Noname manuscript No.
(will be inserted by the editor)

A semantic deconstruction of session types

Massimo Bartoletti · Alceste Scalas · Roberto Zunino

the date of receipt and acceptance should be inserted later

Keywords concurrency, session types, contracts

Abstract *Session types* are a well-established typing discipline for verifying that a *process* correctly implements a structured protocol (i.e., a *session*). Their main property is that typed processes "never go wrong": they interact correctly, without communication mismatches.

In this paper, we revisit the theory of session types starting from their semantic foundations, with the goal of capturing and generalising their underlying notions of "correct interaction" and "safe process substitution".

We develop a unified semantic framework based on *labelled transition systems*, encompassing session types and more general behaviours, endowed with both *synchronous* and *asynchronous* (i.e., buffered) semantics.

In this framework, we formalise the notion of "correct interaction" by introducing the *I/O compliance* and *safety* relations: I/O compliance refines the notion of *progress* between behaviours, while safety is inspired by Communicating Finite State Machines. We prove that I/O compliance and safety coincide on session types (both synchronous and asynchronous), and that they are preserved when passing from synchronous to asynchronous semantics.

We then study two preorders between behaviours, aimed at safe process substitution: the standard semantic subtyping, and the novel *I/O simulation*. The latter generalises the usual syntax-directed notions of typing and subtyping on session types, and it coincides with semantic subtyping on synchronous session types. We show that such preorders are preserved (under suitable conditions) when passing from synchronous to asynchronous semantics.

Finally, we exploit the semantic properties of I/O simulation and compliance to construct a syntax-driven type system, combining typical session typing rules with a more flexible handling of buffered communication. Notably, we derive the type system correctness without an explicit subject reduction result, by relying instead on the general properties of I/O simulation and compliance.

Massimo Bartoletti
Università di Cagliari
E-mail: bart@unica.it

Alceste Scalas
Università di Cagliari and Imperial College London
E-mail: alceste.scalas@imperial.ac.uk

Roberto Zunino
Università di Trento
E-mail: roberto.zunino@unitn.it

1 Introduction

Session typing is a well-established approach to the design of concurrent applications [50,51,53,72]. In a nutshell, a (binary) session type specifies a communication protocol, and dictates how two components are expected to interact through a bi-directional communication channel. The structure of the resulting interaction, called *session*, includes *choices* and *recursion*. In order to guarantee correct interaction, each endpoint of the channel is associated with a session type, so that the two endpoint types are *compliant*. Intuitively, this means that whenever one endpoint sends some data, the other endpoint is able to receive it — and conversely, whenever one endpoint expects to receive some data, the other one is willing to send something. While a session type specifies the interaction protocol between two components, the actual implementation of a component is represented by a *process*, usually formalised in a dialect of the π-calculus [60,71].

The correspondence between a session type T and a process P can be statically established by a type system,

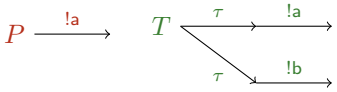
i.e. a set of syntax-driven rules that relate processes with types. Intuitively, the process P is typeable when it is possible to construct a proof of $\vdash P : T$, for some T, using the typing rules. If P is typeable, then its runtime interactions will respect those statically described by T. As a consequence, if both processes at the endpoints of a session type-check, and their types are compliant, then the interaction between the processes is deadlock-free.

Another relation typically studied in the session types community is that of *subtyping*: the *desideratum* is that if a session type T is subtype of U, and we have two processes P, Q such that $\vdash P : T$ and $\vdash Q : U$, then P can safely "replace" Q: i.e., any process that interacts correctly with Q will also interact correctly with P. Similarly to the typing relation, also the subtyping relation is typically defined through syntax-driven rules.

Now, assume that both processes and types are endowed with an operational semantics, in the form of a Labelled Transition System (LTS), as usually done in concurrency theory [59]. In this perspective, it is tempting to interpret the *syntactic* typing/subtyping relations as *semantic* relations, that would only depend on the LTS of process P and the LTS of type T, disregarding their syntax. This would follow the tradition of observational equivalences and preorders between processes, initiated by the study on bisimulation [68,59], where the fundamental relations are given on arbitrary LTSs. This insight opens several research directions, that we discuss below.

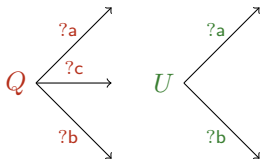
Question 1. *Can we reason on session-based interactions in a language-independent way?*

When $\vdash P : T$ holds, the relation between P and T may resemble a *simulation*: T simulates P iff, whenever $P \xrightarrow{\ell} P'$, then $T \xrightarrow{\ell} T'$, for some T' which still simulates P'. To elaborate further on this idea, consider a session type $T = \mathsf{!a} \oplus \mathsf{!b}$, which models an *internal* choice between two outputs $\mathsf{!a}$ and $\mathsf{!b}$. Further, consider the process $P = \mathsf{!a}$ which just wants to output $\mathsf{!a}$. Intuitively, the process P has type T, because any client who interacts correctly with T does so also with P. More concretely, assume that the processes P and T above are associated with the following LTSs:



We can observe that P is (weakly) simulated by T, in symbols $P \lessapprox T$, because each move of P is matched by a (weak) move of T. This may suggest that weak simulation $P \lessapprox T$ is a suitable semantic counterpart of the (syntactic) typing relation $\vdash P : T$.

To assess this semantic notion of typing, let us consider another example. Let $U = \mathsf{?a} \,\&\, \mathsf{?b}$ be the type modelling an *external* choice between two inputs $\mathsf{?a}$ and $\mathsf{?b}$, and let $Q = \mathsf{?a} + \mathsf{?b} + \mathsf{?c}$ (where $+$ is the standard CCS choice operator) be the process which allows for an additional input $\mathsf{?c}$. Intuitively, Q has type U: indeed, any client interacting correctly with U will send either $\mathsf{!a}$ of $\mathsf{!b}$, and so it will interact correctly also with Q, since the process is able to receive those messages. Assume that Q and U are associated with the LTSs:



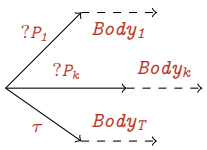
Differently from the previous example, we observe that Q is *not* weakly simulated by U (whereas the converse $U \lessapprox Q$ holds). This dismantles our idea of semantically interpreting the session typing relation as weak simulation. On the positive side, the two examples above seem to suggest that a semantical typing relation should treat input and output capabilities differently: it should be *covariant* w.r.t. outputs and *contravariant* w.r.t. inputs.

Question 2. *Can we devise new syntactic typing rules as instances of some general relation?*

A drawback of the syntax-driven approaches to session types is that they do not usually consider some common programming patterns for communication-oriented applications. For example, consider a server waiting some input which may not arrive on time. In Erlang [43], for instance, one can write:

```
receive  P1  -> Body1...
         Pk  -> Bodyk
after    10  -> BodyT
```

The `receive` statement is aborted if no messages matching $P_1, \ldots, P_k$ arrive within 10 milliseconds; in this case, $Body_T$ is executed. Such a program blurs the distinction between internal/external choices. Intuitively, its LTS has the form:



where the initial state has k input moves $?P_1, \ldots, ?P_k$, and an internal τ-move which abstracts the timeout. This pattern eludes the notion of structured programming at the roots of the session types approach [49,50]. However, the example above shows a use case that one would like to typecheck. This pattern cannot be usually

written in session calculi (like e.g. in many works extending [49,50]), since their grammar does not include τ-branches. Can a syntax-independent approach handle patterns like the one above?

Question 3. *Can we reason on synchronous and asynchronous interactions in a uniform semantic setting?*

Distributed components usually interact in an asynchronous fashion via *FIFO* buffers. Now, consider a type $T = \mathsf{!a}.\mathsf{?b}$, where $\mathsf{?b}$ is fired *after* $\mathsf{!a}$: the "natural" corresponding process is $P = \mathsf{!a}.\mathsf{?b}$, which performs the expected actions in the same order. However, in an *asynchronous* setting, the sequentiality of the resulting interactions cannot be enforced: i.e., if P fires $\mathsf{!a}$ asynchronously, then it will immediately start waiting for $\mathsf{?b}$, without knowing whether $\mathsf{!a}$ is still buffered, or was actually received by the other endpoint. Hence, one might argue that the parallel composition $P' = \mathsf{?b} \mid \mathsf{!a}$ also conforms to T in the presence of asynchrony.

Although many works on asynchronous session types relate P with T, they do not usually relate P' with T. How can we formalise the intuition that both P and P' correctly implement T, in the asynchronous setting?

Question 4. *Is there a language-independent notion of "correct" interaction?*

In the discussion so far, we relied on the intuition of a "correct" interaction between processes. This notion can be defined in different ways, depending on the calculus in use, and what is considered to be an error. In the realm of synchronous session types a typical notion of correct interaction is *progress*, which holds when two processes keep interacting until they terminatation [4,6]. Instead, in the realm of asynchronous interactions, like e.g. in Communicating Finite State Machines (CFSM) [19], progress is too large, since it relates e.g. two machines which just send outputs without ever synchronising. In the asynchronous setting, correct interactions are usually defined in terms of the absence of *orphan messages* [56, 42,36] and *unspecified reception configurations* [34]. Can we achieve, in the semantic framework we are envisioning, a general notion of "correct" interaction which is coherent with the usual ones, *both* in the synchronous and in the asynchronous settings?

Contributions. To address the questions above, we introduce a behavioural theory of (first-order) session types, which unifies the notions of typing and subtyping, in both the synchronous and asynchronous cases.

We represent processes (called *behaviours* in this work) as states in an LTS whose labels are inputs, outputs, or τ-actions. We treat input and output labels in an asymmetric way, similarly to the semantics of CFSMs and interface automata [3]. All the notions sought after in our questions are formalised as relations between behaviours, in a purely semantic fashion. To the best of our knowledge, ours is the first approach that extends session types theory to *arbitrary* behaviours in an LTS.

Our theory unifies synchronous and asynchronous behaviours within the same semantic framework, by defining relations which can be applied to both settings. This allows us to obtain several results about the *preservation* of our relations when passing from synchronous to asynchronous semantics.

More specifically, our main contributions are the following:

- a unifying semantic framework wherein we give synchronous/asynchronous semantics to session types (Definition 2.9) and CCS (Definition 2.14).
- a syntax-independent notion of "correct" interaction between behaviours, that we call *I/O compliance* (Definition 4.4). We prove that I/O compliance is stricter than progress, albeit coinciding with it on synchronous session types (Theorem 4.9).
- a formalisation in our semantic setting of the notion of *safety* in CFSMs [56] (Definition 5.15). We prove that I/O compliance is stricter than safety (Theorem 5.17), and that the two relations are equivalent on asynchronous session types (Theorem 5.20).
- preservation results showing that both I/O compliance and safety are preserved when passing from synchronous to asynchronous semantics of session types (Theorems 4.13 and 5.19).
- a syntax-independent notion of refinement between behaviours, that we call *I/O simulation* (Definition 6.4). We show that it is an I/O compliance-preserving preorder (Theorems 6.10 and 6.12), and that is coincides with the semantic subtyping relation [27,44] on synchronous session types (Theorem 6.13). Under suitable conditions, I/O simulation is preserved when passing from synchronous to asynchronous semantics of session types (Theorem 6.20).
- a syntax-driven session-type systems derived from the semantic properties of I/O simulation (Section 7). We show in Corollary 7.11 that the typing relation obtained in this way is stricter than I/O simulation. The main correctness property of our type system is stated in Theorem 7.14: in the asynchronous setting, if two CCS processes are typeable with I/O compliant session types, then also the processes are I/O compliant, and therefore they can safely interact. This result does not require a proof of subject reduction, but it only relies upon the soundness of the type system w.r.t. the I/O simulation relation, and on the general semantic properties of I/O simulation.

We prove all our results either in the main text or in the appendix.

2 Behaviours

We consider a labelled transition system (LTS) where labels are partitioned into *internal*, *input*, and *output* actions. We call it *I/O LTS*, and we refer to its states as *behaviours*. In Section 2.1 we introduce some basic relations and operators on behaviours. We then define the synchronous and asynchronous semantics of binary session types (Section 2.2) and of CCS (Section 2.3) in terms of I/O LTSs.

2.1 The I/O LTS

We assume a *universal* I/O LTS, which includes all possible LTSs one can be interested in. We denote it as:

$$\left(\mathbb{U}, \mathsf{A}_\tau, \left\{ \xrightarrow{\ell_\tau} \;\middle|\; \ell_\tau \in \mathsf{A}_\tau \right\}\right)$$

where:

- $\mathbb{U}$ is the set of all *behaviours* (ranged over by $p, q, \ldots$);
- A_τ is the set of all *labels*;
- $\xrightarrow{\ell_\tau} \subseteq \mathbb{U} \times \mathbb{U}$ is the *transition relation*.

We partition A_τ into *input actions* $\mathsf{A}^? = \{?\mathsf{a}, ?\mathsf{b}, \ldots\}$, *output actions* $\mathsf{A}^! = \{!\mathsf{a}, !\mathsf{b}, \ldots\}$, and the *internal action* τ. We postulate an involution $\mathrm{co}(\cdot)$ such that $\mathrm{co}(?\mathsf{a}) = !\mathsf{a}$ and $\mathrm{co}(!\mathsf{a}) = ?\mathsf{a}$. Table 1 summarises the main syntactic categories and some notation. Hereafter, unless explicitly stated otherwise, we will always consider behaviours up-to (label-preserving) isomorphism $\cong$.

Notation 2.1. *We write:*

- $p \xrightarrow{\ell_\tau}$ *when* $\exists p' \,.\, p \xrightarrow{\ell_\tau} p'$
- $p \to$ *when* $\exists \ell_\tau \,.\, p \xrightarrow{\ell_\tau}$
- $p \xrightarrow{!} q$ *when* $\exists \mathsf{a} \,.\, p \xrightarrow{!\mathsf{a}} p'$
- $\mathbf{0}$ *for any behaviour without outgoing transitions*
- $L^? = L \cap \mathsf{A}^?$ *and* $L^! = L \cap \mathsf{A}^!$, *for all* $L \subseteq \mathsf{A}$

Definition 2.2 (Parallel composition of behaviours)**.** *For all* $p, q \in \mathbb{U}$, *we define the* parallel composition $p \parallel q$ *as the behaviour in* $\mathbb{U}$ *with transitions given by the rules:*

$$\frac{p \xrightarrow{\ell_\tau} p'}{p \parallel q \xrightarrow{\ell_\tau} p' \parallel q} \qquad \frac{q \xrightarrow{\ell_\tau} q'}{p \parallel q \xrightarrow{\ell_\tau} p \parallel q'} \qquad \frac{p \xrightarrow{\ell} p' \quad q \xrightarrow{\mathrm{co}(\ell)} q'}{p \parallel q \xrightarrow{\tau} p' \parallel q'}$$

We introduce below all the basic relations that will be used along the paper.

Definition 2.3 (Weak transitions)**.** *We define the relation* $\Rightarrow$ *as* $(\xrightarrow{\tau})^*$, *and the relation* $\overset{\ell_\tau}{\Rightarrow}$ *as* $\Rightarrow\xrightarrow{\ell_\tau}\Rightarrow$. *Given a sequence of actions* $w = \ell_1, \ldots, \ell_n$, *we write* $\overset{w}{\Rightarrow}$ *for* $\overset{\ell_1}{\Rightarrow} \cdots \overset{\ell_n}{\Rightarrow}$. *Further, we write* $\overset{!}{\Rightarrow}$ *for* $\Rightarrow\xrightarrow{!}\Rightarrow$, *and for a set of behaviours* $\mathbb{Q}$, *we write* $\mathbb{Q} \overset{\ell_\tau}{\Rightarrow} q'$ *iff* $\exists q \in \mathbb{Q} \,.\, q \overset{\ell_\tau}{\Rightarrow} q'$, *and similarly for* $\mathbb{Q} \Rightarrow q'$.

Definition 2.4 (Weak barbs)**.** *We define* $p\Downarrow$ *as* $\{\ell \mid p \overset{\ell}{\Rightarrow}\}$. *By extension, we define* $\mathbb{Q}\Downarrow$ *as* $\bigcup_{q\in\mathbb{Q}} q\Downarrow$.

Definition 2.5 (Weakly persistent inputs/outputs)**.** *We write* $p \overset{??\mathsf{a}}{\Longrightarrow}$ *(resp.* $p \overset{!!\mathsf{a}}{\Longrightarrow}$*) iff for all* p', *we have that* $p \Rightarrow p'$ *implies* $p' \overset{?\mathsf{a}}{\Rightarrow}$ *(resp.* $p' \overset{!\mathsf{a}}{\Rightarrow}$*).*

Definition 2.6 (Weakly persistent barbs)**.** *We define:*

$$p\Downarrow^{??} = \{?\mathsf{a} \mid p \overset{??\mathsf{a}}{\Longrightarrow}\} \qquad p\Downarrow^{!!} = \{!\mathsf{a} \mid p \overset{!!\mathsf{a}}{\Longrightarrow}\}$$

and, by extension:

$$\mathbb{Q}\Downarrow^{??} = \bigcap_{q\in\mathbb{Q}} q\Downarrow^{??} \qquad \mathbb{Q}\Downarrow^{!!} = \bigcap_{q\in\mathbb{Q}} q\Downarrow^{!!}$$

Definition 2.7 (Set transition relation)**.** *We write* $q \Rrightarrow \mathbb{Q}$ *iff* $\emptyset \neq \mathbb{Q} \subseteq \{q' \mid q \Rightarrow q'\}$. *We write* $\mathbb{Q} \Rrightarrow \mathbb{Q}'$ *iff* $\forall q' \in \mathbb{Q}' \,.\, \exists q \in \mathbb{Q} \,.\, q \Rightarrow q'$.

Proposition 2.8 recalls inclusions between standard observational relations, where $\sim/\approx$ stand for strong/weak bisimilarity, and $\lessapprox$ is weak similarity [59].

Proposition 2.8. $\cong \subsetneq \sim \subsetneq \approx \subsetneq \lessapprox$.

2.2 Session types

A session type abstractly describes the behaviour of a process interacting with other parties. Here we consider the simple case of *binary* session types, where only two parties are involved. Our formalisation slightly adapts (and extends to the asynchronous case) the one in [4].

Formally, a session type is either an *internal choice* $\bigoplus_{i\in I} !\mathsf{a}_i.T_i$, an *external choice* $\&_{i\in I} ?\mathsf{a}_i.T_i$, or a recursion. In an internal choice, the process can choose one of the branches (say, the i-th), fire the output $(!\mathsf{a}_i)$, and then proceed with the continuation T_i. In an external choice, instead, the branch is chosen by the context; the process waits until receiving an input $(?\mathsf{a}_i)$, and then continues as T_i. Empty choices (either internal or external) represent successful termination.

