_I}{\Longrightarrow}$ is *canonical*, and we denote it as $P \overset{\lceil A \rceil_I}{\Longrightarrow} P'$ if it includes just the administrative steps required to enable the non administrative synchronization in P'. More formally,

Definition 10 (Canonical sequence). *The sequence $P \overset{\lceil A \rceil_I}{\Longrightarrow} P'$ of administrative actions is canonical when $P' \xrightarrow{\tau_I} Q$ with a synchronization between two input/output actions on a channel n and $P \overset{\lceil A \rceil_I}{\Longrightarrow} P' \xrightarrow{\tau_I} Q$ has the minimum length among all the reductions $P \overset{A_I}{\Longrightarrow} P'' \xrightarrow{\tau_I} Q$ in which $P'' \xrightarrow{\tau_I} Q$ is a synchronization between the same input/output actions on n.*

We can then prove a preparatory lemma.

Lemma 19. *Let $\mathcal{I} \rhd P$ a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. Define $\mathcal{J} = (\!|\, \mathcal{I} \,|\!)$. If $(\!|\, P \,|\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow} H \xrightarrow{\tau_\mathcal{J}} K$, then there exists H' such that $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H' \xrightarrow{\tau_\mathcal{J}} K'$ with $K \approx_A K'$.*

Proof. Let $(\!|\, P \,|\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow} H \xrightarrow{\tau_\mathcal{J}} K$ then in $\overset{A_\mathcal{J}}{\Longrightarrow}$ there is a synchronization on a channel p_A that start the communication protocol that leads (exactly) to the desired committing action $\xrightarrow{\tau_I}$. Then take this synchronization on p_A as the first action and build the canonical reduction by following the steps 1–7, 8(a) of Tables 13 and 14. Thus obtain $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H' \xrightarrow{\tau_\mathcal{J}} K'$. Now, all the administrative steps in $(\!|\, P \,|\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow} H$ that are not in $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H'$ can be performed starting by K'. Hence $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H' \xrightarrow{\tau_\mathcal{J}} K' \overset{A_I}{\Longrightarrow} K$, and so $\mathcal{I} \vDash K' \approx_A K$ thanks to Lemma 17. □

This results can be extended to general aPi processes, just by considering the derivative processes.

Corollary 7. *Let $\mathcal{J} \rhd H$ a configuration in* aPi. *If $H \overset{A_\mathcal{J}}{\Longrightarrow} Y \xrightarrow{\tau_\mathcal{J}} K$, then there exists H' such that $H \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H' \xrightarrow{\tau_\mathcal{J}} K'$ with $K \approx_A K'$.*

Thanks to the canonical administrative reductions, we have a correspondence between the actions of the encoding and its source process.

Lemma 20. *Let $\mathcal{I} \rhd P$ a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. Define $\mathcal{J} = (\!|\, \mathcal{I} \,|\!)$. If $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H \xrightarrow{\tau_\mathcal{J}} K$ then there exists P' such that $P \xrightarrow{\tau} P'$ and $\mathcal{J} \vDash K \approx_A (\!|\, P' \,|\!)^*_\Gamma$.*

Proof. Assume that $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H \xrightarrow{\tau_\mathcal{J}} K$. Thanks to canonical reduction $(\!|\, P \,|\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} H$, we can infer the shape of P and in turn the shape of H. In fact, this reduction mimics a single synchronization in aPi (see Lemma 19). Thus $P \equiv C[\overline{a}\langle b@B\rangle \mid a(x@B').Q]$ with $B <: B'$. Consequently H is ready to reduce, via a non administrative reduction, to $(\nu \underline{k} : \mathbb{S})\left((\!|\, C[Q\{b/a\}] \,|\!)_\Gamma \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}])(\textsc{Server}(t) \mid \mathtt{TABLE}(t, [(n, \underline{k})]))\right) \equiv K$. Now let P' be $C[Q\{b/a\}]$. Then conclude that $P \xrightarrow{\tau} P'$ and $\mathcal{J} \vDash K \approx_A (\!|\, P' \,|\!)^*_\Gamma$ as the association table does not influence the behavior of the encoded process. □

