A Pi-calculus with Dynamic Typing ⋆

Michele Bugliesi, Marco Giunti, Damiano Macedonio, and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari, Venice {michele,giunti,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.
Here we develop a new typing discipline for the asynchronous pi-calculus, which we call API@. It 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 define a typed equational theory, and we give a co-inductive proof technique useful to prove equivalences among processes.
We study the relationships between our dynamic approach and the static one of the asynchronous pi calculuS, referred as API, which comes with an entirely standard static typing system. On the one hand, we show that API can be encoded in API@ in a sound manner. On the other hand, 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 and motivations

Static typing systems have long been established as an effective device to control the interaction among processes in the pi-calculus and related calculi [12, 13, 18, 19]. 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.

The idea was first introduced in [18], and it is best illustrated by the following example. Consider the two pi-calculus processes:

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

where S represents a print spooler and C is a client. The print spooler serves requests from a private channel s that it communicates to its clients via the public channel d. The client receives s and uses it to print the job j.

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

$$S\,|\,C \quad \stackrel{?}{\cong}^{@} \quad S\,|\,\overline{print}\langle j\rangle \tag{1}$$

Here we take $P \cong^{@} Q$ to mean that P and Q are behaviorally indistinguishable, i.e. they have the same observable behavior when executed in any arbitrary context. The equation (1) is easily disproved by exhibiting a context that interferes with the intended protocol between S and C. A first example is the context $\mathscr{C}_1[-] = -\,|\,d(x).!x(y).\mathbf{0}$, that initially behaves as the client, to receive s, but then it steals the jobs intended

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

for S. A second example is the context $\mathscr{C}_2[-] = -\,|\,(\nu s')\overline{d}\langle s'\rangle$, which may succeed in transmitting to C a dead-ended channel that will never serve the purpose C expected of it.

As shown in [18], hostile contexts such as those described above can be ruled out by resorting to a system of capability types to control the transmission and/or reception of values over channels based on the possession of corresponding type capabilities. In the example, the type system may protect against contexts like $\mathscr{C}_1$ by requiring that clients be only granted write capabilities on the channel s, and by reserving read capabilities on s to the spooler. Similarly, safeguards against attackers like $\mathscr{C}_2$ may be built by demanding that clients only have read access on d.

Both the requirements are expressed formally by the typing assumption $d : \mathsf{r}\langle\mathsf{w}\langle T\rangle;\mathsf{w}\langle T\rangle\rangle$, which grants read-only access on d and write-only access to any name received on d, as desired. We may then refine equation (1) into its typed version:

$$d : \mathsf{r}\langle\mathsf{w}\langle T\rangle\rangle \;\models\; S\,|\,C \;\cong^{@}\; S\,|\,\overline{print}\langle j\rangle. \tag{2}$$

In general, typed equations of the form $\mathrm{I} \models P \cong^{@} Q$ express behavioural equivalences between processes in any context that typechecks in the type environment I. Here I represents the context's view of the processes under observation, given in terms of a set of typing assumptions on the names shared between the processes and the context itself. Incidentally, but importantly, the typing assumptions on the shared names may in general be different – in fact, more accurate – for the processes than they are for the context. To illustrate, in (2), a context is only assumed to have read-capabilities on d, while for the system to typecheck the name d must be known at the lower (hence more accurate) type $\mathsf{rw}\langle\mathsf{w}\langle T\rangle;\mathsf{w}\langle T\rangle\rangle$, so that to allow S to write and C to read. Similarly, for the system $S\,|\,C$ to typecheck, the name s must be known at the type $\mathsf{rw}\langle T;T\rangle$ including both a write-capability, granted to S, and read-capability, granted to any process that receives s: the context, instead, will only acquire s at the super-type $\mathsf{w}\langle T\rangle$ determined by the type of the transmission channel d.

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. Next section outlines this fact.

1.1 Typed processes in untyped contexts

Given the type for d available to the context, it is not difficult to be convinced that (2) above (under appropriate hypotheses on the context's view of the name *print*, see Section 2.3) represents a valid equivalence as no context that typechecks under $d : \mathsf{r}\langle\mathsf{w}\langle T\rangle\rangle$ may tell the two processes apart.

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

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

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

The implementation schema we envision here is one in which the statically checked possession and distribution of type capabilities in the source-level processes is realized in terms of the possession and the dynamic distribution of corresponding term-level capabilities in the implementation agents. For instance, each channel could be implemented by means of a pair of cryptographic keys representing the write and read capabilities. If designed carefully, and instrumented with adequate measures to protect against hostile contexts (see [1, 3]) this represents a viable idea to pursue.

The problem remains, however, to make sure that the implementation preserves the desired typed equations of the source calculus: for that to be the case, one must guarantee that for each name, the distribution

of the term capabilities in the low-level agents match the corresponding type capabilities in the source level process. While this is possible for the names that are statically shared with the context, ensuring such correspondence is much harder, if at all possible, for the names that are dynamically acquired by the context. There are two fundamental difficulties:

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

The dynamic evolution is particularly problematic in a calculus with matching because, as noticed in [13], matching makes it possible to progressively refine the type at which a name is known during the computation. This is best illustrated by the following typed labelled transition, borrowed from the calculus APi of [13], that formalises the effect of emitting a name on a public channel.

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

The configuration $\mathrm{I} \rhd P$ represents a process P operating in a context that typechecks under I, and $\mathrm{I}^{\mathsf{r}}(a)\downarrow$ indicates that I (hence the context) has a read capability on the name a. The meet $\mathrm{I} \sqcap n : \mathrm{I}^{\mathsf{r}}(a)$ in the resulting configuration represents the ability of the context to "merge" its current type for n with the type determined by receiving n on a. As explained in [13], taking the meet mimics the ability of the context to match n with the names already known to it (possibly, at different types), and obtain a more informative type based on that. For instance, if I contains the typing assumptions $n : \mathsf{r}\langle S\rangle$, $a : \mathsf{r}\langle \mathsf{w}\langle S\rangle\rangle$, $q : S$, then the context

$$\mathscr{C}[-] \;=\; - \mid a(x).[x{=}n]\overline{n}\langle q\rangle \mid n(y).\mathbf{0};\mathbf{0}$$

is typechecked by the typing system of [13]. In this case, the ability to check if the channel received through a at type $\mathsf{w}\langle S\rangle$ is equal to the channel n known at type $\mathsf{r}\langle S\rangle$ let the context use both the read and the write capability of n.

The problem is that the effect of this type refinement may propagate dynamically in ways that cannot be determined statically. To illustrate, let I and P be the typing environment and the process defined as follows:

$$\mathrm{I} \;= n : \mathsf{r}\langle \mathsf{w}\langle T\rangle\rangle, a : \mathsf{r}\langle \mathsf{r}\langle \mathsf{r}\langle T\rangle\rangle\rangle$$
$$P = \overline{a}\langle n\rangle \mid (\nu p : \mathsf{rw}\langle T;T\rangle)\overline{n}\langle p\rangle$$

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

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

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

1.2 Access control with dynamic typing

While the problem with the previous example is a direct consequence of the presence of matching, a more fundamental obstacle against full abstraction is in the very structure of the capability types adopted in the source calculus, and in the way that structure determines the acquisition of new capabilities on a name.

As we have shown, acquiring a name, say n, at a certain type not only informs on how n will be used, but also determines how other names transmitted over n will be circulated and used in the system. These invariants are all encoded in the static type of the channel at which n is received, and clearly they will not be guaranteed if that channel is shared with an untyped context. The solution we propose here is to adopt a new typing discipline for the source calculus, based on a combination of static and dynamic typing to control the interaction with the context.

We formalize our approach by introducing a typed variant of the (asynchronous) pi-calculus [5]. In this calculus, named APi@, the types at which the emitted values are to be received by the context are decided the output sites. This is accomplished by introducing a new output construct, noted $\overline{a}\langle v@A\rangle$, that relies on type coercion to enforce the delivery of v at the type A, regardless of the type of the communication channel a. A static typing system will ensure that v has indeed the type A to which it is coerced, while a mechanisms of dynamically typed synchronization guarantees that v is received only at supertypes of A, so as to guarantee the type soundness of each exchange of values.

By breaking the dependency between the types of the transmission channels and the types of the names transmitted, in APi@ we may safely dispense with the nested types of [18, 13], and rely instead on channel types with a flat structure that only exhibits the read/write access rights associated with the channels, regardless of the types of the values they transmit. Needless to say, the resulting discipline of static typing is much looser: to compensate for that one then needs a dynamically typed operational semantics to ensure type soundness.

In this paper we study the consequences of the new typing discipline and we show that the new typing system succeeds in its goal to provide reasoning methods for typed processes in arbitrary contexts. Indeed, [7] shows that the processes of APi@ may be implemented as low-level principals of a cryptographic process calculus based on the applied pi-calculus [2], while preserving their behavioural invariants.

Then we investigate how the combination of static and dynamic typing in APi@ impacts on the ability to control the behaviour of processes with respect to traditional systems relying solely on static typing, as in APi [13]. In particular, we show that there exists a sub-type preserving encoding of APi@ into APi which is fully abstract, i.e., preserves the dynamically typed equivalences of APi@ in all APi 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 APi and the fully abstract implementation developed in [7]. In particular, it allows us to isolate the fragment of APi for which the implementation of [7] is fully abstract.[1]

1. Meaning?

Plan of the paper. Section 2 introduces the calculus APi@ with its syntax, its semantics, its type theory and its typed behavioural theory. Section 3 reviews the statically typed calculus APi. Section 4 introduces a sound encoding of APi into APi@. Section 5 defines a fully abstract encoding of APi@ into APi. Section 6 concludes with final remarks. The appendix presents more details on the proofs.

2 Asynchronous pi-calculus with dynamic typing: APi@

The calculus APi@ is a typed dialect of the asynchronous pi-calculus [5, 15] with matching. The choice of an asynchronous calculus is only meant to ease the implementation of [7]. The results we present here can be obtained also for the synchronous case.

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

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

We use $\tilde{u}$ and $\tilde{A}$ to note (possibly empty) tuples of values and types, respectively. 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 $\tilde{v} : \tilde{A}$.

Table 1 Typing rules for API@.

Types

$$A,B ::= \top \mid \mathsf{rw} \mid \mathsf{r} \mid \mathsf{w}$$

Subtyping

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

Typing Rules

$$\text{(T-PRO)}\ \frac{\Gamma(u) <: A}{\Gamma \vdash u : A} \qquad \text{(T-TUPLE)}\ \frac{\Gamma \vdash v_i : A_i \quad \forall i = 1\ldots n}{\Gamma \vdash (v_1,\ldots,v_n) : (A_1,\ldots,A_n)}$$

$$\text{(T-NEW)}\ \frac{\Gamma, n : A \vdash P}{\Gamma \vdash (\nu n : A)P} \qquad \text{(T-PAR)}\ \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \text{(T-REPL)}\ \frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad \text{(T-NIL)}\ \frac{}{\Gamma \vdash \mathbf{0}}$$

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

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

The syntax of process is standard, with the exception of the constructs for input/output. As we anticipated, $\overline{u}\langle \tilde{v}@\tilde{A} \rangle$ represents the output on u of the values $\tilde{v}$ at the types $\tilde{A}$: the rules of the operational semantics will ensure that outputs at this type only synchronises with input prefixes expecting values at types higher than (or equal to) $\tilde{A}$.

The types of API@ include capability types for names and a top type $\top$. As anticipated in Section 1.2, the channel types we use have the flat structure defined in Table 1, that informs on the access rights associated with the channel: read (r), write (w) or both (rw). The subtype relation, also in Table 1, is the smallest preorder that satisfies the rules in Table 1, and admits a meet operator $\sqcap$ with a top type $\top$ and a bottom type rw, which, differently from [13], is always defined. The subtype relation is extended to tuples as usual.

2.1 Typing System

Most of the typing rules are standard and self explained, with the two exceptions we discuss next. The typing of matching is inherited, unchanged, from [13]. As in that case, we need few preliminary definitions to formalise the structure of *typing environments*. Typing environments, ranged over by $\Gamma, \Delta, \ldots$, are finite mappings from values to types. We use the operator $\sqcap$ to build type environments. We define that the type environment $\Gamma \sqcap u : A$ is $\Gamma, u : A$ if $u \notin dom(\Gamma)$; otherwise it is the type environment Γ' such that $\Gamma'(v) = \Gamma(v)$ for $v \neq u$ and $\Gamma'(u) = \Gamma(u) \sqcap A$. Given $\tilde{v} = v_1,\ldots,v_n$ and $\tilde{A} = A_1,\ldots,A_n$ we use the notation $\Gamma \sqcap \tilde{v} : \tilde{A}$ for $\Gamma \sqcap v_1 : A_1 \sqcap \ldots \sqcap v_n : A_n$, and we say that $\Gamma(\tilde{v}) = \tilde{A}$ if $\Gamma(v_i) = A_i$ for $i = 1,\ldots,n$. In case $dom(\Gamma) \cap dom(\Gamma') = \emptyset$, we write Γ, Γ' to indicate the type environment containing all the mappings in Γ and Γ'.

Subtyping is extended to type environments as expected: $\Gamma <: \Gamma'$ whenever $dom(\Gamma) = dom(\Gamma')$ and for all $v \in dom(\Gamma)$ it holds $\Gamma(v) <: \Gamma'(v)$. When $\Gamma <: \Gamma'$ we say that Γ is *compatible* with Γ'. We write $\Gamma \vdash P$ to mean that P is well typed in the typing environment Γ.

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

In the following we list the basic properties of the type system. Their proofs are left to the reader, as they are similar to the corresponding proofs in [14, 18].

Proposition 1 (Weakening). *If* $\Gamma \vdash u : A$ *then* $\Gamma, v : B \vdash u : A$.

Proposition 2 (Strengthening). *If* $\Gamma, v : B \vdash u : A$ *and* $u \neq v$ *then* $\Gamma \vdash u : A$.

Proposition 3 (Subtyping). *If* $\Gamma \vdash v : A$ *and* $\Gamma' <: \Gamma$ *then* $\Gamma' \vdash v : A$.

Proposition 4 (Meet). *If* $\Gamma \vdash P$ *then* $\Gamma \sqcap \tilde{v} : \tilde{B} \vdash P$.

The following results state that typing judgments are preserved by the substitution of values into variables. Again, the proofs follows the lines of [14, 18].

Lemma 1 (Substitution). *Assume* $\Gamma \vdash v : A$.

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

Corollary 1. *If* $\Gamma \vdash \tilde{v} : \tilde{A}$ *and* $\Gamma, \tilde{x} : \tilde{A} \vdash P$ *then* $\Gamma \vdash P\{\tilde{v}/\tilde{x}\}$.

2.2 Operational Semantics

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

$$\alpha ::= \tau \mid u(\tilde{v}@\tilde{B}) \mid (\tilde{c} : \tilde{C})\overline{u}\langle \tilde{v}@\tilde{B} \rangle \qquad \textit{actions}$$

Most of the transitions, all depicted in Table 2, are inherited from asynchronous pi-calculus. The output action $(\tilde{c} : \tilde{C})\overline{u}\langle \tilde{v}@\tilde{B} \rangle$ carries a type tag along with the output value: it represents the output of (a tuple, possibly including fresh) values $\tilde{v}$ at the types $\tilde{B}$. Dually, the input action $u(\tilde{v}@\tilde{B})$ represents the input of $\tilde{v}$ at the types $\tilde{B}$. As anticipated, synchronising input and output requires a dynamic type check: in (PI-COM@), complementary labels synchronise only if they agree on the type of the values exchanged. We do not list the usual symmetric rules for (PI-COM@), and (PI-PAR).

Subject reduction is a direct consequence of the dynamic checks.

Theorem 1 (Subject Reduction). *If* $\Gamma \vdash P$, *then*

1. $P \xrightarrow{\tau} P'$ *implies* $\Gamma \vdash P'$;
2. $P \xrightarrow{a(\tilde{v}@\tilde{A})} P'$ *implies* $\Gamma \vdash a : r$ *and* $\Gamma \sqcap \tilde{v} : \tilde{A} \vdash P'$;
3. $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle \tilde{v}@\tilde{A} \rangle} P'$ *implies* $\Gamma \vdash a : w$ *and* $\Gamma, \tilde{c} : \tilde{C} \vdash P'$, *and* $\Gamma, \tilde{c} : \tilde{C} \vdash \tilde{v} : \tilde{A}$ *with* $\tilde{c} \subseteq \tilde{v}$.

Proof. Induction on the length of the derivation for $P \xrightarrow{\alpha} P'$. We check the last applied rule, and we outline the significative cases: input, output, open and communication. Let $\tilde{v} = v_1, \ldots, v_n$ and $\tilde{A} = A_1, \ldots, A_n$.

(PI-IN@) In this case the reduction has the form $a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}$. Since $\Gamma \vdash a(\tilde{x}@\tilde{A}).P$ by (T-IN@), we infer $\Gamma \vdash a : r$ and $\Gamma, \tilde{x} : \tilde{A} \vdash P$. By Proposition 4 we obtain $(\Gamma, \tilde{x} : \tilde{A}) \sqcap \tilde{v} : \tilde{A} \vdash P$, that means $(\Gamma \sqcap \tilde{v} : \tilde{A}), \tilde{x} : \tilde{A} \vdash P$, since the usual convention on binders says that $\tilde{x}$ is disjunct from $\tilde{v}$. Clearly $\Gamma \sqcap \tilde{v} : \tilde{A} \vdash \tilde{v} : \tilde{A}$ by (T-PRO). Then we conclude $\Gamma \sqcap \tilde{v} : \tilde{A} \vdash P\{\tilde{v}/\tilde{x}\}$ by Corollary 1.

Table 2 Operational Semantics for APi@.

(Pi-Out@) $\dfrac{}{\overline{a}\langle\tilde{v}@\tilde{B}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathbf{0}}$

(Pi-Open@) $\dfrac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} P' \qquad b \neq a,\ b \in fn(\tilde{v})}{(\nu b:B)P \xrightarrow{(b:B,\ \tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} P'}$

(Pi-In@) $\dfrac{}{a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}}$

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

(Pi-Match) $\dfrac{a = b}{[a=b]P;Q \xrightarrow{\tau} P}$ (Pi-Mismatch) $\dfrac{a \neq b}{[a=b]P;Q \xrightarrow{\tau} Q}$

(Pi-Par) $\dfrac{P \xrightarrow{\alpha} P' \quad bn(\alpha) \cap fn(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$ (Pi-Res) $\dfrac{P \xrightarrow{\alpha} P' \qquad a \notin fn(\alpha) \cup bn(\alpha)}{(\nu a:A)P \xrightarrow{\alpha} (\nu a:A)P'}$ (Pi-Repl) $\dfrac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}$

(Pi-Out@) Then $\overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}$. Since $\Gamma \vdash \overline{a}\langle\tilde{v}@\tilde{A}\rangle$ by (T-Out@), we infer $\Gamma \vdash a : \mathsf{w}$ and $\Gamma \vdash \tilde{v} : \tilde{A}$. Finally $\Gamma \vdash \mathbf{0}$ by (T-Nil).

(Pi-Open@) Then $(\nu b:B)P \xrightarrow{(b:B,\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'$ with $a \neq b$, and it has been inferred from $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'$. Since $\Gamma \vdash (\nu b:B)P$ by (T-new), we infer $\Gamma, b:B \vdash P$. By induction: $\Gamma, b:B \vdash a : \mathsf{w}$ and $\Gamma, b:B, \tilde{c}:\tilde{C} \vdash \tilde{v} : \tilde{A}$ and $\Gamma, b:B, \tilde{c}:\tilde{C} \vdash P'$. Since $a \neq b$ we have $\Gamma \vdash a : \mathsf{w}$ by Proposition 2.

(Pi-Com@) Then $P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')$ is inferred from $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'$ and from $Q \xrightarrow{a(\tilde{v}@\tilde{A}')} Q'$ with $\tilde{A} <: \tilde{A}'$ and $\tilde{c} \cap fn(Q) = \emptyset$. Since $\Gamma \vdash P \mid Q$ by (T-Par), we infer $\Gamma \vdash P$ and $\Gamma \vdash Q$. By induction on $\Gamma \vdash P$ we have: *(i)* $\Gamma \vdash a : \mathsf{w}$, *(ii)* $\Gamma, \tilde{c} : \tilde{C} \vdash \tilde{v} : \tilde{A}$, and *(iii)* $\Gamma, \tilde{c} : \tilde{C} \vdash P'$. By induction on $\Gamma \vdash Q$ we have: *(iv)* $\Gamma \vdash a : \mathsf{r}$ and *(v)* $\Gamma \sqcap \tilde{v} : \tilde{A}' \vdash Q'$. From *(ii)* ad $\tilde{A} <: \tilde{A}'$ we have $(\Gamma, \tilde{c} : \tilde{C})(\tilde{v}) <: \tilde{A}'$. Hence $(\Gamma, \tilde{c} : \tilde{C}) \sqcap \tilde{v} : \tilde{A}' = \Gamma, \tilde{c} : \tilde{C}$. By Proposition 4 and *(v)* we obtain $\Gamma \sqcap \tilde{v} : \tilde{A}' \sqcap \tilde{c} : \tilde{C} \vdash Q'$. By $\tilde{c} \cap dom(\Gamma) = \emptyset$ we have $\Gamma \sqcap \tilde{v} : \tilde{A}' \sqcap \tilde{c} : \tilde{C} = (\Gamma, \tilde{c} : \tilde{C}) \sqcap \tilde{v} : \tilde{A}'$ and in turn $\Gamma, \tilde{c} : \tilde{C} \vdash Q'$. We now consider *(iii)* and apply (T-Par) to obtain $\Gamma, \tilde{c} : \tilde{C} \vdash P' \mid Q'$. We conclude $\Gamma \vdash (\nu\tilde{c}:\tilde{C})(P' \mid Q')$ by (T-New). □

2.3 Observational Equivalence

The notion of observational equivalence is based on weak bisimulation and it is inherited from [13]. An interesting aspect of this relation is that it looks at the behavior of the processes by means of contexts that have a certain knowledge of the processes. As we noted, the typing information available to the context may be different (less informative) than the information available to the system. Thus, while the system processes may perform certain action because they posses the required (type) capabilities, the same may not be true of the context. We formalise these intuitions below.

Definition 1. *A pair* $\mathrm{I} \rhd P$ *is a configuration if and only if there is a closed type environment* Γ *compatible with* I *such that* $\Gamma \vdash P$.

As in [13], we use knowledge-indexed relations in order to define the behavioural equivalence appropriate to this setting. We follow the established settings [15, 14, 13] and we propose an equivalence which respects the reductions and a suitable notion of observation barb, and it is closed under system contexts.

Definition 2 (Type-indexed relation). *A type-indexed relation $\mathscr{R}$ is a family of binary relations between processes indexed by type environments. We write $\mathrm{I} \models P \mathscr{R} Q$ to mean that (i) P and Q are related by $\mathscr{R}$ at I and (ii) that there exist Γ and Δ compatible with I such that $\Gamma \vdash P$ and $\Delta \vdash Q$. We often write $\mathrm{I} \models P \mathscr{R} Q$ as $P \mathscr{R}_I Q$.*

The following proposition is a straight consequence of the definition for type-indexed relations and configurations.