Definition 2.9 (Session types)**.** Session types *are terms with the syntax:*

$$T \;::=\; \&_{i\in I} ?\mathsf{a}_i.T_i \;\mid\; \bigoplus_{i\in I} !\mathsf{a}_i.T_i \;\mid\; \mathrm{rec}_X\, T \;\mid\; X$$

where (i) *the set* I *is finite,* (ii) *actions in internal/external choices are pairwise distinct, and* (iii) *recursion is guarded. We omit the symbol* $\oplus/\&$ *in singleton choices. We write* $\mathbf{0}$

$?\mathsf{a}, ?\mathsf{b}, \ldots \in \mathsf{A}^?$	Input actions
$!\mathsf{a}, !\mathsf{b}, \ldots \in \mathsf{A}^!$	Output actions
$\mathsf{A} = \mathsf{A}^? \cup \mathsf{A}^!$	Set of I/O actions
$\mathrm{co}(\cdot)$	Involution of I/O actions
$\ell, \ell', \ldots \in \mathsf{A}$	Actions (visible labels)
τ	Internal action
$\mathsf{A}_\tau = \mathsf{A} \cup \{\tau\}$	Set of labels
$\ell_\tau, \ell'_\tau, \ldots \in \mathsf{A}_\tau$	Labels
$\mathbb{U}$	Set of all behaviours:
$\mathbb{U}_{\mathrm{ST}}$	Sync session behaviours
$\mathbb{U}_{\mathrm{aST}}$	Async session behaviours
$\mathbb{U}_{\mathrm{aCCS}}$	Async CCS behaviours
$p, q, r, \ldots \in \mathbb{U}$	Behaviours
$T, U, V \ldots \in \mathbb{U}_{\mathrm{ST}}$	Session types (sync)
$T[\sigma], U[\rho], \ldots \in \mathbb{U}_{\mathrm{aST}}$	Session types (async)
$P[\sigma], Q[\rho], \ldots \in \mathbb{U}_{\mathrm{aCCS}}$	Async CCS behaviours
$\mathbb{P}, \mathbb{Q}, \mathbb{R}, \ldots \subseteq \mathbb{U}$	Sets of behaviours
$\xrightarrow{\ell_\tau}, \rightarrow, \xrightarrow{!}$	Transitions
$\Rightarrow$	Weak τ transition
$\overset{\ell_\tau}{\Longrightarrow}$	Weak transition
$\overset{??\mathsf{a}}{\Longrightarrow}, \overset{!!\mathsf{a}}{\Longrightarrow}$	Weakly persistent I/O actions
$\Rrightarrow$	Set transition
$p\!\Downarrow$	Weak barbs
$p\!\Downarrow^?, p\!\Downarrow^!$	Weak I/O barbs
$p\!\Downarrow^{??}, p\!\Downarrow^{!!}$	Weakly persistent I/O barbs
$\cong$	Isomorphism
$\sim$	Strong bisimilarity
$\approx$	Weak bisimilarity
$\lessapprox$	Weak similarity

Table 1: Summary of notation.

for the empty (internal/external) choice, and will usually omit its trailing occurrences. Unless otherwise specified, we assume session types to be closed*, i.e. without free recursion variables.*

We present a *synchronous* semantics for session types (Definition 2.10), and one *asynchronous* (Definition 2.11). In both cases, internal choices first commit to one of the branches $!\mathsf{a}.T$ before enabling $!\mathsf{a}$, while external choices enable all their actions.

Definition 2.10 (Synchronous session behaviours)**.** *We denote with $\mathbb{U}_{ST}$ the set of behaviours of the form T, with transitions given by the rules:*

$$\frac{k \in I}{\&_{i\in I} ?\mathsf{a}_k.T_i \xrightarrow{?\mathsf{a}_k} T_k}\ (\textsc{TExt}) \qquad \frac{k \in I \quad |I| > 1}{\oplus_{i\in I} !\mathsf{a}_i.T_i \xrightarrow{\tau} !\mathsf{a}_k.T_k}\ (\textsc{TInt})$$

$$\frac{}{!\mathsf{a}.T \xrightarrow{!\mathsf{a}} T}\ (\textsc{TOut}) \qquad \frac{T[\mathrm{rec}_X\, T/X] \xrightarrow{\ell_\tau} T'}{\mathrm{rec}_X\, T \xrightarrow{\ell_\tau} T'}\ (\textsc{TRec})$$

For the asynchronous semantics, we consider behaviours of the form $T[\sigma]$ where σ is a finite sequence of output actions, modelling an unbounded *FIFO buffer*. We denote with ϵ the empty buffer, and with $[!\mathsf{a}.\sigma]$ a buffer with head $!\mathsf{a}$ and tail σ.

Definition 2.11 (Asynchronous session behaviours)**.** *We denote with $\mathbb{U}_{aST}$ the set of behaviours of the form $T[\sigma]$, with transition given by the rules:*

$$\frac{k \in I}{(\oplus_{i\in I} !\mathsf{a}_i.T_i)[\sigma] \xrightarrow{\tau} T_k[\sigma\,.!\mathsf{a}_k]}\ (\textsc{TInt}_{\mathrm{A}}) \qquad \frac{}{T[!\mathsf{a}.\sigma] \xrightarrow{!\mathsf{a}} T[\sigma]}\ (\textsc{TOut}_{\mathrm{A}})$$

$$\frac{k \in I}{(\&_{i\in I} ?\mathsf{a}_i.T_i)[\sigma] \xrightarrow{?\mathsf{a}_k} T_k[\sigma]}\ (\textsc{TExt}_{\mathrm{A}})$$

$$\frac{T[\mathrm{rec}_X\, T/X][\sigma] \xrightarrow{\ell_\tau} T'[\sigma']}{\mathrm{rec}_X\, T[\sigma] \xrightarrow{\ell_\tau} T'[\sigma']}\ (\textsc{TRec}_{\mathrm{A}})$$

Rule $(\textsc{TInt}_{\mathrm{A}})$ enqueues the selected output with a τ-move, while rule $(\textsc{TOut}_{\mathrm{A}})$ dequeues an output $!\mathsf{a}$ with a $!\mathsf{a}$-transition. The rules $(\textsc{TExt}_{\mathrm{A}})$ and $(\textsc{TRec}_{\mathrm{A}})$ are similar to their synchronous counterparts.

Example 2.12. *Let $T_1 = !\mathsf{a}.?\mathsf{b} \oplus !\mathsf{a}'.?\mathsf{b}'$, and let $T_2 = ?\mathsf{a}.(!\mathsf{b} \oplus !\mathsf{c})$. Their LTSs are shown in Figure 1 (note that the sync/async behaviours of T_2 are isomorphic).*

Similarly to [36], we adopt an *equi-recursive* view [69], by considering session types equivalent (denoted by the relation $\equiv$) up-to unfolding of recursion (see Definition A.1 and Proposition A.2).

Proposition 2.13 says that (up-to isomorphism) asynchronous session behaviours are *not* more general than synchronous ones, and *vice versa*: e.g., for the session types in Example 2.12, $T_1 \notin \mathbb{U}_{\mathrm{aST}}$ while $T_1\,[] \notin \mathbb{U}_{\mathrm{ST}}$.

Proposition 2.13. $\mathbb{U}_{aST} \not\subseteq \not\supseteq \mathbb{U}_{ST}$.

2.3 CCS

We consider a fragment of CCS [59] without delimitations and relabelling, and we provide it with synchronous and asynchronous semantics.

Definition 2.14. CCS terms *have the syntax:*

$$P, Q, R, S ::= \mathbf{0} \mid \ell_\tau.P \mid P + Q \mid P \,|\, Q \mid X \mid \mu_X P$$

where recursion is guarded.

Definition 2.15 (Synchronous CCS behaviours)**.** *We denote with $\mathbb{U}_{CCS}$ the set of behaviours of the form P, with transitions given by the following rules (symmetric ones are omitted):*

$$\frac{}{\ell_\tau.P \xrightarrow{\ell_\tau} P} \qquad \frac{P \xrightarrow{\ell_\tau} P'}{P + Q \xrightarrow{\ell_\tau} P'} \qquad \frac{P \xrightarrow{\ell_\tau} P'}{P \,|\, Q \xrightarrow{\ell_\tau} P' \,|\, Q} \qquad \frac{P[\mu_X P/X] \xrightarrow{\ell_\tau} P'}{\mu_X P \xrightarrow{\ell_\tau} P'}$$

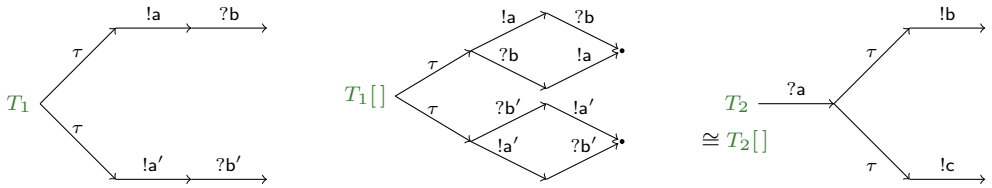


Figure 1: Three session behaviours.

The synchronous semantics of CCS terms is standard, except that the operator $\mid$ allows interleaving but not synchronisation, which instead is allowed by the LTS-level operator $\|$.

Definition 2.16 (Asynchronous CCS behaviours)**.** *We denote with $\mathbb{U}_{aCCS}$ the set of behaviours of the form $P[\sigma]$, with transitions given by the following rules (symmetric ones are omitted):*

$$\frac{\ell_\tau \in \{\tau\} \cup \mathsf{A}^?}{(\ell_\tau.P)\,[\sigma] \xrightarrow{\ell_\tau} P[\sigma]} \qquad \frac{}{(!\mathsf{a}.P)\,[\sigma] \xrightarrow{\tau} P[\sigma.!\mathsf{a}]}$$

$$\frac{P[\mu_X P/X][\sigma] \xrightarrow{\ell_\tau} P'[\sigma']}{\mu_X P[\sigma] \xrightarrow{\ell_\tau} P'[\sigma']} \qquad \frac{}{P\,[!\mathsf{a}.\sigma] \xrightarrow{!\mathsf{a}} P[\sigma]}$$

$$\frac{P[\sigma] \xrightarrow{\ell_\tau} P'[\sigma']}{(P+Q)[\sigma] \xrightarrow{\ell_\tau} P'[\sigma']} \qquad \frac{P[\sigma] \xrightarrow{\ell_\tau} P'[\sigma']}{(P \mid Q)[\sigma] \xrightarrow{\ell_\tau} (P' \mid Q)[\sigma']}$$

As in async session behaviours, an output $!\mathsf{a}$ is enqueued before it can be fired, and behaviours *cannot* read from their own buffers. In fact, in $P[\sigma] \,\|\, Q[\sigma']$ each behaviour reads from the other's buffer.

Example 2.17. *The behaviour of $!\mathsf{a}.\tau[\,]$ is shown as p_1 in Figure 2 at page 8.*

Proposition 2.18 states that async CCS behaviours strictly include async session behaviours, while they do not include synch CCS, and synch session behaviours.

Proposition 2.18. *$\mathbb{U}_{aST} \subsetneq \mathbb{U}_{aCCS} \not\supseteq \mathbb{U}_{CCS} \supsetneq \mathbb{U}_{ST}$, and $\mathbb{U}_{aCCS} \not\subseteq\not\supseteq \mathbb{U}_{ST}$ and $\mathbb{U}_{CCS} \not\subseteq \mathbb{U}_{aST}$.*

Session types can be encoded into CCS as follows.

Definition 2.19. *We define an encoding $[\![]\!]$ of session types into CCS terms as follows:*

$$[\![\textstyle\bigoplus_{i\in I} !\mathsf{a}_i.T_i]\!] = \sum_{i\in I} !\mathsf{a}_i.[\![T_i]\!] \qquad [\![\mathrm{rec}_X\, T]\!] = \mu_X [\![T]\!]$$
$$[\![\textstyle\&_{i\in I} ?\mathsf{a}_i.T_i]\!] = \sum_{i\in I} ?\mathsf{a}_i.[\![T_i]\!] \qquad [\![X]\!] = X$$

Proposition 2.20 states that, in the asynchronous semantics, a session type is isomorphic to its encoding.

Proposition 2.20. *$T[\,] = [\![T]\!][\,]$.*

Example 2.21. *The session type $!\mathsf{a} \oplus !\mathsf{b}$ in $\mathbb{U}_{ST}$ is encoded in $\mathbb{U}_{CCS}$ as the isomorphic behaviour $\tau.!\mathsf{a} + \tau.!\mathsf{b}$, while $?\mathsf{a}\,\&\,?\mathsf{b}$ is encoded as $?\mathsf{a} + ?\mathsf{b}$. By Definition 2.19, the session type $!\mathsf{a} \oplus !\mathsf{b}[\sigma]$ in $\mathbb{U}_{aST}$ is encoded in $\mathbb{U}_{aCCS}$ as the isomorphic behaviour $!\mathsf{a} + !\mathsf{b}[\sigma]$, while $?\mathsf{a}\,\&\,?\mathsf{b}[\sigma]$ is encoded as $?\mathsf{a} + ?\mathsf{b}[\sigma]$. Instead, $?\mathsf{a} + !\mathsf{b}$ in $\mathbb{U}_{CCS}$ is not isomorphic to any session type in $\mathbb{U}_{ST}$ (due to the mixed choice between an input and an output), nor to any behaviour in $\mathbb{U}_{aCCS}$ (because buffered semantics would introduce a τ-transition on the $!\mathsf{b}$-branch). Moreover, $?\mathsf{a} + \tau$ in $\mathbb{U}_{CCS}$ and $?\mathsf{a} + \tau[\,]$ in $\mathbb{U}_{aCCS}$ are not isomorphic to any asynchronous session type in $\mathbb{U}_{ST}$, nor in $\mathbb{U}_{aST}$ (due to the choice between an input and an internal move without continuation). Finally, $!\mathsf{a}.?\mathsf{b}$ from $\mathbb{U}_{ST}$ is not isomorphic to any behaviour in $\mathbb{U}_{aCCS}$ (because the buffered semantics of the latter do not allow the $!\mathsf{a}$-transition to preempt the $?\mathsf{b}$-transition).*

3 A motivating example: Alice and bartender

We now introduce a running example, used throughout the paper. The following types describe the behaviours of a bartender (B), and of a customer Alice (A):

$$U_\mathsf{B} = \mathrm{rec}_X\,(?\mathsf{aCoffee}.!\mathsf{coffee}.X \;\&\; ?\mathsf{aBeer}.(!\mathsf{beer}.X \oplus !\mathsf{no}.X) \;\&\; ?\mathsf{pay})$$
$$T_\mathsf{A} = !\mathsf{aCoffee}.?\mathsf{coffee}.!\mathsf{pay} \;\oplus\; !\mathsf{aBeer}.(?\mathsf{beer}.!\mathsf{pay} \,\&\, ?\mathsf{no}.!\mathsf{pay})$$

The bartender presents an external choice $\&$, allowing a customer to order either coffee or beer, or to eventually pay; in the first case, he will serve the coffee and then recursively wait for more orders; in the second case, he uses the internal choice $\oplus$ to decide whether to serve the beer or not — and then waits for more orders; in the third case, after the due amount (possibly 0) is paid, the interaction ends.

Alice internally chooses between coffee or beer; in the first case, she waits to get the coffee and then pays; in the second case, she lets the bartender choose between serving the beer, or saying no — and in both cases, she will check out.

Intuitively, T_A and U_B are *compliant* — i.e., their parallel composition $T_\mathsf{A} \| U_\mathsf{B}$ (under Definitions 2.2 and 2.10) gives rise to a "correct" interaction such that:

1. each output of T_A is matched by an input of U_B (and *vice versa*), and
2. when they stop synchronising (after pay), they are both "successful" — i.e., they reach a final configuration $\mathbf{0} \,\|\, \mathbf{0}$ without further enabled transitions.

Moreover, the following processes type-check — roughly, because the pairs $U_\mathsf{B}, Q_\mathsf{B}$ and $T_\mathsf{A}, P_\mathsf{A}$ expose the same interaction labels, and have similar branching structures:

$$Q_\mathsf{B} = \mu_Y(?\mathsf{aCoffee}.!\mathsf{coffee}.Y \,+\, ?\mathsf{aBeer}\,.\,(!\mathsf{beer}.Y + !\mathsf{no}.Y) \,+\, ?\mathsf{pay})$$
$$P_\mathsf{A} = !\mathsf{aCoffee}.?\mathsf{coffee}.!\mathsf{pay} \,+\, !\mathsf{aBeer}\,.\,(?\mathsf{beer}.!\mathsf{pay} + ?\mathsf{no}.!\mathsf{pay})$$

From typing and compliance, we can deduce that $P_\mathsf{A}[] \parallel Q_\mathsf{B}[]$ synchronise "correctly" (reflecting the "correct" interaction of $T_\mathsf{A} \parallel U_\mathsf{B}$), and reach the successful configuration $\mathbf{0}[] \parallel \mathbf{0}[]$, where Alice and the bartender agree in stopping their interaction.

Alice may also implement a *subtype* of T_A only asking for coffee: $T'_\mathsf{A} = !\mathsf{aCoffee}.?\mathsf{coffee}.!\mathsf{pay}$, with a corresponding process $P'_\mathsf{A} = !\mathsf{aCoffee}.?\mathsf{coffee}.!\mathsf{pay}$: also in this case, we would have that T'_A and U_B are compliant, and thus $P'_\mathsf{A}[] \parallel Q_\mathsf{B}[]$ also gives rise to a "correct" interaction.

So far, the structure of A's and B's processes have matched the structure of the respective types. This is a common situation in the session types literature: processes are usually written using calculi inheriting the structured communication approach pioneered by Honda *et al.* [49,50], thus reflecting the internal/external choices of types. However, sometimes things may be more complex. The bartender might have other incumbencies, and he may need to stop selling beer after a certain hour:

$$Q''_\mathsf{B} = \mu_Y\big((?\mathsf{aCoffee}.!\mathsf{coffee}.Y \,+\, ?\mathsf{aBeer}\,.\,(!\mathsf{beer}.Y + !\mathsf{no}.Y) \,+\, ?\mathsf{pay}) \\ +\tau.\mu_Z(?\mathsf{aCoffee}.!\mathsf{coffee}.Z \,+\, ?\mathsf{aBeer}.!\mathsf{no}.Z \,+\, ?\mathsf{pay})\big)$$

This reminds us of the small Erlang code sample given in Question 1: the τ branch represents the bartender's decision to stop waiting for customer orders, perform some internal duties (e.g. clean up the bar) and then serve again — this time, refusing to sell beer. Intuitively, we would like Q''_B to still have the type U_B, since compliant customer processes (e.g. Alice's one) will still be able to interact (either before or after the τ). A process like Q''_B, however, cannot usually be written (and typed) using classical session calculi: their grammar does not offer a τ prefix, since it would allow for processes where the distinction between internal/external choices is blurred (contrary to the expected program structure).

Let us consider another scenario: Alice is late for work. But she realises that the bartender-customer system is *asynchronous*: the counter is a bidirectional *buffer* where drinks and money can be placed. Thus, she tries to save time by implementing the following type and process:

$$T''_\mathsf{A} = !\mathsf{aCoffee}.!\mathsf{pay}.?\mathsf{coffee} \qquad P''_\mathsf{A} = !\mathsf{aCoffee}.(?\mathsf{coffee} \mid !\mathsf{pay})$$

i.e., in her type she plans to order a coffee, put her money on the counter while the bartender prepares her drink, and take it as soon as it is ready; in her process, she orders a coffee, and tries to grab the coffee with one hand, while putting the money on the counter with the other. P''_A represents an optimised program exploiting buffered communication semantics, thus diverging from the syntactic structure of T''_A, and reminds us of the issues discussed in Question 3. Therefore, is T''_A a type for P''_A? Is T''_A compliant with U_B, and will P''_A interact smoothly with Q_B and Q''_B? We shall answer these questions later on in Section 7.