Lemma 21 (Operational Reflection). *Let $\mathcal{I} \rhd P$ a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. Define $\mathcal{J} = (\!|\, \mathcal{I} \,|\!)$. Assume that $\mathcal{J} \vDash H \approx_A (\!|\, P \,|\!)^*_\Gamma$ and $H \xrightarrow{\tau} K$. Then either $\mathcal{J} \vDash H \approx_A K$ or there exists P' such that $P \xrightarrow{\tau} P'$ and $\mathcal{J} \vDash K \approx_A (\!|\, P' \,|\!)^*_\Gamma$.*

Proof. Assume that $\mathcal{J} \models H \approx_A (\!| P |\!)^*_\Gamma$ and $H \xrightarrow{\tau} K$. If $H \xrightarrow{A_I} K$ then we apply Lemma 17 and we conclude that $\mathcal{J} \models H \approx_A K$. Otherwise, it is the case that $H \xrightarrow{\tau_I} K$ and, since $\mathcal{J} \models H \approx_A (\!| P |\!)^*_\Gamma$, there exist Y, Z such that $(\!| P |\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{J}} Y \overset{A_J}{\Longrightarrow} Z$ and $\mathcal{J} \models Z \approx_A K$. Then, by Lemma 19, there exists X such that $(\!| P |\!)^*_\Gamma \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{J}} X$ with $\mathcal{J} \models X \approx_A Y$ and hence $\mathcal{J} \models X \approx_A Z$ by Corollary 6. Now, By Lemma 20, there exists P' such that $P \xrightarrow{\tau} P'$ and $\mathcal{J} \models (\!| P' |\!)^*_\Gamma \approx_A X$. Then $\mathcal{J} \models (\!| P' |\!)^*_\Gamma \approx_A X \approx_A Z \approx_A K$ and conclude the thesis by transitivity. □

Lemma 1 (Operational Correspondence). *Let $\mathcal{I} \triangleright P$ be a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. Define $\mathcal{J} = (\!| \mathcal{I} |\!)$. The following hold:*

1. *If $P \xrightarrow{\tau} P'$ then $(\!| P |\!)^*_\Gamma \overset{A_\mathcal{J}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{J}} H$ with $\mathcal{J} \models H \approx_A (\!| P' |\!)^*_\Gamma$.*
2. *If $(\!| \mathcal{I} |\!) \models H \approx_A (\!| P |\!)^*_\Gamma$ and $H \xrightarrow{\tau} K$, then either $(\!| \mathcal{I} |\!) \models H \approx_A K$ or there exists P' such that $P \xrightarrow{\tau} P'$ and $(\!| \mathcal{I} |\!) \models K \approx_A (\!| P' |\!)^*_\Gamma$.*

Proof. The operational preservation in item 1 is given by Lemma 18. The operational reflection in item 2 is given by Lemma 21. □

Item 2 of the previous lemma can be extended to an unbounded number of reductions, as said by the following corollary.

Corollary 8. *Let $\mathcal{I} \triangleright P$ a configuration in* aPi@ *and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$. If $(\!| P |\!)^*_\Gamma \Longrightarrow H$ then there exists P' such that $P \Longrightarrow P'$ and $(\!| \mathcal{I} |\!) \models H \approx_A (\!| P' |\!)^*_\Gamma$.*

Theorem 5 (Soundness). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: \mathcal{I}$ be such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$. Then $(\!| \mathcal{I} |\!) \models (\!| P |\!)^*_\Gamma \approx (\!| Q |\!)^*_{\Gamma'}$ implies $\mathcal{I} \models P \approx_@ Q$.*

Proof. We consider the type indexed relation $\mathcal{R}$ defined as

$$\mathcal{I} \models P \,\mathcal{R}\, Q \quad \text{if and only if} \quad (\!| I |\!) \models (\!| P |\!)^*_{\Gamma_1} \approx (\!| Q |\!)^*_{\Gamma_2} \tag{2}$$

for some $\Gamma_1, \Gamma_2 <: \mathcal{I}$ such that $\Gamma_1 \vdash P$ and $\Gamma_2 \vdash Q$. Then we show that $\mathcal{R}$ is symmetric, reduction closed, barb preserving and contextual.