Proposition 5. *If $\mathscr{R}$ is a type-indexed relation and $\mathrm{I} \models P \mathscr{R} Q$, then $\mathrm{I} \rhd P$ and $\mathrm{I} \rhd Q$ are configurations.*

Definition 3 (Contextuality). *A type-indexed relation $\mathscr{R}$ is contextual when:*

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

As usual, *barbs* denote that a process is ready to output a particular public channel, but here their definition is based on the actual capability of the context to read from the considered channel. The notation $\mathrm{I}^{\mathsf{r}}(a) \downarrow$ indicates that I (hence the context) has a read capability on the name a, i.e., $\mathrm{I}(a) <: \mathsf{r}$, only in that case the context can observe the output action performed by the process. A similar notation will apply for $\mathrm{I}^{\mathsf{w}}(a) \downarrow$, which means $\mathrm{I}(a) <: \mathsf{w}$. Moreover, $\Longrightarrow$ denotes the reflexive and transitive closure of $\xrightarrow{\tau}$.

Definition 4 (Barbs). *Given a configuration $\mathrm{I} \rhd P$, we say that*

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

The definition of typed behavioral equivalence is now standard, as in [13].

Definition 5 (Typed Behavioral Equivalence). *Typed behavioral equivalence, noted by $\cong^{@}$, is the largest symmetric, contextual and type-indexed relation $\mathscr{R}$ such that $\mathrm{I} \models P \mathscr{R} Q$ implies*

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

As a simple illustration of the calculus and its behavioural theory we give the API@ version of our running example. Letting J be the type of jobs, the two processes S and C may be defined as follows:

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

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

$$j : \mathsf{J}, print : \top, d : \mathsf{r} \;\models\; S \,|\, C \;\cong^{@}\; S \,|\, \overline{print}\langle j@\mathsf{J}\rangle \tag{3}$$

As usual, proving such equivalences takes some effort and is often not easy, as it requires induction to capture all the typed contexts. Luckily (but not surprisingly), the construction from [13] works just as well for our calculus in providing a purely coinductive characterization for reduction barbed congruence.

Table 3 Typed Actions for API@.

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

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

(G-REDUCE)
$$\frac{P \xrightarrow{\tau} P'}{\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I} \rhd P'}$$

(G-REPL)
$$\frac{\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'}{\mathrm{I} \rhd !P \xrightarrow{\alpha} \mathrm{I}' \rhd P' \mid !P}$$

(G-OPEN@)
$$\frac{\mathrm{I}, b : \top \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle \tilde{v}@\tilde{A}\rangle} \mathrm{I}' \rhd P' \qquad b \neq a,\ b \in fn(\tilde{v})}{\mathrm{I} \rhd (\nu b{:}B)P \xrightarrow{(b,\tilde{c})\overline{a}\langle \tilde{v}@\tilde{A}\rangle} \mathrm{I}' \rhd P'}$$

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

(G-PAR)
$$\frac{\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P' \qquad bn(\alpha) \cap fn(Q) = \emptyset}{\mathrm{I} \rhd P \mid Q \xrightarrow{\alpha} \mathrm{I}' \rhd P' \mid Q}$$

(G-RES)
$$\frac{\mathrm{I}, a : \top \rhd P \xrightarrow{\alpha} \mathrm{I}', a : \top \rhd P' \qquad a \notin n(\alpha)}{\mathrm{I} \rhd (\nu a : A)P \xrightarrow{\alpha} \mathrm{I}' \rhd (\nu a : A)P'}$$

2.4 A coinductive proof technique

The characterisation draws on the definition of a set of typed labelled transitions in which the interaction between a process and its context is mediated by the type capabilities that the context possesses on the shared names. The typed actions, in Table 3, encode transitions over configurations of the form $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$, and identify the actions that can be performed by the process P only in the case they are allowed by the environment I. Not surprisingly, most of the typed transitions in the system are derived directly from their companion transitions of [13], to which we refer for the underlying intuitions and full details. Again, we leave out the symmetric rule for (G-PAR).

The only differences are in the input/output transitions, as they reflect the nature of the interactions with the context which are distinctive of the present calculus. Specifically, the (G-OUT@) rule formalises the fact that a context willing to observe an output action performed by a process may only do so by guessing a super-type of the actual type which is used in the type coercion: the super-type is also the type at which the context acquires the values emitted. Dually, the (G-IN@) rule shows that an input by a process may in general only be observed at a lower type than the one actually performed by the process. This limitation is a consequence of the subtype-based synchronisation rule, (PI-COMM@) in Table 2.

The next lemma ensures that configurations progress to valid configurations.

Lemma 2. *If* $\mathrm{I} \rhd P$ *is a configuration and* $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$*, then* $\mathrm{I}' \rhd P'$ *is a configuration.*

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

(G-IN@) Then the transition has the form $\mathrm{I} \rhd a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A}')} \mathrm{I} \rhd P\{\tilde{v}/\tilde{x}\}$ with $\tilde{A}' <: \tilde{A}$. Moreover, we know that $\mathrm{I}(a)^{\mathsf{w}}\downarrow$ and $\mathrm{I} \vdash \tilde{v} : \tilde{A}'$. By (PI-IN@) we have $a(\tilde{x}@\tilde{A}).P \xrightarrow{a(\tilde{v}@\tilde{A})} P\{\tilde{v}/\tilde{x}\}$. From $\Gamma \vdash P$ we have $\Gamma \sqcap \tilde{v} : \tilde{A} \vdash P\{\tilde{v}/\tilde{x}\}$ by subject reduction. Then, from $\mathrm{I}(\tilde{v}) <: \tilde{A}'$ and $\tilde{A}' <: \tilde{A}$ we infer $\mathrm{I}(\tilde{v}) <: \tilde{A}$ by transitivity. Since $\Gamma(\tilde{v}) <: \mathrm{I}(\tilde{v})$ we infer $\Gamma \sqcap \tilde{v} : \tilde{A} = \Gamma$, then we conclude $\Gamma \vdash P\{\tilde{v}/\tilde{x}\}$.

(G-OPEN) Then the transition has the form $\mathrm{I} \rhd P \xrightarrow{(b,\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I}' \rhd P'$ and it was derived from $\mathrm{I}, b : \top \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I}' \rhd P'$. We conclude that $\mathrm{I}' \rhd P'$ thanks to the induction hypothesis. □

We introduce the following lemmas that establish a number of useful properties about the typed actions of Table 3 and connect them with the corresponding untyped actions. In every statement $\tilde{b}$ represents the tuple of names $b_1, \ldots b_n$, and $\tilde{B}$ represents the tuple of types $B_1, \ldots B_n$.

Proposition 6 (Meet). *If* $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$ *then* $\mathrm{I} \sqcap \tilde{b} : \tilde{B} \rhd P \xrightarrow{\alpha} \mathrm{I}' \sqcap \tilde{b} : \tilde{B} \rhd P'$.

Proof. Induction on the derivation. We check the last applied rule. In case of (G-OUT@), we have $\alpha = \overline{a}\langle\tilde{v}@\tilde{A}\rangle$. Then the premisses say that $\mathrm{I}(a)^{\mathsf{r}}\downarrow$. Since $\mathrm{I} \sqcap \tilde{b} : \tilde{B} = \mathrm{I}', \Delta$ for some type environments I', Δ such that $\mathrm{I}' <: \mathrm{I}$, we conclude that $(\mathrm{I} \sqcap \tilde{b} : \tilde{B})^{\mathsf{r}}(a)\downarrow$. The thesis is a straightforward consequence of this fact. In case of (G-IN@), we have $\alpha = a(\tilde{b}@\tilde{A})$. From the premisses we infer that $(\mathrm{I} \sqcap \tilde{b} : \tilde{B})^{\mathsf{w}}(a)\downarrow$ and $\mathrm{I} \sqcap \tilde{b} : \tilde{B} \vdash v_1 : A_1, \ldots \mathrm{I} \sqcap \tilde{b} : \tilde{B} \vdash v_n : A_n$, and so we conclude the thesis. The remaining cases follow directly by induction hypothesis. □

Lemma 3 (Action correspondence). *Assume that there exists* Γ *compatible with* I *such that* $\Gamma \vdash P$.

1. *If* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I}' \rhd P'$ *then* $\mathrm{I}^{\mathsf{r}}(a)\downarrow$ *and* $\mathrm{I}' = \mathrm{I} \sqcap \tilde{v} : \tilde{B}$ *and there exist tuples* $\tilde{A} <: \tilde{B}$ *and* $\tilde{C}$ *such that* $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{b}@\tilde{A}\rangle} P'$. *Conversely, if* $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{b}@\tilde{A}\rangle} P'$ *and* $\mathrm{I}^{\mathsf{r}}(a)\downarrow$ *then for every tuple* $\tilde{B} :> \tilde{A}$, *it holds* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v} : \tilde{B} \rhd P'$.
2. *If* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{b}@\tilde{A})} \mathrm{I}' \rhd P'$ *then* $\mathrm{I}^{\mathsf{w}}(a)\downarrow$, $\mathrm{I}' = (\mathrm{I}, \tilde{c} : \tilde{C})$, $\mathrm{I}' \vdash \tilde{b} : \tilde{A}$ *and there exists a tuple* $\tilde{A}' :> \tilde{A}$ *such that* $P \xrightarrow{a(\tilde{b}@\tilde{A}')} P'$. *Conversely, if* $P \xrightarrow{a(\tilde{v}@\tilde{A})} P'$ *and* $\mathrm{I}(a)^{\mathsf{w}}\downarrow$ *then for every tuple* $\tilde{A}' <: \tilde{A}$ *such that* $\mathrm{I} \vdash \tilde{v} : \tilde{A}'$, *it holds* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A}')} \mathrm{I}, \tilde{c} : \tilde{C} \rhd P'$, *for some tuple of types* $\tilde{C}$.
3. $\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I}' \rhd P'$ *if and only if* $\mathrm{I}' = \mathrm{I}$ *and* $P \xrightarrow{\tau} P'$.

Proof. See Appendix.

The role of the environment in a configuration is to enable the communications with the context and to collect the information on the capability the context receives. This intuition is formally given by the following lemma.

Lemma 4. *Let* I, J *be closed type environments and* P, Q *be closed processes, and assume* $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. *If* $\mathrm{J} \rhd Q \xrightarrow{\alpha} \mathrm{J}' \rhd Q'$ *then also* $\mathrm{I} \rhd Q \xrightarrow{\alpha} \mathrm{I}' \rhd Q'$.

Proof. It is a consequence of Lemma 3. □

The notion of asynchronous bisimilarity arises as expected [4]. We define $\stackrel{\widehat{\alpha}}{\Longrightarrow}$ to be $\Longrightarrow$ whenever $\alpha = \tau$, and $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$ otherwise.

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

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

The asynchronous labelled bisimilarity, noted $\approx^{@}$, *is the largest type indexed asynchronous bisimulation.*

Table 4 Structural congruence

$P \mid \mathbf{0} \equiv P$	$(\nu a:T)\mathbf{0} \equiv \mathbf{0}$
$P \mid Q \equiv Q \mid P$	$P \mid (\nu a:A)Q \equiv (\nu a:A)(P \mid Q) \quad (\text{if } a \notin fn(P))$
$!P \equiv !P \mid P$	$(\nu a_1:T_1)(\nu a_2:T_2)P \equiv (\nu a_2:T_2)(\nu a_1:T_1)P$
$!!P \equiv !P$	$[a=a]P;Q \equiv P$
$!(P \mid Q) \equiv !P \mid !Q$	$[a=b]P;Q \equiv Q \quad (\text{if } a \neq b)$

Perhaps interestingly, the reader will notice that the types $\tilde{C}$ and $\tilde{A}$ chosen to match the input transition are, in both cases, exactly the types occurring in the label of the transition to be matched. Labelled bisimilarity, as defined above, can be shown to coincide with barbed congruence. The proof of this fact is mostly standard. In the following, we list the most significative results, and we leave the full details in the Appendix.

When proving that two processes are observational equivalent, it is often convenient to seek for a relation that is included in the behavioural equivalence. This technique is called *up to technique* and it need to be carefully analysed in order to ensure its soundness, that is: we need to prove that the smaller relation we introduce is actually contained in behavioural equivalence. The proof o soundness for typed labelled bisimilarity will be simplified by the "up to" technique that follows from the introduction of the usual notion of structural congruence of the pi calculus, which we note by $\equiv$. 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 axioms in Table 4.

Definition 7 (Typed labelled bisimilarity up to structural equivalence). *A symmetric type indexed relation $\mathscr{R}$ over processes is an asynchronous bisimulation up to structural equivalence if whenever* $\mathrm{I} \models P \mathscr{R} Q$*, one has:*

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

As one would expect, we obtain that the asynchronous bisimulation up to $\equiv$ is a sound technique.

Proposition 7. *If $\mathscr{R}$ is an an asynchronous bisimulation up to $\equiv$, then $\mathscr{R} \subseteq \approx^{@}$.*

Proof. See Appendix.

2.5 Soundness and completeness of typed labelled bisimilarity

We show several properties of $\approx^{@}$, useful to prove that the relation is contained in $\cong^{@}$.

Proposition 8. *$\approx^{@}$ is reduction closed and barb preserving.*

Proof. Let $\mathrm{I} \models P \approx^{@} Q$. If $\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I}' \rhd P'$, then, by definition, there exists Q' such that $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I}' \rhd Q'$ and $\mathrm{I} \models P' \approx^{@} Q'$, and this says that $\approx^{@}$ is reduction closed. Moreover, if $\mathrm{I} \models P \Downarrow a$ then, by definition, $\mathrm{I}(a)^{\mathrm{r}}\downarrow$ and $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'$ for some P' and $\tilde{A}, \tilde{C}$. By Lemma 3 there exists I' such that $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I}' \rhd P'$. Hence there exists Q' such that $\mathrm{I} \rhd Q \xRightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I}' \rhd Q'$ with $\mathrm{I}' \models P' \approx^{@} Q'$. Since $\mathrm{I}(a)^{\mathrm{r}}\downarrow$, we conclude $\mathrm{I} \models Q \Downarrow_a$ and this means that $\approx^{@}$ preserves barbs. □

Then we we focus on the proof that $\approx^{@}$ is contextual.

Lemma 5 (Subtyping closure). *If* $\mathrm{I} \models P \approx^{@} Q$ *and* $\mathrm{I} <: \mathrm{I}'$*, then* $\mathrm{I}' \models P \approx^{@} Q$.

Proof. We fix a type environment I, and we define $\mathscr{R}$ to be the type-indexed relation such that $\mathrm{J} \models P \,\mathscr{R}\, Q$ whenever $\mathrm{I} <: \mathrm{J}$ and $\mathrm{I} \models P \approx^{@} Q$. We can prove the thesis just by showing that $\mathscr{R} \subseteq \approx^{@}$. Then we assume that $\mathrm{J} \rhd P \xrightarrow{\alpha} \mathrm{J}' \rhd P'$. Since $\mathrm{I} <: \mathrm{J}$ we have $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$ with $\mathrm{I}' <: \mathrm{J}'$. In case α is a silent or an output transition, the hypothesis $\mathrm{I} \models P \approx^{@} Q$ implies that $\mathrm{I} \rhd Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{I}' \rhd Q'$ and $\mathrm{I}' \models P' \approx^{@} Q'$. Lemma 4 says that also $\mathrm{J} \rhd Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{J}' \rhd Q'$. Moreover $\mathrm{I}' \models P' \approx Q'$ implies $\mathrm{J}' \models P' \,\mathscr{R}\, Q'$. In case α is an input transition we proceed in the same way, and we conclude that $\mathscr{R} \subseteq \approx^{@}$. □

Proposition 9 (Weakening closure). *If* $\mathrm{I} \models P \approx^{@} Q$ *and* $dom(\mathrm{I}) \cap dom(\mathrm{I}') = \emptyset$ *then* $\mathrm{I}, \mathrm{I}' \models P \approx^{@} Q$.

Proof. See Appendix.

Proposition 10 (Closure under ν operator). *If* $\mathrm{I}, a : A \models P \approx^{@} Q$ *then* $\mathrm{I} \models (\nu a{:}A)P \approx^{@} (\nu a{:}A)Q$.

Proof. See Appendix.

Proposition 11 (Closure under composition). *If* $\mathrm{I} \models P \approx^{@} Q$ *and* $\mathrm{I} \vdash R$ *then* $\mathrm{I} \models (P \,|\, R) \approx^{@} (Q \,|\, R)$.

Proof. See Appendix.

Remark 1. We note that the hypothesis $\mathrm{I} \vdash R$ is crucial in the previous proof. Furthermore we observe that some of the standard properties of asynchronous bisimilarity do not hold for the type indexed counterpart. In particular, $\mathrm{I} \models P \approx^{@} Q$ does not imply $\mathrm{I} \models P \,|\, P \approx^{@} Q \,|\, Q$. To see why, we consider the following counter-example:

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

The context environment I says that the environment may not observe any action on n because it does not have capabilities on n. Thus the only available action for P and Q is the internal synchronisation on n. Hence $\mathrm{I} \models P \approx^{@} Q$. On the other hand, $\mathrm{I} \not\models P \,|\, P \approx^{@} Q \,|\, Q$ as in this case there exists the transition $\mathrm{I} \rhd P \,|\, P \xrightarrow{\overline{m}\langle\rangle} \mathrm{I} \rhd P'$ that cannot be matched by $\mathrm{I} \rhd Q \,|\, Q$.

In a similar way, we can prove that $\mathrm{I} \models P \approx^{@} Q$ does not imply $\mathrm{I} \models !P \approx^{@} !Q$.

The previous propositions provides all the results we need to prove soundness.

Theorem 2 (Soundness). *If* $\mathrm{I} \models P \approx^{@} Q$ *then* $\mathrm{I} \models P \cong^{@} Q$.

Proof. The relation $\approx^{@}$ is reduction closed and barb preserving by Proposition 8, and it is contextual by Propositions 9, 10 and 11. Hence it is contained in $\cong^{@}$. □

In order to show that the bisimulation coincides with the behavioural equivalence we need to show the converse of Theorem 2. As in [13] this result relies on the fact that the observations of processes we make with the typed actions of Table 3 are contextually valid. In general a context interacts with the processes via typed actions and can learn new capabilities from the processes or may send them (possibly fresh) capabilities. This behaviour can be actually represented through well-typed contexts. We formalise this fact by associating each typed input and output action to a typed context which mimics the behaviour of the environment in the typed transition.

Notations. In the following, we use some abbreviations. We assume I to be $\tilde{a} : \tilde{T}$. We write $\overline{n}\langle \mathrm{I} \rangle$ to identify the process $\overline{n}\langle \tilde{a}@\tilde{T} \rangle$. We write I_{types} and I_{names} to identify the tuples $\tilde{T}$ and $\tilde{a}$ respectively. We define

$$\mathrm{I}\,\mathsf{after}\,(\tilde{c})\overline{a}\langle \tilde{b}@\tilde{B} \rangle \stackrel{def}{=} \mathrm{I} \sqcap \tilde{b}@\tilde{B} \qquad \mathrm{I}\,\mathsf{after}\,(\tilde{c} : \tilde{C})a(\tilde{b}@\tilde{B}) \stackrel{def}{=} \mathrm{I}, \tilde{c} : \tilde{C} \qquad \mathrm{I}\,\mathsf{after}\,\tau \stackrel{def}{=} \mathrm{I}$$

Proposition 12. *If* $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$, *then* $\mathrm{I}' = \mathrm{I}\,\mathsf{after}\,\alpha$.

Proof. Induction on the length of the inference $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$ by checking the the last applied rule.

Proposition 13 (Label contextuality). *For every label* $\alpha \neq \tau$ *and environment* I, *there exists a process* C^{I}_{α} *and a name* $e \notin dom(\mathrm{I})$ *such that* $\mathrm{I}, e : \mathsf{rw} \vdash C^{\mathrm{I}}_{\alpha}$ *and for all* P *for which* $\mathrm{I} \rhd P$ *is a configuration it holds:*

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

The next result says that the names and the capabilities communicated along the fresh channel e of Proposition 13 are actually collected by the environment.

Proposition 14. *Assume that* $\Delta, \tilde{c} : \tilde{C}$ *and* $\Delta', \tilde{c} : \tilde{C}'$ *are compatible with* I', *that* $\Delta, \tilde{c} : \tilde{C} \vdash P$ *and* $\Delta', \tilde{c} : \tilde{C}' \vdash Q$, *and that* $e \notin fn(P) \cup fn(Q)$. *If* $\mathrm{I}, e : \mathsf{rw} \models (\nu\tilde{c} : \tilde{C})P \mid \overline{e}\langle \mathrm{I}' \rangle \cong^{@} (\nu\tilde{c} : \tilde{C}')Q \mid \overline{e}\langle \mathrm{I}' \rangle$ *for every environment* I, *then* $\mathrm{I}' \models P \cong^{@} Q$.

Proof. By co-induction. We define the type-indexed relation $\mathscr{R}$ as follows. We say that $\mathrm{I}' \models P \mathscr{R} Q$ if and only if there exist an environment I and a name $e \notin fn(P) \cup fn(Q)$ such that

$$\mathrm{I}, e : \mathsf{rw} \models (\nu\tilde{c} : \tilde{C})(P \mid \overline{e}\langle \mathrm{I}' \rangle) \cong^{@} (\nu\tilde{c} : \tilde{C}')(Q \mid \overline{e}\langle \mathrm{I}' \rangle)$$

and there exist Δ and Δ' such that $\Delta, \tilde{c} : \tilde{C}$ and $\Delta', \tilde{c} : \tilde{C}'$ are compatible with I'.

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

Barb preservation We assume $\mathrm{I}' \models P \mathscr{R} Q$. To show that $\mathscr{R}$ preserves barbs, we assume $\mathrm{I}' \models P \downarrow_a$. Intuitively, we pick a process R as a testing context which receives the environment I' from e and afterwards unlock the (un-typed) barb $P \downarrow_a$ by means of an input on a. We use a fresh channel to sign the success of the operation. To simplify the notation, we write $a().P$ for the process $a(\tilde{x}@\top).P$ whenever the variables in $\tilde{x}$ do not occur in P and the arity of $\tilde{x}$ can be deduced from the context.

We first observe that $a \in dom(\mathrm{I}')$, as $\mathrm{I}' \models P \downarrow_a$. We assume that j is the position of the name a in I'_{names}. Then we choose $e' : \mathsf{rw}$ fresh to $\mathrm{I}, e : \mathsf{rw}$ and we consider the process $R \stackrel{def}{=} e(\tilde{x}@\mathrm{I}'_{types}).x_j().\overline{e'}\langle\rangle$ where the arity of $\tilde{x}$ is the one of I'_{names}. We note that x_j will be bound to a whenever R synchronises with $\overline{e}\langle I' \rangle$.

Due to the contextuality of $\cong^{@}$, in particular the weakening clause, we have

$$\mathrm{I}, e : \mathsf{rw}, e' : \mathsf{rw} \models (\nu\tilde{c} : \tilde{c})(P \mid \overline{e}\langle \mathrm{I}' \rangle) \cong^{@} (\nu\tilde{c} : \tilde{c}')(Q \mid \overline{e}\langle \mathrm{I}' \rangle).$$

Then we will use the closure under parallel composition in order to compose both sides with R. Before doing so, we must show that $e : \mathsf{rw}, e' : \mathsf{rw} \vdash R$. This is a consequence of the fact that $\mathrm{I}' \models P \downarrow_a$ implies $\mathrm{I}'(x_a)^{\mathsf{r}} \downarrow$ and that $\mathrm{I}', e : \mathsf{rw}, e' : \mathsf{rw}, \tilde{x} : \mathrm{I}'_{types} \vdash x_j().\overline{e'}\langle\rangle$. Hence we obtain

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

By (PI@-COM) rule, we have

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

Then, as $P \downarrow_a$, by (PI-COM@) rule we infer

$$P \mid \overline{e}\langle \mathrm{I}' \rangle \mid R \Longrightarrow \Downarrow_{e'} .$$

By Lemma 3:

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

Since $\cong^{@}$ preserves barbs, it must be

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

Since e' is fresh to Q, this barb has to come from R. Moreover, in order to allow this barb, R has to input on channel. The only possibility is that $Q \Downarrow_a$. This fact, along with $\mathrm{I}(a)^{\mathsf{r}} \downarrow$, says that $\mathrm{I} \models Q \Downarrow_a$.