4 I/O compliance

We introduce *I/O compliance*, a language-independent relation between arbitrary behaviours which formalises the intuitive notion of "correct interaction". We start by reviewing *progress*, a standard notion of compliance which identifies "correct interaction" with the absence of deadlock. Then, we observe its limitations in Section 4.1, as well as some limitations of other well-known notions of compliance. In Sections 4.2 and 4.3 we show how I/O compliance overcomes these limitations.

4.1 Compliance as progress

We start by discussing *progress*, a notion of compliance which interprets correctness as the absence of (unsuccessful) stuck states. To model the client/server asymmetry [31,4], we consider the *asymmetric* progress relation: a behaviour can stop interacting whenever it reaches success (regardless of the state of its counterpart).

Definition 4.1 (Progress)**.** *We write* $p \dashv q$ *whenever* $p \parallel q \Rightarrow p' \parallel q' \not\xrightarrow{\tau}$ *implies* $p' = \mathbf{0}$*. We write* $\vdash$ *for* $\dashv^{-1}$*, and* $\dashv\vdash$ *for* $\dashv \cap \vdash$*.*

Example 4.2. *We have the following relations:*

$!\mathsf{a}.?\mathsf{b} \dashv\vdash ?\mathsf{a}.!\mathsf{b}$	$\mathrm{rec}_X\, !\mathsf{a}.X \dashv\vdash \mathrm{rec}_Y\, ?\mathsf{a}.Y$
$!\mathsf{a}.?\mathsf{b} \not\dashv\not\vdash !\mathsf{b}.?\mathsf{a}$	$(!\mathsf{a}.?\mathsf{b})[] \dashv\vdash (!\mathsf{b}.?\mathsf{a})[]$
$!\mathsf{a}.!\mathsf{b} \not\dashv\not\vdash ?\mathsf{c}$	$(!\mathsf{a}.!\mathsf{b})[] \dashv\not\vdash ?\mathsf{c}[]$
$!\mathsf{a}.?\mathsf{b} \not\dashv\vdash ?\mathsf{a}$	$(\mathrm{rec}_X\, ?\mathsf{a}.X)[] \not\dashv\not\vdash !\mathsf{b}[]$
$\mathrm{rec}_X\, !\mathsf{a}.X \not\dashv\vdash ?\mathsf{a}$	$(\mathrm{rec}_X\, !\mathsf{a}.X)[] \dashv\vdash ?\mathsf{b}[]$

We now restate a result from [4]: progress between two synchronous session behaviours can be characterised in a syntactic way.

Proposition 4.3 (⊣/⊣⊢-induced shapes of session types)**.** *If* $T \dashv U$*, then exactly one of the following cases holds:*

a. $T = \mathbf{0}$*;*
b. $T \equiv \&_{i\in I} ?\mathsf{a}_i.T_i$ *and* $U \equiv \bigoplus_{j\in J} !\mathsf{a}_j.U_j$*, with* $\emptyset \neq J \subseteq I$*, and* $\forall j \in J\,.\,T_j \dashv U_j$*;*
c. $T \equiv \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ *and* $U \equiv \&_{j\in J} ?\mathsf{a}_j.U_j$*, with* $\emptyset \neq I \subseteq J$*, and* $\forall i \in I\,.\,T_i \dashv U_i$*.*

Furthermore, if $T \dashv\vdash U$*, we have either:*

a. $T = U = \mathbf{0}$*;*

b. $T \equiv \&_{i\in I}?\mathsf{a}_i.T_i$ *and* $U \equiv \bigoplus_{j\in J}!\mathsf{a}_j.U_j$, *with* $\emptyset \neq J \subseteq I$, *and* $\forall j \in J\,.\, T_j \dashv\!\vdash U_j$;
c. $T \equiv \bigoplus_{i\in I}!\mathsf{a}_i.T_i$ *and* $U \equiv \&_{j\in J}?\mathsf{a}_j.U_j$, *with* $\emptyset \neq I \subseteq J$, *and* $\forall i \in I\,.\, T_i \dashv\!\vdash U_i$.

Note that, when leaving synchronous session types, progress also relate behaviours with arguably incorrect interactions. For instance, we have $(\mathrm{rec}_X\, !\mathsf{a}.X)\,[] \dashv\!\vdash ?\mathsf{b}\,[]$, because the two asynchronous behaviours never reach a stuck state, even if no interaction ever happens. Ideally, we would like a stricter compliance relation, avoiding such "vacuous" progress.

To this aim one may consider, e.g., the *may*-, *must*- or *should*-testing compliance [10,24,38]. These relations do not hold in the above-mentioned case, since they all require that the success state $\mathbf{0} \parallel \mathbf{0}$ is reachable. On the other hand, these relations do not consider as compliant the asynchronous behaviours $(\mathrm{rec}_X\, !\mathsf{a}.X)\,[]$ and $(\mathrm{rec}_Y\, ?\mathsf{a}.Y)\,[]$, precisely because they never reach success. The last pair of behaviours models a producer/consumer system, where each produced message is consumed. Indeed, in the CFSM literature [19,56] these two behaviours would be considered compliant.

4.2 I/O compliance: definition and examples

We now introduce I/O compliance (recall from Section 2 that $p\Downarrow^! = (p\Downarrow)^! = p\Downarrow \cap \mathsf{A}^!$).

Definition 4.4 (I/O compliance)**.** *We say that a relation* $\mathcal{R}$ *is an* I/O compliance *iff, whenever* $p\ \mathcal{R}\ q$:

a. $p\Downarrow^! \subseteq \mathrm{co}(q\Downarrow^?)$ *and* $\Big((p\Downarrow^! = \emptyset \wedge p\Downarrow^? \neq \emptyset)$ *implies* $\emptyset \neq q\Downarrow^! \subseteq \mathrm{co}(p\Downarrow^?)\Big)$;
b. $p \xrightarrow{\ell} p' \wedge q \xrightarrow{\mathrm{co}(\ell)} q'$ *implies* $p'\ \mathcal{R}\ q'$;
c. $p \xrightarrow{\tau} p'$ *implies* $p'\ \mathcal{R}\ q$;
d. $q \xrightarrow{\tau} q'$ *implies* $p\ \mathcal{R}\ q'$.

We write $\ddot{\lhd}$ *for the largest I/O compliance relation,* $\ddot{\rhd}$ *for* $(\ddot{\lhd})^{-1}$, *and* $\ddot{\bowtie}$ *for the largest* symmetric *I/O compliance. We say that* p and q are I/O compliant *iff* $p \ddot{\bowtie} q$.

Item *a* captures the different role of inputs and outputs: each weak output must be matched by a corresponding input, so that the output can be dispatched. Although it is not required to match *every* weak input, the presence of weak inputs (without weak outputs) requires the counterpart to offer some matching weak output. Similarly to Proposition 4.3, when $p \ddot{\bowtie} q$, item *a* allows to enlarge the inputs of p (when non-empty), or to restrict its outputs (to a non-empty subset), while preserving I/O compliance. Items *b*, *c*, *d* require I/O compliance to be preserved if p and q synchronise or do internal moves.

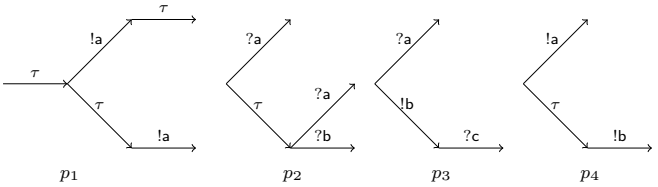


Figure 2: Some (non-session) behaviours.

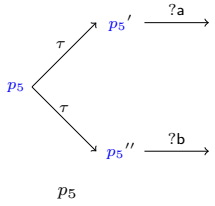


Figure 3: A behaviour without an I/O compliant one.

Lemma 4.5 states that the symmetric relation $\ddot{\bowtie}$ can be obtained by intersecting the two asymmetric relations (unlike e.g. bisimilarity [59]).

Lemma 4.5. $\ddot{\bowtie} \;=\; \ddot{\rhd} \cap \ddot{\lhd}$.

Example 4.6. *Consider the behaviours in Figure 2. We have that* $p_1 \ddot{\bowtie} p_2$, $p_2 \ddot{\bowtie} p_4$, $p_1 \ddot{\lhd} p_3$, *and* $p_2 \ddot{\lhd} p_3$, *while all the other pairs of behaviours are not compliant.*

The following example shows that some behaviours do not admit a (symmetric) I/O compliant one.

Example 4.7. *Consider the behaviour* p_5 *in Figure 3. We show that, for all behaviours* q, *it must be* $q \not\ddot{\bowtie} p_5$. *By contradiction, assume that* $q \ddot{\bowtie} p_5$, *for some* q. *Since* $p_5 \xrightarrow{\tau} p_5'$ *and* $p_5 \xrightarrow{\tau} p_5''$, *by applying twice clause* d *of Definition 4.4 it follows that* $q \ddot{\bowtie} p_5'$ *and* $q \ddot{\bowtie} p_5''$. *Now, by applying twice the first part of clause* a, *we get:*

$$q\Downarrow^! \;\subseteq\; \mathrm{co}(p_5'\Downarrow^?) \cap \mathrm{co}(p_5''\Downarrow^?) \;=\; \{?\mathsf{a}\} \cap \{?\mathsf{b}\} \;=\; \emptyset$$

And since $p_5' \ddot{\bowtie} q$ *and* $p_5'\Downarrow^! = \emptyset \neq p_5'\Downarrow^?$, *by the second part of clause* a *we would get* $\emptyset \neq q\Downarrow^!$ *(contradiction).*

The following example shows that I/O compliance does not hold when the progress is vacuous.

Example 4.8. *We have that* $(\mathrm{rec}_X\, !\mathsf{a}.X)\,[] \dashv ?\mathsf{b}\,[]$, *but* $(\mathrm{rec}_X\, !\mathsf{a}.X)\,[] \not\ddot{\bowtie} ?\mathsf{b}\,[]$: *in fact, the weak output barbs of the LHS contain* $!\mathsf{a}$, *not matched by the weak input barbs of the RHS (that only contain* $?\mathsf{b}$*) — thus violating item* a *of Definition 4.4.*

In CCS, let $P = !\mathsf{a} + !\mathsf{b}$, *and let* $Q = ?\mathsf{a}$. *We have that* $P \dashv Q$ *(because the LHS and RHS can synchronise on* a, *and then terminate), but* $P \not\ddot{\bowtie} Q$ *(because* $!\mathsf{b}$ *on the LHS is not matched by the inputs of the RHS).*

4.3 Main properties

Theorem 4.9 relates I/O compliance with progress. If two behaviours are I/O compliant, then they enjoy progress, while the *vice versa* is false (see Example 4.8). Still, $\ddot{\lhd}$ and $\dashv$ coincide for *synchronous* session behaviours. The intuition is the following. Take $T, U \in \mathbb{U}_{\mathrm{ST}}$: by Definition 2.10, they cannot generate infinite τ-traces, and whenever they output, they always commit to a *single* output transition. These properties allow to avoid situations similar to the ones in Example 4.8.

Theorem 4.9. *(a) If $p \ddot{\lhd} q$, then $p \dashv q$; (b) if $p, q \in \mathbb{U}_{ST}$ and $p \dashv q$, then $p \ddot{\lhd} q$.*

Example 4.10 (Alice and bartender)**.** *Recall the types and processes in Section 3. In the sync case, $U_{\mathsf{B}} \dashv\!\vdash T_{\mathsf{A}}$, $U_{\mathsf{B}} \bowtie T_{\mathsf{A}}$, $U_{\mathsf{B}} \dashv\!\vdash T'_{\mathsf{A}}$ and $U_{\mathsf{B}} \bowtie T'_{\mathsf{A}}$. The same holds for their async versions. When Alice is late for work, in the sync case we have $U_{\mathsf{B}} \not\dashv\!\vdash T''_{\mathsf{A}}$ and $U_{\mathsf{B}} \not\bowtie T''_{\mathsf{A}}$, due to the wrong order of Alice's actions. In the async case, instead, we have $U_{\mathsf{B}}[] \dashv\!\vdash T''_{\mathsf{A}}[]$ and $U_{\mathsf{B}}[] \bowtie T''_{\mathsf{A}}[]$. The latter relation is detailed in Table 2. Note that $U_{\mathsf{B}}[]$ has an input (?aBeer) which is not matched by an output of $T''_{\mathsf{A}}[]$, and therefore leads to states not considered in the relation (omitted in the Table). The same goes for $U_{\mathsf{B}}{}^{(3)}$, with two unmatched inputs (?aBeer and ?aCoffee).*

When two synchronous session behaviours are I/O compliant, the following lemma characterises the form of buffers in their asynchronous computations. In particular, in each reachable configuration one of the queues must be empty (thus resulting in a *half-duplex* communication [34]).

Lemma 4.11. *Let $T \circ U$, with $\circ \in \{\dashv, \vdash, \dashv\!\vdash, \ddot{\lhd}, \ddot{\rhd}, \bowtie\}$. If $T[] \parallel U[] \Rightarrow T'[\sigma] \parallel U'[\rho]$, then:*

(i) *$\sigma = \epsilon$ or $\rho = \epsilon$.*
(ii) *if $\sigma = \rho = \epsilon$ then $T' \circ U'$.*

Lemma 4.12 states that if two synchronous session behaviours are I/O compliant, then they are such also when passing to the asynchronous semantics. The inverse implication is false: I/O compliance relates asynchronous session behaviours which are *not* compliant under the synchronous semantics, e.g. $(!\mathsf{a}.?\mathsf{b})[] \bowtie (!\mathsf{b}.?\mathsf{a})[]$.

Lemma 4.12. *If $T \ddot{\lhd} U$, then $T[] \ddot{\lhd} U[]$.*

The following theorem extends Lemma 4.12 to all the notions of compliance studied in this section.

Theorem 4.13. *Let $\circ \in \{\vdash, \dashv\!\vdash, \dashv, \ddot{\rhd}, \bowtie, \ddot{\lhd}\}$. If $T \circ U$, then $T[] \circ U[]$.*

Proof. The case $\circ = \ddot{\lhd}$ follows by Lemma 4.12. For $\circ = \dashv$, we have:

$$\begin{array}{ccc} T \dashv U & \dashrightarrow & T[] \dashv U[] \\ \text{Theorem 4.9(b)} \Downarrow & & \Uparrow \text{Theorem 4.9(a)} \\ T \ddot{\lhd} U & \xRightarrow{\text{Lemma 4.12}} & T[] \ddot{\lhd} U[] \end{array}$$

The case for $\circ \in \{\vdash, \ddot{\rhd}\}$ follow by symmetry. Instead, the cases for $\circ \in \{\dashv\!\vdash, \bowtie\}$ follow respectively from $\dashv\!\vdash = \vdash \cap \dashv$ (by Definition 4.1) and $\bowtie = \ddot{\rhd} \cap \ddot{\lhd}$ (by Lemma 4.5). □

5 Safety

In the realm of Communicating Finite State Machines, where communication is asynchronous, the notion of correct interaction (called *safety* in [56]) is stricter than progress: e.g., it requires that enqueued messages are eventually consumed. In this section we interpret safety in our semantic setting, and we relate it with I/O compliance. The main result is that I/O compliance implies safety (Theorem 5.17), and that the two notions coincide on asynchronous session behaviours (Theorem 5.20).

5.1 Orphan messages

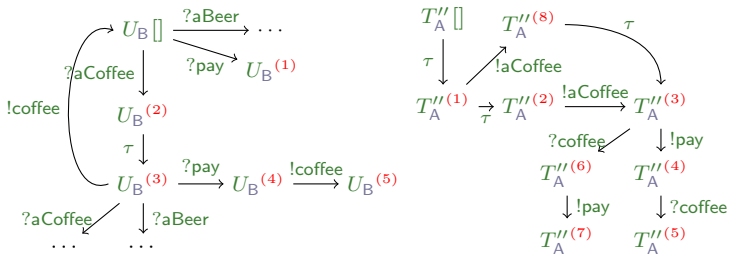
In CFSMs, *orphan messages* are outputs that are sent, but never received. To formalise this notion in our semantic setting, consider a parallel composition $p \parallel q$. If p always exposes a (weak) output transition $!\mathsf{a}$ (i.e., $p \overset{!\mathsf{a}}{\Longrightarrow}$, as per Definition 2.5) but q does not expose a matching input transition $?\mathsf{a}$, then we can say that $!\mathsf{a}$ is "orphan".

Definition 5.1 (Orphan message configuration)**.** *We say that $p \parallel q$ is an orphan message configuration of p iff $\exists \mathsf{a} \,.\, p \overset{!\mathsf{a}}{\Longrightarrow}$ and $q \overset{?\mathsf{a}}{\not\Longrightarrow}$. In this case we say that $!\mathsf{a}$ is an orphan message of p.*

Proposition 5.2. *Let $p \parallel q$ be an orphan message configuration, with $!\mathsf{a}$ orphan message of p. If $p \Rightarrow p'$ and $q \Rightarrow q'$, then $p' \parallel q'$ is an orphan message configuration, with $!\mathsf{a}$ orphan message of p'.*

Proof. By Definition 5.1, we have $p \overset{!\mathsf{a}}{\Longrightarrow}$ and $q \overset{?\mathsf{a}}{\not\Longrightarrow}$. From Definition 2.5, we have $p' \overset{!\mathsf{a}}{\Longrightarrow}$; furthermore, $q \overset{?\mathsf{a}}{\not\Longrightarrow}$ implies $q' \overset{?\mathsf{a}}{\not\Longrightarrow}$. Therefore, by Definition 5.1 we conclude that $p' \parallel q'$ is an orphan message configuration, with $!\mathsf{a}$ orphan message of p'. □

Note that the usual notion of orphan message from CFSM literature [56, 42] requires that there exists at least a non-empty buffer, and each machine is in a final state. Our definition relaxes the second requirement, while it uses persistent outputs to formalise the first one.



I/O compliance relation

$(U_B[], T_A''[])$	$(U_B[], T_A''^{(1)})$
$(U_B[], T_A''^{(2)})$	$(U_B{}^{(2)}, T_A''^{(8)})$
$(U_B{}^{(2)}, T_A''^{(3)})$	$(U_B{}^{(3)}, T_A''^{(8)})$
$(U_B{}^{(3)}, T_A''^{(3)})$	$(U_B{}^{(4)}, T_A''^{(4)})$
$(U_B{}^{(5)}, T_A''^{(5)})$	$(U_B[], T_A''^{(6)})$
$(U_B{}^{(1)}, T_A''^{(7)})$	

Table 2: I/O compliance for the asynchronous Alice and bartender.

The intuition is clear when considering asynchronous session behaviours: there, persistent outputs exist whenever the buffer is non-empty. In those behaviours, orphan messages are characterised by Lemma 5.3.

Lemma 5.3. *$T[\sigma] \parallel U[\rho]$ is an orphan message configuration with $!a$ orphan message of $T[\sigma]$ iff both the following conditions hold:*

(i) *$\sigma = !a.\sigma'$, or $\sigma = \epsilon$ and $T \equiv !a.T'$;*
(ii) *$U[\rho] \not\overset{?a}{\Longrightarrow}$.*

Proof. Direct by Definition 5.1 and Proposition A.6. □

Lemma 5.4 states that, in asynchronous session behaviours, orphan messages are preserved not only after individual τ-moves (as ensured by Proposition 5.2), but even after synchronisation.