Symmetry. It holds by definition

Reduction closure. We use the operational correspondence of Lemma 1 and its Corollary 8. Assume that $\mathcal{I} \models P \,\mathcal{R}\, Q$ and let $P \xrightarrow{\tau} Q$. Define $\mathcal{J} \overset{def}{=} (\!| \mathcal{I} |\!)$. Item 1 of Lemma 1 says that there exists H in aPi such that $(\!| P |\!)^*_{\Gamma_1} \overset{A_\mathcal{J}}{\Longrightarrow} \xrightarrow{\tau_\mathcal{J}} H$ and $\mathcal{J} \models H \approx_A (\!| P' |\!)^*_{\Gamma_1}$. As $\approx_A$ is finer than $\approx$ we have $\mathcal{J} \models H \approx (\!| P' |\!)^*_{\Gamma_1}$. Due to the definition in (2) we deduce that there exists K such that $\mathcal{J} \models K \approx H$ and $(\!| Q |\!)^*_{\Gamma_2} \Longrightarrow K$. Now, Corollary 8 says that there exists Q' such that $Q \Longrightarrow Q'$ and $\mathcal{J} \models (\!| Q' |\!)^*_{\Gamma_2} \approx_A K$. Again, as $\approx_A \subseteq \approx$, it holds $\mathcal{J} \models (\!| Q' |\!)^*_{\Gamma_2} \approx K$ and thus $\mathcal{J} \models (\!| Q' |\!)^*_{\Gamma_2} \approx (\!| P' |\!)^*_{\Gamma_2}$ by transitivity.

Then we found Q' such that $Q \Longrightarrow Q'$ and $(\!| \mathcal{I} |\!) \models (\!| Q' |\!)^*_{\Gamma_2} \approx (\!| P' |\!)^*_{\Gamma_2}$. This means that $\mathcal{I} \models P' \,\mathcal{R}\, Q'$, hence $\mathcal{R}$ is reduction closed.

Barb Preservation. It is obtained directly by observing that for every configuration $\mathcal{I} \triangleright P$ and $\Gamma <: \mathcal{I}$ such that $\Gamma \vdash P$ we have

$$\mathcal{I} \triangleright P{\downarrow_a} \quad \text{if and only if} \quad \begin{array}{l}\text{there exists } A \text{ such that} \\ (\!| \mathcal{I} |\!), h : \mathsf{rw}\langle \mathbb{T}_A \rangle \triangleright \textsc{Link}_\Gamma\ (a, \underline{x}) \text{ in } \overline{x_{\mathsf{r}@A}}\langle h \rangle \mid (\!| P |\!)^*_\Gamma \Downarrow_h\end{array}$$