The reduction closure for $\mathscr{R}$ is trivial. Then we consider contextuality. We focus on closure under parallel composition and new. Weakening is straightforward.

Closure under parallel composition We assume that $\mathrm{I}' \models P \mathscr{R} Q$ and $\mathrm{I}' \vdash R$. We need to show that $\mathrm{I}' \models P \,|\, R \mathscr{R} Q \,|\, R$. The definition says that there exists $e \notin dom(\mathrm{I}')$, and hence $e \notin fn(P) \cup fn(Q) \cup fn(R)$, such that $\mathrm{I}, e : \mathsf{rw} \models (\nu \tilde{C} : \tilde{c})(P \,|\, \overline{e}\langle \mathrm{I}' \rangle) \cong^{@} (\nu \tilde{c} : \tilde{C}')(Q \,|\, \overline{e}\langle \mathrm{I}' \rangle)$. Now we choose $e' : \mathsf{rw}$ fresh to P and Q and R and we define $R' \stackrel{def}{=} e(\tilde{x}@\mathrm{I}'_{types}).(R \,|\, \overline{e'}\langle \mathrm{I}' \rangle)\,\{\tilde{x}/\mathrm{I}'_{types}\}$. Since $fn(e(\tilde{x}@\mathrm{I}'_{types}).(R \,|\, \overline{e'}\langle \mathrm{I}' \rangle)) \subseteq \{e, e'\} \cup dom(\mathrm{I}')$ we have $e : \mathsf{rw}, e' : \mathsf{rw} \vdash R'$. Thanks to the $\cong^{@}$ we obtain

$$\mathrm{I}, e' : \mathsf{rw} \models (\nu \tilde{c} : \tilde{C})(\nu e : \mathsf{rw})(P \,|\, \overline{e}\langle \mathrm{I}' \rangle \,|\, R') \cong^{@} (\nu \tilde{c} : \tilde{C}')(\nu e : \mathsf{rw})(Q \,|\, \overline{e}\langle \mathrm{I}' \rangle \,|\, R').$$

And, since $e \notin dom(\mathrm{I}')$,

$$\mathrm{I}, e' : \mathsf{rw} \models (\nu \tilde{c} : \tilde{C})(\nu e : \mathsf{rw})(P \,|\, \overline{e}\langle \mathrm{I}' \rangle \,|\, R') \cong^{@} (\nu \tilde{c} : \tilde{C})(P \,|\, \overline{e'}\langle \mathrm{I}' \rangle) \,|\, R)$$

Indeed, the process on the left hand side can reach in one reduction the term on the right hand side, and the reduction cannot be locked as P cannot interact with R'. In a similar way, we obtain

$$\mathrm{I}, e' : \mathsf{rw} \models (\nu \tilde{c} : \tilde{c})(\nu e : \mathsf{rw})(Q \,|\, \overline{e}\langle \mathrm{I}' \rangle \,|\, R') \cong^{@} (\nu \tilde{c} : \tilde{c})(Q \,|\, \overline{e'}\langle \mathrm{I}' \rangle) \,|\, R)$$

The two equalities imply that

$$\mathrm{I}, e' : \mathsf{rw} \models (\nu \tilde{c} : \tilde{C})(P \,|\, \overline{e'}\langle \mathrm{I}' \rangle) \,|\, R) \cong^{@} (\nu \tilde{c} : \tilde{C})(Q \,|\, \overline{e'}\langle \mathrm{I}' \rangle) \,|\, R).$$

The fact that $e' : \mathsf{rw}$ is fresh to P, Q, and the hypothesis on I prove that $\mathrm{I} \models P \,|\, R \mathscr{R} Q \,|\, R$, as desired.

Closure under new We assume $\mathrm{I}' = \mathrm{J}, a : A$ and $\mathrm{I}' \models P \mathscr{R} Q$. We let $(\Delta, \tilde{c} : \tilde{C})(a) = B$ and $(\Delta', \tilde{c} : \tilde{C}')(a) = D$. We need to show that $\mathrm{J} \models (\nu a : B)P \mathscr{R} (\nu a : D)Q$. We define $\Gamma \stackrel{def}{=} (\Delta, \tilde{c} : \tilde{C}) \setminus a : B$ and $\Gamma' \stackrel{def}{=} (\Delta', \tilde{c} : \tilde{C}') \setminus a : D$. The definition says that $\Delta, \tilde{c} : \tilde{C} \vdash P$ and $\Delta', \tilde{c} : \tilde{C}' \vdash Q$, and both $\Delta, \tilde{c} : \tilde{C}$ and $\Delta', \tilde{c} : \tilde{C}'$ are compatible with I'. Then we have $\Gamma \vdash (\nu a : B)P$ and $\Gamma' \vdash (\nu a : C)Q$. Moreover, from the compatibility hypotheses we obtain that Γ and Γ' are compatible with J.

The definition says also that $\mathrm{I}, e : \mathsf{rw} \models (\nu \tilde{c} : \tilde{C})(P \,|\, \overline{e}\langle \mathrm{I}' \rangle) \cong^{@} (\nu \tilde{c} : \tilde{C}')(Q \,|\, \overline{e}\langle \mathrm{I}' \rangle)$. Since $a \in dom(\mathrm{I}')$ and $\cong^{@}$ is a type indexed relation, we must distinguish two cases: *(i)* $a \in \tilde{c}$ or *(ii)* $a \in \mathrm{dom}(\mathrm{I})$. In case *(i)* we infer $\mathrm{J} \models (\nu a : B)P \mathscr{R} (\nu a : D)Q$ as $\Gamma \vdash (\nu a : B)P$ and both $\Gamma' \vdash (\nu a : D)Q$ and Γ, Γ' are compatible with J. In case *(ii)* the contextuality of $\cong^{@}$ says that

$$(\mathrm{I} \setminus a : \mathrm{I}(a)), e : \mathsf{rw} \models (\nu \tilde{c} : \tilde{C})(\nu a : B)(P \,|\, \overline{e}\langle \mathrm{I}' \rangle) \cong^{@} (\nu \tilde{c} : \tilde{C}')(\nu a : D)(Q \,|\, \overline{e}\langle \mathrm{I}' \rangle).$$

Again, since $\Gamma \vdash (\nu a : B)P$ and both $\Gamma'(\nu a : D)Q$ and Γ, Γ' are compatible with J, we conclude $\mathrm{J} \models (\nu a : B)P \mathscr{R} (\nu a : D)Q$, as desired. □

The two previous propositions are the basis to prove that $\approx^{@}$ provides a characterisation of the typed behavioural equivalence $\cong^{@}$.

Theorem 3 (Completeness). *If* $\mathrm{I} \models P \cong^{@} Q$, *then* $\mathrm{I} \models P \approx^{@} Q$.

Proof. We prove that $\cong^{@}$ is an asynchronous bisimulation up to $\equiv$. We assume that $\mathrm{I} \models P \cong^{@} Q$ and $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. If α is an output or a silent transition, we must find Q' such that $\mathrm{I} \rhd Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{I}' \rhd Q'$ and $\mathrm{I} \models P' \cong^{@} Q'$.

When $\alpha = \tau$, the reduction $\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I}' \rhd P'$ and Lemma 3 say that $P \xrightarrow{\tau} P'$ and $\mathrm{I}' = \mathrm{I}$. Moreover, since $\mathrm{I} \models P \cong^{@} Q$ we find Q' such that $Q \Longrightarrow Q'$, hence $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I} \rhd Q'$ by Lemma 3, and $\mathrm{I} \models P' \cong^{@} Q'$, as desired.

When $\alpha = (\tilde{c})\overline{a}\langle \tilde{b}@\tilde{B} \rangle$, we pick $e \notin \mathrm{I}$ such that $\mathrm{I}, e : \mathsf{rw} \vdash C^{\mathrm{I}}_{\alpha}$, where C^{I}_{α} is the testing process of Proposition 13. By Lemma 3 we have $\mathrm{I}' = \mathrm{I} \sqcap \tilde{b}@\tilde{B}$ and that there are types $\tilde{A} <: \tilde{B}$ and $\tilde{C}$ such that $P \xrightarrow{(\tilde{c}:\tilde{c})\overline{a}\langle \tilde{b}@\tilde{A} \rangle} P'$. Now, let Δ be such that $\Delta \vdash P$ and $\Delta <: \mathrm{I}$. By subject reduction: $\Delta, \tilde{c} : \tilde{C} \vdash P'$. Since $\tilde{c} \subseteq \tilde{b}$, we infer $\Delta, \tilde{c} : \tilde{C} <: \mathrm{I}'$. Then the assumption $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle \tilde{b}@\tilde{B} \rangle} \mathrm{I}' \rhd P'$ implies $C^{\mathrm{I}}_{\alpha} \,|\, P \Longrightarrow (\nu \tilde{c} : \tilde{C})\, P' \,|\, \overline{e}\langle \mathrm{I}' \rangle$ by Proposition 13. Now we define $C_f \stackrel{def}{=} \overline{g}\langle\rangle \,|\, e(\tilde{x}@\mathrm{I}'_{types}).g().\overline{f}\langle \tilde{x}@\mathrm{I}'_{types} \rangle$ with $f, g \notin dom(\mathrm{I})$. From $\mathrm{I} \models P \cong^{@} Q$, the contextuality of $\cong^{@}$ says that

$$\mathrm{I}, f : \mathsf{rw}, g : \mathsf{rw} \models (\nu e : \mathsf{rw})(C_f \,|\, C^{\mathrm{I}}_{\alpha} \,|\, P) \cong^{@} (\nu e : \mathsf{rw})(C_f \,|\, C^{\mathrm{I}}_{\alpha} \,|\, Q) \tag{4}$$

For the left hand side of the equivalence we derive the following sequence of reductions, up to structural congruence:

$$(\nu e:\mathsf{rw})(C_f \mid C^{\mathrm{I}}_{\alpha} \mid P) \Longrightarrow (\nu e:\mathsf{rw})(C_f \mid (\nu \tilde{c}:\tilde{C})\ P' \mid \overline{e}\langle \mathrm{I}' \rangle \Longrightarrow (\nu \tilde{c}:\tilde{C})\ P' \mid \overline{f}\langle \mathrm{I}' \rangle \stackrel{def}{=} \mathscr{P}$$

We notice that $\mathscr{P} \not\Downarrow_g$, since g is not in the free names of P. on the other hand $\mathscr{P} \Downarrow_f$. Then, by applying repeatedly (G-REDUCE), we obtain

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

Hence, from (4) there exists $\mathscr{Q}$ such that

$$\mathrm{I}, f:\mathsf{rw}, g:\mathsf{rw} \triangleright (\nu e:\mathsf{rw})(C_f \mid C_{\alpha} \mid Q) \Longrightarrow \mathrm{I}, f:\mathsf{rw}, g:\mathsf{rw} \triangleright \mathscr{Q}$$

with $\mathrm{I}, f:\mathsf{rw}, g:\mathsf{rw} \models \mathscr{P} \cong^{@} \mathscr{Q}$. This, in particular, means that $\mathrm{I}, f:\mathsf{rw}, g:\mathsf{rw} \not\models \mathscr{Q} \Downarrow_g$, $\mathrm{I}, f:\mathsf{rw} \models \mathscr{Q} \Downarrow_f$. Hence the prefix $e(\tilde{x}@\mathrm{I}'_{types}).g()$ must have been consumed in $\mathscr{Q}$. By means of minor structural rearrangements, we find Q' such that $\mathscr{Q} \equiv (\nu \tilde{c}:\tilde{C}')Q' \mid \overline{f}\langle \tilde{v}@\tilde{T} \rangle$. Furthermore, due to the definition of C_f, we see that in order to consume the prefix $e(\tilde{x}@\mathrm{I}'_{types}).g()$ and the output $\overline{g}\langle\rangle$, the sequence of transitions leading to $\mathscr{Q}$ must have been derived from $C^{\mathrm{I}}_{\alpha} \mid Q$ with the following (untyped) transition sequence

$$C^{\mathrm{I}}_{\alpha} \mid Q \Longrightarrow (\nu \tilde{c}:\tilde{C}')\ Q' \mid \overline{e}\langle \mathrm{I}' \rangle.$$

Thus we deduce that

$$\mathscr{Q} \equiv (\nu \tilde{c}:\tilde{C}')Q' \mid \overline{f}\langle \mathrm{I}' \rangle.$$

Now we apply Proposition 13 and we conclude

$$\mathrm{I} \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'.$$

Similarly to $\Delta \vdash P$, in case Δ' is a type environment such that $\Delta' \vdash Q$ and $\Delta' <: \mathrm{I}$, then we have $\Delta', \tilde{c}:\tilde{C}' \vdash Q'$ by subject reduction. Moreover we know that $\Delta', \tilde{c}:\tilde{C}' <: \mathrm{I}'$. Since $g \notin fn(\mathscr{P}, \mathscr{Q})$ and $\mathrm{I}, f:\mathsf{rw}, g:\mathsf{rw} \models \mathscr{P} \cong^{@} \mathscr{Q}$ we infer $\mathrm{I}, f:\mathsf{rw} \models \mathscr{P} \cong^{@} \mathscr{Q}$. We then apply Proposition 14 to $\mathrm{I}, f:\mathsf{rw} \models \mathscr{P} \cong^{@} \mathscr{Q}$, and we conclude $\mathrm{I}' \models P' \cong^{@} Q'$, as desired.

We conclude the proof by examining the input case. Then we assume $\alpha = (\tilde{c})\tilde{C}(a@\tilde{b})\tilde{B}$. The proof proceeds as in the output case, with the difference that in this case $\tilde{C}' = \tilde{C}$, and looks up the sequence:

$$C^{\mathrm{I}}_{\alpha} \mid Q \Longrightarrow (\nu \tilde{c}:\tilde{C})\ Q' \mid \overline{e}\langle \mathrm{I}' \rangle.$$

Then, due to Proposition 13, we need to check two sub-cases: either $\mathrm{I} \triangleright Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \triangleright Q'$ or Q' is $Q'' \mid \overline{a}\langle \tilde{b}@\tilde{B} \rangle$ with $Q \Longrightarrow Q''$. In the former case, we proceed as for output transitions. In the latter case, Lemma 3 says that from $Q \Longrightarrow Q''$ we have $\mathrm{I}, \tilde{c}:\tilde{C} \triangleright Q \Longrightarrow \mathrm{I}, \tilde{c}:\tilde{C} \triangleright Q''$. Then $\mathrm{I}' \models P' \cong^{@} Q'$ is a consequence of Proposition 14 as in the previous cases. □

Then, by combining Theorems 2 and 3 we obtain the following characterisation.

Theorem 4 (Soundness and Completeness). $\mathrm{I} \models P \approx^{@} Q$ *if and only if* $\mathrm{I} \models P \cong^{@} Q$.

Using this characterisation, the equation (3) can now be proved co-inductively. We leave the details to the interested reader, and focus instead on some distinguishing equations for API@. As we show below, the presence of typed synchronisation has some noticeable consequences on the behavioural theory.

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

$$a:\mathsf{r} \models \overline{a}\langle n@T \rangle \mid !\overline{a}\langle n@S \rangle \cong^{@} !\overline{a}\langle n@S \rangle \qquad \text{whenever } S <: T \tag{5}$$

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

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

Another interesting consequence of the typed semantics is observed in the behavior of *forwarder* processes like $a(x).\overline{b}\langle x\rangle$. In particular, we show that one of the distinguishing equations of the asynchronous pi-calculus holds only in very specific cases in API@. Specifically, we have:

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

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

$$\begin{aligned}\mathscr{R} \stackrel{def}{=} \{&a : \mathsf{rw} \models a(x@T).\overline{a}\langle x@T\rangle \approx^{@} \mathbf{0}\} \cup \\ &\{a : \mathsf{rw}, n : T \models \overline{a}\langle n@T\rangle \approx^{@} \overline{a}\langle n@T\rangle \mid n \text{ name}\}\end{aligned}$$

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

$$R \stackrel{def}{=} \overline{a}\langle v@S\rangle \,|\, a(x@S).\overline{\omega}\langle\rangle$$

It is easy to see that this process tells the two processes apart if S is a proper subtype of T. Let $P \stackrel{def}{=} a(x@T).\overline{a}\langle x@T\rangle$. For all $S <: T$, we have $R \,|\, P \xrightarrow{\tau} \overline{a}\langle x@T\rangle \,|\, a(x@S).\overline{\omega}\langle\rangle$. Now the process $R \,|\, \mathbf{0}$ has just two possibilities to match this move: either with *(i)* $R \,|\, \mathbf{0} \xrightarrow{\tau} \overline{\omega}\langle\rangle \,|\, \mathbf{0}$, or with *(ii)* $R \,|\, \mathbf{0} \Longrightarrow R \,|\, \mathbf{0}$. In case *(i)*, then $\overline{a}\langle x@T\rangle \,|\, a(x@S).\overline{\omega}\langle\rangle$ and $\overline{\omega}\langle\rangle \,|\, \mathbf{0}$ cannot be behavioural equivalent in the typing environment $a : \mathsf{rw}$. In fact, $a : \mathsf{rw}, \omega : \mathsf{rw} \rhd \overline{\omega}\langle\rangle \,|\, \mathbf{0}\downarrow_{\omega}$ and $a : \mathsf{rw}, \omega : \mathsf{rw} \rhd \overline{a}\langle x@T\rangle \,|\, a(x@S).\overline{\omega}\langle\rangle \not\Downarrow_{\omega}$. In case *(ii)* the same arguments apply to the processes $\overline{a}\langle x@T\rangle \,|\, a(x@S).\overline{\omega}\langle\rangle$ and $R \,|\, \mathbf{0}$.

2.6 Equivalence relation

Finally we prove that $\approx^{@}$ is an equivalence relation. This fact, along with the soundness and completeness, guarantees that $\cong^{@}$ is an equivalence relation. We first need a preliminary result.

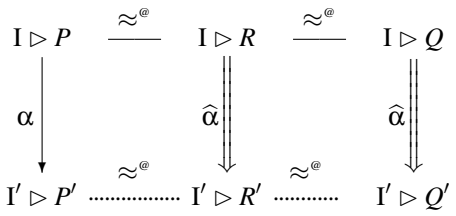
Fact 5 *If* $\mathrm{I} \models P \approx^{@} Q$ *and* $\mathrm{I} \rhd P \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd P'$ *then there exists* Q' *such that either* (i) $\mathrm{I} \rhd Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{I}' \rhd Q'$ *with* $\mathrm{I}' \models P' \approx^{@} Q'$, *or* (ii) $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I} \rhd Q'$ *with* $\mathrm{I}' \models P' \approx^{@} Q' \,|\, \overline{\alpha}$ *where* α *is an input action and* $\overline{\alpha}$ *is the corresponding output co-action.*

Proof. By induction on the number of τ reductions in $\mathrm{I} \rhd P \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd P'$.

Proposition 15. *The relation* $\approx^{@}$ *is an equivalence.*

Proof. The reflexivity of $\approx^{@}$ is a consequence of the fact that the identity relation is a bisimulation. Moreover $\approx^{@}$ is symmetric by definition. Then we just need to show transitivity. We prove that $\approx^{@}\approx^{@}$ is indeed a bisimulation. We outline the proof with diagrams, with the conventions given in the proof of Proposition 7.

We assume that $P \approx^{@}_{\mathrm{I}} R \approx^{@}_{\mathrm{I}} Q$. From $\mathrm{I} \models P \approx^{@} R$ and $\mathrm{I} \models R \approx Q$ we obtain Γ and Γ' such that $\Gamma <: \mathrm{I}$ and $\Gamma' <: \mathrm{I}$ with $\Gamma \vdash P$ and $\Gamma' \vdash Q$. This means that $\approx\approx$ is a type-indexed relation. Then we assume that $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I} \rhd P'$. In case α is an output of a τ action, we have the following situation:



In case α is the input action $(\tilde{c} : \tilde{C})a(\tilde{v}@\tilde{A})$, then:

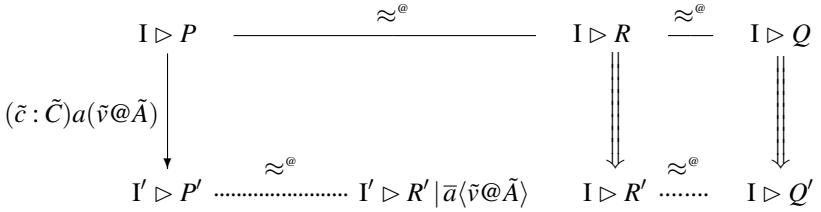


Table 5 Typing rules for API.

Types

$$S,T \;::=\; \top \mid \mathsf{r}\langle\tilde{T}\rangle \mid \mathsf{w}\langle\tilde{T}\rangle \mid \mathsf{rw}\langle\tilde{S};\tilde{T}\rangle \mid X \mid \mu X.T$$

Subtyping

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

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

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

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

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

Typing Rules for communication

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

Hence we need to show that $\mathrm{I}' \models P' \approx^{@} \approx^{@} Q' \,|\, \overline{\alpha}$, that means we have to prove $\mathrm{I}' \models R' \,|\, \overline{a}\langle\tilde{v}@\tilde{A}\rangle \approx^{@} Q' \,|\, \overline{a}\langle\tilde{v}@\tilde{A}\rangle$. Lemma 3 says that $\mathrm{I}' = \mathrm{I}, \tilde{c} : \tilde{C}$ and $\mathrm{I}(a)^{\mathsf{w}} \downarrow$ and $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v}@\tilde{A}$. Hence we infer $\mathrm{I}' \vdash \overline{a}\langle\tilde{v}@\tilde{A}\rangle$. Proposition 9 applied to $\mathrm{I} \models R' \approx^{@} Q'$ says that $\mathrm{I}' \models R' \approx^{@} Q'$. Then we conclude $\mathrm{I}' \models R' \,|\, \overline{a}\langle\tilde{v}@\tilde{A}\rangle \approx^{@} Q' \,|\, \overline{a}\langle\tilde{v}@\tilde{A}\rangle$ by Proposition 11.

Now we can repeat the previous proof with the hypothesis $\mathrm{I} \rhd Q \xrightarrow{\alpha} \mathrm{I}' \rhd Q'$ in order to show the symmetry of $\approx^{@}\approx^{@}$ and in turn that $\approx^{@}\approx^{@}$ is an asynchronous bisimulation. □

3 Asynchronous pi-calculus with static typing: API

After the detailed presentation of our dynamically typed pi-calculus, we can now draw a more precise comparison with the statically typed pi calculus of [13] to which we have referred throughout. Syntactically, API and API@ differ only in the form of the communication primitive as and in the structures of the types. The structure of processes is defined as follows:

$$P,Q,\ldots ::= \mathbf{0} \mid P|Q \mid (\nu n{:}S)P \mid !P \mid [u{=}v]P;Q \mid \overline{u}\langle\tilde{v}\rangle \mid u(\tilde{x}).P$$

The type system for API is depicted in Table 5. As outlined in the introduction, the types for API have the form $\mathsf{r}\langle S\rangle$, $\mathsf{w}\langle T\rangle$ and $\mathsf{rw}\langle S;T\rangle$, where S, T are in turn types. Values allocated at these types, respectively, are to be thought of as channel names with the capability to read values of type T, write values of type T or both. We will often use the shorthand $\mathsf{rw}\langle S\rangle$ to mean $\mathsf{rw}\langle S;S\rangle$. In order to present the significative result of this section, we extend the syntax of [12, 13] with recursion as done in [18][1]. We outline the full syntax of API types in Table 5.