Lemma 5.4. *Let $T[\sigma] \parallel U[\rho]$ be an orphan message configuration, with $!a$ orphan message of $T[\sigma]$. Then, $T[\sigma] \parallel U[\rho] \Rightarrow T'[\sigma'] \parallel U'[\rho']$ implies that $T'[\sigma'] \parallel U'[\rho']$ is an orphan message configuration, with $!a$ orphan message of $T'[\sigma']$. Furthermore, $\sigma' = \sigma.\sigma''$ (for some σ'').*

The following lemma states that I/O compliance does *not* hold in orphan message configurations.

Lemma 5.5. *If $p \parallel q$ is an orphan message configuration for p, then $p \not\ddot{\asymp} q$.*

Proof. By Definition 5.1, we have $p \overset{!!a}{\Longrightarrow}$ and $q \not\overset{?a}{\Longrightarrow}$: this implies $!a \in p\Downarrow^{!} \not\subseteq q\Downarrow^{?}$, which violates clause *a* of Definition 4.4. Therefore, $p \not\ddot{\asymp} q$. □

Example 5.6. *The inverse implication of Lemma 5.5 does not hold in general. E.g., let $P = !a + \tau.\mathbf{0}$. We have $P \not\ddot{\asymp} \mathbf{0}$, because $!a \in P\Downarrow^{!} \not\subseteq \mathrm{co}(\mathbf{0}\Downarrow^{?}) = \emptyset$. However, $P \parallel \mathbf{0}$ is* not *an orphan message configuration, because $P \not\overset{!!a}{\Longrightarrow}$.*

5.2 Unspecified reception

In CFSMs, *unspecified receptions* are configurations where some machine can only perform receptions, but the heads of buffers do not match the required input. In our setting, we say that a parallel composition $p \parallel q$ is an unspecified reception when p can only interact via input transitions, but q is not going to offer any matching output. Before formalising this in Definition 5.10, we study the auxiliary notion of *input behaviours*.

Definition 5.7 (Input behaviour)**.** *We say that p* is an input behaviour *(written $p \overset{??}{\Longrightarrow}$) whenever:*

$$p\Downarrow^{!} = \emptyset \qquad \text{and} \qquad p \Rightarrow p' \text{ implies } p'\Downarrow^{?} \neq \emptyset$$

Proposition 5.8 below states that an input behaviour can only τ-reduce to an input behaviour.

Proposition 5.8. *If $p \overset{??}{\Longrightarrow}$ and $p \Rightarrow p'$, then $p' \overset{??}{\Longrightarrow}$.*

Proof. From $p\Downarrow^{!} = \emptyset$ we have $p'\Downarrow^{!} = \emptyset$; moreover, since $\forall p'' \,.\, p \Rightarrow p''$ implies $p''\Downarrow^{?} \neq \emptyset$, and $p \Rightarrow p'$ (by hypothesis), we have $\forall p'' \,.\, p' \Rightarrow p''$ implies $p''\Downarrow^{?} \neq \emptyset$. Hence, by Definition 5.7, we conclude $p' \overset{??}{\Longrightarrow}$. □

In synchronous session types, input behaviours correspond to external choices; in *asynchronous* session types, they must also have an empty buffer.

Proposition 5.9. *$T[\sigma] \overset{??}{\Longrightarrow}$ iff $T \equiv \&_{i \in I} ?a_i.T_i$ with $I \neq \emptyset$, and $\sigma = \epsilon$.*

Proof. Straightforward, by Definition 2.11. □

We can now formalise unspecified reception.

Definition 5.10 (Unspecified reception configuration)**.** *We say that $p \parallel q$* is an unspecified reception configuration for p *iff $p \overset{??}{\Longrightarrow}$ and $\nexists a \,.\, p \overset{?a}{\Longrightarrow} \wedge q \overset{!a}{\Longrightarrow}$.*

Example 5.11. *We have that:*

- *$\mathrm{rec}_X\, !a.X[] \parallel ?b[]$ is an unspecified reception configuration for the RHS behaviour (and also an orphan message configuration for the LHS behaviour).*[1]
- *$\mathbf{0}[] \parallel ?b[]$ is an unspecified reception configuration for the RHS behaviour*

[1] In [56,42] orphan message configuration only contain final states, hence this configuration would not be considered an orphan message in CFSMs.

– $?\mathsf{a}[] \parallel ?\mathsf{b}[]$ *is an unspecified reception configuration for both LHS and RHS behaviours.*

Proposition 5.12 shows that unspecified reception is preserved by τ-transitions.

Proposition 5.12. *Let $p \parallel q$ be an unspecified reception configuration for p. If $p \parallel q \Rightarrow p' \parallel q'$ then $p' \parallel q'$ is an unspecified reception configuration for p'.*

Proof. Note that, by Definition 5.10, p and q cannot synchronise, because no output of q matches p's inputs, — and p has no weakly reachable outputs. Therefore, the τ transitions along $p \parallel q \Rightarrow p' \parallel q'$ can only be originated by *internal* moves. Now, since $p \Rightarrow p'$, by Proposition 5.8 we have $p' \overset{??}{\Longrightarrow}$; furthermore, since $\nexists \mathsf{a} . p \overset{?\mathsf{a}}{\Longrightarrow} \wedge q \overset{!\mathsf{a}}{\Longrightarrow}$, considering that $p'\Downarrow^{?} \subseteq p\Downarrow^{?}$ and $q'\Downarrow^{!} \subseteq q\Downarrow^{!}$ (because $q \Rightarrow q'$), we obtain $\nexists \mathsf{a} \,.\, p' \overset{?\mathsf{a}}{\Longrightarrow} \wedge\ q' \overset{!\mathsf{a}}{\Longrightarrow}$. □

Lemma 5.13 syntactically characterises unspecified reception configurations for asynchronous session types (similarly to Lemma 5.3 for orphan messages).

Lemma 5.13. *$T[\sigma] \parallel U[\rho]$ is an unspecified reception configuration for $T[\sigma]$ iff $T \equiv \&_{i\in I} ?\mathsf{a}_i.T_i$ with $I \neq \emptyset$, $\sigma = \epsilon$ and one of the following holds:*

a. *$\rho = !\mathsf{b}.\rho'$, for some ρ' and b such that $\forall i \in I \,.\, \mathsf{a}_i \neq \mathsf{b}$;*
b. *$\rho = \epsilon$ and $U \equiv \bigoplus_{j\in J} !\mathsf{b}_j.U_j$, where $\forall i \in I, j \in J \,.\, \mathsf{a}_i \neq \mathsf{b}_j$;*
c. *$\rho = \epsilon$ and $U \equiv \&_{j\in J} ?\mathsf{b}_j.U_j$.*

Proof. The $\Longrightarrow$ direction follows from Proposition 5.9 and Definition 2.11, considering that, by Definition 5.10:

(i) since $T[\sigma] \overset{??}{\Longrightarrow}$, by Proposition 5.9 T can only be equivalent to a non-empty external choice, and $\sigma = \epsilon$;
(ii) $U[\rho]$ does *not* offer any output matching T's inputs: hence, either U is equivalent to a (possibly empty) external choice and $\rho = \epsilon$ (case *c.* in the statement), or ρ's head exposes an unmatching output — either immediately (case *a.*) or after a τ-move (case *b.*).

The $\Longleftarrow$ direction follows from Definition 2.11: in all cases *a.*–*c.* of the statement, we conclude that $T[\sigma] \parallel U[\rho]$ is an unspecified reception configuration. □

Lemma 5.14 shows that two behaviours in an unspecified reception configuration are *not* I/O compliant.

Lemma 5.14. *If $p \parallel q$ is an unspecified reception configuration for p, then $p \not\dashv q$.*

Proof. By Definition 5.10 we have $p \overset{??}{\Longrightarrow}$ and $\nexists \mathsf{a} \,.\, p \overset{?\mathsf{a}}{\Longrightarrow} \wedge\ q \overset{!\mathsf{a}}{\Longrightarrow}$. Then, by Definition 5.7 we have $p\Downarrow^{!} = \emptyset$ and $p\Downarrow^{?} \neq \emptyset$. Furthermore, $q\Downarrow^{!} \cap \mathrm{co}(p\Downarrow^{?}) = \emptyset$: this violates clause *a* of Definition 4.4. Therefore, we have $p \not\dashv q$. □

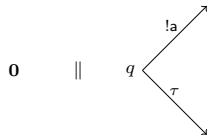


Figure 4: The behaviour $\mathbf{0} \parallel q$ is safe, but $\mathbf{0}$ and q are not I/O compliant.

5.3 Safety: definition and main properties

Definition 5.15 (Safety)**.** *We say that $p \parallel q$* is safe *iff $p \parallel q \Rightarrow p' \parallel q'$ implies that $p' \parallel q'$ is neither an orphan message nor an unspecified reception configuration.*

We relate safety with progress and I/O compliance.

Proposition 5.16. *If $p \parallel q$ is safe, then $p \dashv\!\vdash q$.*

Proof. We prove the contrapositive. Assume that $p \not\dashv\!\vdash q$. By definition of progress, there exist p' and q' such that $p \parallel q \Rightarrow p' \parallel q' \not\xrightarrow{\tau}$ and $p' \neq \mathbf{0}$ or $q' \neq \mathbf{0}$. Therefore:

$$p' \parallel q' \rightarrow \qquad \text{and} \qquad p' \not\xrightarrow{\tau} \qquad \text{and} \qquad q' \not\xrightarrow{\tau} \tag{1}$$

Since $p' \parallel q' \rightarrow$, at least one of the behaviours p', q' must take a move, which cannot be labelled τ. Without loss of generality, assume that p' moves. There are two possible (not mutually exclusive) cases:

– $p' \xrightarrow{!\mathsf{a}}$. Since $p' \not\xrightarrow{\tau}$, then by Definition 2.5 it must be $p \overset{!!\mathsf{a}}{\Longrightarrow}$. Now, by (1) it follows that $q' \not\xrightarrow{?\mathsf{a}}$, otherwise p' and q' would synchronise. Since $q' \not\xrightarrow{\tau}$, this implies that $q' \not\overset{?\mathsf{a}}{\Longrightarrow}$. By Definition 5.1, we obtain that $p' \parallel q'$ is an orphan message configuration, hence by Definition 5.15 we conclude that $p \parallel q$ is not safe.
– $p' \xrightarrow{?\mathsf{a}}$. Since $p' \not\xrightarrow{\tau}$, then by Definition 5.7 it must be $p' \overset{??}{\Longrightarrow}$. Now, by (1) it follows that $\nexists \mathsf{b} : p' \overset{?\mathsf{b}}{\Longrightarrow} \wedge\ q' \overset{!\mathsf{b}}{\Longrightarrow}$, otherwise p' and q' would synchronise. By Definition 5.10, we obtain that $p' \parallel q'$ is an unspecified reception configuration, hence by Definition 5.15 we conclude that $p \parallel q$ is not safe. □

Theorem 5.17. *If $p \bowtie q$, then $p \parallel q$ is safe.*

Proof. Let $p \parallel q \Rightarrow p' \parallel q'$. By Lemmas 4.5 and 5.5, $p' \parallel q'$ is *not* an orphan message configuration. By Lemmas 4.5 and 5.14, $p' \parallel q'$ is *not* an unspecified reception configuration. Hence, by Definition 5.15, $p \parallel q$ is safe. □

The following example shows that the converse of Theorem 5.17 does not hold in general.

Example 5.18. *The behaviour $\mathbf{0} \parallel q$ in Figure 4 is safe:*

– *neither $\mathbf{0}$ nor q have persistent outputs, hence $\mathbf{0} \parallel q$ has no orphan messages;*

– *neither* $\mathbf{0}$ *nor* q *is an input state, hence* $\mathbf{0} \parallel q$ *is not an unspecified reception configuration.*

Note however that $\mathbf{0} \not\bowtie q$. *Indeed,* $\mathsf{!a} \in q{\Downarrow^!} \not\subseteq \mathrm{co}(\mathbf{0}{\Downarrow^?}) = \emptyset$, *and therefore item a of Definition 4.4 is false.*

The following theorem extends Theorem 4.13 to the notion of safety. Namely, if the composition of two synchronous session behaviours is safe, then it is such also when passing to the asynchronous semantics.

Theorem 5.19. *If* $T \parallel U$ *is safe, then* $T[] \parallel U[]$ *is safe.*

Proof. If $T \parallel U$ is safe, then by Proposition 5.16 $T \dashv\vdash U$, and so by item (b) of Theorem 4.9 we also have $T \bowtie U$. By Theorem 4.13, $T[] \bowtie U[]$, and so by Theorem 5.17 we conclude that $T[] \parallel U[]$ is safe. □

Theorem 5.20 shows that the converse of Theorem 5.17 holds for asynchronous session behaviours. Summing up, safety and I/O compliance coincide in the setting of asynchronous session types.

Theorem 5.20. *If* $T[\sigma] \| U[\rho]$ *is safe, then* $T[\sigma] \bowtie U[\rho]$.

Proof. We prove the contrapositive. Assume $T[\sigma] \not\bowtie U[\rho]$. By Definition 4.4, this implies that there exist T', U', σ', ρ' such that $T[\sigma] \parallel U[\rho] \Rightarrow T'[\sigma'] \parallel U'[\rho']$ and item *a* is false for $p = T'[\sigma']$ and $q = U'[\rho']$. There are the following two cases:

1. $p{\Downarrow^!} \not\subseteq \mathrm{co}(q{\Downarrow^?})$. Then, there exists some a such that $\mathsf{!a} \in p{\Downarrow^!}$ and $\mathsf{?a} \notin q{\Downarrow^?}$. Since $\mathsf{!a} \in p{\Downarrow^!}$, one of the following two cases must hold:
   (i) $\sigma' = \mathsf{!a}.\sigma''$. By Proposition A.6 it follows that $T'[\sigma'] \overset{\mathsf{!!a}}{\Longrightarrow}$. Since $\mathsf{?a} \notin q{\Downarrow^?}$, we have that $U'[\rho'] \not\overset{\mathsf{?a}}{\Longrightarrow}$. Therefore, by Lemma 5.3 we obtain that $T'[\sigma'] \parallel U'[\rho']$ is an orphan message configuration. By Definition 5.15, $T[\sigma] \parallel U[\rho]$ is not safe.
   (ii) $\sigma' = \epsilon$ and $T' \equiv \mathsf{!a}.T'' \oplus T'''$. By Definition 2.11 we have the transition:

   $$T'[\epsilon] \parallel U'[\rho'] \xrightarrow{\tau} T''[\mathsf{!a}] \parallel U'[\rho']$$

   and we fall back to the previous case.
2. $\big((p{\Downarrow^!} = \emptyset \wedge p{\Downarrow^?} \neq \emptyset) \implies \emptyset \neq q{\Downarrow^!} \subseteq \mathrm{co}(p{\Downarrow^?})\big)$. Then, $p{\Downarrow^!} = \emptyset \wedge p{\Downarrow^?} \neq \emptyset$. This implies that T' is equivalent to an external choice, and that $\sigma' = \epsilon$. Further, one of the following two cases must hold:
   – $q{\Downarrow^!} = \emptyset$. This implies that $\rho' = \epsilon$, and U' is equivalent to an external choice (possibly empty). By item *c.* of Lemma 5.13, it follows that $T'[\sigma'] \parallel U'[\rho']$ is an unspecified reception configuration. By Definition 5.15, $T[\sigma] \parallel U[\rho]$ is not safe.
   – there exists some a such that $\mathsf{!a} \in q{\Downarrow^!}$ and $\mathsf{?a} \notin p{\Downarrow^?}$. Therefore, either one of the following subcases applies:
   – $\rho' = \mathsf{!a}.\rho''$. Then, by item *a.* of Lemma 5.13, it follows that $T'[\sigma'] \parallel U'[\rho']$ is an unspecified reception configuration. By Definition 5.15 we conclude that $T[\sigma] \parallel U[\rho]$ is not safe.
   – $\rho' = \epsilon$ and $U' \equiv \mathsf{!a}.U'' \oplus U'''$. By Definition 2.11 we have the transition:

   $$T'[\epsilon] \parallel U'[\epsilon] \xrightarrow{\tau} T'[\epsilon] \parallel U''[\mathsf{!a}]$$

   and we fall back to the previous case. □

6 I/O simulation

We now introduce a preorder between behaviours that generalises the usual notions of session typing and subtyping, with the typical co/contra-variance of outputs and inputs. We define the relation in Section 6.1, and study its main properties in Sections 6.2 and 6.3 — notably, proving that it preserves I/O compliance. In Section 6.4, we study its preservation when passing to the asynchronous semantics.

We start by adapting to our framework a classical preorder from the literature on behavioural contracts: intuitively, p is a *subcontract* of q when p can replace q in all possible contexts while preserving I/O compliance[2].

Definition 6.1 (Subcontract relation)**.** *We define the relation* $\sqsubseteq_{\mathbb{R}}$ *between behaviours as:*

$$p \sqsubseteq_{\mathbb{R}} q \qquad \textit{iff} \qquad \forall r \in \mathbb{R}\,.\; q \bowtie r \;\textit{ implies }\; p \bowtie r$$

We write $p \sqsubseteq q$ *when* $p \sqsubseteq_{\mathbb{R}} q$ *with* $\mathbb{R} = \mathbb{U}$.

The following lemma syntactically characterises the subcontract relation on synchronous session behaviours.

Lemma 6.2 ($\sqsubseteq_{\mathbb{U}_{ST}}$-induced shapes of session types)**.** $T \sqsubseteq_{\mathbb{U}_{ST}} U$ *implies either:*

a. $T = U = \mathbf{0}$;
b. $T \equiv \&_{k \in K} \mathsf{?a}_k.T_k$ *and* $U \equiv \&_{i \in I} \mathsf{?a}_i.T_i$, *with* $\emptyset \neq I \subseteq K$ *and* $\forall i \in I\,.\, T_i \sqsubseteq_{\mathbb{U}_{ST}} U_i$;
c. $T \equiv \bigoplus_{k \in K} \mathsf{!a}_k.T_k$ *and* $U \equiv \bigoplus_{i \in I} \mathsf{!a}_i.T_i$, *with* $\emptyset \neq K \subseteq I$, *and* $\forall k \in K\,.\, T_k \sqsubseteq_{\mathbb{U}_{ST}} U_k$.

To determine if two synchronous session behaviours are in the subcontract relation (i.e., $T \sqsubseteq U$), we do *not* need to consider all possible behaviours $r \in \mathbb{U}$ such that $U \bowtie r$: indeed, we can restrict ourselves to the case where r is a synchronous session behaviour itself.

Lemma 6.3. $T \sqsubseteq U \iff T \sqsubseteq_{\mathbb{U}_{ST}} U$.

[2] Note that the direction of $\sqsubseteq$ is opposite w.r.t. the strong subcontract relation in [31]. The other difference between $\sqsubseteq$ and the strong subcontract relation is that $\sqsubseteq$ requires to preserve I/O compliance, while [31] requires progress.

6.1 I/O simulation: definition and examples

Despite its elegance, Definition 6.1 cannot be directly exploited to establish whether two behaviours are related, due to the universal quantification over all contexts. We aim at extending the syntax-based coinductive characterisation on session types (Lemma 6.2) to arbitrary behaviours, without resorting to a universal quantification over contexts. To do that, we define an *I/O simulation* relation on behaviours, denoted by $\ddot{\lesssim}$. We show that it is a preorder (Theorem 6.10), and it preserves I/O compliance (Theorem 6.12). $\ddot{\lesssim}$ is equivalent to the subtype relation on sync session behaviours (Theorem 6.13), albeit stricter on arbitrary behaviours.