Contextuality. Essentially this is a consequence of the contextuality of $\approx_A$. We need to prove the three requirements of Definition 2. Consider for instance item 1. Assume that $\mathcal{I}, a : A \models P \mathcal{R} Q$. Let $\Gamma_1 \vdash P$ and $\Gamma_2 \vdash Q$, with $\Gamma_1, \Gamma_2 <: \mathcal{I}, a : A$. We have to prove that $\mathcal{I} \models (\nu a : A)P \mathcal{R} (\nu a : A)Q$. By the definition in (2) we have $(\![\mathcal{I}, a : A]\!) \models (\![P]\!)^*_{\Gamma_1} \approx (\![Q]\!)^*_{\Gamma_2}$ and then $(\![\mathcal{I}]\!) \models (\nu \underline{a} : \mathbb{T}_A)(\![P]\!)^*_{\Gamma_1} \approx (\nu \underline{a} : \mathbb{T}_A)(\![Q]\!)^*_{\Gamma_2}$ by contextuality of $\approx$. Thanks to the axioms of structural congruence and the definition of the encoding we have $(\nu \underline{a} : \mathbb{T}_A)(\![P]\!)^*_{\Gamma_1} \equiv (\![(\nu a : A)P]\!)^*_{\Gamma'_1}$ with $\Gamma'_1 = \Gamma_1 \setminus \{a\}$ and $(\nu \underline{a} : \mathbb{T}_A)(\![Q]\!)^*_{\Gamma_2} \equiv (\![(\nu a : A)Q]\!)^*_{\Gamma'_2}$ with $\Gamma'_1 = \Gamma_1 \setminus \{a\}$. Then we conclude $(\![\mathcal{I}]\!) \models (\![(\nu a : A)P]\!)^*_{\Gamma'_1} \approx (\![(\nu a : A)Q]\!)^*_{\Gamma'_2}$ with $\Gamma'_1, \Gamma'_2 <: \mathcal{I}$, hence $\mathcal{I} \models (\nu a : A)P \mathcal{R} (\nu a : A)Q$. The other items are analogous. □

Theorem 6 (Completeness). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: \mathcal{I}$ be such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$. Then $\mathcal{I} \models P \approx_@ Q$ implies $(\![\mathcal{I}]\!) \models (\![P]\!)^*_\Gamma \approx (\![Q]\!)^*_{\Gamma'}$.*

Proof. We recall that $(\![P]\!)^*_\Gamma$ is $(\![P]\!)_\Gamma \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}])(\textsc{Server}(t) \mid \mathtt{TABLE}(t, \mathcal{T}))$ with the list $\mathcal{T}$ empty. The encoding represents the initial configuration without pre-synchronization between the client $(\![P]\!)_\Gamma$ and the proxy server. Due to the communication protocol, the proxy server will fulfill the requesters from the client by generating new entries in $\mathcal{T}$ along with the respective channel managers. The system will then evolve to configurations of the form $(\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S})(\, Q_1 \mid Q_2)$ where Q_1 is a derivative of the original process $(\![P]\!)_\Gamma$, which can be typed as $(\![\Gamma]\!), \underline{k}_1 : \mathbb{T}_{\mathsf{rw}}, \ldots \underline{k}_p : \mathbb{T}_{\mathsf{rw}} \vdash Q_1$, and Q_2 represents the state of the system server/table:

$$Q_2 = \prod_{i=1\ldots p} \textsc{Chan}(k_i) \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}]) \Big(\textsc{Server}(t) \mid \mathtt{TABLE}(t, (n_1, \underline{k}_1) :: \ldots :: (n_p, \underline{k}_p)) \Big)$$

In order to account this fact, we introduce the process $\mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}$ which represents the state of an arbitrary list $\mathcal{T} = (n_1, \underline{k}_1) :: \ldots :: (n_p, \underline{k}_p)$ and it is defined as

$$\mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p} \stackrel{def}{=} \prod_{i=1\ldots p} \textsc{Chan}(k_i) \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}]) \Big(\textsc{Server}(t) \mid \mathtt{TABLE}(t, (n_1, \underline{k}_1) :: \ldots :: (n_p, \underline{k}_p)) \Big).$$