[1] Although in [12, 13] the types are not recursive, as far as we see, the generalisation is harmless for the results relevant to our present endeavour.

Table 6 Operational Semantics for APi.

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

(Pi-Open)
$$\frac{P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}\rangle} P' \qquad b \neq a,\ b \in fn(\tilde{v})}{(\nu b{:}B)P \xrightarrow{(b:B,\ \tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}\rangle} P'}$$

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

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

As in [18], we require that the body of a recursive type $\mu X.T$ be *contractive* in the recursion variable X: either X does not appear at all in T, or else it appears inside at least one set of brackets $\langle\,\rangle$. In particular the type $\mu X.X$ is not allowed. The subtype relation of [12, 13] is here extended with the standard rules needed to handle the presence of recursive types [18]. In this case the relation $<:$ is a partially complete pre-order. Hence the type environment $\Gamma \sqcap u : T$ 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. The typing rules for processes are the same of APi@ except those for communication primitives, (T-Out) and (T-In), which are depicted in Table 5.

The dynamics of APi is expressed by the usual labelled transition system of the asynchronous pi-calculus. In Table 6 we report only the rules for synchronisation: the remaining rules are common to both APi and APi@ and are already depicted in Table 2. 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 receiving type is decided by the sender.

As in APi@, the notion of observational equivalence in APi is based on the notion of reduction barbed congruence and is mediated by the type capabilities that the contexts possess on the names shared with the processes. All the definition we gave for APi@ can be rephrased for APi. We just specify that the notation $\Gamma^{\mathsf{r}}(a)\downarrow$ in the static case means that the type environment Γ at a has a type of the form $\mathsf{r}\langle S\rangle$ or $\mathsf{rw}\langle S;T\rangle$ and, in this situation we write $\Gamma^{\mathsf{r}}(a)$ to refer to the type S. Similarly for $\Gamma^{\mathsf{w}}(a)$ for types with the write capability. We refer to [12, 13] for further details. Finally, the typed behavioural equivalence is denoted as $\approx$ in APi.

4 Encoding APi into APi@

The calculus APi can be naturally encoded in APi@. Intuitively, we need to drop all the additional information carried by types, just by by considering the top level capabilities. Below we give the relevant clauses of a type-directed translation from well-typed pi processes to processes of APi@: input and output. The other cases are obtained just by requiring the translation to respect the structure of processes.

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

This encoding is parametric on the typing environment, as in APi@ the syntax requires some knowledge on the use of the transmitted values. Moreover, the encoding underpins on an auxiliary definition: if T is a closed fully fledged capability type of APi, we define $|T|$ to denote the outermost capability in T. In detail:

$$|\top| \stackrel{def}{=} \top \qquad |\mathsf{r}\langle T\rangle| \stackrel{def}{=} \mathsf{r} \qquad |\mathsf{w}\langle T\rangle| \stackrel{def}{=} \mathsf{w} \qquad |\mathsf{rw}\langle T_1;T_2\rangle| \stackrel{def}{=} \mathsf{rw} \qquad |\mu X.T| \stackrel{def}{=} |T|$$

In particular we observe that this definition is well defined thanks to the contractive requirement on recursive types.

A first consequence of the definition is the fact that the encoding is type-preserving, as stated by the following proposition that can be proved by checking the correspondence between the typing rules of the two calculi

Proposition 16. *If* $\Gamma \vdash P$ *in* API, *then* $|\Gamma| \vdash [\![P]\!]_\Gamma$ *in* API@.

Moreover we can establish an operational correspondence between an API process and it encoding in API@.

Lemma 6 (Operational correspondence). *Let* P *be an* API *process and* $\Gamma <: \mathrm{I}$ *be a typing environment in* API *such that* $\Gamma \vdash P$.

(Output API*)* *If* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}\rangle} \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \rhd P'$ *in* API, *then* $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@|\Gamma^{\mathsf{w}}(a)|\rangle} |\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \rhd [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$ *in* API@ *with* $\tilde{C}$ *types in* API.

(Output API@*)* *If* $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} |\mathrm{I}| \sqcap \tilde{v} : \tilde{A} \rhd R$ *in* API@ *and* $|\Gamma, \tilde{c} : \tilde{C}| \vdash \tilde{v} : \tilde{A}$, *then* $\exists P'$ *such that* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}\rangle} \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \rhd P'$ *in* API *and* $R \equiv [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$.

(Tau API*)* *If* $\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I} \rhd P'$ *in* API, *then* $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{\tau} |\mathrm{I}| \rhd [\![P']\!]_\Gamma$ *in* API@.

(Tau API@*)* *If* $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{\tau} |\mathrm{I}| \rhd R$ *in* API@ *then there exists* P' *such that* $\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I} \rhd P'$ *in* API *and* $R \equiv [\![P']\!]_\Gamma$.

(Input API*)* *If* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})} \mathrm{I}, \tilde{c} : \tilde{C} \rhd P'$ *in* API *then* $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \rhd [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$ *in* API@.

(Input API@*)* *If* $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@\tilde{A})} |\mathrm{I}, \tilde{c} : \tilde{C}| \rhd R$ *in* API@ *and* $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \mathrm{I}^{\mathsf{w}}(a)$ *then there exists* P' *such that* $\mathrm{I} \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})} \mathrm{I}' \rhd P'$ *in* API *and* $R \equiv [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$.

Proof. The only subtlety arises in the input case. We assume that $\mathrm{I} \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})} \mathrm{I}, \tilde{c} : \tilde{C} \rhd P'$. Then the typed semantics of [13] implies $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \mathrm{I}^{\mathsf{w}}(a)$. By $\Gamma <: \mathrm{I}$ in API we have $\mathrm{I}^{\mathsf{w}}(a) <: \Gamma^{\mathsf{w}}(a)$ in API due to contra-variance. Moreover $\Gamma^{\mathsf{r}}(a)\downarrow$, due to typing assumptions, and $\Gamma^{\mathsf{w}}(a) <:_\pi \Gamma^{\mathsf{r}}(a)$ in API by definition. Then $\mathrm{I}^{\mathsf{w}}(a) <:_\pi \Gamma^{\mathsf{r}}(a)$ in API by definition, and $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \Gamma^{\mathsf{r}}(a)$ by projection. Now, type preservation says that $|\mathrm{I}, \tilde{c} : \tilde{C}| \vdash \tilde{v} : |\Gamma^{\mathsf{r}}(a)|$ and we obtain the thesis by applying (G-IN),(G-WEAK). □

Finally, the encoding $[\![\ ]\!]_\Gamma$ is sound in the following sense.

Theorem 6 (Soundness for $[\![\ ]\!]_\Gamma$). *Let* $\Gamma, \Gamma' <: \mathrm{I}$ *in* API *such that* $\Gamma \vdash P$ *and* $\Gamma' \vdash Q$ *in* API. *Then* $|\mathrm{I}| \models [\![P]\!]_\Gamma \cong^{@} [\![Q]\!]_{\Gamma'}$ *implies* $\mathrm{I} \models P \cong Q$.

Proof. The proof is based on Lemma 6. We define $\mathscr{R}$ to be the type indexed relation in API such that $\mathrm{I} \models P \mathscr{R} Q$ if and only if

$$|\mathrm{I}| \models [\![P]\!]_\Gamma \approx^{@} [\![Q]\!]_{\Gamma'} \text{ in API@, with } \Gamma, \Gamma' <: \mathrm{I} \text{ and such that } \Gamma \vdash P \text{ and } \Gamma \vdash Q \text{ in API.}$$

Then we show that $\mathscr{R}$ is an asynchronous bisimulation. The soundness and completeness results of [13] provide the desired result. We assume that $\mathrm{I} \models P \mathscr{R} Q$ and $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$ in API. We check the most significative cases: input and output.

(Output) The reduction is $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}\rangle} \mathrm{I}' \rhd P'$. From [13, Lemma 4.3], that corresponds to Proposition 12, we have $\mathrm{I}' = \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a)$. Then by Lemma 6:

$$|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@|\Gamma^{\mathsf{w}}(a)|\rangle} |\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \rhd [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$$

for some types $\tilde{C}$ in API. Since $|\mathrm{I}| \models [\![P]\!]_\Gamma \approx^{@} [\![Q]\!]_{\Gamma'}$ in API@ by definition, then there exists R such that

$$|\mathrm{I}| \rhd [\![Q]\!]_{\Gamma'} \xRightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@|\Gamma^{\mathsf{w}}(a)|\rangle} |\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \rhd R$$

with $|\mathrm{I}| \sqcap \tilde{v} : |\Gamma^{\mathsf{w}}(a)| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx R$. Then we have $|\Gamma', \tilde{c} : \tilde{C}| \vdash \tilde{v} : |\Gamma^{\mathsf{w}}(a)|$, and again by Lemma 6 we obtain $\mathrm{I} \rhd Q \xRightarrow{(\tilde{c})\overline{a}\langle\tilde{v}\rangle} \mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \rhd Q'$ and $R \equiv [\![Q']\!]_{\Gamma',\tilde{c}:\tilde{C}}$ in API. Then we conclude $\mathrm{I} \sqcap \tilde{v} : \mathrm{I}^{\mathsf{r}}(a) \models P' \mathscr{R} Q'$, as requested.

(Input) The reduction is $\mathrm{I} \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})} \mathrm{I}' \rhd P'$. From [13, Lemma 4.3] we have $\mathrm{I}' = \mathrm{I}, \tilde{c} : \tilde{C}$. By Lemma 6, in APi@ we have $|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \rhd [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}$. We consider two cases: either *(i)* $\Gamma'^{\mathsf{r}}(a)$ is a proper subtype of $\Gamma^{\mathsf{r}}(a)$ or *(ii)* not. In case *(ii)*, From $|\mathrm{I}| \models [\![P]\!]_\Gamma \approx^{@} [\![Q]\!]_{\Gamma'}$ we have two possible matching moves from $|\mathrm{I}| \rhd [\![Q]\!]_{\Gamma'}$. If the move is is matched by a weak input, then

$$|\mathrm{I}| \rhd [\![Q]\!]'_\Gamma \xRightarrow{(\tilde{c})|\tilde{C}|(a@\tilde{v})|\Gamma^{\mathsf{r}}(a)|} |\mathrm{I}, \tilde{c} : \tilde{C}| \rhd R$$

with $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx^{@} R$. Now, since $\mathrm{I} \vdash \tilde{v} : \mathrm{I}^{\mathsf{w}}(a)$, Lemma 6 says that $\mathrm{I} \rhd Q \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v})} \mathrm{I}, \tilde{c} : \tilde{C} \rhd Q'$ in API, and $R \equiv [\![Q']\!]_{\Gamma',\tilde{c}:\tilde{C}}$. Since $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx^{@} R$ we conclude $\mathrm{I}, \tilde{c} : \tilde{C} \models P' \mathscr{R} Q'$ in API. In case *(i)* the matching move is $|\mathrm{I}| \rhd [\![Q]\!]_{\Gamma'} \Longrightarrow |\mathrm{I}| \rhd R$ with $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx^{@} R \,|\, \overline{a}\langle \tilde{v}@|\Gamma^{\mathsf{r}}(a)|\rangle$. From $\Gamma', \tilde{c} : \tilde{C} \vdash \tilde{v} : |\Gamma^{\mathsf{r}}(a)|$ and the hypothesis above on $\Gamma'^{\mathsf{r}}(a)$ we have $\Gamma'^{\mathsf{r}}(a) = \Gamma^{\mathsf{r}}(a)$. Thus $R \,|\, \overline{a}\langle \tilde{v}@|\Gamma^{\mathsf{r}}(a)|\rangle \equiv [\![Q' \,|\, \overline{a}\langle \tilde{v}\rangle]\!]_{\Gamma',\tilde{c}:\tilde{C}}$ and we conclude $\mathrm{I}, \tilde{c} : \tilde{C} \models P' \mathscr{R} Q' \,|\, \overline{a}\langle \tilde{v}\rangle$ as needed.

In case *(i)* $\Gamma'^{\mathsf{r}}(a)$ is a proper subtype of $\Gamma^{\mathsf{r}}(a)$. By Lemma 6 we have $\mathrm{I}^{\mathsf{w}}(a)\downarrow$ and $\mathrm{I}^{\mathsf{w}}(a) <: \Gamma'^{\mathsf{w}}(a) <: \Gamma'^{\mathsf{r}}(a)$ in API. Thus $\mathrm{I}, \tilde{c} : \tilde{C} \vdash \tilde{v} : \Gamma'^{\mathsf{r}}(a)$ and in turn

$$|\mathrm{I}| \rhd [\![P]\!]_\Gamma \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma'^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \rhd [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}}.$$

We proceed as in the previous case and we conclude that one of the following cases holds:

1. $|\mathrm{I}| \rhd [\![Q]\!]_{\Gamma'} \xrightarrow{(\tilde{c}:|\tilde{C}|)a(\tilde{v}@|\Gamma'^{\mathsf{r}}(a)|)} |\mathrm{I}, \tilde{c} : \tilde{C}| \rhd R$ and $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx^{@} R$;
2. $|\mathrm{I}| \rhd [\![Q]\!]_{\Gamma'} \Longrightarrow |\mathrm{I}| \rhd R$ and $|\mathrm{I}, \tilde{c} : \tilde{C}| \models [\![P']\!]_{\Gamma,\tilde{c}:\tilde{C}} \approx^{@} R \,|\, \overline{a}\langle \tilde{v}@|\Gamma'^{\mathsf{r}}(a)|\rangle$

In case 1. we proceed as above by operational correspondence. In case 2. Lemma 6 says that $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I} \rhd Q'$ and $R \equiv [\![Q']\!]_{\Gamma'}$. From $R \,|\, \overline{a}\langle \tilde{v}@|\Gamma'^{\mathsf{r}}(a)|\rangle \equiv [\![Q' \,|\, \overline{a}\langle \tilde{v}\rangle]\!]_{\Gamma',\tilde{c}:\tilde{C}}$ we conclude $\mathrm{I}, \tilde{c} : \tilde{C} \models P' \mathscr{R} Q' \,|\, \overline{a}\langle v\rangle$, as needed. □

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

While we do not have a formal separation result between the two calculi, it appears that achieving a fully abstract encoding is just as hard as giving a fully abstract implementation of the typed pi calculus. In fact, as we observed, the flat capability types of API@ provide much looser control over the dynamic invariants of execution than the fully fledged capability types of [13]. Clearly, this affects the notion of typed equivalence, as the representation of contexts in terms of the typing assumptions they satisfy is much less informative on the behavior of those contexts that it is with traditional typing systems. This loss of control is compensated by the type coercions available for API@ processes to determine the types at which a context receives the emitted values: still, as the example above shows, the underlying equational theory remains affected.

On the other hand, just because its typing system makes looser assumptions on the structure of the typed contexts, API@ lends itself to be implemented into low-level calculi while preserving the typed behavior, as shown in [7].

5 Encoding API@ into API

In this section we complement the previous one by showing that there exists a sub-type preserving encoding of API@ into API which is fully abstract, i.e., preserves the dynamically typed equivalences of API@ in all API contexts. 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

Table 7 Encoding functions (with $Q = u(\tilde{x}).P$)

$$
\begin{array}{rcll}
\textsc{Read}_l\,\langle Q\rangle & \stackrel{def}{=} & u(\tilde{x}).\textsc{Test}_l\,\langle Q\rangle & \\
\textsc{Test}_l\,\langle Q\rangle & \stackrel{def}{=} & l(z).[z=\mathtt{t}]\textsc{Commit}_l\,\langle Q\rangle \oplus \textsc{Undo}_l\,\langle Q\rangle\,;\textsc{Abort}_l\,\langle Q\rangle & z \text{ fresh} \\
\textsc{Undo}_l\,\langle Q\rangle & \stackrel{def}{=} & \bar{l}\langle\mathtt{t}\rangle \,|\, \overline{u}\langle\tilde{x}\rangle & \\
\textsc{Commit}_l\,\langle Q\rangle & \stackrel{def}{=} & \bar{l}\langle\mathtt{f}\rangle \,|\, P & \\
\textsc{Abort}_l\,\langle Q\rangle & \stackrel{def}{=} & \bar{l}\langle\mathtt{f}\rangle \,|\, \overline{u}\langle\tilde{x}\rangle & \\
R_1 \oplus R_2 & \stackrel{def}{=} & (\nu i{:}\mathsf{rw}\langle\rangle)(\bar{i}\langle\rangle \mid i().R_1 \mid i().R_2) & i \text{ fresh}
\end{array}
$$

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 6, we briefly discuss how the approach can be generalized to the polyadic calculus.

5.1 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 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 [16] yields the following definition.

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

Just as in [16], 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 7, which we inherit, essentially unchanged, from [16] (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. In the general case, 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 8 A correct encoding of processes

Channel Servers

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

Clients

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

Complete Systems

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

5.2 Fixing the typing problem

The solution is inspired by [7] 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 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 8 formalize these intuitions: each 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, 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.

The new version of the encoding solves the typing problem of our initial attempt. Table 9, 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 $n_{\mathsf{r}@A} : T_{\mathsf{r}@A}$ and $n_{\mathsf{w}@A} : T_{\mathsf{w}@A}$) and $\mathsf{T} \stackrel{def}{=} (\top,\top,\top,\top)$ is the representation of the top type in API@. Notice that clients are only granted write capabilities on the channels involved in the protocols,

Table 9 Encoding of types.

Client types

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

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

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

Server types

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

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

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

Type Environments

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

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 now show that the encoding preserves the expected properties about typing. To do that, we need some auxiliary results. A direct consequence of the definition is that the subtyping lattice of API@ is preserved by the recursive types of Table 9.

Lemma 7 (Subtyping Preservation). $\mathbb{T}_{rw} <: \{\mathbb{T}_r,\mathbb{T}_w\} <: \mathbb{T}_{\top}$.

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

Corollary 2. *If* $\Gamma <: \Gamma'$ *in* API@, *then* $\{|\Gamma|\} <: \{|\Gamma'|\}$ *in* API.

As we noticed, Table 9 introduces also the server types $S_{\mathsf{r}@A}$ and $S_{\mathsf{w}@s}$, 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 8. $S_{r@A} <: T_{r@A}$ *and* $S_{w@A} <: T_{w@A}$ *for every type* A *in* API@.

Server types provide an auxiliary encoding for type environments, which is useful for type preservation. 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 7:

$$\mathbb{S} <: T \text{ for every } T \in \{\mathbb{T}_{\mathsf{rw}},\mathbb{T}_{\mathsf{r}},\mathbb{T}_{\mathsf{w}},\mathbb{T}_{\top}\}. \tag{6}$$

In order to prove the type preservation for the encoding $\{|-|\}_{\Gamma}$ defined in Table 8, we need the auxiliary encoding $\{|\,|\}_{\mathbb{S}}$ defined as

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

Then (6) 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 3.

Lemma 9. *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 $\textsc{Chan}(a) \stackrel{def}{=} \prod_{A\in\{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}} !u_{\mathsf{r}@A}(h).\textsc{Choose}(u,A,h)$. 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 $\Delta_1 \vdash u_{\mathsf{w}@\mathsf{w}} : \mathsf{rw}\langle \mathsf{T},\mathbb{W}\rangle <: \mathsf{w}\langle \mathsf{T},\mathbb{W}\rangle$ and $\Delta_1 \vdash h : \mathsf{w}\langle \mathsf{T},\mathbb{W}\rangle$, the rules of Table 5 derive $\Delta_1 \vdash \textsc{Test}_l \left\langle u_{\mathsf{w}@\mathsf{w}}(\underline{z}).\overline{h}\langle \underline{z}\rangle \right\rangle$. Moreover $\Delta_1 \vdash u_{\mathsf{w}@\mathsf{w}} : \mathsf{rw}\langle \mathsf{T},\mathbb{W}\rangle <: \mathsf{r}\langle \mathsf{T},\mathbb{W}\rangle$, hence item 1 follows by (T-In). 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 $\Delta_2 \vdash u_{\mathsf{w}@\mathsf{rw}} : \mathsf{rw}\langle \mathbb{R},\mathbb{W}\rangle <: \mathsf{w}\langle \mathbb{R},\mathbb{W}\rangle$ 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 5 derive $\Delta_2 \vdash \textsc{Test}_l \left\langle u_{\mathsf{w}@\mathsf{rw}}(\underline{z}).\overline{h}\langle \underline{z}\rangle \right\rangle$. Moreover $\Delta_2 \vdash u_{\mathsf{w}@\mathsf{rw}} : \mathsf{rw}\langle \mathbb{R},\mathbb{W}\rangle <: \mathsf{r}\langle \mathbb{R},\mathbb{W}\rangle$, hence item 2 follows by (T-In). 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 (T-Repl), (T-Par) and (T-New). Then, since $\{|\Gamma|\}, h : \mathsf{w}\langle \mathbb{R},\mathbb{W}\rangle, \langle \underline{a} : \mathbb{S}\rangle \vdash u_{\mathsf{r}@\mathsf{w}} : \mathsf{rw}\langle \mathsf{w}\langle \mathbb{R},\mathbb{W}\rangle\rangle <: \mathsf{r}\langle \mathsf{w}\langle \mathbb{R},\mathbb{W}\rangle\rangle$, an application of the rule (T-In) concludes that

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

As we 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 (T-Repl) and (T-Par). □

Then we can conclude the following result.

Corollary 3. *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 10. *If* $\Gamma \vdash P$ *in* API@, *then* $\{|\Gamma|\} \vdash \{|P|\}$ *in* API.

Proof. First check that if Γ is well define then also $\{|\Gamma|\}$ is well defined. Then, thanks to Lemma 7, 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 (T-In@), (T-Out@) and (T-New).

When (T-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 $\{|\Gamma|\} \vdash u_{\mathsf{r}@\mathsf{w}} : \mathsf{w}\langle \mathsf{w}\langle \mathbb{T}_{\mathsf{w}}\rangle\rangle$; 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 5), (T-In) and (T-Out) in particular, allow the derivation of the typing judgment $\{|\Gamma|\} \vdash (\nu h : \mathsf{rw}\langle \mathbb{T}_{\mathsf{w}}\rangle)\,(\overline{u_{\mathsf{r}@\mathsf{w}}}\langle h\rangle \,|\, h(\underline{x}).\{|P|\})$, that is $\{|\Gamma|\} \vdash \{|P|\}$. The cases $A = \mathsf{rw}, \mathsf{r}, \top$ are similar.

When (T-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 (T-Out@), and so $\{|\Gamma|\} \vdash \{|\overline{u}\langle v@A\rangle|\}$.