Definition 6.4 (I/O simulation)**.** *We say that $\ddot{\mathcal{R}}$ is an* I/O simulation *iff, whenever $p\ \ddot{\mathcal{R}}\ q$, then $\exists \mathbb{Q}$ (called* predictive set*) such that $q \Rightarrow \mathbb{Q}$, and:*

a. $p\Downarrow^{!} = \emptyset$ *implies* $\mathbb{Q}\Downarrow^{!} = \emptyset$
b. $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^{?}$ $\wedge$ $(\mathbb{Q}\Downarrow^{?} = \emptyset$ *implies* $p\Downarrow^{?} = \emptyset)$
c. $p \xrightarrow{\tau} p'$ *implies* $\exists q' \,.\, \mathbb{Q} \Rightarrow q' \wedge p'\ \ddot{\mathcal{R}}\ q'$
d. $p \xrightarrow{!\mathsf{a}} p'$ *implies* $\exists q' \,.\, \mathbb{Q} \overset{!\mathsf{a}}{\Rightarrow} q' \wedge p'\ \ddot{\mathcal{R}}\ q'$
e. $p \xrightarrow{?\mathsf{a}} p' \wedge \mathbb{Q} \overset{??\mathsf{a}}{\Longrightarrow}$ *implies* $\exists q' \,.\, \mathbb{Q} \overset{?\mathsf{a}}{\Rightarrow} q' \wedge p'\ \ddot{\mathcal{R}}\ q'$.

We write $\ddot{\lesssim}$ for the largest I/O simulation, $\ddot{\approx}$ for the largest symmetric I/O simulation, $\ddot{\gtrsim}$ for its inverse, and $\ddot{=}$ for $\ddot{\lesssim} \cap \ddot{\gtrsim}$.

Definition 6.4 generalises to our semantic setting the (syntax-based) coinductive subtyping rules for session types (Lemma 6.2). Similarly to Lemma 6.2, adding further external choices (inputs) or removing some internal choices (outputs) leads to a subtype. However, I/O simulation is more permissive than this, as it can relate general behaviours. These can have mixed choices, made of both inputs and outputs. Further, choices (possibly of inputs) are internal when led by τ transitions; dually, choices of outputs can be external. To cope with this complexity, in Definition 6.4 we use the predictive set $\mathbb{Q}$ as a subset of the internal choices of q. Once this set is fixed, the simulation game occurs between p and $\mathbb{Q}$.

Example 6.5. *In Table 3 we exemplify $\ddot{\lesssim}$. The first pair of behaviours correspond to the sync session types $T = ?\mathsf{a}.!\mathsf{b} \,\&\, ?\mathsf{c}$ and $U = ?\mathsf{a}.(!\mathsf{b} \oplus !\mathsf{c})$. The other pairs deal with behaviours outside $\mathbb{U}_{ST}$. The third one corresponds to Alice's asynchronous type T''_{A} and her process P''_{A}, from Section 3: we show that the former I/O simulates the latter. The last one shows an I/O simulation where the predictive set for the pair $(p, q) \in \ddot{\lesssim}$ contains more than one element.*

Example 6.6. *To establish whether, in Figure 5, $p \ddot{\lesssim} q$, we choose a predictive set $\mathbb{Q}$ that mandates the inputs of p, and includes its outputs (note that p has an additional input $?\mathsf{c}'$ not offered by $\mathbb{Q}$). The same happens with the predictive set $\mathbb{R}$, establishing $q \ddot{\lesssim} r$. Note that $\mathbb{R}$ and the small set inside are also predictive sets for $p \ddot{\lesssim} r$.*

6.2 Basic properties

The following lemma ensures that the largest I/O simulation $\ddot{\lesssim}$ indeed exists.

Lemma 6.7. *Let $\ddot{\mathbb{R}}$ be a set of I/O simulations. Then, $\bigcup \ddot{\mathbb{R}}$ is an I/O simulation.*

We now study preservation of I/O simulation under weak moves. When $p \ddot{\lesssim} q$, the relation $\ddot{\lesssim}$ is preserved by forward τ-moves of p and *backward* τ-moves of q.

Lemma 6.8. *If $p \ddot{\lesssim} q$, $p \Rightarrow p'$ and $q' \Rightarrow q$, then $p' \ddot{\lesssim} q'$.*

A consequence of Lemma 6.8 is that we can obtain a behaviour I/O simulated by p by reducing the internal non-determinism of p (i.e., by making p take τ moves).

Corollary 6.9. *If $p \Rightarrow p'$, then $p' \ddot{\lesssim} p$.*

Proof. From reflexivity of $\ddot{\lesssim}$ (Lemma D.1) we have $p \ddot{\lesssim} p$. From Lemma 6.8 we conclude $p' \ddot{\lesssim} p$. □

Theorem 6.10 establishes that $\ddot{\lesssim}$ is a preorder. The proof is not as straightforward as proving that $\sqsubseteq$ is a preorder, because we have to deal with the existential quantification on the predictive set $\mathbb{Q}$. Note that constructing the set $\mathbb{Q}$ is not always as easy as in Example 6.6, where the predictive set used for $q \ddot{\lesssim} r$ was valid also for proving $p \ddot{\lesssim} r$. Our general construction of $\mathbb{Q}$ is detailed in the proof of Theorem 6.10.

Theorem 6.10. *$(\mathbb{U}, \ddot{\lesssim})$ is a preorder.*

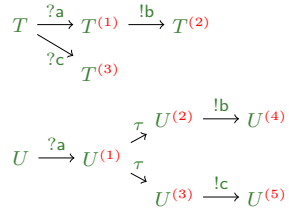
Weak simulation $\gtrsim$ and I/O simulation are unrelated, i.e. $\ddot{\lesssim} \not\subseteq \gtrsim \not\subseteq \ddot{\lesssim}$ (see Proposition D.8). However, weak *bisimulation* $\approx$ is strictly stronger than I/O bisimulation.

Theorem 6.11. $\approx \subsetneq \ddot{\approx}$

6.3 On I/O simulation and I/O compliance

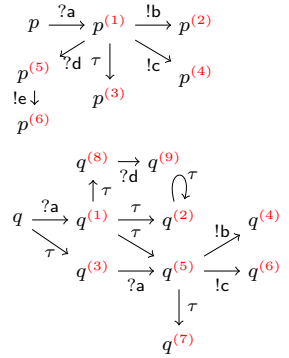
Theorem 6.12 is one of our main results: it states that $\ddot{\lesssim}$ preserves *I/O compliance*. Instead, $\ddot{\lesssim}$ does not preserve progress $\dashv$. For instance, consider the behaviours in Figure 6. We have that $p_6 \ddot{\lesssim} p_7$ and $p_7 \dashv p_8$. However, $p_6 \not\dashv p_8$: indeed, if p_8 chooses the branch $!\mathsf{b}$, then p_6 is stuck waiting for $?\mathsf{c}$.

Theorem 6.12. $p \ddot{\lesssim} q \circ r \implies p \circ r$, *for* $\circ \in \{\ddot{\triangleright}, \ddot{\bowtie}\}$.



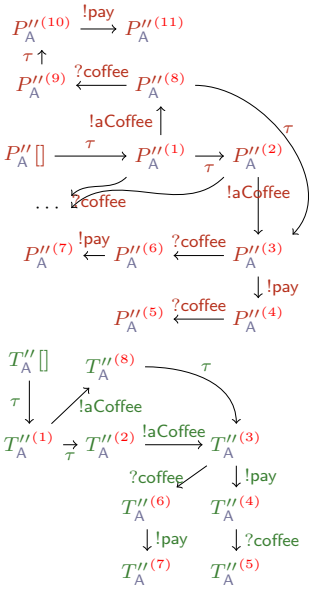
Relation	Pred. set
(T, U)	$\{U\}$
$(T^{(1)}, U^{(1)})$	$\{U^{(1)}\}$
$(T^{(1)}, U^{(2)})$	$\{U^{(2)}\}$
$(T^{(2)}, U^{(4)})$	$\{U^{(4)}\}$

$T^{(1)}$ is in relation with $U^{(1)}, U^{(2)}$: both the latter match the outputs of the first. $T^{(3)}$ is *not* in the relation: it is reached via an input ?c unmatched by U. $U^{(3)}, U^{(5)}$ are *not* in the relation: they are reached via a τ and an output !c unmatched by T.



Relation	Pred. set
(p, q)	$\{q\}$
$(p, q^{(3)})$	$\{q^{(3)}\}$
$(p^{(1)}, q^{(1)})$	$\{q^{(1)}\}$
$(p^{(1)}, q^{(5)})$	$\{q^{(5)}\}$
$(p^{(2)}, q^{(4)})$	$\{q^{(4)}\}$
$(p^{(3)}, q^{(2)})$	$\{q^{(2)}\}$
$(p^{(3)}, q^{(7)})$	$\{q^{(7)}\}$
$(p^{(4)}, q^{(6)})$	$\{q^{(6)}\}$

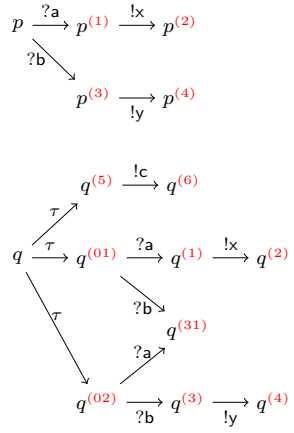
p is in relation with $q, q^{(3)}$, matching their ?a. Then, $p^{(1)}$ can either perform !b, !c, ?d or quit: this is matched by $q^{(1)}$, $q^{(5)}$; if $p^{(1)}$ follows its τ-branch to $p^{(3)}$, the latter is related with $q^{(2)}$, $q^{(7)}$. Note that $q^{(2)}$ does not stop, but performs τ-loop. Also note that the relation does *not* include $p^{(5)}$ since $q^{(1)} \stackrel{?d}{\Longrightarrow}$ but $\{q^{(1)}\} \not\stackrel{??d}{\Longrightarrow}$; indeed, it *cannot* include $p^{(5)}$, due to its !e (unmatched in q's reductions).



Relation	Pred. set
$(P''_A[], T''_A[])$	$\{T''^{(1)}_A\}$
$(P''^{(1)}_A, T''^{(1)}_A)$	$\{T''^{(1)}_A\}$
$(P''^{(2)}_A, T''^{(2)}_A)$	$\{T''^{(2)}_A\}$
$(P''^{(3)}_A, T''^{(3)}_A)$	$\{T''^{(3)}_A\}$
$(P''^{(4)}_A, T''^{(4)}_A)$	$\{T''^{(4)}_A\}$
$(P''^{(5)}_A, T''^{(5)}_A)$	$\{T''^{(5)}_A\}$
$(P''^{(6)}_A, T''^{(6)}_A)$	$\{T''^{(6)}_A\}$
$(P''^{(7)}_A, T''^{(7)}_A)$	$\{T''^{(7)}_A\}$
$(P''^{(8)}_A, T''^{(3)}_A)$	$\{T''^{(3)}_A\}$
$(P''^{(9)}_A, T''^{(6)}_A)$	$\{T''^{(6)}_A\}$
$(P''^{(10)}_A, T''^{(6)}_A)$	$\{T''^{(6)}_A\}$
$(P''^{(11)}_A, T''^{(7)}_A)$	$\{T''^{(7)}_A\}$

Note that $P''^{(1)}_A$ and $P''^{(2)}_A$ have an input branch (?coffee) unmatched by $T''^{(1)}_A$ and $T''^{(2)}_A$, and thus leading to states not considered in the relation (and here omitted).

$T''_A[] = \text{!aCoffee.!pay.?coffee } []$
$T''^{(1)}_A = \text{!pay.?coffee [!aCoffee]}$
$T''^{(2)}_A = \text{?coffee [!aCoffee.!pay]}$
$T''^{(3)}_A = \text{?coffee [!pay]}$
$T''^{(4)}_A = \text{?coffee } []$
$T''^{(5)}_A = \mathbf{0}\ [] = T''^{(7)}_A$
$T''^{(6)}_A = \mathbf{0}\ [\text{!pay}]$
$T''^{(7)}_A = \mathbf{0}\ [] = T''^{(5)}_A$
$T''^{(8)}_A = \text{!pay.?coffee } []$



Relation	Pred. set
(p, q)	$\{q^{(01)}, q^{(02)}\}$
$(p^{(1)}, q^{(1)})$	$\{q^{(1)}\}$
$(p^{(2)}, q^{(2)})$	$\{q^{(2)}\}$
$(p^{(3)}, q^{(3)})$	$\{q^{(3)}\}$
$(p^{(4)}, q^{(4)})$	$\{q^{(4)}\}$

p is related with q, with a predictive set containing 2 elements. This is the only possible predictive set supporting the relation: in fact, if the predictive set included q or $q^{(5)}$, then p would be forced to offer some output (by clause a of Definition 6.4); moreover, if the predictive set did *not* contain $q^{(01)}$, then the ?a-branch of p would need to match the ?a-branch of $q^{(02)}$ (by clause e of Definition 6.4) — but in this case we would need $(p^{(1)}, q^{(31)}) \in \dot{\lesssim}$, which is false. A similar problem would arise if the predictive set did not contain $q^{(02)}$.

Table 3: Examples of I/O simulation. For each pair of behaviours on the left, the table shows an I/O simulation relation, and the predictive sets supporting each pair of related states.

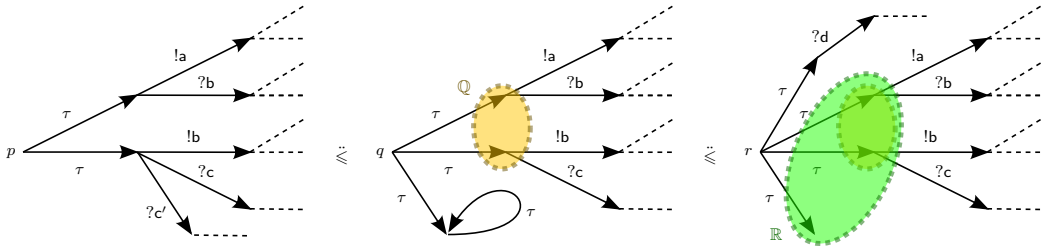


Figure 5: I/O simulation. $\mathbb{Q}$, $\mathbb{R}$ are the predictive sets resp. for $p \mathrel{\ddot{\leqslant}} q$ and $q \mathrel{\ddot{\leqslant}} r$.

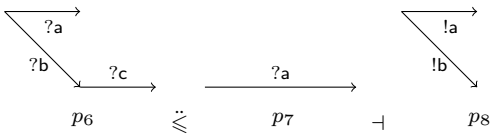


Figure 6: Progress is not preserved by I/O simulation (on general behaviours).

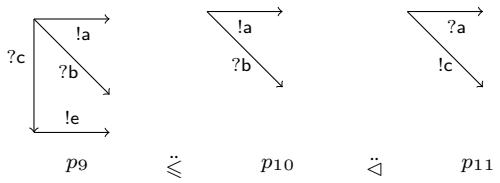


Figure 7: $\ddot{\lhd}$ is not preserved by I/O simulation (on general behaviours).

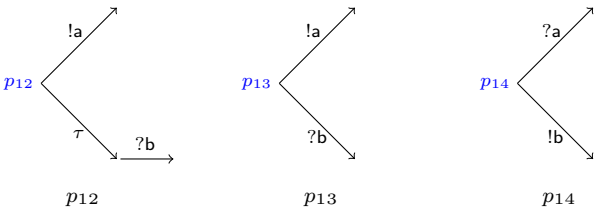


Figure 8: Three compliant-admitting behaviours showing that $\ddot{\leqslant} \subsetneq \sqsubseteq$.

Theorem 6.13 below establishes that I/O simulation can be seen as a subtyping/subcontract relation on general behaviours, that is $p \mathrel{\ddot{\leqslant}} q$ allows p to be always used in place of q. For instance, assume that p is an asynchronous CCS process typed with a session type q, which in turn complies with the session type r. Then, Theorem 6.12 states that I/O compliance is preserved by $\ddot{\leqslant}$, i.e. p is also I/O compliant with r, notwithstanding with the fact that p and r are specified in different calculi. Although I/O simulation is stricter than the subcontract relation on general behaviours, the two relations coincide on synchronous session behaviours, hence $\ddot{\leqslant}$ can be interpreted as the subtyping relation in [44] (albeit in the inverse direction, corresponding to that adopted e.g. in [27]).

Theorem 6.13. *(a)* $\ddot{\leqslant} \subsetneq \sqsubseteq$. *(b)* $T \mathrel{\ddot{\leqslant}} U$ *iff* $T \sqsubseteq U$.

Theorem 6.12 above does *not* generally hold for $\ddot{\lhd}$. Let $p \mathrel{\ddot{\leqslant}} q \mathrel{\ddot{\lhd}} r$: intuitively, $\ddot{\lhd}$ does *not* guarantee that r's outputs will be matched by q's inputs; therefore, if p adds an input branch which is matched by an output of r, then their synchronisation may reduce to *non*-compliant behaviours, as shown in Example 6.14 below.

Example 6.14 (On $\ddot{\leqslant}$ and $\ddot{\lhd}$)**.** *The behaviours* p_{10} *and* p_{11} *in Figure 7 are* $\ddot{\lhd}$ *compliant. Indeed, even though* p_{11} *offers an additional* !c *output, this is immaterial for I/O compliance, since it only checks that the* !a *output of* p_{10} *is received by a corresponding* ?a *of* p_{11}*. If we add a* ?c *branch to* p_{10}*, e.g. as in* p_9*, by* $\ddot{\leqslant}$ *we obtain a behaviour which is simulated by* p_{10}*. However* p_9 *and* p_{11} *are not I/O compliant, since the* c*-branches now can synchronise, leading to a stuck output* !e.

The situation described in Example 6.14 does *not* arise on synchronous session behaviours, as established in Lemma 6.15 below, which extends Theorem 6.12. Its proof exploits the fact that synchronous session behaviours do not have *mixed choices*.

Lemma 6.15. $T \mathrel{\ddot{\leqslant}} U \mathrel{\ddot{\lhd}} V$ *implies* $T \mathrel{\ddot{\lhd}} V$.

6.4 I/O simulation and asynchrony

We now address the problem of establishing whether I/O simulation and the subcontract relation are preserved by buffering. That is, if $T \mathrel{\ddot{\leqslant}} U$, (resp. $T \sqsubseteq U$), is it the case that $T[] \mathrel{\ddot{\leqslant}} U[]$ ($T[] \sqsubseteq U[]$)? Were the answer positive, we could reason about I/O simulation/subcontracting in the (finite-state) synchronous semantics, and then transfer these results to the asynchronous setting, where both I/O simulation and subcontracting are undecidable.

Unfortunately, the answer to both questions is negative, as shown in the following examples.

Example 6.16. *Consider the following session types:*

$$T = \mathrm{rec}_X\, !\mathsf{a}.X \qquad U = \mathrm{rec}_Y\, !\mathsf{a}.X \oplus !\mathsf{b}.?\mathsf{c}$$

We have $T \mathrel{\ddot{\leqslant}} U$*, but* $T[] \not\mathrel{\ddot{\leqslant}} U[]$*. To prove it, assume by contradiction that* $T[] \mathrel{\ddot{\leqslant}} U[]$*, with some predictive set* $\mathbb{U}$.