Observe that $\mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p} \mid (\![P]\!)_\Gamma \equiv (\![P]\!)^*_\Gamma$ when $p = 0$. Thus the candidate relation $\mathcal{R}$ can be defined as:

$$\mathcal{J} \models C\Big[(\![P]\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p} \Big] \;\; \mathcal{R} \;\; C\Big[(\![Q]\!)_{\Gamma'} \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p} \Big]$$

whenever

1. $\mathcal{I} \models P \approx_@ Q$ for some typing context $\mathcal{I}$ in $\textsc{api}@$;
2. $\Gamma, \Gamma' <: \mathcal{I}$ with $\Gamma \vdash P$ and $\Gamma' \vdash Q$;
3. $\mathcal{J} = (\![\mathcal{I}]\!), a_1 : T_1, \ldots a_n : T_n$ with $a_i \notin dom((\![\mathcal{I}]\!))$;
4. $C[-] = (\nu \tilde{b} : \tilde{T})(\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S})(- \mid R)$ with
   (a) $dom(\mathcal{J}) \cap \{\tilde{b}\} = \emptyset$
   (b) $n_1 \ldots n_p \in dom(\mathcal{J}) \cup \{\tilde{b}\}$
   (c) $\mathcal{J}, \tilde{b} : \tilde{T}, \underline{k}_1 : \mathbb{T}_{\mathsf{rw}}, \ldots \underline{k}_p : \mathbb{T}_{\mathsf{rw}} \vdash R$.

If we prove that $\mathcal{R}$ is a behavioral equivalence up to administrative equivalence (see Definition 8), then $\mathcal{R} \subseteq \approx$ by Lemma 13. Hence $\mathcal{I} \models P \approx_@ Q$ implies $(\!|\, \mathcal{I} \,|\!) \models (\!|\, P \,|\!)^*_\Gamma \approx (\!|\, Q \,|\!)^*_{\Gamma'}$ by defining the context $C[-]$ to be empty. So we conclude the completeness of the encoding.

Next we show that $\mathcal{R}$ barb preserving, reduction closed and contextual in the sense of Definition 8. We use the following conventions: $\mathcal{J}$ is the typing context $(\!|\, \mathcal{I} \,|\!), a_1 : T_1, \ldots a_n : T_n$, and $C[-]$ is the evaluating context. We let $H = C[(\!|\, P \,|\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}]$ and $K = C[(\!|\, Q \,|\!)_{\Gamma'} \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}]$.

Barb Preservation. Assume that $H \mathcal{R} K$ and $\mathcal{J} \rhd H \downarrow_a$ then $\mathcal{J} \rhd C[- \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}] \downarrow_a$ as $\mathcal{J} \rhd (\!|\, P \,|\!)_\Gamma$ does not exhibit barbs, thus we conclude that $\mathcal{J} \rhd K \downarrow_a$.

Reduction closure. Assume that $\mathcal{J} \models H \mathcal{R} K$. We check the cases 2 (with $H \overset{A_\mathcal{J}}{\Longrightarrow} H'$) and 3 (with $H \overset{A_\mathcal{J}}{\Longrightarrow} \overset{\tau_\mathcal{J}}{\longrightarrow} \overset{A_\mathcal{J}}{\Longrightarrow} H'$) of Definition 8.

Assume $H \overset{A_\mathcal{J}}{\Longrightarrow} H'$. Then $\mathcal{J} \models H \approx_A H'$ by Corollary 6, hence we conclude $\mathcal{J} \models H' \approx_A H \mathcal{R} K$.

Otherwise, assume $H \overset{A_\mathcal{J}}{\Longrightarrow} Y \overset{\tau_\mathcal{J}}{\longrightarrow} Y' \overset{A_\mathcal{J}}{\Longrightarrow} H'$. Then it is sufficient to find K' such that $K \Longrightarrow K'$ and $\mathcal{J} \models Y' \approx_A \mathcal{R} \approx_A K'$. The required result $\mathcal{J} \models H' \approx_A \mathcal{R} \approx_A K'$ is again a consequence of Corollary 6, as $\approx_A$ is closed under administrative reductions and transitive.