When (T-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 $\{|\Gamma|\}, \underline{a} : T_A \vdash \{|P|\}$, hence $\{|\Gamma|\}, \underline{a} : \mathbb{S} \vdash \{|P|\}$. Moreover $\{|\Gamma|\}, \underline{a} : \mathbb{S} \vdash \textsc{Chan}(a)$ by Lemma 9, then $\{|\Gamma|\}, \underline{a} : \mathbb{S} \vdash \{|P|\} \,|\, \textsc{Chan}(a)$ by (T-Par). Finally $\{|\Gamma|\} \vdash \{|(\nu a{:}A)P|\}$ by (T-New). □

Theorem 8 (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 7. For the second part, let $\Gamma \vdash P$ in API@. Lemma 10 says that $\{|\Gamma|\} \vdash \{|P|\}$ in API, hence $\{|\Gamma|\}_{\mathbb{S}} \vdash \{|P|\}$ by Fact 7. Moreover $\{|\Gamma|\}_{\mathbb{S}} \vdash \text{MANAGER}_\Gamma$ by Corollary 3. Thus $\{|\Gamma|\}_{\mathbb{S}} \vdash \{|P|\}_\Gamma$ by (T-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 4. *If $\mathrm{I} \rhd P$ is a configuration in* API@*, with $\Gamma <: \mathrm{I}$ such that $\Gamma \vdash P$, then $\{|\mathrm{I}|\} \rhd \{|P|\}_\Gamma$ is a configuration in* API*.*

Proof. Let $\mathrm{I} \rhd P$ be a configuration, with $\Gamma <: \mathrm{I}$ such that $\Gamma \vdash P$ in API@. Theorem 8 says that there exists $\Gamma' <: \{|\Gamma|\}$ such that $\Gamma' \vdash \{|P|\}_\Gamma$ in API. Lemma 7 says that $\{|\Gamma|\} <: \{|\mathrm{I}|\}$, hence $\Gamma' <: \{|\mathrm{I}|\}$. Thus $\{|\mathrm{I}|\} \rhd \{|P|\}_\Gamma$ is a configuration in API. □

Also, we can show that the encoding is sound, in the sense made precise below. We omit the proof as it follows the same pattern of the one for the more general Theorem 11. In particular it can be proved just by considering the administrative reductions (see Section 5.5) not involving the proxy server that will be introduced to gain completeness.

Theorem 9 (Soundness). *Let $\Gamma <: \mathrm{I}$ and $\Gamma' <: \mathrm{I}$ be two type environments such that $\Gamma \vdash P$ and $\Gamma' \vdash Q$ in* API@*. Then $\{|\mathrm{I}|\} \models \{|P|\}_\Gamma \cong \{|Q|\}_{\Gamma'}$ implies $\mathrm{I} \models P \cong^@ Q$.*

The converse direction of Theorem 9 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 9).

5.3 Failure of full abstraction

Take the following two API@ processes

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

As shown in [6], one has $\mathrm{I} \models n(y@\mathsf{rw}).\bar{n}\langle y@\mathsf{rw}\rangle \cong^@ \mathbf{0}$ for all I such that $n{:}\mathsf{rw} \in \mathrm{I}$. From this, one easily derives $a : \mathsf{w} \models P \cong^@ Q$. 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 $\mathrm{I} = a : \mathsf{w}$, so that $\{|\mathrm{I}|\} = \underline{a} : \mathbb{T}_\mathsf{w}$, and assume $\underline{a} : \mathbb{T}_\mathsf{w} \models \{|P|\}_\mathrm{I} \cong \{|Q|\}_\mathrm{I}$. Let also $\mathbb{S}$ be the server type defined in Table 9. Then, by contextuality, one would have:

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

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

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

with

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

This is 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 done by the process. On the other hand, clearly,

$$\underline{a} : \mathbb{T}_\mathsf{w}, \underline{b} : \mathbb{S} \rhd \{|Q|\}_\mathrm{I} \,|\, \overline{a_{\mathsf{w}@\mathsf{rw}}}\langle \underline{b}\rangle \not\Downarrow_{b_{\mathsf{r}@\mathsf{rw}}}$$

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 $\underline{a} : \mathbb{T}_\mathsf{w} \not\models \{|P|\}_\mathrm{I} \cong \{|Q|\}_\mathrm{I}$ as we anticipated.

Table 10 Fully Abstract Encoding of API@ into API.

Proxy

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

Clients

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

Complete Systems

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

5.4 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 [7], which relies on a *proxy* service 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 PROXY receives a client name for the first time, it maps it to a fresh name, and allocates a channel server for the new proxy name.

The definition of the proxy server is reported in Table 10. 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.

- $\mathtt{TABLE}(t,\mathscr{T})$ is a process parameterized over a structure $\mathscr{T}$ 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 $\mathscr{T}$ as a list of pairs that map channel names to server names: we write $(n,\underline{k}) \in \mathscr{T}$ when n is associated to the tuple $\underline{k}$, and $n \in dom(\mathscr{T})$ to say that there exists $\underline{k}$ with $(n,\underline{k}) \in \mathscr{T}$. Initially, the list $\mathscr{T}$ is empty (cf. Table 10).
- $\mathtt{LOOKUP}(n,t,r)$ is a process employed by the proxy to access the table referenced to by $t : \mathtt{TBL}[\top,\mathbb{S}]$. Here, $n : \top$ is the (client) name to be looked up in the table, and $r : \mathsf{w}\langle\top,\mathbb{S}\rangle$ 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 with the client name together with a boolean flag that says whether a new entry was created in the table as a result of the search. Thus, $\mathtt{LOOKUP}(n,t,r)$ replies on r the pair $(x,\underline{k})$ where $\underline{k}$ is the tuple associated with n, and x is $\mathtt{t}$ 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:

$$\frac{(n,\underline{k}) \in \mathscr{T}}{\texttt{LOOKUP}(n,t,r) \mid \texttt{TABLE}(t,\mathscr{T}) \xrightarrow{\tau} \overline{r}\langle \texttt{t},\underline{k}\rangle \mid \texttt{TABLE}(t,\mathscr{T})} \quad \text{(TABLE-LOOKUP-FOUND)}$$

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

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:

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

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 8 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 9. 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:

$$([\Gamma]) \stackrel{def}{=} \{|\,\Gamma\,|\} \cup \{p_A : \mathsf{w}\langle \mathsf{w}\langle \mathbb{T}_A\rangle, \mathbb{T}_A\rangle\}_{A\in\{\mathsf{rw},\mathsf{r},\mathsf{w},\top\}} \tag{7}$$

with $\{|\,\Gamma\,|\}$ as in Table 9. Lower types, precisely the 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 $\texttt{TBL}[\top,\mathbb{S}]$ 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:

$$\frac{\Gamma \vdash t : \texttt{TBL}[\top,\mathbb{S}], n : \top, r : \mathsf{rw}\langle \top,\mathbb{S}\rangle}{\Gamma \vdash \texttt{LOOKUP}(n,t,r)} \quad \text{(T-LOOKUP)}$$

$$\frac{\Gamma \vdash t : \texttt{TBL}[\top,\mathbb{S}], n_1 : \top, \underline{k}_1 : \mathbb{S}, \ldots n_l : \top, \underline{k}_l : \mathbb{S}}{\Gamma \vdash \texttt{TABLE}(t,[(n_1,\underline{k}_1) :: \cdots (n_l,\underline{k}_l)])} \quad \text{(T-TABLE)}$$

Based on these definitions, we can show that the encoding has the desired properties of type and subtype preservation. Essentially, the proof is a consequence of the type preservation of the encoding $\{|-|\}_\Gamma$ given in Section 5.2. As explained in (7), the encoding of typing contexts $([-])$ extends $\{|-|\}$ with the four types that describe the behavior of the proxy channels p_A, with $A = \mathsf{rw},\mathsf{r},\mathsf{w},\top$. Again, this definition preserves subtyping.

Lemma 11. *If* $\Gamma <: \Gamma'$ *in* API@, *then* $([\Gamma]) <: ([\Gamma'])$ *in* API.

Moreover, the definition ensures that the encoding of API@ processes as clients (see Table 10) is well typed.

Lemma 12. *If* $\Gamma \vdash P$ *in* API@, *then* $([\Gamma]) \vdash ([P])^*_\Gamma$ *in* API.

Proof. First observe that the definition $\text{LINK}_\Gamma\ (a,\underline{x})$ IN $P \stackrel{def}{=} (\nu h : \mathsf{rw}\langle \mathbb{T}_A \rangle)(\overline{p_A}\langle h, a_{\mathsf{r@r}} \rangle \mid h(\underline{x}).P)$, 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 10. □

Corollary 5. *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 5.2, in order to show that the proxy manager is well typed, we need to resort to auxiliary types. Contrarily to Section 5.2, 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:

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

With these assumptions it is straight to prove the following lemma.

Lemma 13. *For every* Γ *in* API@ *it holds* $([\Gamma])_{rw} \vdash$ PROXY *in* API.

Theorem 10 (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 7. For the second part, take $\Gamma' \stackrel{def}{=} ([\Gamma])_{\mathsf{rw}}$. Clearly $\Gamma' <: ([\Gamma])$. Moreover $\Gamma' \vdash ([P])^*_\Gamma$ by Lemma 12 and $\Gamma' \vdash$ PROXY by Lemma 13. Then conclude $\Gamma' \vdash ([P])^*_\Gamma$ by (T-PAR). □

Again, the encoding is well defined.

Corollary 6. *If* $\mathrm{I} \rhd P$ *is a configuration in* API@, *with* $\Gamma <: \mathrm{I}$ *such that* $\Gamma \vdash P$, *then* $([\mathrm{I}]) \rhd ([P])^*_\Gamma$ *is a configuration in* API.

Proof. Let $\mathrm{I} \rhd P$ be a configuration, with $\Gamma <: \mathrm{I}$ such that $\Gamma \vdash P$ in API@. Theorem 10 says that there exists $\Gamma' <: ([\Gamma])$ such that $\Gamma' \vdash ([P])^*_\Gamma$ in API. Lemma 7 says that $([\Gamma]) <: ([\mathrm{I}])$, hence $\Gamma' <: ([\mathrm{I}])$. Thus $([\mathrm{I}]) \rhd ([P])^*_\Gamma$ is a configuration in API. □

Now, the presence of the proxy makes the encoding fully abstract. Before giving the full abstraction proof, it is instructive to look at how the new encoding solves the problem with the example discussed in Section 5.2. In that case, the encodings of the two processes

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

are 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:

$$([P])_\Gamma \mid \overline{a_{\mathsf{w@rw}}}\langle \underline{b} \rangle \Longrightarrow\cong \text{LINK}_\Gamma\ (b,x)\ \text{IN}\ (\nu k)(\overline{x_{\mathsf{r@rw}}}\langle k \rangle \mid k(\underline{y}).\overline{x_{\mathsf{w@rw}}}\langle \underline{y} \rangle)$$

where now

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

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

5.5 Soundness and completeness

The full abstraction theorem will state that $\mathrm{I} \models P \cong^{@} Q$ in API@ if and only if $([\![\mathrm{I}]\!]) \models ([\![P]\!])^*_\Gamma \cong ([\![Q]\!])^*_{\Gamma'}$ in API. Its 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 $([\![\mathrm{I}]\!]) \models K \cong ([\![P']\!])^*_\Gamma$.
- If $([\![P]\!])^*_\Gamma \Longrightarrow K$ then there exists P' such that $P \Longrightarrow P'$ and $([\![\mathrm{I}]\!]) \models K \cong ([\![P']\!])^*_\Gamma$.

The "reflection" direction, stated by the second item above, is subtle, because our encoding is not "prompt" [16]. 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 $\cong$. 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 a variant of the operational correspondence, which will be stated in Lemma 25.

The guideline of the whole line of argument is the proof of full abstraction in [7, 10]. As we said above, the proof is based on the administrative equivalence which is is defined on the administrative reductions. 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 8 (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 $([\![P]\!])^*_\Gamma \Longrightarrow K\rho$.*

Just for the formal Definition 9, 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 $\mathfrak{m}, \mathfrak{n} \ldots$ In the encoding, signed names are those sent on the channels $u_{\mathsf{r}@A}$, 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:

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

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

Definition 9 (Administrative Reduction). *We say that $P \xrightarrow{\tau} P'$ is an* I*-administrative reduction, noted $P \xrightarrow{A_\mathrm{I}} P'$, when $\mathrm{I} \rhd P \downarrow_a$ iff $\mathrm{I} \rhd P' \downarrow_a$ 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 in the Appendix (c.f. Lemma 29), we will use a single sorted set of names and abandon the idea of signed channels. Moreover, we define $\stackrel{A_\mathrm{I}}{\Longrightarrow}$ to be the reflexive and transitive closure of $\xrightarrow{A_\mathrm{I}}$. Moreover we mark as $\xrightarrow{\tau_\mathrm{I}}$ a τ-transition which is not I-administrative.

Definition 10 (Administrative Equivalence). *The administrative equivalence $\approx_A$ is the largest symmetric, contextual and term indexed relation $\mathscr{R}$ such that $\mathrm{I} \models P \mathscr{R} Q$ implies*

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

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

Lemma 14. *Given* $\mathrm{I} \models P \approx_A Q$ *then:*

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

Lemma 15. *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 10. Symmetry holds by definition. Finally, Lemma 14 is useful to show that $\approx_A \approx_A$ satisfies Definition 10 and thus to prove transitivity. □

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

Lemma 16. *If* $\mathrm{I} \models P \approx_A Q$ *then* $\mathrm{I} \models P \cong Q$.

Proof. We show that the relation $\approx_A$ satisfies the properties of Definition 5. The contextuality of $\approx_A$ is a consequence of Definition 10. To see that $\approx_A$ preserves barbs, assume that $\mathrm{I} \models P \approx_A Q$ and $\mathrm{I} \rhd P\downarrow_a$, then $\mathrm{I} \rhd Q\downarrow_a$ by Definition 10, and thus $\mathrm{I} \rhd Q\Downarrow_a$. To see that $\approx_A$ is reduction closed, assume that $\mathrm{I} \models P \approx_A Q$ and $P \xrightarrow{\tau} P'$. In case $P \xrightarrow{A_\mathrm{I}} P'$, then there exists Q' such that $\mathrm{I} \models P' \approx_A Q'$ and $Q \overset{A_\mathrm{I}}{\Longrightarrow} Q'$, hence $Q \Longrightarrow Q'$. In case $P \xrightarrow{\tau_\mathrm{I}} P'$, then there exists Q' such that $\mathrm{I} \models P' \approx_A Q'$ and $Q \overset{A_\mathrm{I}}{\Longrightarrow} \xrightarrow{\tau_\mathrm{I}} \overset{A_\mathrm{I}}{\Longrightarrow} Q'$, 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 12 (completeness).

Definition 11 (Behavioral Equivalence up to Administrative Equivalence). *A type-indexed relation* $\mathscr{R}$ *is a behavioral equivalence up to administrative equivalence if it is symmetric, contextual up to* $\approx_A$ *and such that* $\mathrm{I} \models P \mathscr{R} Q$ *implies*

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

Next lemma says that a behavioral equivalence up to administrative equivalence is finer than the behavioral equivalence $\cong$. For its proof we refer to [7].

Lemma 17. *If* $\mathscr{R}$ *is a behavioral equivalence up to administrative equivalence, then* $\mathrm{I} \models P \mathscr{R} Q$ *implies* $\mathrm{I} \models P \approx_A Q$.

Again, the usual structural congruence of pi-calculus, depicted in Table 4, simplifies the proofs in the following of the paper. It is easy to see that the congruence is sound with respect the operational semantics given in Table 6. 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 and follows the corrispettive one in [10].

Lemma 18. *If* $P \equiv Q$ *then* $\mathrm{I} \models P \approx_A Q$ *for every* I.

The congruence relation provides a characterisation of administrative reductions. The full details are outlined in the Appendix (see Lemma 29).

The key property of administrative reductions is that they are closed under administrative equivalence, as stated by Lemma 20. The notion of non-overlapping transitions and Lemma 19 are useful to prove Lemma 20.

Definition 12 (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 $P'' \xrightarrow{\tau} R''$.*

Lemma 19. *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 20. *If $P \xrightarrow{A_{\mathrm{I}}} P'$ then $\mathrm{I} \models P \approx_A P'$.*

Proof. See Appendix.

The result of Lemma 20 can be extended to an arbitrary number of administrative reductions.

Corollary 7. *If $P \xRightarrow{A_{\mathrm{I}}} Q$ then $\mathrm{I} \models P \approx_A Q$.*

Proof. Extend Lemma 20 by induction on the number of reductions $P \xRightarrow{A_{\mathrm{I}}} Q$. □

Soundness relies on a operational correspondence between the source process P and its encoding $([P])^*_\Gamma$. The 'preservation' direction (Lemma 21) is fairly standard, whereas the 'reflection' (Lemma 24) direction needs more care as the encoding is not 'prompt' (see [16]): 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 21 (Operational Preservation). *Let $\mathrm{I} \rhd P$ a configuration in* APi@ *and $\Gamma <: \mathrm{I}$ such that $\Gamma \vdash P$. If $P \xrightarrow{\tau} P'$ then $([P])^*_\Gamma \xRightarrow{A_{\mathrm{J}}} \xrightarrow{\tau_{\mathrm{J}}} K$, with $\mathrm{J} = ([\mathrm{I}])$ and $\mathrm{J} \models K \approx_A ([P'])^*_\Gamma$.*

Proof. See Appendix.

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

Definition 13 (Canonical sequence). *The sequence $P \xRightarrow{\lceil A \rceil_{\mathrm{I}}} P'$ of administrative actions is canonical when $P' \xrightarrow{\tau_{\mathrm{I}}} Q$ with a synchronization between two input/output actions on a channel n and $P \xRightarrow{\lceil A \rceil_{\mathrm{I}}} P' \xrightarrow{\tau_{\mathrm{I}}} Q$ has the minimum length among all the reductions $P \xRightarrow{A_{\mathrm{I}}} P'' \xrightarrow{\tau_{\mathrm{I}}} Q$ in which $P'' \xrightarrow{\tau_{\mathrm{I}}} Q$ is a synchronization between the same input/output actions on n.*

We can then prove a preparatory lemma.

Lemma 22. *Let $\mathrm{I} \rhd P$ a configuration in* APi@ *and $\Gamma <: \mathrm{I}$ such that $\Gamma \vdash P$. Define $\mathrm{J} = ([\mathrm{I}])$. If $([P])^*_\Gamma \xRightarrow{A_{\mathrm{J}}} H \xrightarrow{\tau_{\mathrm{J}}} K$, then there exists H' such that $([P])^*_\Gamma \xRightarrow{\lceil A \rceil_{\mathrm{J}}} H' \xrightarrow{\tau_{\mathrm{J}}} K'$ with $K \approx_A K'$.*

Proof. See Appendix

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

Corollary 8. *Let $\mathrm{J} \rhd H$ a configuration in* APi. *If $H \xRightarrow{A_{\mathrm{J}}} Y \xrightarrow{\tau_{\mathrm{J}}} K$, then there exists H' such that $H \xRightarrow{\lceil A \rceil_{\mathrm{J}}} H' \xrightarrow{\tau_{\mathrm{J}}} K'$ 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 23. *Let* $\mathrm{I} \rhd P$ *a configuration in* API@ *and* $\Gamma <: \mathrm{I}$ *such that* $\Gamma \vdash P$. *Define* $\mathrm{J} = (\!|\mathrm{I}|\!)$. *If* $(\!|P|\!)^*_\Gamma \overset{\lceil A \rceil_\mathrm{J}}{\Longrightarrow} H \xrightarrow{\tau_\mathrm{J}} K$ *then there exists* P' *such that* $P \xrightarrow{\tau} P'$ *and* $\mathrm{J} \models K \approx_A (\!|P'|\!)^*_\Gamma$.

Proof. Assume that $(\!|P|\!)^*_\Gamma \overset{\lceil A \rceil_\mathrm{J}}{\Longrightarrow} H \xrightarrow{\tau_\mathrm{J}} K$. Thanks to canonical reduction $(\!|P|\!)^*_\Gamma \overset{\lceil A \rceil_\mathrm{J}}{\Longrightarrow} H$, 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 22). 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 $(\nu \underline{k} : \mathbb{S})\,((\!|C[Q\{b/a\}]|\!)_\Gamma \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}])(\textsc{Server}(t) \mid \mathtt{TABLE}(t, [(n, \underline{k})])))) \equiv K$. Now let P' be $C[Q\{b/a\}]$. Then conclude that $P \xrightarrow{\tau} P'$ and $\mathrm{J} \models K \approx_A (\!|P'|\!)^*_\Gamma$ as the association table does not influence the behavior of the encoded process. □

Lemma 24 (Operational Reflection). *Let* $\mathrm{I} \rhd P$ *a configuration in* API@ *and* $\Gamma <: \mathrm{I}$ *such that* $\Gamma \vdash P$. *Define* $\mathrm{J} = (\!|\mathrm{I}|\!)$. *Assume that* $\mathrm{J} \models H \approx_A (\!|P|\!)^*_\Gamma$ *and* $H \xrightarrow{\tau} K$. *Then either* $\mathrm{J} \models H \approx_A K$ *or there exists* P' *such that* $P \xrightarrow{\tau} P'$ *and* $\mathrm{J} \models K \approx_A (\!|P'|\!)^*_\Gamma$.

Proof. Assume that $\mathrm{J} \models H \approx_A (\!|P|\!)^*_\Gamma$ and $H \xrightarrow{\tau} K$. If $H \xrightarrow{A_\mathrm{I}} K$ then we apply Lemma 20 and we conclude that $\mathrm{J} \models H \approx_A K$. Otherwise, it is the case that $H \xrightarrow{\tau_\mathrm{I}} K$ and, since $\mathrm{J} \models H \approx_A (\!|P|\!)^*_\Gamma$, there exist Y, Z such that $(\!|P|\!)^*_\Gamma \overset{A_\mathrm{J}}{\Longrightarrow} \xrightarrow{\tau_\mathrm{J}} Y \overset{A_J}{\Longrightarrow} Z$ and $\mathrm{J} \models Z \approx_A K$. Then, by Lemma 22, there exists X such that $(\!|P|\!)^*_\Gamma \overset{\lceil A \rceil_\mathrm{J}}{\Longrightarrow} \xrightarrow{\tau_\mathrm{J}} X$ with $\mathrm{J} \models X \approx_A Y$ and hence $\mathrm{J} \models X \approx_A Z$ by Corollary 7. Now, By Lemma 23, there exists P' such that $P \xrightarrow{\tau} P'$ and $\mathrm{J} \models (\!|P'|\!)^*_\Gamma \approx_A X$. Then $\mathrm{J} \models (\!|P'|\!)^*_\Gamma \approx_A X \approx_A Z \approx_A K$ and conclude the thesis by transitivity. □

Lemma 25 (Operational Correspondence). *Let* $\mathrm{I} \rhd P$ *be a configuration in* API@ *and* $\Gamma <: \mathrm{I}$ *such that* $\Gamma \vdash P$. *Then:*

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

Proof. The operational preservation in item 1 is given by Lemma 21. The operational reflection in item 2 is given by Lemma 24. □

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

Corollary 9. *Let* $\mathrm{I} \rhd P$ *a configuration in* API@ *and* $\Gamma <: \mathrm{I}$ *such that* $\Gamma \vdash P$. *If* $(\!|P|\!)^*_\Gamma \Longrightarrow H$ *then there exists* P' *such that* $P \Longrightarrow P'$ *and* $(\!|\mathrm{I}|\!) \models H \approx_A (\!|P'|\!)^*_\Gamma$.