Since $U[] \overset{??_c}{\Longrightarrow}$, *we must have* $\mathbb{U} \overset{??_c}{\Longrightarrow}$ *(see Proposition D.6). Then, by the first part of clause* b *in Definition 6.4, it must be* $?c \in T[]\Downarrow^{?} = \emptyset$ — *contradiction.*

Example 6.17. *Let* T, U *as in Example 6.16, from which we know that* $T \ddot{\leqslant} U$*: hence, by item (a) of Theorem 6.13 we also know that* $T \sqsubseteq U$*. We now prove that* $T[] \not\sqsubseteq U[]$*. To this purpose, define:*

$$V \;=\; !c.\mathrm{rec}_Y\, ?a.Y \;\&\; ?b$$

Note that $U[] \bowtie V[]$ *(albeit* $U \not\bowtie V$*). However,* $T[] \not\bowtie V[]$*, since clause* a *of Definition 4.4 is violated because* $V[] \overset{!c}{\Rightarrow}$ *but* $T[] \overset{?c}{\not\Rightarrow}$[3]*. Therefore,* $T[] \not\sqsubseteq U[]$*.*

Intuitively, the non-preservation of $\ddot{\leqslant}$ under buffering is due to the fact that, in the asynchronous semantics, output prefixes can always be fired. Consequently, the inputs that are guarded by a sequence of outputs in a session type T may become weakly persistent (as per Definition 2.5), and thus relevant for clauses b and e of Definition 6.4. Similarly, $\sqsubseteq$ is not preserved because output-guarded inputs may become weakly reachable in $T[\sigma]$ and thus relevant for clause a of Definition 4.4.

Following this intuition, we provide in Definition 6.18 below a sufficient condition which guarantees the preservation of $\ddot{\leqslant}$ when passing from the synchronous to the asynchronous semantics of session types.

Definition 6.18 (Input-preserving behaviours)**.** *We say that* p is input-preserving w.r.t. q *(in symbols* $p \preceq_? q$*) iff for all* $w = \ell_1, \ldots, \ell_n, p', q', \mathsf{a}$*, whenever*

$$p \overset{w}{\Rightarrow} p' \;\; \textit{and} \;\; q \overset{w}{\Rightarrow} q' \tag{2a}$$

$$\textit{and} \quad \left(\forall q'' \,.\, q' \overset{!}{\Rightarrow}^{*} q'' \;\textit{implies}\; q'' \overset{!}{\Rightarrow}^{*} \overset{?a}{\longrightarrow}\right), \tag{2b}$$

$$\textit{then}\; p' \overset{!}{\Rightarrow}^{*} \overset{?a}{\longrightarrow}. \tag{2c}$$

Intuitively, $p \preceq_? q$ requires that, whenever p and q reach some states p', q' through the *same* weak transitions and $?a$ is persistent in q' (possibly after some outputs), then it must also be reachable in p' (possibly after some outputs).

Example 6.19. *Consider* T *and* U *from Example 6.16. We have that:*

a. $T \not\preceq_? U$
b. $!a.!a.!b.?c \;\left(\ddot{\leqslant} \cap \preceq_?\right) U$
c. $\mathrm{rec}_X\, !a.?c'.X \;\left(\ddot{\leqslant} \cap \preceq_?\right) \mathrm{rec}_Y\, !a.?c'.Y \oplus !b.?c$
d. $!a.?c \preceq_? !b.?c' \oplus !b'$ *(while* $\ddot{\leqslant}$ *does not hold).*

When the relation $\preceq_?$ holds, both I/O simulation and the subcontract relation are preserved by buffering.

[3] Note also that $T[] \parallel V[]$ is *not* safe: in fact, according to Definition 5.1, it is an orphan message configuration, with $!c$ orphan message of $V[]$.

Theorem 6.20. *If* $T \,(\circ \cap \preceq_?)\, U$ *then* $T[] \circ U[]$*, for* $\circ \in \{\ddot{\leqslant}, \sqsubseteq\}$*.*

Proof. The proof for $\circ = \ddot{\leqslant}$ exploits four key properties:

1. whenever $p \ddot{\leqslant} q$, each weak sequence of outputs of p is I/O simulated by q (Proposition D.12);
2. whenever $p \preceq_? q$, and the two behaviours reach p', q' through the same weak transitions, then $p' \preceq_? q'$ (Proposition D.13);
3. each weak sequence of ouptuts from a session behaviour T corresponds to a sequence of τs in its asynchronous version $T[\sigma]$, for all σ (Proposition D.14);
4. *vice versa*, each sequence of τs from $T[\sigma]$ corresponds to a weak sequence of outputs from T (Proposition D.15).

To prove $\circ = \sqsubseteq$ we exploit the previous case and Theorem 6.13. See Appendix D.3 for full details. □

We now show how the three main notions studied so far (I/O simulation, I/O compliance and safety) can be combined in order to determine that two processes can safely interact. Assume that $T \ddot{\leqslant} U$ and $U \bowtie V$. Using Theorem 6.12 we can deduce $T \bowtie V$; then, from Lemma 4.12 we obtain $T[] \bowtie V[]$, and so by Theorem 5.17, we conclude that $T[] \parallel V[]$ is safe.

Corollary 6.21. *If* $T \ddot{\leqslant} U \bowtie V$*, then* $T[] \parallel V[]$ *is safe.*

7 Session types without types

In this section we show how typing rules can arise by applying I/O simulation on specific process and type languages — in our case, (a)sync CCS and session types.

7.1 From semantics to syntax

Our treatment so far does not depend on a syntactic representation of behaviours in $\mathbb{U}$. In the resulting unifying view, there are no inherent distinctions between processes and types: they are just behaviours in an LTS. This allows us to define relations between objects which belong to different realms: e.g. $p \ddot{\leqslant} q$ may relate, say, an async CCS process with a session type.

The price for this generalisation seems to be the loss of compositional type systems, which assign types to processes by only reasoning on their syntax. However, we can restore this feature in our framework as follows:

1. choosing a process language and a type language (with their syntax and semantics);
2. encoding types into processes;
3. devising a set of properties for $\ddot{\leqslant}$ on processes, and interpreting them as syntax-based typing rules.

$$P \;\ddot{\approx}\; P + \mathbf{0} \qquad P \;\ddot{\approx}\; P \mid \mathbf{0}$$
$$P + Q \;\ddot{\approx}\; Q + P \qquad P \mid Q \;\ddot{\approx}\; Q \mid P$$
$$P + (Q + R) \;\ddot{\approx}\; (P + Q) + R \qquad P \mid (Q \mid R) \;\ddot{\approx}\; (P \mid Q) \mid R$$
$$P \;\ddot{\approx}\; P + P$$

Table 4: Some properties of $\ddot{\approx}$ in $\mathbb{U}_{\text{aCCS}}$ (recall that $\ddot{\approx}$ is the largest symmetric I/O simulation, by Definition 6.4).

We apply this methodology to a concrete case, where at step 1 we choose async CCS ($\mathbb{U}_{\text{aCCS}}$) for processes[4], and async session behaviours ($\mathbb{U}_{\text{aST}}$) for types. For step 2, the encoding $[\![\cdot]\!]$ from types to processes is just the one given in Definition 2.19. To improve readability, hereafter we shall sometimes omit the buffers $[\sigma]$ appearing in $\mathbb{U}_{\text{aCCS}}$ processes. Moreover, when $P, Q \in \mathbb{U}_{\text{aCCS}}$, we will write $P \;\ddot{\lesssim}\; Q$ to mean $\forall \sigma \,.\, P[\sigma] \;\ddot{\lesssim}\; Q[\sigma]$.

For step 3, we aim at a $\ddot{\lesssim}$-based inductive relation for $\mathbb{U}_{\text{aCCS}}$. We start by selecting in Table 4 a few semantic equivalences between $\mathbb{U}_{\text{aCCS}}$ behaviours.

Lemma 7.1. *The relations in Table 4 hold in $\mathbb{U}_{aCCS}$.*

We then interpret the semantic equivalences in Table 4 between $\mathbb{U}_{\text{aCCS}}$ *behaviours* as a set of syntactic axioms for a relation $\equiv$ between $\mathbb{U}_{\text{aCCS}}$ *terms*[5].

Definition 7.2. *We define $\equiv$ as the least congruence relation on $\mathbb{U}_{aCCS}$ satisfying the axioms in Table 4 (by replacing $\ddot{\approx}$ with $\equiv$).*

As usual, $\equiv$ embodies the commutative monoidal laws for $+$ and $\mid$ (with $\mathbf{0}$ as neutral element), and the absorption law for $+$.

We now follow a similar approach, by selecting in Table 5 a set of properties satisfied by $\ddot{\lesssim}$ on $\mathbb{U}_{\text{aCCS}}$ behaviours. We present these properties as a set of inference rules, whose correctness is established by Lemma 7.3. The intuition behind these rules is the following:

- (+Ctx) says that $\ddot{\lesssim}$ is a precongruence with respect to guarded choices;
- (+Int) and (+Ext) with $K = \emptyset$ correspond to the typical session typing rules for internal/external choices, which add inputs and remove outputs on the LHS;
- (+Ext) with $K \neq \emptyset$ handles an external choice with timeout (modelled as a τ action), similarly to the Erlang-style `receive...after...` behaviour discussed in Question 2;

[4] The process calculi used in session types literature are typically variants of the π-calculus. In this work, we focus on the behavioural properties of structured first-order interaction; we thus focus on (async) CCS, abstracting orthogonal aspects such as value and channel passing (see also Section 8.1).

[5] We overload the symbol $\equiv$, which was already introduced as a relation between session types in Definition A.1. The relation being used is always clear from the context.

$$\frac{\forall i \in I \,.\, P_i \;\ddot{\lesssim}\; Q_i}{\sum_{i\in I} \ell_{\tau i}.P_i \;\ddot{\lesssim}\; \sum_{i\in I} \ell_{\tau i}.Q_i}\ (+\text{Ctx}) \qquad \frac{Q = \sum_{i\in I} !\mathsf{a}_i.Q_i \quad \forall i \in I \,.\, P_i \;\ddot{\lesssim}\; Q_i \quad I \neq \emptyset}{\sum_{i\in I} !\mathsf{a}_i.P_i \;\ddot{\lesssim}\; Q + !\mathsf{b}.Q'}\ (+\text{Int})$$

$$\frac{Q = \sum_{i\in I} ?\mathsf{a}_i.Q_i \quad \forall i \in I \,.\, P_i \;\ddot{\lesssim}\; Q_i \quad \emptyset \neq I \subseteq J \quad \forall k \in K \,.\, P_k \;\ddot{\lesssim}\; Q \quad \forall j \in J \setminus I \,.\, \mathsf{a}_j \notin \{\mathsf{a}_i\}_{i\in I}}{\sum_{j\in J}(?\mathsf{a}_j.P_j) + \sum_{k\in K} \tau.P_k \;\ddot{\lesssim}\; Q}\ (+\text{Ext})$$

$$\frac{\forall i \in I \,.\, P_i \;\ddot{\lesssim}\; Q}{\sum_{i\in I} \tau.P_i \;\ddot{\lesssim}\; Q}\ (+\tau) \qquad \frac{P \;\ddot{\lesssim}\; Q \quad R \;\ddot{\lesssim}\; S \quad \begin{array}{c}\text{ins}(P) \cap \text{ins}(R) = \emptyset \\ \text{ins}(Q) \cap \text{ins}(S) = \emptyset\end{array}}{P \mid R \;\ddot{\lesssim}\; Q \mid S}\ (|\text{LR})$$

$$\frac{Q = \sum_{i\in I} !\mathsf{c}_i.Q_i \quad P' \;\ddot{\lesssim}\; !\mathsf{a} \quad P'' \;\ddot{\lesssim}\; ?\mathsf{b}}{P' \mid P'' \;\ddot{\lesssim}\; !\mathsf{a}.?\mathsf{b} + Q}\ (|\text{L}) \qquad \frac{P \;\ddot{\lesssim}\; ?\mathsf{b}.Q \quad R = \sum_{i\in I} !\mathsf{c}_i.R_i}{!\mathsf{a}.P \;\ddot{\lesssim}\; !\mathsf{a} + R \mid ?\mathsf{b}.Q}\ (|\text{R})$$

$$\frac{P \equiv P' \quad P' \;\ddot{\lesssim}\; Q' \quad Q' \equiv Q}{P \;\ddot{\lesssim}\; Q}\ (\text{Congr})$$

Table 5: Rules for $\ddot{\lesssim}$ in $\mathbb{U}_{\text{aCCS}}$. We denote with $\text{ins}(P)$ the set of inputs appearing in the body of P.

- (+τ) allows the LHS to aggregate internal choices, provided that all of them are I/O simulated by Q;
- (|LR) states that $\ddot{\lesssim}$ is almost a precongruence w.r.t. parallel composition — but further requires that the inputs of the parallel components do not overlap (and thus, cannot compete for synchronising on the same output);
- (|L) exploits the asynchronicity of $\mathbb{U}_{\text{aCCS}}$, by allowing an output and an input in parallel on the LHS, when they are sequentially composed in the RHS;
- (|R) is roughly symmetrical, by allowing the LHS to be the sequential composition $!\mathsf{a}.P$ when $!\mathsf{a}$ and $?\mathsf{b}$ are in parallel on the RHS.

Lemma 7.3. *The rules in Table 5 hold in $\mathbb{U}_{aCCS}$.*

As prescribed by step 3, we now interpret the properties in Table 5 as syntax-based typing rules for $\mathbb{U}_{\text{aCCS}}$.

Definition 7.4. *We define $\ddot{\preccurlyeq}$ as the least relation between $\mathbb{U}_{aCCS}$ terms satisfying the rules in Table 5.*

Example 7.5. *From Section 3, recall Alice's type T''_{A} and process P''_{A} when she is late for work.*

$$T''_{\mathsf{A}} = !\mathsf{aCoffee}.!\mathsf{pay}.?\mathsf{coffee} \qquad P''_{\mathsf{A}} = !\mathsf{aCoffee}.(?\mathsf{coffee} \mid !\mathsf{pay})$$

We have the following type encoding in CCS:

$$[\![T''_{\mathsf{A}}]\!] = !\mathsf{aCoffee}.!\mathsf{pay}.?\mathsf{coffee}$$

We have the following typing derivation:

$$\dfrac{\dfrac{\dfrac{}{\mathbf{0} \;\ddot{\lesssim}\; [\![\mathbf{0}]\!]}\,(+\text{Ctx})}{!\mathsf{pay} \;\ddot{\lesssim}\; [\![!\mathsf{pay}]\!]}\,(+\text{Ctx}) \quad \dfrac{\dfrac{}{\mathbf{0} \;\ddot{\lesssim}\; [\![\mathbf{0}]\!]}\,(+\text{Ctx})}{?\mathsf{coffee} \;\ddot{\lesssim}\; [\![?\mathsf{coffee}]\!]}\,(+\text{Ctx}) \quad \mathbf{0} \equiv \sum_{i\in\emptyset} !\mathsf{c}_i.Q_i}{?\mathsf{coffee} \mid !\mathsf{pay} \;\equiv\; !\mathsf{pay} \mid ?\mathsf{coffee} \;\ddot{\lesssim}\; [\![!\mathsf{pay}.?\mathsf{coffee}]\!] + \mathbf{0} \;\equiv\; [\![!\mathsf{pay}.?\mathsf{coffee}]\!]}\,(|\text{L})}{?\mathsf{coffee} \mid !\mathsf{pay} \;\ddot{\lesssim}\; [\![!\mathsf{pay}.?\mathsf{coffee}]\!]}\,(\text{Congr})}{!\mathsf{aCoffee}.(?\mathsf{coffee} \mid !\mathsf{pay}) \;\ddot{\lesssim}\; [\![T''_{\mathsf{A}}]\!]}\,(+\text{Ctx})$$

7.2 Handling recursion

The typing rules introduced so far can be applied to all $\mathbb{U}_{\text{aCCS}}$ terms (including those involving recursion), yet no rule allows to operate *under* recursion. In this section we show how to overcome this limitation. E.g., consider:

$$P = \mu_X !a.X \qquad Q = \mu_X !a.X + !b \tag{3}$$

If we add the axiom $X \mathrel{\ddot{\preccurlyeq}} X$ (based on $X \mathrel{\ddot{\lessdot}} X$, that holds since X has no transitions), by applying rule (+Int) we would obtain $!a.X \mathrel{\ddot{\preccurlyeq}} !a.X + !b$; from this, we would like to deduce $P \mathrel{\ddot{\preccurlyeq}} Q$ — coherently with the fact that $P \mathrel{\ddot{\lessdot}} Q$ holds. By considering this example, it would be tempting to also add a typing rule which allows to deduce $\mu_X P' \mathrel{\ddot{\preccurlyeq}} \mu_X Q'$ whenever $P' \mathrel{\ddot{\preccurlyeq}} Q'$. However, in the general case, this would be unsound. For instance:

$$P = \mu_X !a.X \qquad R = \mu_X !a.X + !b.?c \tag{4}$$

would be a counterexample. Indeed, using the proposed axiom and rule, together with (+Int), we would obtain $P \mathrel{\ddot{\preccurlyeq}} R$ — yet we do *not* have $P \mathrel{\ddot{\lessdot}} R$. Recall that we chose to operate in the asynchronous setting, and we have $R[\sigma] \overset{??c}{\Longrightarrow}$ but $P[\sigma] \overset{?c}{\not\Longrightarrow}$.

To rule out these problematic cases, we restrict our syntax-based typing to CCS$^-$, a fragment of async CCS which is expressive enough for the examples in Section 3.

Definition 7.6 (CCS$^-$). *CCS$^-$ terms have the syntax:*

$$P, Q \;::=\; \textstyle\sum_{i\in I} \ell_{\tau i}.P_i \;\mid\; P \mid Q \;\mid\; X \;\mid\; \mu_X P$$

where we stipulate that:

a. *in $\mu_X P$, each X is* guarded *in some subterm $\ell.X$;*
b. *if a choice $\sum_{i\in I} P_i$ appears under μ_X, and some P_i contains an input, then each X occurs after an input.*

The semantics of CCS$^-$ is inherited from that of CCS. We denote by $\mathbb{U}^-_{aCCS}$ the resulting async behaviours. CCS$^-$ processes not containing $\mid$ are said sequential. *Unless otherwise specified, hereafter we will use P, Q, R, S to denote CCS$^-$ terms, only. We will also adopt the Barendregt convention [7, §2.1.13] that no bound X can occur free or in two different bindings.*

The issue highlighted in Equation (4) is solved, because R does not satisy condition b. Note that, e.g., both Q in Equation (3) and $Q' = \mu_X !a.\mathbf{0} + !b.?c.!d.X$ are CCS$^-$ terms. Indeed, in Q' the branch without inputs is non-recursive, while X occurs after an input. The guardedness condition for recursion variables (clause a) comes from our original CCS fragment.

Lemma 7.7 provides an induction principle for $\ddot{\lessdot}$.

Lemma 7.7. *Let $\mu_X P$ and Q be sequential CCS$^-$ terms. If $P[Q/X] \mathrel{\ddot{\lessdot}} Q$, then $\mu_X P \mathrel{\ddot{\lessdot}} Q$.*

Intuitively, Lemma 7.7 is reminiscent of the standard argument in domain theory, stating that if x is a prefixed point of f, i.e. $f(x) \sqsubseteq x$, then $fix f \sqsubseteq x$.