Consider $H \overset{A_\mathcal{J}}{\Longrightarrow} Y \overset{\tau_\mathcal{J}}{\longrightarrow} Y'$. Then Corollary 7 says that there exists X' such that $H \overset{\lceil A \rceil_\mathcal{J}}{\Longrightarrow} X \overset{\tau_\mathcal{J}}{\longrightarrow} X'$ with $\mathcal{J} \models Y' \approx_A X'$. Now we have three possibilities: (i) the reduction is exclusively made by the context $C[-]$, (ii) the reduction is exclusively made by the encoded process $(\!|\, P \,|\!)_\Gamma$ interacting with $\mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}$ or (iii) the reduction is generated by the interaction between the encoded process $(\!|\, P \,|\!)_\Gamma$ and the full context.

(i) In case the reduction is generated by the context; $H \Longrightarrow X'$ with $X' = C'[(\!|\, P \,|\!)_\Gamma]$ and so $K \Longrightarrow K'$ with $K' = C'[(\!|\, Q \,|\!)_{\Gamma'}]$. Hence $X' \mathcal{R} K'$ according to the definition. We conclude that $K \Longrightarrow K'$ with $\mathcal{J} \models Y' \approx_A X' \mathcal{R} K'$.

(ii) In case the reduction is generated by the process $(\!|\, P \,|\!)_\Gamma$ interacting with the proxy server and the channel manager; the reduction corresponds to a synchronization on a single channel n for the original process P. There are two sub-cases depending on n appearing in the association table or not. Here we assume that n is in the association table, the other case is analogous. To ease the notation, given $C[-] = (\nu \tilde{b} : \tilde{T})(\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S})(- \mid R)$ define $C'[-] = (\nu \tilde{b} : \tilde{T})(- \mid R)$, then $C[-] = C'[(\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S})(-)]$. Since n is in the association table, then $X' \equiv C'[Z]$ with

$$(\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S}) ((\!|\, P \,|\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}) \overset{A_\mathcal{J}}{\Longrightarrow} \overset{\tau_\mathcal{J}}{\longrightarrow} Z$$

As for Lemma 20, due to the presence of the last non administrative transition, the original process P must perform a τ action. Thus $P \overset{\tau}{\longrightarrow} P'$ and $\mathcal{J} \models (\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S}) ((\!|\, P' \,|\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}) \approx_A Z$. Since $\mathcal{I} \models P \approx_@ Q$, there exists Q' such that $Q \Longrightarrow Q'$ and $\mathcal{I} \models P' \approx_@ Q'$. As for Lemma 18, we have that $(\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S}) ((\!|\, Q \,|\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}) \Longrightarrow (\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S}) ((\!|\, Q' \,|\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p})$. Now we define K' to be $C'[(\nu \underline{k}_1 :$

$\mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S}) (([\![Q']\!]_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p})]$ and we conclude that $K \Longrightarrow K'$ with $\mathcal{J} \models K' = C[[\![Q']\!]_\Gamma] \mathcal{R} C[[\![P']\!]_\Gamma] \approx_A C'[Z] \equiv X' \approx_A Y'$ as required.

(iii) If the reduction is generated by the interaction between the encoded process $[\![P]\!]_\Gamma$ and the full context, it means that P performs either an input or an output in $\text{A}\pi\text{@}$. In this case, the reasoning is analogous to item (ii), we just need to consider the contextuality of the behavioral equivalence that forces the encoding to mime the action on the $\text{A}\pi$ side, and the typing of the process R that cannot read on server channels.

Contextuality. It follows straightforwardly from the definition of $\mathcal{R}$, thanks to the evaluating contexts $C[-]$. □

Theorem 4 (Full Abstraction). *Let $\Gamma <: \mathcal{I}$ and $\Gamma' <: I$ be two type environments such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$. Then $\mathcal{I} \models P \approx_@ Q$ if and only if $[\![\mathcal{I}]\!] \models [\![P]\!]^*_\Gamma \approx [\![Q]\!]^*_{\Gamma'}$.*

Proof. The 'if' direction is given by Theorem 5. The 'only if' direction is given by Theorem 6. □