The proof of soundness requires a further preliminary lemma:

Lemma 26 (Barb Correspondence). *Let* $\mathrm{I} \rhd P$ *a configuration in* API@ *and* $\Gamma <: \mathrm{I}$ *such that* $\Gamma \vdash P$. *Then there exists* I', $h \in dom(\mathrm{I}')$ *and* $C[\cdot]$ *such that*

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

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

Relying on Lemmas 25 and 26, we have:

Theorem 11 (Soundness). *Let* $\Gamma <: \mathrm{I}$ *and* $\Gamma' <: \mathrm{I}$ *be such that* $\Gamma \vdash P$ *and* $\Gamma' \vdash Q$. *Then* $(\!|\mathrm{I}|\!) \models (\!|P|\!)^*_\Gamma \cong (\!|Q|\!)^*_{\Gamma'}$ *implies* $\mathrm{I} \models P \cong^{@} Q$.

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

$$\mathrm{I} \models P \mathscr{R} Q \quad \text{if and only if} \quad (\!|I|\!) \models (\!|P|\!)^*_{\Gamma_1} \cong (\!|Q|\!)^*_{\Gamma_2} \tag{8}$$

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

Symmetry. It holds by definition

Reduction closure. We use the operational correspondence of Lemma 25 and its Corollary 9. Assume that $\mathrm{I} \models P \mathscr{R} Q$ and let $P \xrightarrow{\tau} Q$. Define $\mathrm{J} \stackrel{def}{=} (\!|\mathrm{I}|\!)$. Item 1 of Lemma 25 says that there exists H in API such that $(\!|P|\!)^*_{\Gamma_1} \stackrel{A_\mathrm{J}}{\Longrightarrow} \xrightarrow{\tau_\mathrm{J}} H$ and $\mathrm{J} \models H \approx_A (\!|P'|\!)^*_{\Gamma_1}$. As $\approx_A$ is finer than $\cong$ we have $\mathrm{J} \models H \cong (\!|P'|\!)^*_{\Gamma_1}$. Due to the definition in (8) we deduce that there exists K such that $\mathrm{J} \models K \cong H$ and $(\!|Q|\!)^*_{\Gamma_2} \Longrightarrow K$. Now, Corollary 9 says that there exists Q' such that $Q \Longrightarrow Q'$ and $\mathrm{J} \models (\!|Q'|\!)^*_{\Gamma_2} \approx_A K$. Again, as $\approx_A \subseteq \cong$, it holds $\mathrm{J} \models (\!|Q'|\!)^*_{\Gamma_2} \cong K$ and thus $\mathrm{J} \models (\!|Q'|\!)^*_{\Gamma_2} \cong (\!|P'|\!)^*_{\Gamma_2}$ by transitivity.

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

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

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

Contextuality. Essentially this is a consequence of the contextuality of $\approx_A$. We need to prove the three requirements of Definition 3. Consider for instance item 1. Assume that $\mathrm{I}, a : A \models P \mathscr{R} Q$. Let $\Gamma_1 \vdash P$ and $\Gamma_2 \vdash Q$, with $\Gamma_1, \Gamma_2 <: \mathrm{I}, a : A$. We have to prove that $\mathrm{I} \models (\nu a{:}A) P \mathscr{R} (\nu a{:}A) Q$. By the definition in (8) we have $(\!|\mathrm{I}, a : A|\!) \models (\!|P|\!)^*_{\Gamma_1} \cong (\!|Q|\!)^*_{\Gamma_2}$ and then $(\!|\mathrm{I}|\!) \models (\nu \underline{a} : \mathbb{T}_A)(\!|P|\!)^*_{\Gamma_1} \cong (\nu \underline{a} : \mathbb{T}_A)(\!|Q|\!)^*_{\Gamma_2}$ by contextuality of $\cong$. 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 $(\!|\mathrm{I}|\!) \models (\!|(\nu a{:}A) P|\!)^*_{\Gamma'_1} \cong (\!|(\nu a{:}A) Q|\!)^*_{\Gamma'_2}$ with $\Gamma'_1, \Gamma'_2 <: \mathrm{I}$, hence $\mathrm{I} \models (\nu a{:}A) P \mathscr{R} (\nu a{:}A) Q$. The other items are analogous. □

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 PROXY as a result of the interactions with its clients.

Theorem 12 (Completeness). *Let* $\Gamma <: \mathrm{I}$ *and* $\Gamma' <: \mathrm{I}$ *be two type environments such that* $\Gamma \vdash P$ *and* $\Gamma' \vdash Q$. *Then* $\mathrm{I} \models P \cong^@ Q$ *implies* $(\!|\mathrm{I}|\!) \models (\!|P|\!)^*_\Gamma \cong (\!|Q|\!)^*_{\Gamma'}$.

Proof. We recall that $(\!|P|\!)^*_\Gamma$ is $(\!|P|\!)_\Gamma \mid (\nu t : \mathtt{TBL}[\top, \mathbb{S}])(\textsc{Server}(t) \mid \mathtt{TABLE}(t, \mathscr{T}))$ with the list $\mathscr{T}$ 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 $\mathscr{T}$ 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:

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

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 $\mathscr{T} = (n_1, \underline{k}_1) :: \ldots :: (n_p, \underline{k}_p)$ and it is defined as

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

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 $\mathscr{R}$ can be defined as:

$$\mathrm{J} \models C\left[(\!|P|\!)_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p} \right] \ \mathscr{R} \ C\left[(\!|Q|\!)_{\Gamma'} \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p} \right]$$

whenever

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

If we prove that $\mathscr{R}$ is a behavioral equivalence up to administrative equivalence (see Definition 11), then $\mathscr{R} \subseteq \cong$ by Lemma 17. Hence $\mathrm{I} \models P \cong^{@} Q$ implies $([\mathrm{I}]) \models ([P])^*_\Gamma \cong ([Q])^*_{\Gamma'}$ by defining the context $C[-]$ to be empty. So we conclude the completeness of the encoding.

Next we show that $\mathscr{R}$ barb preserving, reduction closed and contextual in the sense of Definition 11. We use the following conventions: J is the typing context $([\mathrm{I}]), a_1 : T_1, \ldots a_n : T_n$, 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 $H \mathscr{R} K$ and $\mathrm{J} \rhd H \downarrow_a$ then $\mathrm{J} \rhd C[- \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}] \downarrow_a$ as $\mathrm{J} \rhd ([P])_\Gamma$ does not exhibit barbs, thus we conclude that $\mathrm{J} \rhd K \downarrow_a$.

Reduction closure. Assume that $\mathrm{J} \models H \mathscr{R} K$. We check the cases 2 (with $H \overset{A_\mathrm{J}}{\Longrightarrow} H'$) and 3 (with $H \overset{A_\mathrm{J}}{\Longrightarrow} \overset{\tau_\mathrm{J}}{\longrightarrow} \overset{A_\mathrm{J}}{\Longrightarrow} H'$) of Definition 11.

Assume $H \overset{A_\mathrm{J}}{\Longrightarrow} H'$. Then $\mathrm{J} \models H \approx_A H'$ by Corollary 7, hence we conclude $\mathrm{J} \models H' \approx_A H \mathscr{R} K$.

Otherwise, assume $H \overset{A_\mathrm{J}}{\Longrightarrow} Y \overset{\tau_\mathrm{J}}{\longrightarrow} Y' \overset{A_\mathrm{J}}{\Longrightarrow} H'$. Then it is sufficient to find K' such that $K \Longrightarrow K'$ and $\mathrm{J} \models Y' \approx_A \mathscr{R} \approx_A K'$. The required result $\mathrm{J} \models H' \approx_A \mathscr{R} \approx_A K'$ is again a consequence of Corollary 7, as $\approx_A$ is closed under administrative reductions and transitive.

Consider $H \overset{A_\mathrm{J}}{\Longrightarrow} Y \overset{\tau_\mathrm{J}}{\longrightarrow} Y'$. Then Corollary 8 says that there exists X' such that $H \overset{[A]_\mathrm{J}}{\Longrightarrow} X \overset{\tau_\mathrm{J}}{\longrightarrow} X'$ with $\mathrm{J} \models Y' \approx_A X'$. 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 $X' \mathscr{R} K'$ according to the definition. We conclude that $K \Longrightarrow K'$ with $\mathrm{J} \models Y' \approx_A X' \mathscr{R} K'$.

(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 synchronisation 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

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

As for Lemma 23, 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 $\mathrm{J} \models (\nu \underline{k}_1 : \mathbb{S}) \ldots (\nu \underline{k}_p : \mathbb{S}) \, (([P'])_\Gamma \mid \mathsf{E}^{n_1 \ldots n_p}_{\underline{k}_1 \ldots \underline{k}_p}) \approx_A Z$. Since $\mathrm{I} \models P \cong^{@} Q$, there exists Q' such that $Q \Longrightarrow Q'$ and $\mathrm{I} \models P' \cong^{@} Q'$. As for Lemma 21, 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 $\mathrm{J} \models K' = C[([Q'])_\Gamma] \mathscr{R} C[([P'])_\Gamma] \approx_A C'[Z] \equiv X' \approx_A Y'$ 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 API@. 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 API side, and the typing of the process R that cannot read on server channels.

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

Theorem 13 (Full Abstraction). *Let* $\Gamma <: \mathrm{I}$ *and* $\Gamma' <: I$ *be two type environments such that* $\Gamma \vdash P$ *and* $\Gamma' \vdash Q$. *Then* $\mathrm{I} \models P \cong^{@} Q$ *if and only if* $([\mathrm{I}]) \models ([P])^{*}_{\Gamma} \cong ([Q])^{*}_{\Gamma'}$.

Proof. The 'if' direction is given by Theorem 11. The 'only if' direction is given by Theorem 12. □

6 Conclusions

Typed behavioral theories provide a powerful technique for reasoning on the behavior of typed processes in typed contexts. However, they are of little use in the more general case of typed processes interacting with untyped contexts. That is unfortunate, as the gap translates directly into a fundamental impediment to fully abstract implementation of the typed calculi.

We have proposed a new dialect of the asynchronous pi-calculus for which this gap can be filled, and fully abstract implementations recovered [7]. As in previous attempts of this kind, filling the gap has a price (in [3] we must give up the ability to communicate read access rights, in our case we need dynamic typing): on the other hand, it is worthwhile, as reasoning on the high-level calculus is feasible, and relatively simple, while reasoning on the implementation is utterly complex. The result gained in [7] shows indeed that the dynamic management of capabilities enforced by using a global type system, can be effectively enforced in untyped, open, distributed environments by using cryptography without any assumption on the behavior or trust of the contexts.[2]

2. Added some words on fully abstract implementation, POPL

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 [12, 13] 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 [7].

Types and advanced techniques for behavioral observation have been used extensively in the analysis of distributed computations and open systems. Types have been employed to describe resources and their usage [8, 9, 11, 14, 17] and typed equational theories have been studied to characterize the observational properties of processes [6, 12, 13, 19]. In particular, the type systems introduced in [8, 9], for Ambient calculus, and in [11], 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. Future work may include extending our present results to other calculi such as Ambients and KLAIM.

References

1. M. Abadi. Protection in programming-language translations. In *Proc. of the International Colloquium on Automata, Languages and Programming (ICALP)*, number 1443 in LNCS, pages 868–883. Springer, 1998.
2. 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.
3. M. Abadi, C. Fournet, and G. Gonthier. Secure implementation of channel abstractions. *Information and Computation*, 174(1):37–83, 2002.
4. R. M. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous pi-calculus. *Theoretical Computer Science*, 195(2):291–324, 1998.

5. G. Boudol. Asynchrony and the π-calculus. Research Report 1702, INRIA, 1992.
6. 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.
7. 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.
8. 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.
9. 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.
10. M. Giunti. *Secure Implementations of Typed Channel Abstractions*. PhD Thesis TD-2007-1, Informatics Department, University Ca Foscari of Venice, 2007.
11. 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.
12. M. Hennessy. *A Distributed Pi-Calculus*. Cambridge University Press, 2007.
13. 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.
14. M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173(1):82–120, 2002.
15. K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.
16. U. Nestmann and B C. Pierce. Decoding choice encodings. *Information and Computation*, 163(1):1–59, 2000.
17. R. De Nicola, G. L. Ferrari, R. Pugliese, and B. Venneri. Types for access control. *Theoretical Computer Science*, 240(1):215–254, 2000.
18. B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.
19. B. C. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *Journal of the ACM*, 47(3):531–584, 2000.
20. D. Sangiorgi D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge Universtity Press, 2001.

A Proofs: coinductive characterisation

Proof of Lemma 3. Induction on derivations. We first show the main cases involving the typed semantics.

(G-OUT@) Then the transition has the form $\mathrm{I} \rhd \overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v}:\tilde{B} \rhd \mathbf{0}$ where $\tilde{A} <: \tilde{B}$ and $\mathrm{I}^{\mathsf{r}}(a)\downarrow$. We conclude $\overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}$ by (PI-OUT@).

(G-IN@) Then the transition has the form $\mathrm{I} \rhd a(\tilde{x}@\tilde{A}').P \xrightarrow{a(\tilde{v}@\tilde{A})} \mathrm{I} \rhd P\{\tilde{v}/\tilde{x}\}$ where $\tilde{A} <: \tilde{A}'$ and $\mathrm{I} \vdash \tilde{v}:\tilde{A}$ and $\mathrm{I}^{\mathsf{w}}(a)\downarrow$. We conclude $a(\tilde{v}@\tilde{A}').P \xrightarrow{a(\tilde{v}@\tilde{A}')} P\{\tilde{v}/\tilde{x}\}$ by (PI-IN@).

(G-WEAK) Then the transition has the form $\mathrm{I} \rhd P \xrightarrow{(b:B,\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}' \rhd P'$, and it is inferred from $\mathrm{I}, b:B \rhd P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}' \rhd P'$ with b not occurring in $a, \tilde{c}$. The inductive hypothesis says that $(\mathrm{I}, b:B)(a)^{\mathsf{w}}\downarrow$ and $\mathrm{I}' = \mathrm{I}, b:B, \tilde{c}:\tilde{c}$ and $\mathrm{I}' \vdash \tilde{v}:\tilde{A}$ and $P \xrightarrow{a(\tilde{v}@\tilde{A}')} P'$, with $\tilde{A}' :> \tilde{A}$. Since $a \neq b$ and $(\mathrm{I}, b:B)(a)^{\mathsf{w}}\downarrow$ we conclude $\mathrm{I}(a)^{\mathsf{w}}\downarrow$ by strengthening (Proposition 2).

(G-MATCH) Then the transition has the form $\mathrm{I} \rhd [a=a]P;Q \xrightarrow{\alpha} \mathrm{I}' \rhd P'$, and it is inferred from $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. The inductive hypothesis provides the desired conditions on environment types. We need to check the condition for untyped transitions. Let α' be the untyped action which corresponds to α in the statement of the Lemma. Then, the inductive hypothesis says that $P \xrightarrow{\alpha'} P'$. We apply (PI-MATCH) to conclude $[a=a]P;Q \xrightarrow{\alpha'} P'$, as desired.

We then prove the most interesting cases involving the untyped semantics.

(PI-OUT@) Then the transition has the form $\overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathbf{0}$. We consider now the tuple $\tilde{B}$ such that $\tilde{A} <: \tilde{B}$. Whenever $\mathrm{I}(a)^{\mathsf{r}}\downarrow$ we conclude $\mathrm{I} \rhd \overline{a}\langle\tilde{v}@\tilde{A}\rangle \xrightarrow{\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v}:\tilde{B} \rhd \mathbf{0}$ by (G-OUT@).

(PI-INP@) Then the transition has the form $a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{B})} P\{\tilde{v}/\tilde{x}\}$. We consider now $\tilde{A} <: \tilde{B}$. If $\mathrm{I}(a)^{\mathsf{w}}\downarrow$ and $\mathrm{I} \vdash \tilde{v}:\tilde{A}$, then $\mathrm{I} \rhd a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{A})} \mathrm{I} \rhd P\{\tilde{v}/\tilde{x}\}$ by (G-IN@). By Proposition 6 we have $\mathrm{I}, \tilde{c}:\tilde{C} \rhd a(\tilde{x}@\tilde{B}).P \xrightarrow{a(\tilde{v}@\tilde{A})} \mathrm{I}, \tilde{c}:\tilde{C} \rhd P\{\tilde{v}/\tilde{x}\}$ for every tuple $\tilde{C}$ disjoint from I. Hence we apply (G-WEAK@) and we obtain $\mathrm{I} \rhd a(\tilde{x})\tilde{B}.P \xrightarrow{(\tilde{c}:\tilde{C})a(\tilde{v}@\tilde{A})} \mathrm{I}, \tilde{c}:\tilde{C} \rhd P\{\tilde{v}/\tilde{x}\}$, as desired.

(PI-OPEN@) Then the transition has the form $(\nu b:B)P \xrightarrow{(b:B,\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'$ and it has been inferred from $P \xrightarrow{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} P'$ with b occurring in $\tilde{v}$, but different form a. The induction says that if $\mathrm{I}(a)^{\mathsf{r}}\downarrow$ then for every tuple $\tilde{B} :> \tilde{A}$, it is the case that $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v}:\tilde{B} \rhd P'$. Now, Proposition 6 says that $\mathrm{I}, b:\top \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} (\mathrm{I}, b:\top) \sqcap \tilde{v}:\tilde{B} \rhd P'$. Since $b \in \tilde{v}$ we have $(\mathrm{I}, b:\top) \sqcap \tilde{v}:\tilde{B} = \mathrm{I} \sqcap \tilde{v}:\tilde{B}$. Then we conclude $\mathrm{I} \rhd (\nu b:B)P \xrightarrow{(b,\tilde{c})\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \tilde{v}:\tilde{B} \rhd P'$ by (G-OPEN@).

(PI-COM@) Then the transition has the form $P \mid Q \xrightarrow{\tau} (\nu\tilde{c}:\tilde{C})(P' \mid Q')$, and we apply (G-REDUCE) to conclude $\mathrm{I} \rhd P \mid Q \xrightarrow{\tau} \mathrm{I} \rhd (\nu\tilde{c}:\tilde{C})(P' \mid Q')$. □

Proof of Proposition 9. We proceed by co-induction, but first we underline some properties for typed action. They are useful in the proof. If $\mathrm{I} \rhd P$ is a configuration, then for every I' such that that $\mathrm{I}, \mathrm{I}' \rhd P \xrightarrow{\alpha} \mathrm{J} \rhd P'$ and α is a silent or output transition it must be the case that $fn(\alpha) \subseteq dom(\mathrm{I})$, and:

(i) if $\mathrm{I} \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathrm{I} \sqcap \tilde{v}:\tilde{A} \rhd P'$ then $\mathrm{I}, \mathrm{I}' \rhd P \xrightarrow{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle} \mathrm{J} \rhd P'$ with $\mathrm{J} = (\mathrm{I} \sqcap \tilde{v}:\tilde{A}), \mathrm{I}'$;

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

In particular, item *(i)* specialises Proposition 6 in case of output actions and environments with empty intersection of domains.

The we fix the a type environment I' and we define $\mathscr{R}$ to be the type-indexed relation such that $\mathrm{I},\mathrm{I}' \models P \mathscr{R} Q$ whenever $dom(\mathrm{I}) \cap dom(\mathrm{I}') = \emptyset$ and $\mathrm{I},\mathrm{I}' \models P \approx^{@} Q$ or $\mathrm{I} \models P \approx^{@} Q$. If we show that $\mathscr{R}$ is an asynchronous bisimulation, and hence $\mathscr{R} \subseteq \approx^{@}$, then we can conclude the thesis.

We assume $\mathrm{I},\mathrm{I}' \models P \mathscr{R} Q$ and $\mathrm{I},\mathrm{I}' \rhd P \xrightarrow{\alpha} \mathrm{J} \rhd P'$. Then we must find Q' such that $\mathrm{I},\mathrm{I}' \rhd Q \stackrel{\widehat{\alpha}}{\Longrightarrow} \mathrm{J} \rhd Q'$ and $\mathrm{J} \models P' \mathscr{R} Q'$. In case $\mathrm{I},\mathrm{I}' \models P \mathscr{R} Q$ from $\mathrm{I},\mathrm{I}' \models P \approx^{@} Q$ the claim is immediate. In case $\mathrm{I},\mathrm{I}' \models P \mathscr{R} Q$ from $\mathrm{I} \models P \approx^{@} Q$ we check the action α.

Case $\alpha = (\tilde{c})\overline{a}\langle \tilde{v}@\tilde{A}\rangle$. Since $\mathrm{I} \models P \approx^{@} Q$ we know that $\mathrm{I} \rhd P$ is a configuration. From item *(ii)* above, we have $\mathrm{J} = (\mathrm{I} \sqcap \tilde{v} : \tilde{A}), \mathrm{I}'$ and also

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

Since $\mathrm{I} \models P \approx^{@} Q$ there exists Q' such that

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

and $\mathrm{I} \sqcap \tilde{v} : \tilde{A} \models P' \approx^{@} Q'$ that implies $\mathrm{J} \models P' \mathscr{R} Q'$. Finally, from *(i)* we have

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

Case $\alpha = \tau$. Then $\mathrm{I},\mathrm{I}' \rhd P \xrightarrow{\tau} \mathrm{I},\mathrm{I}' \rhd P'$. By Lemma 3 we have $P \xrightarrow{\tau} P'$ and, again by Lemma 3, $\mathrm{I} \rhd P \xrightarrow{\tau} \mathrm{I} \rhd P$. Since $\mathrm{I} \models P \approx^{@} Q$, there exists Q' such that $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I} \rhd Q'$ and $\mathrm{I} \models P' \approx^{@} Q'$. We conclude that $\mathrm{I},\mathrm{I}' \rhd Q \Longrightarrow \mathrm{I},\mathrm{I}' \rhd Q'$, by Lemma 3, and $\mathrm{I},\mathrm{I}' \models P' \mathscr{R} Q'$, by definition.

Case $\alpha = (\tilde{c} : \tilde{C})a(\tilde{v}@\tilde{A})$. From $\mathrm{I},\mathrm{I}' \rhd P \xrightarrow{\alpha} \mathrm{J} \rhd P'$, we have $\mathrm{I} \rhd P \xrightarrow{(\mathrm{I}')\alpha} \mathrm{J} \rhd P'$, by (G-WEAK). Then Lemma 3 says that $\mathrm{J} = \mathrm{I},\mathrm{I}', \tilde{c} : \tilde{C}$. From $\mathrm{I} \models P \approx^{@} Q$ we find Q' such that either *(a)* $\mathrm{I} \rhd Q \xrightarrow{(\mathrm{I}')\alpha} \mathrm{J} \rhd Q'$ and $\mathrm{J} \models P' \approx^{@} Q'$, or *(b)* $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I} \rhd Q'$ with $\mathrm{J} \models P' \approx^{@} (Q' \,|\, \overline{a}\langle \tilde{v}@\tilde{A}\rangle)$. In case *(a)* by (G-WEAK) we have $\mathrm{I},\mathrm{I}' \rhd Q \xrightarrow{\alpha} \mathrm{J} \rhd Q'$, then we can conclude as $\mathrm{J} \models P' \approx^{@} Q'$ implies $\mathrm{J} \models P' \mathscr{R} Q'$ as desired. In case *(b)* we can conclude as well, in fact we found Q' such that $\mathrm{I},\mathrm{I}', \tilde{c} : \tilde{C} \rhd Q \Longrightarrow \mathrm{J} \rhd Q'$ with $\mathrm{J} \models P' \mathscr{R} (Q' \,|\, \overline{a}\langle \tilde{v}@\tilde{A}\rangle)$. □

Proof of Proposition 10. We show the most general case

$$\text{If } \mathrm{I}, a : \top \models P \approx^{@} Q, \text{ then } \mathrm{I} \models (\nu a{:}A)P \approx^{@} (\nu a{:}A)Q. \tag{9}$$

The thesis is a consequence of Lemma 5, which says that $\mathrm{I}, a : A \models P \approx^{@} Q$ implies $\mathrm{I}, a : \top \models P \approx Q$.