We now extend the type system $\ddot{\preccurlyeq}$ of Section 7.1 to deal with recursive CCS$^-$ terms.

Definition 7.8. *Let Γ be a mapping from recursion variables to CCS$^-$ terms. Then, $\ddot{\preccurlyeq}_\Gamma$ is the relation between CCS$^-$ terms inductively defined by the rules obtained by replacing $\ddot{\lessdot}$ with $\ddot{\preccurlyeq}_\Gamma$ in Table 5, and by the following additional rules:*

$$\frac{\Gamma(X) = Q}{X \mathrel{\ddot{\preccurlyeq}}_\Gamma Q}\ \text{(S-Var)}$$

$$\frac{P \mathrel{\ddot{\preccurlyeq}}_{\Gamma, X:Q} Q \quad P, Q \text{ sequential}}{\mu_X P \mathrel{\ddot{\preccurlyeq}}_\Gamma Q}\ (\text{S-}\mu\text{L}) \qquad \frac{P \mathrel{\ddot{\preccurlyeq}}_\Gamma Q[\mu_X Q/X]}{P \mathrel{\ddot{\preccurlyeq}}_\Gamma \mu_X Q}\ (\text{S-}\mu\text{R})$$

We will often write $P \mathrel{\ddot{\preccurlyeq}} Q$ instead of $P \mathrel{\ddot{\preccurlyeq}}_\emptyset Q$.

The rules in Definition 7.8 are mostly straightforward:

- (S-Var) uses the environment, requiring the hypothesis that X expands into a behaviour I/O simulated by Q;
- (S-μL) consumes such an hypothesis, introducing recursion on the LHS: the intuition is that, in the rule premise, the LHS is I/O simulated by the RHS when Q replaces X in P's body, similarly to the premise of Lemma 7.7;
- (S-μR) allows to unfold a recursion on the RHS, when going upwards in a derivation.

The relation $\ddot{\preccurlyeq}$ satisfies the following weakening property.

Proposition 7.9 (Weakening for $\ddot{\preccurlyeq}$). *For all Γ such that $X \notin \mathrm{dom}(\Gamma)$, $P \mathrel{\ddot{\preccurlyeq}}_\Gamma Q$ implies $P \mathrel{\ddot{\preccurlyeq}}_{\Gamma, X:R} Q$.*

Proof. By induction on the derivation of $P \mathrel{\ddot{\preccurlyeq}}_\Gamma Q$, the result is immediate on all rules. □

7.3 An I/O simulation-based type system

We can now show that the syntactic rules for $\ddot{\preccurlyeq}_\Gamma$ can be a basis for a type system for $\mathbb{U}^-_{\text{aCCS}}$. The correctness of $\ddot{\preccurlyeq}$ w.r.t. $\ddot{\lessdot}$ is formalised in Corollary 7.11 below; this, in turn, is based on Theorem 7.10, which shows how the environment in each derivation step is used to construct an intermediate I/O simulation relation.

Theorem 7.10. *If $P \mathrel{\ddot{\preccurlyeq}}_\Gamma Q$, then $P \Gamma \mathrel{\ddot{\lessdot}} Q$.*

When $\Gamma = \emptyset$, Theorem 7.10 has the following corollary:

Corollary 7.11. *If $P \mathrel{\ddot{\preccurlyeq}} Q$, then $P \mathrel{\ddot{\lessdot}} Q$.*

We can now define a *syntax-directed* typing judgement, relating CCS$^-$ terms with session types. To this purpose, we exploit the encoding in Definition 2.19.

Definition 7.12 (Type system). *When $[\![T]\!]$ is a CCS$^-$ term, we write $\Gamma \vdash P : T$ whenever $P \ddot{\preccurlyeq}_\Gamma [\![T]\!]$.*

The condition on $[\![T]\!]$ in Definition 7.12 is necessary because the encoding of recursive session types does not always respect condition *b* of Definition 7.6: see, for instance, $U = \mathrm{rec}_Y\, !\mathsf{a}.X \oplus !\mathsf{b}.?\mathsf{c}$ (from Example 6.16) and its encoding $[\![U]\!] = \mu_X !\mathsf{a}.X + !\mathsf{b}.?\mathsf{c}$.

Example 7.13. *From Section 3, recall the bartender's process Q''_B that stops selling beer after a certain hour, the bartender type U_B, and its encoding in CCS (which is also a CCS$^-$ term):*

$$Q''_\mathsf{B} = \mu_Y \big((?\mathsf{aCoffee}.!\mathsf{coffee}.Y \,+\, ?\mathsf{aBeer}\,.\,(!\mathsf{beer}.Y + !\mathsf{no}.Y) \,+\, ?\mathsf{pay}) \\ +\tau.\mu_Z(?\mathsf{aCoffee}.!\mathsf{coffee}.Z \,+\, ?\mathsf{aBeer}.!\mathsf{no}.Z \,+\, ?\mathsf{pay}) \big)$$

$$U_\mathsf{B} = \mathrm{rec}_X\, (?\mathsf{aCoffee}.!\mathsf{coffee}.X \;\&\; ?\mathsf{aBeer}.(!\mathsf{beer}.X \oplus !\mathsf{no}.X) \;\&\; ?\mathsf{pay})$$

$$[\![U_\mathsf{B}]\!] = \mu_X (?\mathsf{aCoffee}.!\mathsf{coffee}.X \,+\, ?\mathsf{aBeer}\,.\,(!\mathsf{beer}.X + !\mathsf{no}.X) \,+\, ?\mathsf{pay})$$

We want to prove $\vdash Q''_\mathsf{B} : U_\mathsf{B}$. First of all, for Q''_B we show a typing derivation for the term under Z-recursion $\mu_Z ?\mathsf{aCoffee}\ldots$. Letting $\Gamma = Z : [\![U_\mathsf{B}]\!]$, we have the derivation shown in Table 6. Secondly, let $\mathcal{D}$ indicate such a derivation, starting from the instance of rule (S-μL) *(i.e., excluding the application of Definition 7.12). We reuse $\mathcal{D}$ in the derivation for the term under Y-recursion in Q''_B, where it provides a premise for applying rule* (+Ext) *from Table 5. Letting $\Gamma' = Y : [\![U_\mathsf{B}]\!]$, we obtain the derivation in Table 7.*

Theorem 7.14 below states the correctness of our typing discipline. Suppose you have a process P with type T, and a process Q with type U. If $T[]$ and $U[]$ are I/O compliant, then $P[]$ and $Q[]$ are I/O compliant, too. Thus, we have that $P[] \parallel Q[]$ is safe (by Theorem 5.17).

Theorem 7.14 (Correctness). *If $\vdash P : T$ and $\vdash Q : U$ with $T[] \ddot{\bowtie} U[]$, then $P[] \ddot{\bowtie} Q[]$.*

Proof. From Definition 7.12 we have $P \ddot{\preccurlyeq} [\![T]\!]$; by Proposition 2.20, Definition 7.8 and Corollary 7.11 it follows $P[] \ddot{\preccurlyeq} T[]$. Similarly, $Q[] \ddot{\preccurlyeq} U[]$. Since $P[] \ddot{\preccurlyeq} T[] \ddot{\bowtie} U[]$, by Theorem 6.12 it follows $P[] \ddot{\bowtie} U[]$. Since $Q[] \ddot{\preccurlyeq} U[] \ddot{\bowtie} P[]$, then by Theorem 6.12 we conclude $Q[] \ddot{\bowtie} P[]$. □

We stress that the above result is obtained just by exploiting the properties of I/O simulation, without explicitly proving subject reduction.

In the synchronous setting we can deduce $T \ddot{\bowtie} U$ either via model checking (since T and U are finite state), or using syntax-driven techniques (e.g., those in [4]); this result is lifted to the async case by Lemma 4.12. If both T and U can be encoded in CCS$^-$, we can reason on $\vdash P : T$ and $\vdash Q : U$ on a syntax-driven basis, through the rules in Definition 7.8.

Note, however, that Theorem 7.14 does *not* require compliance between *synchronous* session types. Therefore, the result also holds e.g. for $T = !\mathsf{a}.?\mathsf{b}$ and $U = !\mathsf{b}.?\mathsf{a}$ — since in the asynchronous setting we have $T[] \ddot{\bowtie} U[]$ (even though $T \not\ddot{\bowtie} U$). This is a departure from traditional session typing systems, that restrict compliance to (synchronous) *duality*; for further discussion, see Examples 7.15 and 7.16 below.

Example 7.15. *Recall $\vdash P''_\mathsf{A} : T''_\mathsf{A}$ from Example 7.5, and consider the bartender processes Q_B, Q''_B and type U_B from Section 3. We can easily obtain $\vdash Q_\mathsf{B} : U_\mathsf{B}$. Therefore, since in Example 4.10 we determined $U_\mathsf{B}[] \ddot{\bowtie} T''_\mathsf{A}[]$, by Theorem 7.14 we have $Q_\mathsf{B}[] \ddot{\bowtie} P''_\mathsf{A}[]$; hence, by Theorem 5.17 we have that $Q_\mathsf{B}[] \parallel P''_\mathsf{A}[]$ is safe. Note that this result exploits asynchrony* both *via I/O compliance and via typing: in fact, this is based on $U_\mathsf{B}[] \ddot{\bowtie} T''_\mathsf{A}[]$, albeit $U_\mathsf{B} \not\ddot{\bowtie} T''_\mathsf{A}$ (as discussed in Example 4.10); moreover, the typing judgement $\vdash P''_\mathsf{A} : T''_\mathsf{A}$ uses rule* (|L).

Example 7.16. *Recall $\vdash P''_\mathsf{A} : T''_\mathsf{A}$ from Example 7.5, and consider Example 7.13, where we show $\vdash Q''_\mathsf{B} : U_\mathsf{B}$. By Theorem 7.14 we have $Q''_\mathsf{B}[] \ddot{\bowtie} P''_\mathsf{A}[]$, and by Theorem 5.17 we conclude that $Q''_\mathsf{B}[] \parallel P''_\mathsf{A}[]$ is safe. Note that this result, as in Example 7.15, is based on* asynchronous *semantics, and would not hold in the synchronous setting.*

The previous examples (in particular, Example 7.13) show that our syntax-driven rules allow to type an Erlang-style `receive...after...` behaviour, featured in the bartender process.

8 Related work

This work corresponds to a thoroughly revised and improved version of [12]. The main differences between the two papers are summarized as follows:

- the definition and results on safety (Section 5) are new;
- the I/O simulation $\ddot{\preccurlyeq}$ is now larger, and thus able to relate more compliance-preserving behaviours;
- the typing relation in Section 7 has been also enlarged, and its definition streamlined and simplified, bringing it closer to an algorithm;
- we correct a mistake in [12], i.e. the fact that the counterpart of Theorem 6.20 lacked the $\preceq_?$ relation in its hypotheses;
- we include the proofs of all results.

Session types were introduced by Honda *et al.* in [49, 50, 72], as a type system for communication channels in a variant of the π-calculus. The resulting concept of *structured communication-based programming* has been the cornerstone on which a thriving research trend has been developed throughout the following decades. Beyond the π-calculus, session types have been later on applied

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma(Z) = [\![U_{\mathrm{B}}]\!]}{Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![U_{\mathrm{B}}]\!]}\ (\text{S-Var})}{!\mathsf{coffee}.Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![!\mathsf{coffee}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx})}{?\mathsf{aCoffee}.!\mathsf{coffee}.Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![?\mathsf{aCoffee}.!\mathsf{coffee}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx}) \quad \dfrac{\dfrac{\dfrac{\dfrac{\Gamma(Z) = [\![U_{\mathrm{B}}]\!]}{Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![U_{\mathrm{B}}]\!]}\ (\text{S-Var})}{!\mathsf{no}.Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![!\mathsf{no}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx})}{!\mathsf{no}.Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![!\mathsf{beer}.U_{\mathrm{B}} \oplus !\mathsf{no}.U_{\mathrm{B}}]\!]}\ (+\text{Int})}{?\mathsf{aBeer}.!\mathsf{no}.Z \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![?\mathsf{aBeer}.(!\mathsf{beer}.U_{\mathrm{B}} \oplus !\mathsf{no}.U_{\mathrm{B}})]\!]}\ (+\text{Ctx}) \quad \dfrac{\dfrac{}{\mathbf{0} \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![\mathbf{0}]\!]}\ (+\text{Ctx})}{?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![?\mathsf{pay}]\!]}\ (+\text{Ctx})}{?\mathsf{aBeer}.!\mathsf{no}.Z + ?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![?\mathsf{aBeer}.(!\mathsf{beer}.U_{\mathrm{B}} \oplus !\mathsf{no}.U_{\mathrm{B}}) \,\&\, ?\mathsf{pay}]\!]}\ (+\text{Ext})}{?\mathsf{aCoffee}.!\mathsf{coffee}.Z + ?\mathsf{aBeer}.!\mathsf{no}.Z + ?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![?\mathsf{aCoffee}.!\mathsf{coffee}.U_{\mathrm{B}} \,\&\, ?\mathsf{aBeer}.(!\mathsf{beer}.U_{\mathrm{B}} \oplus !\mathsf{no}.U_{\mathrm{B}}) \,\&\, ?\mathsf{pay}]\!]}\ (\text{S-}\mu\text{R})$$

$$\dfrac{?\mathsf{aCoffee}.!\mathsf{coffee}.Z + ?\mathsf{aBeer}.!\mathsf{no}.Z + ?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}}_\Gamma [\![U_{\mathrm{B}}]\!]}{\mu_Y ?\mathsf{aCoffee}.!\mathsf{coffee}.Z + ?\mathsf{aBeer}.!\mathsf{no}.Z + ?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}} [\![U_{\mathrm{B}}]\!]}\ (\text{S-}\mu\text{L})}{\vdash \mu_Z ?\mathsf{aCoffee}.!\mathsf{coffee}.Z + ?\mathsf{aBeer}.!\mathsf{no}.Z + ?\mathsf{pay} : U_{\mathrm{B}}}\ (\text{Definition 7.12})$$

Table 6: Typing derivation for the running example (I).

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma'(Y) = [\![U_{\mathrm{B}}]\!]}{Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![U_{\mathrm{B}}]\!]}\ (\text{S-Var})}{!\mathsf{coffee}.Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![!\mathsf{coffee}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx}) \quad \dfrac{\dfrac{\dfrac{\Gamma'(Y) = [\![U_{\mathrm{B}}]\!]}{Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![U_{\mathrm{B}}]\!]}\ (\text{S-Var})}{!\mathsf{beer}.Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![!\mathsf{beer}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx}) \quad \dfrac{\dfrac{\Gamma'(Y) = [\![U_{\mathrm{B}}]\!]}{Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![U_{\mathrm{B}}]\!]}\ (\text{S-Var})}{!\mathsf{no}.Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![!\mathsf{no}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx})}{!\mathsf{beer}.Y + !\mathsf{no}.Y \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![!\mathsf{beer}.U_{\mathrm{B}} \oplus !\mathsf{no}.U_{\mathrm{B}}]\!]}\ (+\text{Ctx})}{?\mathsf{aBeer}.(!\mathsf{beer}.Y + !\mathsf{no}.Y) \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![?\mathsf{aBeer}.(!\mathsf{beer}.U_{\mathrm{B}} \oplus !\mathsf{no}.U_{\mathrm{B}})]\!]}\ (+\text{Ctx}) \quad \dfrac{\dfrac{}{\mathbf{0} \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![\mathbf{0}]\!]}\ (+\text{Ctx})}{?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![?\mathsf{pay}]\!]}\ (+\text{Ctx})}{?\mathsf{aCoffee}.!\mathsf{coffee}.Y + ?\mathsf{aBeer}.(!\mathsf{beer}.Y + !\mathsf{no}.Y) + ?\mathsf{pay} + \tau.\mu_Z ?\mathsf{aCoffee}.!\mathsf{coffee}.Z + ?\mathsf{aBeer}.!\mathsf{no}.Z + ?\mathsf{pay} \mathrel{\ddot{\preccurlyeq}}_{\Gamma'} [\![U_{\mathrm{B}}]\!]}\ (\text{S-}\mu\text{L})}{Q''_{\mathrm{B}} \mathrel{\ddot{\preccurlyeq}} [\![U_{\mathrm{B}}]\!]}{\vdash Q''_{\mathrm{B}} : U_{\mathrm{B}}}\ (\text{Definition 7.12})}$$

Table 7: Typing derivation for the running example (II). $\mathcal{D}$ is the derivation starting from (S-μL) in Table 6. Note that in this derivation, the environment Γ' is larger than the empty environment at the root of $\mathcal{D}$. However, $\mathcal{D}$ "still works", and can be simply rewritten with Γ' at its root, by Proposition 7.9.

to other calculi and programming languages. However, most approaches do not allow to express the Erlang `receive...after...` construct discussed in Question 1. Also in [62], which devises a session type system for the featherweight Erlang language, such problematic pattern is omitted. While adapting the type system of [62] to cope with the `receive...after...` construct should be feasible, our approach allows the construction of the type system (in our case, the rules for $\ddot{\preccurlyeq}$) to be driven by an explicit underlying semantic notion (I/O simulation).

8.1 Other foundational approaches to session types

Most approaches to session types (like e.g. [73]) focus on a specific syntax of processes, which constrains their LTS; further, they often give semantics to processes, but not to types. A more semantics-oriented approach is the one in [29]. While types and processes follow a given syntax, they are both equipped with a higher-order LTS semantics, and most definitions depend on the LTS semantics, only, without inspecting the syntax of types and processes. For instance, the duality and subtyping relations are based on the *"may output"* and *"must input"* relations, which only rely on the LTS semantics. Our approach shares similar insights: for instance, also our theory deals with inputs and outputs in an asymmetric way, and uses "persistent inputs" in key definitions, such as I/O simulation and safety.

The main difference between our work and [29] is that we do not fix a syntax of processes/types, by focusing instead on a first-order I/O LTS populated by *both* processes and types. In this way, we can reason on behaviours which are not directly expressible in [29], e.g. the Erlang pattern `receive...after...`. However, while our types are *first*-order LTSs, those in [29] are *higher*-order, as action labels can be types themselves. Both our theory and [29] consider some form of asynchrony: while we deal (in the same framework) with synchronous and FIFO-buffered behaviours, [29] implements a weaker form of asynchrony, inspired by [33].

Our approach is similar to [28,31,32] with respect to various aspects: for instance, (a) the common inspiration to [39] for the (synchronous) session types semantics, (b) the idea of representing processes and contracts/types in the same LTS, (c) the aim to overcome the rigid internal/external choices dichotomy required by session types, emerging both at type and process levels. In [31], it is assumed that some type system can *abstract* processes P, Q (expressed in *any* calculus) into contracts. This type system must be *"consistent"* and *"informative"*, by preserving some essential properties

like e.g. visible actions and internal non-determinism. A result in [31] is that if the abstractions of P, Q are compliant, then P, Q will be compliant as well. A similar abstraction is at the root of our approach.