In order to prove (9) we define $\mathscr{R}$ be the type indexed relation such that $\mathrm{I} \models P \mathscr{R} Q$ whenever *(i)* $\mathrm{I} \models P \approx^{@} Q$ or *(ii)* $P = (\nu a{:}A)P'$ and $Q = (\nu a{:}A)Q'$ with $\mathrm{I}, a : \top \models P' \approx^{@} Q'$. Now, let $\Gamma = \Gamma', a : A$. If $\Gamma \vdash P$, then $\Gamma' \vdash (\nu a{:}A)P'$ by (T-NEW), and if $\Gamma <: \mathrm{I}, a : \top$ we have $\Gamma' <: \mathrm{I}$. The same reasoning holds for $(\nu a{:}A)Q'$. Hence we conclude that $\mathscr{R}$ is a type indexed relation.

We then show that $\mathscr{R}$ is a type indexed bisimulation. In detail, we show that if $\mathrm{I} \models P \mathscr{R} Q$ then every transition of the configuration $\mathrm{I} \rhd P$ can be matched by $\mathrm{I} \rhd Q$. When $\mathrm{I} \models P \mathscr{R} Q$ by *(i)* the claim is trivial. We consider *(ii)*, and we assume that $\mathrm{I}, a : \top \models P \approx^{@} Q$ and $\mathrm{I} \rhd (\nu a{:}A)P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. We proceed by induction on the length of the derivation for the transition α.

(G-OPEN) Then the transition is $\mathrm{I} \rhd (\nu a{:}A)P \xrightarrow{(a)\alpha} \mathrm{I}' \rhd P'$, and the premise of the rule is $\mathrm{I}, a : \top \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. Since $\mathrm{I}, a : \top \models P \approx Q$ then there exists Q' such that $\mathrm{I}' \models P' \approx^{@} Q'$, and $\mathrm{I}, a : \top \rhd Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q'$. We have just found the right process Q' such that $\mathrm{I}' \models P' \mathscr{R} Q'$ by *(i)*, and $\mathrm{I} \rhd (\nu a{:}A)Q \stackrel{(a)\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q'$ by (G-OPEN).

(G-RES) Then the transition is $\mathrm{I} \rhd (\nu a{:}A)P \xrightarrow{\alpha} \mathrm{I}' \rhd (\nu a{:}A)P'$ with $a \notin n(\alpha)$, and the premise of the rule is $\mathrm{I}, a : \top \rhd P \xrightarrow{\alpha} \mathrm{I}', a : \top \rhd P'$. Since $\mathrm{I}, a : \top \models P \approx^{@} Q$, there exists Q' such that $\mathrm{I}, a : \top \rhd Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}', a : \top \rhd Q'$

with $\mathrm{I}, a:\top \models P' \approx^{@} Q'$. Thus, by (G-RES) along with the fact $a \notin n(\alpha)$, we have $\mathrm{I} \rhd (\nu a{:}A)Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd (\nu a{:}A)Q'$. Since $\mathrm{I}, a:\top \models P' \approx^{@} Q'$ we conclude $\mathrm{I} \models (\nu a{:}A)P' \mathrel{\mathscr{R}} (\nu a{:}A)Q'$.

(G-REDUCE) Then the transition is $\mathrm{I} \rhd (\nu a:A)P \stackrel{\tau}{\longrightarrow} \mathrm{I}' \rhd (\nu a:A)P'$, and the premise of the rule is $(\nu a:A)P \stackrel{\tau}{\longrightarrow} (\nu a:A)P'$, which in turn is deduced from $P \stackrel{\tau}{\longrightarrow} P'$. We now show that there exists Q' such that $\mathrm{I} \rhd (\nu a:A)Q \Longrightarrow \mathrm{I}' \rhd (\nu a:A)Q'$ and $\mathrm{I}, a:\top \models P' \approx^{@} Q'$. Since $P \stackrel{\tau}{\longrightarrow} P'$, Lemma 3 says that $\mathrm{I}, a:\top \rhd P \stackrel{\tau}{\longrightarrow} \mathrm{I}, a:\top \rhd P'$. Since $\mathrm{I}, a:\top \models P \approx^{@} Q$ then there exists Q' such that $\mathrm{I}, a:\top \rhd Q \Longrightarrow \mathrm{I}, a:\top \rhd Q'$ and $\mathrm{I}, a:\top \models P' \approx Q'$. The process Q' is the required one. In fact, from $\mathrm{I}, a:\top \rhd Q \Longrightarrow \mathrm{I}, a:\top \rhd Q'$, we conclude $\mathrm{I} \rhd (\nu a:A)Q \Longrightarrow \mathrm{I} \rhd (\nu a:A)Q'$ by (G-RES) and several instances of (G-REDUCE).

(G-WEAK) Then the transition is $\mathrm{I} \rhd (\nu a:A)P \stackrel{(b)\alpha}{\longrightarrow} \mathrm{I}' \rhd P'$, and the premise of the rule is $\mathrm{I}, b:D \rhd (\nu a:A)P \stackrel{\alpha}{\longrightarrow} \mathrm{I}' \rhd P'$. Since $\mathrm{I}, a:\top \models P \approx^{@} Q$ we have $\mathrm{I}, b:D, a:\top \models P \approx^{@} Q$, by Proposition 9. Then the induction hypothesis says that there exists a weak transition $\mathrm{I}, b:D \rhd (\nu a:A)Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q'$ (or else the weak transition corresponding to second case of the definition of asynchronous bisimulation), with $\mathrm{I}' \models P' \approx^{@} Q'$, and hence $\mathrm{I}' \models P' \mathrel{\mathscr{R}} Q'$. Form the last transition we conclude $\mathrm{I} \rhd (\nu a:A)Q \stackrel{(b)\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q'$, as desired. □

To prove the closure of labelled bisimilarity under parallel composition (Proposition 11) we need two technical lemmas; the first lemma will be used in the (G-PAR) sub-case, while the second is used in (G-RED).

Lemma 27. *If* $\mathrm{I} \vdash P$ *and* $\mathrm{I} \rhd P \stackrel{\alpha}{\longrightarrow} \mathrm{I}' \rhd P'$ *then there exists* I'' *such that* $\mathrm{I}' = \mathrm{I}, \mathrm{I}''$.

Proof. Intuitively, a process typechecking at I both may add new names to the knowledge of I, and may refine the information on the names that I already knows, namely I may acquire lower type/capabilities on its names. Formally, the proof is by case analysis on the form of the label α. If α is an input or a silent action, then the proof is a consequence of Lemma 3. Otherwise, when α is $(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle$, then Lemma 3 says that $\mathrm{I}' = \mathrm{I} \sqcap \tilde{v}:\tilde{A}$ and $P \stackrel{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}'\rangle}{\longrightarrow} P'$ for some tuples $\tilde{C}$ and $\tilde{A}' <: \tilde{A}$. The fact that $\mathrm{I} \vdash P$ implies $\mathrm{I}, \tilde{c}:\tilde{C} \vdash \tilde{v}:\tilde{A}'$ and $\mathrm{I}, \tilde{c}:\tilde{C} \vdash P'$ by Subject Reduction. Then we define $\mathrm{I}'' = \tilde{c}:\tilde{C}$. □

Lemma 28. *Let* $\Gamma \vdash P$ *with* $\Gamma <: \mathrm{I}$. *If* $P \stackrel{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle}{\longrightarrow} P'$ *then* $\Gamma, \tilde{c}:\tilde{C} <: \mathrm{I} \sqcap \tilde{v}:\tilde{A}$.

Proof. We assume $P \stackrel{(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{v}@\tilde{A}\rangle}{\longrightarrow} P'$. By Lemma 3 we have $\mathrm{I} \rhd P \stackrel{(\tilde{c})\overline{a}\langle\tilde{v}@\tilde{A}\rangle}{\longrightarrow} \mathrm{I} \sqcap \tilde{v}:\tilde{A} \rhd P'$. Since $\Gamma \vdash P$ then $\Gamma, \tilde{c}:\tilde{C} \vdash \tilde{v}:\tilde{A}$ and $\tilde{c} \subseteq \tilde{v}$ by subject reduction. Hence *(i)* $\tilde{v} \in dom(\Gamma, \tilde{c}:\tilde{C})$ and *(ii)* $(\Gamma, \tilde{c}:\tilde{C})(\tilde{v}) <: \tilde{A}$. From *(i)* and $\tilde{c} \subseteq \tilde{v}$ we have $dom(\Gamma, \tilde{c}:\tilde{C}) = dom(\mathrm{I} \sqcap \tilde{v}:\tilde{A})$. From *(ii)* and $\Gamma <: \mathrm{I}$ we conclude $\Gamma, \tilde{c}:\tilde{C} <: \mathrm{I} \sqcap \tilde{v}:\tilde{A}$. □

Proof of Proposition 11. To simplify the notation, we write $(\nu\Delta)P$ whenever Δ is of the form $\tilde{a}:\tilde{A}$, and we write $|\Delta|$ to for the tuple $\tilde{a}$. Then we define be the type-indexed relation $\mathscr{R}$ as follows:

$\mathrm{I} \models (\nu\Delta)(P \mid R) \mathrel{\mathscr{R}} (\nu\Delta')(Q \mid R)$ *if and only if* $dom(\Delta) = dom(\Delta')$ *and there exists* J *such that:*

1. $\mathrm{I} \sqcap \mathrm{J} \models P \approx^{@} Q$;
2. $\mathrm{I} \sqcap \mathrm{J} \vdash R$;
3. *there exists* Γ *such that* $\Gamma, \Delta <: \mathrm{I} \sqcap \mathrm{J}$ *and* $\Gamma, \Delta \vdash P$;
4. *there exists* Γ' *such that* $\Gamma', \Delta' <: \mathrm{I} \sqcap \mathrm{J}$ *and* $\Gamma', \Delta' \vdash Q$.

We note that if $\mathrm{I} \models P \approx^{@} Q$ and $\mathrm{I} \vdash R$ then $\mathrm{I} \models (P \mid R) \mathscr{R} (Q \mid R)$. In fact, we can choose $\Delta = \Delta' = \mathrm{J} = \varnothing$ in the definition above. Now, if we show that $\mathscr{R}$ is an asynchronous bisimulation up to $\equiv$, then we have $\mathscr{R} \subseteq \approx^{@}$ by Proposition 7, and we conclude the current proposition by coinduction.

Then we prove that $\mathscr{R}$ is an asynchronous bisimulation up to $\equiv$. First, we need a preliminary result.

Fact 14 *If* $\mathrm{I} \models (\nu b{:}B)(\nu\Delta)(P \mid R) \mathrel{\mathscr{R}} (\nu b{:}B')(\nu\Delta')(Q \mid R)$, *then* $\mathrm{I}, b:\top \models (\nu\Delta)(P \mid R) \mathrel{\mathscr{R}} (\nu\Delta')(Q \mid R)$.

To show this fact, we assume $\mathrm{I} \models (\nu b{:}B)(\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu b{:}B')(\nu\Delta')(Q\,|\,R)$. By definition there exists J such that $\mathrm{I}\sqcap\mathrm{J} \models P \approx^{@} Q$ and $\mathrm{I}\sqcap\mathrm{J}\vdash R$, and there are Γ and Γ' such that $\Gamma, b:B, \Delta \vdash P$ and $\Gamma', b:B', \Delta' \vdash Q$ with $\Gamma, b:B, \Delta <: \mathrm{I}\sqcap\mathrm{J}$ and $\Gamma', b:B', \Delta' <: \mathrm{I}\sqcap\mathrm{J}$. In particular $\Gamma, b:B, \Delta <: \mathrm{I}\sqcap\mathrm{J}$ implies $b \in dom(\mathrm{I}\sqcap\mathrm{J})$. Hence $(\mathrm{I}, b:\top)\sqcap\mathrm{J} = \mathrm{I}\sqcap\mathrm{J}$. Thus $\mathrm{I}, b:\top \models (\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu\Delta')(Q\,|\,R)$ by definition.

Now we assume that $\mathrm{I} \models (\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu\Delta')(Q\,|\,R)$ and $\mathrm{I} \rhd (\nu\Delta)(P\,|\,R) \xrightarrow{\alpha} \mathrm{I}' \rhd S$. Then we find a corresponding transition for $\mathrm{I} \rhd (\nu\Delta')(Q\,|\,R)$. We proceed by induction on the derivation of the action α, by checking the last rule applied.

(G-OPEN) The hypothesis says that $\mathrm{I} \models (\nu b{:}B)(\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu b{:}B')(\nu\Delta')(Q\,|\,R)$. The transition is $\mathrm{I} \rhd (\nu b{:}B)(\nu\Delta)(P\,|\,R) \xrightarrow{(b)\alpha} \mathrm{I}' \rhd S$, and it was derived from $\mathrm{I}, b:\top \rhd (\nu\Delta)(P\,|\,R) \xrightarrow{\alpha} \mathrm{I}' \rhd S$. Fact 14 says that $\mathrm{I}, b:\top \models (\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu\Delta')(Q\,|\,R)$. Thus, by induction hypothesis there exists a corresponding transition $\mathrm{I}, b:\top \rhd (\nu\Delta)(Q'\,|\,R) \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd S'$ with $\mathrm{I}' \models S\ \mathscr{R}\ S'$ (note that G-OPEN involves only output actions). By (G-OPEN) we conclude $\mathrm{I} \rhd (\nu b{:}B')(\nu\Delta)(Q\,|\,R) \stackrel{(b)\alpha}{\Longrightarrow} \mathrm{I}' \rhd S'$.

(G-WEAK) The proof is similar to the case (G-OPEN).

(G-RES) We assume that $\mathrm{I} \models (\nu a:A)(\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu a:A')(\nu\Delta')(Q\,|\,R)$, and consider the transition $\mathrm{I} \rhd (\nu a:A)(\nu\Delta)(P\,|\,R) \xrightarrow{\alpha} \mathrm{I}' \rhd (\nu a:A)S$, which is derived from $\mathrm{I}, a:\top \rhd (\nu\Delta)(P\,|\,R) \xrightarrow{\alpha} \mathrm{I}', a:\top \rhd S$. Thanks to Fact 14, we have $\mathrm{I}, a:\top \models (\nu\Delta)(P\,|\,R)\ \mathscr{R}\ (\nu\Delta')(Q\,|\,R)$. Thus, by the induction hypothesis, there exists a matching transition. This means that either *(i)* $\mathrm{I}, a:\top \rhd (\nu\Delta')(Q\,|\,R) \stackrel{\alpha}{\Longrightarrow} \mathrm{I}', a:\top \rhd S'$ with $\mathrm{I}', a:\top \models S\,\mathscr{R}\,S'$ (in case α is a silent or output action), or *(ii)* $\mathrm{I}, a:\top \rhd (\nu\Delta')(Q\,|\,R) \Longrightarrow \mathrm{I}, a:\top \rhd S'$ with $\mathrm{I}', a:\top \models S\ \mathscr{R}\ (S'\,|\,\overline{b}\langle\tilde{\nu}@\tilde{B}\rangle)$, in case α is an input action. In case *(i)* we find the transition $\mathrm{I} \rhd (\nu a:A')(\nu\Delta')(Q\,|\,R) \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd (\nu a:A')S'$. In case *(ii)* we find the transition $\mathrm{I}' \rhd (\nu a:A')(\nu\Delta')(Q\,|\,R) \Longrightarrow \mathrm{I}', \rhd (\nu a:A')S'$. To conclude the reasoning, we need to show $\mathrm{I}' \models (\nu a:A)S\ \mathscr{R}\ (\nu a:A')S'$ and $\mathrm{I}' \models (\nu a:A)S\ \mathscr{R}\ (\overline{b}\langle\tilde{\nu}@\tilde{B}\rangle\,|\,(\nu a:A')S')$. The former relation is a consequence of $\mathrm{I}', a:\top \models S\ \mathscr{R}\ S'$ and the of generalization of Proposition 10 used in (9). Similarly for the latter one.

(G-PAR) Then $\mathrm{I} \models (P\,|\,R)\ \mathscr{R}\ (Q\,|\,R)$, and there are two sub-cases depending on whether the action comes from P or from R. *(i)* If P moves, then the transition is $\mathrm{I} \rhd (P\,|\,R) \xrightarrow{\alpha} \mathrm{I}' \rhd (P'\,|\,R)$ and it is deduced from $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. We know that $\mathrm{I} \models P \approx^{@} Q$ by hypothesis. Then, in case α is a silent or output action there exists Q' such that $\mathrm{I} \rhd Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q'$ and $\mathrm{I}' \models P' \approx^{@} Q'$, and we obtain $\mathrm{I} \rhd Q\,|\,R \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q'\,|\,R$. Since $\mathrm{I}' \models P' \approx^{@} Q'$ and $\mathrm{I} \vdash R$ we conclude $\mathrm{I}' \models (P'\,|\,R)\ \mathscr{R}\ (Q'\,|\,R)$ by definition. Otherwise, if $\alpha = (\Delta)a(\tilde{\nu}@\tilde{A})$ we have two more possibilities: either we reason as above, or we find Q' such that $\mathrm{I}, \Delta \rhd Q \Longrightarrow \mathrm{I}' \rhd Q'$ and $\mathrm{I}' \models P' \approx^{@} (Q'\,|\,\overline{a}\langle\tilde{\nu}@A\rangle)$. In the former case we have $\mathrm{I}, \Delta \rhd Q\,|\,R \Longrightarrow \mathrm{I}' \rhd Q'\,|\,R$. Now, from $\mathrm{I}' \models P' \approx^{@} (Q'\,|\,\overline{a}\langle\tilde{\nu}@A\rangle)$ and $\mathrm{I} \vdash R$ we conclude $\mathrm{I}' \models (P'\,|\,R)\ \mathscr{R} \equiv\ (Q'\,|\,R\,|\,\overline{a}\langle\tilde{\nu}@A\rangle$ by definition. *(ii)* If R moves, then the transition is $\mathrm{I} \rhd (P\,|\,R) \xrightarrow{\alpha} \mathrm{I}' \rhd P\,|\,R'$ and it is deduced from $\mathrm{I} \rhd R \xrightarrow{\alpha} \mathrm{I}' \rhd R'$. By hypothesis we have $\mathrm{I} \models P \approx^{@} Q$. By Lemma 27, we know that $\mathrm{I}' = \mathrm{I}, \mathrm{I}''$, for a suitable I''. Thus $\mathrm{I}' \models P \approx Q$ by Proposition 9. Then we can proceed as in the previous case.

(G-REDUCE) Then the transition has the form $\mathrm{I} \rhd (\nu\Delta)(P\,|\,R) \xrightarrow{\tau} \mathrm{I} \rhd (\nu\Delta)T$ and it is deduced from $(\nu\Delta)(P\,|\,R) \xrightarrow{\tau} (\nu\Delta)T$. Moreover the former transition has to be derived from $P\,|\,R \xrightarrow{\tau} T$. This transition may be made autonomously either by P or by R: these two cases are similar to the two sub-cases we just worked out above. In particular, if $P \xrightarrow{\tau} P'$, then $T = P'\,|\,R$. By Lemma 3, we also have $\mathrm{I}\sqcap\mathrm{J} \rhd P \xrightarrow{\tau} \mathrm{I}\sqcap\mathrm{J} \rhd P'$. Then we find a matching transition by observing that $\mathrm{I}\sqcap\mathrm{J} \models P \approx^{@} Q$ implies $\mathrm{I}\sqcap\mathrm{J} \rhd Q \Longrightarrow \mathrm{I}\sqcap\mathrm{J} \rhd Q'$ and $\mathrm{I}\sqcap\mathrm{J} \models P' \approx^{@} Q'$. Again by Lemma 3, we also have $Q \Longrightarrow Q'$, and we can construct the weak transition $\mathrm{I} \rhd (\nu\Delta)(Q\,|\,R) \Longrightarrow \mathrm{I} \rhd (\nu\Delta)(P'\,|\,R)$ which proves this case. The case that R moves is similar. For (G-REDUCE), two further sub-cases arise when the transition is the result of the interaction between the two processes:

(i) $P \xrightarrow{(\Delta_P)\overline{a}\langle\tilde{\nu}@\tilde{B}\rangle} P'$ and $R \xrightarrow{a(\tilde{\nu}@\tilde{B}')} R'$ with $\tilde{B} <: \tilde{B}'$;

(ii) $P \xrightarrow{a(\tilde{\nu}@\tilde{B}')} P'$ and $R \xrightarrow{(\Delta_R)\overline{a}\langle\tilde{\nu}@\tilde{B}\rangle} R'$ with $\tilde{B} <: \tilde{B}'$.

In case *(i)* the transition is $\mathrm{I} \rhd (\nu\Delta)(P\,|\,R) \xrightarrow{\tau} \mathrm{I} \rhd (\nu\Delta, \Delta_P)(P'\,|\,R')$. Then we need to find a matching transition for $\mathrm{I} \rhd (\nu\Delta')(Q\,|\,R)$. Since $R \xrightarrow{a(\tilde{\nu}@\tilde{B}')} R'$ and $\mathrm{I}\sqcap\mathrm{J} \vdash R$, we have $\mathrm{I}\sqcap\mathrm{J}\sqcap\tilde{\nu}:\tilde{B}' \vdash R'$ by Subject

Reduction. Since $P \xrightarrow{(\Delta_P)\overline{a}\langle\tilde{v}@\tilde{B}\rangle} P'$, we have $\mathrm{I} \sqcap \mathrm{J} \rhd P \xrightarrow{(|\Delta_P|)\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \rhd P'$ by Lemma 3, since $\mathrm{I} \sqcap \mathrm{J} \vdash R$ implies $(\mathrm{I} \sqcap \mathrm{J})^{\mathrm{r}}(a)\downarrow$. As $\mathrm{I} \sqcap \mathrm{J} \models P \approx^{@} Q$ by hypothesis, we have $\mathrm{I} \sqcap \mathrm{J} \rhd Q \xRightarrow{(|\Delta_Q|)\overline{a}\langle\tilde{v}@\tilde{B}\rangle} \mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \rhd Q'$ with $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \models P' \approx^{@} Q'$. From $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B}' \vdash R'$, we have $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \vdash R'$ by weakening. Furthermore, by Lemma 3, we know that there are $\tilde{A} <: \tilde{B}$ and Δ_Q such that $Q \xRightarrow{(\Delta_Q)\overline{a}\langle\tilde{v}@\tilde{A}\rangle} Q'$. This fact, along with $R \xrightarrow{a(\tilde{v}@\tilde{B}')} R'$ and $\tilde{B} <: \tilde{B}'$, implies that there exists a weak transition $Q \,|\, R \Longrightarrow (\nu\Delta_Q)(Q' \,|\, R')$, hence $\mathrm{I} \rhd (\nu\Delta')(Q \,|\, R) \Longrightarrow \mathrm{I} \rhd (\nu\Delta, \Delta_Q)(Q' \,|\, R')$. We have now showed that $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \models P' \approx^{@} Q'$ and $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B} \vdash R'$. Then it remains to show that Γ, Δ, Δ_P and $\Gamma, \Delta', \Delta_Q$ are compatible with $\mathrm{I} \sqcap \mathrm{J} \sqcap \tilde{v} : \tilde{B}$. This is a consequence of Lemma 28, and the hypothesis that both Γ, Δ and Γ, Δ' are compatible with $\mathrm{I} \sqcap \mathrm{J}$.

In case *(ii)* the transition is $\mathrm{I} \rhd (\nu\Delta)(P \,|\, R) \xrightarrow{\tau} \mathrm{I} \rhd (\nu\Delta, \Delta_R)(P' \,|\, R')$. Then we need to find a matching transition from $\mathrm{I} \rhd (\nu\Delta')(Q \,|\, R)$. Since $R \xrightarrow{(\Delta_R)\overline{a}\langle\tilde{v}@\tilde{B}\rangle} R'$ and $\mathrm{I} \sqcap \mathrm{J} \vdash R$ we have $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash R'$ and $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash v : B$ by Subject Reduction. Since $B <: B'$ we obtain $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash \tilde{v} : \tilde{B}'$. Moreover $P \xrightarrow{a(\tilde{v}@\tilde{B}')} P'$, hence $\mathrm{I} \sqcap \mathrm{J} \rhd P \xrightarrow{(\Delta_R)a(\tilde{v}@\tilde{B}')} \mathrm{I} \sqcap \mathrm{J}, \Delta_R \rhd P'$ by Lemma 3. Since $\mathrm{I} \sqcap \mathrm{J} \models P \approx^{@} Q$ hypothesis, we have:

(a) either $\mathrm{I} \sqcap \mathrm{J} \rhd Q \xRightarrow{(\Delta_R)a(\tilde{v}@\tilde{B}')} \mathrm{I} \sqcap \mathrm{J}, \Delta_R \rhd Q'$, with $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models P' \approx^{@} Q'$,
(b) or $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \rhd Q \Longrightarrow \mathrm{I} \sqcap \mathrm{J}, \Delta_R \rhd Q'$, with $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models P' \approx^{@} (Q' \,|\, \overline{a}\langle\tilde{v}@\tilde{B}''\rangle)$, for all $\tilde{B}'' <: \tilde{B}'$ such that $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models \tilde{v} : \tilde{B}''$.

In case *(a)*, the the desired matching transition can be found by reasoning as in *(i)* above. We consider case *(b)*. Since $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \rhd Q \Longrightarrow \mathrm{I} \sqcap \mathrm{J}, \Delta_R \rhd Q'$, we obtain $Q \Longrightarrow Q'$ by Lemma 3. Now, an analysis of transition $R \xrightarrow{(\Delta_R)\overline{a}\langle\tilde{v}@\tilde{B}\rangle} R'$ shows that $R \equiv (\nu\Delta_R)(R' \,|\, \overline{a}\langle\tilde{v}@\tilde{B}\rangle)$. Moreover $Q \Longrightarrow Q'$, hence $Q \,|\, R \Longrightarrow \equiv (\nu\Delta_R)(Q' \,|\, \overline{a}\langle\tilde{v}\rangle\tilde{B} \,|\, R')$. Thus we can construct the typed action $\mathrm{I} \rhd (\nu\Delta')(Q \,|\, R) \Longrightarrow \mathrm{I} \rhd (\nu\Delta', \Delta_R)(Q' \,|\, \overline{a}\langle\tilde{v}@\tilde{B}\rangle \,|\, R')$. We observe that $\mathrm{I} \sqcap \mathrm{J} \vdash R$ implies $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash R'$ and $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \vdash \tilde{v} : \tilde{B}$. Hence $\mathrm{I} \sqcap \mathrm{J}, \Delta_R \models P' \approx^{@} (Q' \,|\, \overline{a}\langle\tilde{v}@\tilde{B}\rangle)$. Now we only need to show that Γ, Δ, Δ_R and $\Gamma, \Delta', \Delta_R$ are compatible with $\mathrm{I} \sqcap \mathrm{J}, \Delta_R$. This is consequence of the hypothesis that Γ, Δ and Γ, Δ' are compatible with $\mathrm{I} \sqcap \mathrm{J}$. □

To prove that the asynchronous bisimulation up to $\equiv$ is a sound technique we first need to show that $\equiv$ commutes with the typed dynamics of Table 2.

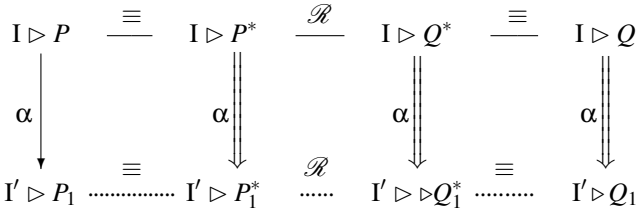
Proposition 17. *If $P \equiv Q$ and $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$ then there exists Q' s.t. $\mathrm{I} \rhd Q \xrightarrow{\alpha} \mathrm{I}' \rhd Q'$ and $P' \equiv Q'$.*

Proof. Let $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I}' \rhd P'$. We analyse the action α and, thanks to Lemma 3, we find an appropriate untyped action α' such that $P \xrightarrow{\alpha'} P'$. We then show that $P \equiv Q$ and $P \xrightarrow{\alpha'} P'$ implies that there is Q' s.t. $Q \xrightarrow{\alpha'} Q'$ and $P' \equiv Q'$. The proof of this result is available in [20]. Finally, by applying Lemma 3 to $Q \xrightarrow{\alpha'} Q'$ we conclude $\mathrm{I} \rhd Q \xrightarrow{\alpha} \mathrm{I}' \rhd Q'$.

Corollary 10. *If $P \equiv Q$ and $\mathrm{I} \rhd P \xRightarrow{\alpha} \mathrm{I}' \rhd P'$ then there exists Q' s.t. $\mathrm{I} \rhd Q \xRightarrow{\alpha} \mathrm{I}' \rhd Q'$ and $P' \equiv Q'$.*

Proof of Proof of Proposition 7 We show that $\equiv\mathscr{R}\equiv$ is an asynchronous bisimulation. We outline the proof with diagrams. By "$\mathrm{I} \rhd P \overset{\equiv}{—} \mathrm{I} \rhd Q$" we mean "$P \equiv Q$" and by "$\mathrm{I} \rhd P \overset{\mathscr{R}}{—} \mathrm{I} \rhd Q$" we mean "$\mathrm{I} \models P \,\mathscr{R}\, Q$". Existential relations are depicted by dotted lines. Then we assume that $\mathrm{I} \rhd P \xrightarrow{\alpha} \mathrm{I} \rhd P_1$.

In case α is an output action, we have:



Proposition 17 provides the first inference on the left. Moreover, since $\mathrm{I} \models P^* \mathscr{R} Q^*$ and $\mathrm{I} \rhd P^* \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd P_1^*$, and since α is an output action, then there exists Q_1^* such that $\mathrm{I} \rhd Q^* \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q_1^*$ and $\mathrm{I}' \models P_1^* \mathscr{R} Q_1^*$. By Corollary 10 there exists Q' such that $\mathrm{I} \rhd Q \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd Q_1$ with $Q_1 \equiv Q_1^*$.

In case α is an input action, then if $\mathrm{I} \rhd Q^*$ reacts with α the situation is analogous to the previous one. If $\mathrm{I} \rhd Q^*$ reacts with a silent move we have:

$$\begin{array}{ccccccccc}
\mathrm{I} \rhd P & \equiv & \mathrm{I} \rhd P^* & \mathscr{R} & & & \mathrm{I} \rhd Q^* & \equiv & \mathrm{I} \rhd Q \\
\alpha \downarrow & & \alpha \Downarrow & & & & \Downarrow & & \Downarrow \\
\mathrm{I}' \rhd P_1 & \equiv & \mathrm{I}' \rhd P_1^* & \mathscr{R} & \mathrm{I}' \rhd Q_1^* \,|\, \overline{\alpha} & & \mathrm{I} \rhd Q_1^* & \equiv & \mathrm{I} \rhd Q_1
\end{array}$$

For simplicity of notation, $\overline{\alpha}$ represents the asynchronous output that corresponds to the input action α in Definition 6. Then, since $\mathrm{I} \models P^* \mathscr{R} Q^*$ and $\mathrm{I} \rhd P^* \stackrel{\alpha}{\Longrightarrow} \mathrm{I}' \rhd P_1^*$ there exists Q_1^* such that $\mathrm{I} \rhd Q^* \Longrightarrow \mathrm{I} \rhd Q_1^*$ and $\mathrm{I}' \models P_1^* \mathscr{R} Q_1^* \,|\, \overline{\alpha}$. As done before, from $Q^* \equiv Q$ we conclude that there is Q' such that $\mathrm{I} \rhd Q \Longrightarrow \mathrm{I} \rhd Q_1$ and $Q_1^* \equiv Q_1$.

The case $\alpha = \tau$ is analogous. □

Proof of Proposition 13 Since it will be useful in the proof, we first note that if $\mathrm{I} \rhd P$ is a configuration and $e \notin \mathrm{I}$ then $e \notin fn(P)$. Now, the testing processes C_α^{I} is defined as follows:

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

Where the process $Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}}$ is defined as follows. Let $\tilde{b} = b_1, \ldots, b_n$, and let $\tilde{j}, \tilde{k}$ the partition of $1, \ldots, n$ such that $b_j \notin \tilde{c}$ for every j in $\tilde{j}$, and $b_k \in \tilde{c}$ for every k in $\tilde{k}$. The process $Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}}$ provides the tuple $\tilde{x}$ of free variables and it is defined in order to ensure the following property:

$$Q^{\mathrm{I}}_{(\tilde{c})\tilde{b}}\{\tilde{v}/\tilde{x}\} \Longrightarrow \overline{e}\langle \mathrm{I} \sqcap \tilde{b}:\tilde{B}\rangle \quad \text{iff} \quad \begin{array}{l} v_j = b_j \text{ for every } j \text{ in } \tilde{j}, \\ v_k \notin dom(\mathrm{I}) \text{ for every } k \in \tilde{k}, \text{ and} \\ \text{given } k,h \text{ in } \tilde{k} \text{ it holds } v_k = v_h \text{ iff } b_k = b_h \end{array}$$

We leave to the reader the detailed definition of a process with this properties. We just hint an example. Let $\alpha = (c)\overline{a}\langle b@B', c@B'', c@B'''\rangle$ and $\tilde{B} = (B', B'', B''')$. Then $C_\alpha^{\mathrm{I}} = a(\tilde{x}@\tilde{B}).Q^{\mathrm{I}}_{(c)b,c,c}$, where the process $Q^{\mathrm{I}}_{(c)(b,c,c)}$ can be defined as $[x_1 = b]\,[x_2 \notin dom(\mathrm{I})]\,[x_3 \notin dom(\mathrm{I})]\,[x_3 = x_2]\,\overline{e}\langle \mathrm{I} \sqcap \tilde{x}:\tilde{B}\rangle$. The test $x \notin dom(\mathrm{I})$ is coded by several nested conditionals to check $x \neq d$ for every $d \in dom(\mathrm{I})$.

Proof of (1.) We first consider the input action: when α is $(\tilde{c}:\tilde{C})a(\tilde{b}@\tilde{B})$. Since $\mathrm{I} \rhd P \stackrel{\alpha}{\longrightarrow} \mathrm{I}' \rhd P'$, Lemma 3 says that $P \stackrel{a(\tilde{b}@\tilde{D})}{\longrightarrow} P'$ with $\tilde{B} <: \tilde{D}$, and Proposition 12 says that $\mathrm{I}' = \mathrm{I}, \tilde{c}:\tilde{C}$ that implies $\tilde{c} \cap fn(P) = \emptyset$. Hence $C_\alpha^{\mathrm{I}} \,|\, P \stackrel{\tau}{\longrightarrow} (\nu\tilde{c}:\tilde{C})(P' \,|\, \overline{e}\langle \mathrm{I}'\rangle)$ by (PI-COM@). Then we consider the output action: when α is $(\tilde{c})\overline{a}\langle \tilde{b}@\tilde{B}\rangle$. Since $\mathrm{I} \rhd P \stackrel{\alpha}{\longrightarrow} \mathrm{I}' \rhd P'$, again Lemma 3 says that $P \stackrel{(\tilde{c}:\tilde{C})\overline{a}\langle \tilde{b}@\tilde{D}\rangle}{\longrightarrow} P'$ with $\tilde{D} <: \tilde{B}$, and by Proposition 12 we have $\mathrm{I}' = \mathrm{I} \sqcap \tilde{b}:\tilde{B}$. Hence $C_\alpha^{\mathrm{I}} \,|\, P \stackrel{\tau}{\longrightarrow} (\nu\tilde{c}:\tilde{C})(P' \,|\, Q^{\mathrm{I}}_{\tilde{b}}\{\tilde{b}/\tilde{x}\})$ by (PI-COM@) rule, and the claim is a consequence of the properties for $Q^{\mathrm{I}}_{\tilde{b}}$.

Proof of (2.) We first consider the input action: when α is $(\tilde{c})\overline{a}\langle \tilde{b}@\tilde{B}\rangle$. Since $e \notin fn(P)$, the transition $C_\alpha^{\mathrm{I}} \,|\, P \Longrightarrow (\nu\Delta)P' \,|\, \overline{e}\langle \tilde{v}@\tilde{T}\rangle$ must involve both P and C_α^{I} and it must be derived from

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

with $\tilde{D} <: \tilde{B}$ and $\Delta = (\tilde{c}:\tilde{C})$ and $\tilde{v}@\tilde{T} = \mathrm{I} \sqcap \tilde{b}:\tilde{B}$. Proposition 12 says that $\mathrm{I} \sqcap \tilde{b}:\tilde{B} = \mathrm{I}\,\mathsf{after}\,\alpha$. Hence we conclude $\mathrm{I} \rhd P \stackrel{(\tilde{c})\overline{a}\langle \tilde{b}@\tilde{B}\rangle}{\Longrightarrow} (\mathrm{I}\,\mathsf{after}\,\alpha) \rhd P'$ by Lemma 3. Then we consider the output action: when α

Table 11 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):
$H \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)\,(h(\underline{x}).P \mid \textsc{Reply}_t(a,h)) \mid \mathtt{TABLE}(t,\mathscr{T})$
$H' \equiv (\nu h{:}\mathsf{rw}\langle\mathbb{T}_A\rangle)$
$\big(h(\underline{x}).P \mid (\nu r{:}\mathsf{rw}\langle\top,\mathbb{S}\rangle)\,\big(\overline{r}\langle\mathtt{t},\underline{k}\rangle \mid r(x,\underline{y})[x{=}\mathtt{t}]\overline{h}\langle\underline{y}\rangle;(\overline{h}\langle\underline{y}\rangle \mid \textsc{Chan}(y))\big)\big) \mid \mathtt{TABLE}(t,\mathscr{T})$

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

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

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

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

is $(\tilde{c}:\tilde{C})\overline{a}\langle\tilde{b}@\tilde{B}\rangle$. Due to the structure of C^{I}_α and the fact that $C^{\mathrm{I}}_\alpha \mid P \Longrightarrow (\nu\Delta)\ P' \mid \overline{e}\langle\tilde{v}@\tilde{T}\rangle$, there are two possible cases:

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

The second item provides the thesis directly. For the first item, the fact that $\mathrm{I},e:\mathsf{rw} \vdash C^{\mathrm{I}}_\alpha$ and the rules (T-New) and (T-Out@) says that $\mathrm{I},\tilde{c}:\tilde{C} \vdash \tilde{b}:\tilde{B}$ and $\mathrm{I}^{\mathsf{w}}(a)\downarrow$. Now Proposition 12 says that $\mathrm{I} \sqcap \tilde{c}:\tilde{C} = \mathrm{I}\,\mathsf{after}\,\alpha$. Hence we conclude $\mathrm{I} \rhd P \xRightarrow{\alpha} (\mathrm{I}\,\mathsf{after}\,\alpha) \rhd P'$ by Lemma 3. □

B Proofs: fully abstract encoding

It is straightforward to see how the congruence relation provides a characterization of administrative reductions. The details are outlined in tables 11 and 12. In particular, Table 11 depicts the administrative reductions due to the interaction with the proxy server, and Table 12 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.

Lemma 29 (Administrative Reduction – Characterization). *The reduction $P \xrightarrow{\tau} P'$ is* I*-administrative when $\mathrm{I} \rhd P\downarrow_a$ iff $\mathrm{I} \rhd P'\downarrow_a$ and $P = C[H]$, $P' = C[H']$ with H,H' satisfying one of the eight cases depicted in tables 11 and 12.*

Table 12 Administrative Reductions – Interaction Client/Channel Manager.

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

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

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

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

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

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

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

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

Proof of Lemma 20. We take the type indexed relation $\mathscr{R} \stackrel{def}{=} (\mathscr{R}_1 \cup \mathscr{R}_2 \cup \equiv)$, where $\mathrm{I} \models C[P]\mathscr{R}_1 C[Q]$ and $\mathrm{I} \models C[Q]\mathscr{R}_2 C[P]$ whenever $P \xrightarrow{A_\mathrm{I}} Q$ and $C[-]$ is an evaluating context defined as $C[-] \stackrel{def}{=} (\nu\tilde{b}{:}\tilde{T})(- \mid R)$, with $\mathrm{I} \vdash R$. Then we show that $\mathscr{R}$ is symmetric, barb preserving, reduction closed and contextual.
Symmetry. It holds by definition
Barb Preservation. Assume that $\mathrm{I} \models C[P]\,\mathscr{R}_1\,C[Q]$. If $\mathrm{I} \rhd C[] \downarrow_a$ then $\mathrm{I} \rhd C[P] \downarrow_a$ if and only if $\mathrm{I} \rhd C[Q] \downarrow_a$. Otherwise $\mathrm{I} \rhd P \downarrow_a$ if and only if $\mathrm{I} \rhd Q \downarrow_a$ by Definition 9, and again $\mathrm{I} \rhd C[P] \downarrow_a$ if and only if $\mathrm{I} \rhd C[Q] \downarrow_a$. The case for $\mathscr{R}_2$ is analogous and the case for $\equiv$ follows from Lemma 18.
Reduction Closure. The relation $\mathscr{R}_1$ is the only significative one. Then we assume that $\mathrm{I} \models C[P]\,\mathscr{R}_1\,C[Q]$, meaning that $P \xrightarrow{A_\mathrm{I}} Q$. We need to prove that *(a)* if $C[P] \xrightarrow{A_\mathrm{I}} H$ then $C[Q] \stackrel{A_\mathrm{I}}{\Longrightarrow} K$ with $H\,\mathscr{R}\,K$, and that *(b)* if $C[P] \xrightarrow{\tau_\mathrm{I}} H$ then $C[Q] \stackrel{A_\mathrm{I}}{\Longrightarrow} \xrightarrow{\tau_\mathrm{I}} \stackrel{A_\mathrm{I}}{\Longrightarrow} K$ with $H\,\mathscr{R}\,K$. The two cases are essentially treated in a similar way, and they rely on Lemma 19 as administrative reductions are in general non-overlapping. In the following we discuss two significative cases

In case $C[P] \xrightarrow{A_\mathrm{I}} H$ and this reduction has been generated by item 5 in Table 12. If the administrative reduction $P \xrightarrow{A_\mathrm{I}} Q$ has been inferred from the items 1–4 of Table 11 or 6–8 of Table 12, then the reductions $C[P] \xrightarrow{A_\mathrm{I}} C[Q]$ and $C[P] \xrightarrow{A_\mathrm{I}} H$ are non overlapping, hence we conclude that there exists K such that $C[Q] \xrightarrow{A_\mathrm{I}} \equiv K$ and $H \xrightarrow{A_\mathrm{I}} \equiv K$ by Lemma 19. On the other hand, if the administrative reduction $P \xrightarrow{A_\mathrm{I}} Q$ has been inferred from the item 5 of Table 12, then it must be a synchronization on a channel $u_{\mathsf{r}@A}$. Also $C[P] \xrightarrow{A_\mathrm{I}} H$ 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 $C[Q] \stackrel{A_\mathrm{I}}{\Longrightarrow} H$. On the other hand, if there are two synchronizations

on two different outputs on $u_{\mathsf{r}@A}$ it must be the case that

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

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

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

In case $C[P] \xrightarrow{A_{\mathrm{I}}} H$ according to item 6 in Table 12. If the administrative reduction $P \xrightarrow{A_{\mathrm{I}}} Q$ has been inferred from the items 1–4 of Table 11 and 5, 7, 8 of Table 12 then we conclude as done above thanks to Lemma 19. On the other hand if the administrative reduction $P \xrightarrow{A_{\mathrm{I}}} Q$ has been inferred from the item 6 of Table 12, then it must be a synchronization on a channel $u_{\mathsf{w}@A}$. Also $C[P] \xrightarrow{A_{\mathrm{I}}} H$ is a synchronization on a channel $u'_{\mathsf{w}@A}$. Again, if $H = C[Q]$ then we conclude that $C[Q] \xRightarrow{A_{\mathrm{I}}} H$. 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 $\prod_{C<:A} !\,u_{\mathsf{w}@C}(\underline{z}).\textsc{Test}_l\langle u_{\mathsf{w}@C}(\underline{z}).\bar{h}\langle\underline{z}\rangle\rangle$ in $R_A\langle h,l\rangle$.
Contextuality. It is a direct consequence of the definition of $\mathscr{R}_1$ and $\mathscr{R}_2$, as we have introduced the evaluating context $C[-]$. □

Proof of Lemma 21 By induction on the derivation of $P \xrightarrow{\tau} P'$ in API@, we see that there exists a reduction $([\![P]\!])^*_\Gamma \xRightarrow{A_{\mathrm{J}}} H$ which follows the steps in (the same order of) Table 11 and Table 12 with $H \xrightarrow{\tau_{\mathrm{J}}} K \equiv (\nu\underline{k}{:}\mathbb{S})\,(([\![P']\!])_\Gamma \mid (\nu t{:}\mathtt{TBL}[\top,\mathbb{S}])(\textsc{Server}(t) \mid \mathtt{TABLE}(t,[(n,\underline{k})])))$ and $\mathrm{J} \models K \approx_A ([\![P']\!])^*_\Gamma$ as the association table does not influence the behavior of the encoded process. □

Proof of Lemma 22 Let $([\![P]\!])^*_\Gamma \xRightarrow{A_{\mathrm{J}}} H \xrightarrow{\tau_{\mathrm{J}}} K$ then in $\xRightarrow{A_{\mathrm{J}}}$ there is a synchronization on a channel p_A that start the communication protocol that leads (exactly) to the desired committing action $\xrightarrow{\tau_{\mathrm{I}}}$. 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 11 and 12. Thus obtain $([\![P]\!])^*_\Gamma \xRightarrow{\lceil A\rceil_{\mathrm{J}}} H' \xrightarrow{\tau_{\mathrm{J}}} K'$. Now, all the administrative steps in $([\![P]\!])^*_\Gamma \xRightarrow{A_{\mathrm{J}}} H$ that are not in $([\![P]\!])^*_\Gamma \xRightarrow{\lceil A\rceil_{\mathrm{J}}} H'$ can be performed starting by K'. Hence $([\![P]\!])^*_\Gamma \xRightarrow{\lceil A\rceil_{\mathrm{J}}} H' \xrightarrow{\tau_{\mathrm{J}}} K' \xRightarrow{A_{\mathrm{I}}} K$, and so $\mathrm{I} \models K' \approx_A K$ thanks to Lemma 20. □