The works [47,44] study subtyping for binary session types, focussing on *communication channel* replaceability [70]. The intuition is the following: consider a type U and its subtype T, and a program P interacting through a channel; then, a channel typed with T is "less demanding" for P w.r.t. a channel typed with U — i.e., if P correctly uses an U-typed channel, then it can also correctly use a T-typed channel. This holds because T allows P to choose among "more" possible outputs, and mandates "less" inputs to be enabled, w.r.t. U. In the present work, we look at behaviours as if we were on the other endpoint of a channel, and so the co/contravariance rules are reversed. As a consequence, a subtype of a behaviour q is I/O compliant with all the behaviours compliant with q (and possibly with more, see Theorem 6.12). Subtyping relations with the same ordering of $\ddot{\leqslant}$ are studied in [26,27,40], while the direction of the subcontract relations in [29,31] is opposite to ours. A discussion on the channel-oriented and process-oriented views of subtyping is in [46]. Besides the direction of the relation, there are other differences between [31] and this paper. For instance, some desirable subtypings like $?\mathsf{a} \,\&\, ?\mathsf{b} \not\sqsubseteq ?\mathsf{a}$ do not hold in the strong subcontract relation of [31]. However, these are restored through a "weak" subcontract relation, which exploits filters to resolve external non-determinism.

In [4,5,6], session types are equipped with first-order and higher-order LTS semantics, and studied under different notions of client-server compliance (e.g., allowing the client to terminate interaction or to skip messages). A similar line of research is taken in [14,15], which shows an encoding of session types into the contracts of [31], which preserves (and reflects) a subtype relation. Our paper is also focussed on preorders, but compared to [4,5,6,14,15] we work in a language-independent setting, restricting to first-order behaviours.

8.2 Multiple participants and multiple sessions

Many recent works extend the session types discipline to the multiparty case, starting from [51]. In this setting, the application designer specifies the overall communication behaviour of a set of participants through a *choreography*, which enjoys some correctness properties (e.g., safety and progress). The overall application is the result of the composition of a set of processes, which are distributed over the network, and interact through a multiparty session. To ensure the correctness of this composition, the choreography is projected into a set of *local session types*, which abstract the end-point communication behaviour of processes: if each process is type-checked against its session type, the composition of services preserves the properties enjoyed by the choreography. The crucial technical difference w.r.t. binary session types is that, in *local* multiparty session types, input/output actions target the specific participant from/to which a message should be received/sent; correspondingly, a process typechecks only if its input/output actions target the correct participants.

Extending our theory of binary session types to the multiparty case would require to extend the compliance and subtyping relations to capture the role of each type/process, and then to produce the syntax-based typing rules. Some insights come from [30], which aims at a semantic understanding of choreographies.

Another orthogonal issue of the multiparty case is how to guarantee correct interactions in the presence of multiple *interleaved* sessions. To deal with this case, one could take inspiration from [11,67], which introduce type systems for ensuring liveness in the presence of multiple interleaved sessions, and [13], which relates deadlock-freedom within single sessions to deadlock-freedom of processes which juggle with multiple sessions.

8.3 Asynchrony and session types

Asynchronous binary session types have been addressed in [65], where type equivalence up-to buffering was defined over traces, and then approximated via syntax-based rules. In the setting of multiparty session types, [61,63,64] study subtyping with "safe" partial commutativity allowed by asynchrony. These works also tackle possible optimisations allowed by buffering, similarly to our asynchronous Alice-Bartender interaction (Section 3). In [35,36] it is noted that the asynchronous subtyping of [61] guarantees progress, but does not offer guarantees about orphan messages: e.g., in case of recursive types, a program matching the subtype may never read some messages from its queues. For this reason, [35,36] propose a new asynchronous subtyping relation for binary session types, and they prove it to be precise w.r.t. orphan messages and other errors.

While our focus is on preservation of compliance and subtyping under buffering, the work [54] studies a bisimulation to relate processes which communicate via unbounded buffers. The expressiveness of (multiparty) session types with different kinds of FIFO buffers is investigated in [41]. In particular, they prove that, in the binary case, one output FIFO buffer per participant (as in our Definition 2.11) allows to obtain the same communication traces as other, more complex representations — e.g., adding *both* input and output FIFOs,

as in [54]. Several other works address asynchronous session types, e.g. [45,37,18].

8.4 Compliance and safety

Many different notions of "correct interaction" have been considered in the literature. These notions typically restrict progress, by preventing infinite traces of two behaviours which never interact [55], or by exploiting *must*-testing [2] and *should*-testing [21,1]. Our I/O compliance is related in [10] to some of these notions.

The notion of compliance in [66] can relate behaviours which enjoy progress after a (finite-state) orchestrator has rearranged their messages. This notion is unrelated to I/O compliance: on the one hand, the latter cannot rearrange messages; on the other hand, I/O compliance has no fixed bound on the size of the buffers. For instance, the async behaviours $!\mathsf{a}.?\mathsf{b}.!\mathsf{a}^2.?\mathsf{b}^2\cdots!\mathsf{a}^n.?\mathsf{b}^n\cdots$ and $!\mathsf{b}.?\mathsf{a}.!\mathsf{b}^2.?\mathsf{a}^2\cdots!\mathsf{b}^n.?\mathsf{a}^n\cdots$ (where $!\mathsf{a}^m$ is a sequence of m $!\mathsf{a}$) are I/O compliant, but they are *not* compliant according to [66], as orchestrators must be finite-state.

Several works define notions of compliance for multiparty interactions [22,23,9,42]. For instance, the notions in [22,23] are based on should-testing on CCS-like processes which interact through FIFO buffers. This approach is similar to our pairing of session types and CCS terms with queues, except that we embed buffers *within* processes, and represent enqueueing with an internal action, whereas [23] adds explicit labels signalling enqueueing/dequeueing of messages.

Compliance among CFSMs [19] is usually called *safety*, and it typically requires deadlock-freedom and the absence of orphan message and unspecified reception configurations. In our framework, two behaviours with unbounded buffers can be seen as a system of CFSMs, albeit the resulting LTS is slightly different: e.g., in our setting, an output is added to a buffer with a τ-transition, and consumed with a τ-synchronisation; in CFSMs, an output $!\mathsf{a}$ is buffered with a visible $!\mathsf{a}$-transition, and is consumed with a visible $?\mathsf{a}$-transition.

The notion of safety introduced in Section 5 is inspired to those for CFSMs, although deadlock-freedom is not explicitly mentioned in Definition 5.15, as it is already implied by the other requirements (Proposition 5.16). As discussed in Example 5.11, our notion of orphan message is stricter than the ones in [56,42]. For instance, consider $T = \mathrm{rec}_X\, !\mathsf{a}.X$ and $U = \mathrm{rec}_Y\, !\mathsf{b}.Y$. The composition $T[] \parallel U[]$ is deadlock-free and does not reach unspecified reception configurations; by Definition 5.1, however, it is an orphan message configuration (because outputs are sent but never received), and therefore not safe. According to the definitions in [56,42], instead, such a configuration is considered safe: in fact, a message is an "orphan" when the machines terminate with non-empty buffers.

A notion of "orphan message" closer to ours is studied in [36,35], where a process is said to "orphan" a buffered message when it has no chance to read it in the future — even though the process itself is not terminated. A difference w.r.t. our approach is that, in [36,35], buffers are not introduced at the types level, but at the level of π-calculus processes. The *asynchronous subtyping* in [36,35] allows to swap the order of input/output actions under certain conditions. Our subtyping (i.e., I/O simulation $\ddot{\leqslant}$) is generally more restrictive in the reordering of input/output actions, even in presence of buffers: e.g., $!\mathsf{a}.?\mathsf{b}[] \not\ddot{\leqslant} ?\mathsf{b}.!\mathsf{a}[]$, but the two behaviours would be related by the subtyping in [36,35]. However, in our setting, we support a similar kind of swapping through I/O compliance: e.g., $!\mathsf{a}.?\mathsf{b}[] \ddot{\bowtie} !\mathsf{b}.?\mathsf{a}[]$, which is *not* allowed by the "compliance" (i.e., the session types duality) adopted in [36,35]. Hence, we conjecture that the type relations captured by type duality and asynchronous subtyping in [36,35] correspond to those captured by combining $\ddot{\bowtie}$ and $\ddot{\leqslant}$ on session types with asynchronous semantics; the latter, however, can cover more general cases of asynchrony (e.g., the parallel composition in Question 3).

8.5 Timeouts and exceptions

The works [8,17,16,25,52] enrich session types with new constructs and messages for handling timeouts and exceptions. In particular, [25,52] mark where an exceptional event might occur within a protocol, how it can be handled at runtime; moreover, they introduce run-time monitoring to ensure proper coordination of the communicating processes. The works [8,17,16] add timing constraints to, respectively, binary session types, multi-party session types, and CFSMs. Although the present work does not fully address these problems, we can model some simple cases of exception handling without requiring dedicated constructs. For instance, our running example in Section 3 shows a case in which a timeout event (abstracted as a τ-transition) interrupts an external choice. Such events can also be expressed in the CCS^- fragment used in Section 7, where subtyping is syntactically defined by the relation $\ddot{\preccurlyeq}_\Gamma$.

9 Conclusions

We have studied session types from a semantic perspective. To this purpose, we have interpreted some standard

notions on session types on the general semantic framework of Labelled Transition Systems. This allowed us to extend some notions and results beyond session types, in a syntax-agnostic way. While we mostly focused on behaviours arising from (synchronous and asynchronous) session types and CCS, we believe that our approach can be applied to analyse the properties of other behaviours — e.g. the LTS semantics of other process calculi and programming languages.

9.1 Summary of the main results

We now briefly recap the main results of our paper. For simplicity, here we only use the symmetric I/O compliance relation $\bowtie$. A first remarkable result, following from Theorem 4.9, is that I/O compliance is stricter than progress, and that the two relations coincide on synchronous session types:

Theorem. *If $p \bowtie q$, then $p \dashv\vdash q$. If p, q are synchronous session types and $p \dashv\vdash q$, then $p \bowtie q$.*

In Theorems 5.17 and 5.20 we have shown that, on general behaviours, I/O compliance implies safety, i.e. absence of deadlocks, orphan messages and unspecified reception configurations. Moreover, the two relations coincide for asynchronous session types:

Theorem. *If $p \bowtie q$, then $p \parallel q$ is safe. If p, q are asynchronous session types and $p \parallel q$ is safe, then $p \bowtie q$.*

In Theorems 6.10 and 6.12 we have shown that I/O simulation $\leqslant$ is an I/O compliance-preserving (and thus, safety-preserving) preorder for general behaviours.

Theorem. *The relation $\leqslant$ is a preorder, and $p \leqslant q \bowtie r$ implies $p \bowtie r$.*

In Theorems 4.13 and 6.20 we have shown that I/O compliance is preserved when passing from synchronous to asynchronous semantics of session types; to preserve I/O simulation, an additional relation $\preceq_?$ (Definition 6.18) is required.

Theorem. *For all session types T, U:*

- *If $T \bowtie U$, then $T[] \bowtie U[]$.*
- *If $T \left(\leqslant \cap \preceq_?\right) U$, then $T[] \leqslant U[]$.*

Our typing system is semantically grounded on $\leqslant$ and $\bowtie$, and it enjoys the following correctness property: in the asynchronous setting, if two CCS processes are typeable with session types which are I/O compliant, then also the processes are I/O compliant. Therefore, they can safely interact.

Theorem 7.14 (Correctness)**.** *If $\vdash P : T$ and $\vdash Q : U$ with $T[] \bowtie U[]$, then $P[] \bowtie Q[]$.*

Remarkably, our type system allows to syntactically type our examples from Section 3. Since these examples feature asynchronous semantics and external choices mixed with τ-transitions, at least in this case we can say that our approach gives a positive answer to Questions 2 and 3 in Section 1.

9.2 Future work

A possible extension to our work is to allow for the communication of channels, which is a common feature in many works on session types [53]. This should be feasible by enriching the labels of I/O LTSs to also cater for channel transmission. Modelling higher-order calculi could be feasible by employing a translation of higher-order LTSs into first-order ones, as done e.g. in [58].

Some questions about I/O compliance/simulation and our type system remain open. For instance, in this work we have not dealt with the decidability of these notions. Clearly, I/O compliance/simulation are decidable for finite-state behaviours, while they seem to be undecidable in the general case, and also in the specific case of asynchronous session types. For instance, the works [48,11] prove that, for (binary, first-order) asynchronous session types, some compliance relations which are close to our notion of safety are undecidable. Instead, the works [57,20] prove undecidability of asynchronous subtyping. We do not expect that these results apply directly to $\leqslant$ on asynchronous session types, since $\leqslant$ is more restrictive and closer to the (decidable) synchronous subtyping (as discussed in Section 8). Some works propose decidable, sound approximate techniques to check safety in CFSMs [56]. Since asynchronous session types are a special case of CFSMs, these techniques can also be applied in this setting. We expect that analogous results can be obtained for I/O simulation: in particular, is should be possible to soundly approximate I/O simulation via an axiomatization, in the spirit of our syntactic relation $\preccurlyeq$, which we conjecture to be decidable.

Note that $\leqslant$ is not a precongruence on CCS: for instance, $\tau.?\mathsf{a}.P \leqslant ?\mathsf{a}.P$ but $?\mathsf{b} + \tau.?\mathsf{a}.P \not\leqslant ?\mathsf{b} + ?\mathsf{a}.P$. A relevant question on I/O simulation is how to devise a stricter relation which is also a precongruence on an expressive calculus, like CCS or CCS$^-$. In CCS$^-$, the task is simplified by the requirement of guarded sums (Definition 7.6), that rules out the example above. The main difficulties, however, are due to parallel composition: e.g., we have $?\mathsf{a} + ?\mathsf{b}.\mathbf{0} \leqslant ?\mathsf{a}$ and $?\mathsf{b}.!\mathsf{c} \leqslant ?\mathsf{b}.!\mathsf{c}$, but $?\mathsf{a} + ?\mathsf{b}.\mathbf{0} \mid ?\mathsf{b}.!\mathsf{c} \not\leqslant ?\mathsf{a} \mid ?\mathsf{b}.!\mathsf{c}$ — because the (persistent) $?\mathsf{b}$-prefix of $?\mathsf{b}.\mathbf{0}$ (on the LHS) and $?\mathsf{b}.!\mathsf{c}$ (on the RHS) cause Definition 6.4 to require $?\mathsf{b}.!\mathsf{c} \mid \mathbf{0} \leqslant ?\mathsf{a} \mid !\mathsf{c}$ (which is false). For this reason, rule (|LR) (Table 5) constrains the

inputs of the composed processes; different approaches might be possible, and can be investigated — albeit they seem to lead to rather strict subsets of $\ddot{\leqslant}$.

Finally, the syntactic relation $\ddot{\preccurlyeq}$ in Section 7.1 can be extended in several directions. The addition of new rules is generally possible and pretty straightforward, under the approach we followed in Table 5: assuming some I/O simulations in the premises, and constructing a new one in the conclusions. As an example, the "τ laws" from [59, §3.2] can be integrated. Another possible extension to $\ddot{\preccurlyeq}$ is an additional rule for "interruptible" external choices, such as:

$$\frac{\begin{array}{ll} Q \equiv \sum_{i\in I} ?\mathsf{a}_i.Q_i & K \neq \emptyset \\ \forall j \in J, i \in I \,.\, \mathsf{a}_j = \mathsf{a}_i \implies P_j \ddot{\leqslant} Q_i & \forall k \in K \,.\, P_k \ddot{\leqslant} Q \end{array}}{\sum_{j\in J}(?\mathsf{a}_j.P_j) + \sum_{k\in K}\tau.P_k \;\ddot{\leqslant}\; Q} \;(+\textsc{Ext2})$$

The difference between (+Ext) in Table 5 and (+Ext2) above is that the latter does *not* require the "immediate" input transitions of P to be a superset of those of Q — provided that the continuations of overlapping inputs are within the relation, and also the continuation after each τ-branch (of which at least one must exist) is within the relation. With such rule, the following bartender process has also type U_{B} (Section 3):

$$\mu_Y\big(?\mathsf{aCoffee}.!\mathsf{coffee}.Y \\ + \tau.\mu_Z(?\mathsf{aCoffee}.!\mathsf{coffee}.Z + ?\mathsf{aBeer}.!\mathsf{beer}.Z + ?\mathsf{pay})\big)$$

This process models a bartender that initially only serves coffees, and after an internal move τ, starts serving both coffees and beers. The further investigation of these extensions of $\ddot{\preccurlyeq}$ is left as future work.

References

1. van der Aalst, W.M.P., Lohmann, N., Massuthe, P., Stahl, C., Wolf, K.: Multiparty contracts: Agreeing and implementing interorganizational processes. Comput. J. **53**(1), 90–106 (2010). DOI 10.1093/comjnl/bxn064
2. Acciai, L., Boreale, M., Zavattaro, G.: Behavioural contracts with request-response operations. In: COORDINATION, *LNCS*, vol. 6116, pp. 16–30. Springer (2010). DOI 10.1007/978-3-642-13414-2_2
3. de Alfaro, L., Henzinger, T.A.: Interface automata. In: ACM SIGSOFT, pp. 109–120. ACM (2001). DOI 10.1145/503209.503226
4. Barbanera, F., de'Liguoro, U.: Two notions of sub-behaviour for session-based client/server systems. In: PPDP, pp. 155–164. ACM (2010). DOI 10.1145/1836089.1836109
5. Barbanera, F., de'Liguoro, U.: Sub-behaviour relations for session-based client/server systems. Mathematical Structures in Computer Science **25**(6), 1339–1381 (2015). DOI 10.1017/S096012951400005X
6. Barbanera, F., de' Liguoro, U.: Loosening the notions of compliance and sub-behaviour in client/server systems. In: ICE, *EPTCS*, vol. 166, pp. 94–110 (2014). DOI 10.4204/EPTCS.166.10
7. Barendregt, H.: The Lambda Calculus: Its Syntax and Semantics. North Holland (1984)
8. Bartoletti, M., Cimoli, T., Murgia, M., Podda, A.S., Pompianu, L.: Compliance and subtyping in timed session types. In: FORTE 2015, pp. 161–177 (2015). DOI 10.1007/978-3-319-19195-9_11
9. Bartoletti, M., Cimoli, T., Pinna, G.M., Zunino, R.: Contracts as games on event structures. J. Log. Algebr. Meth. Program. **85**(3), 399–424 (2016). DOI 10.1016/j.jlamp.2015.05.001
10. Bartoletti, M., Cimoli, T., Zunino, R.: Compliance in behavioural contracts: a brief survey. In: Programming Languages with Applications to Biology and Security, *LNCS*, vol. 9465. Springer (2015). DOI 10.1007/978-3-319-25527-9_9
11. Bartoletti, M., Scalas, A., Tuosto, E., Zunino, R.: Honesty by Typing. Logical Methods in Computer Science **Volume 12, Issue 4** (2016). DOI 10.2168/LMCS-12(4:7)2016
12. Bartoletti, M., Scalas, A., Zunino, R.: A semantic deconstruction of session types. In: CONCUR, *LNCS*, vol. 8704, pp. 402–418. Springer (2014). DOI 10.1007/978-3-662-44584-6_28
13. Bartoletti, M., Zunino, R.: On the decidability of honesty and of its variants. In: Web Services, Formal Methods, and Behavioral Types, *LNCS*, vol. 9421, pp. 143–166. Springer (2015). DOI 10.1007/978-3-319-33612-1_9
14. Bernardi, G., Hennessy, M.: Modelling session types using contracts. In: SAC, pp. 1941–1946. ACM (2012). DOI 10.1145/2245276.2232097
15. Bernardi, G., Hennessy, M.: Using higher-order contracts to model session types (extended abstract). In: CONCUR, *LNCS*, vol. 8704, pp. 387–401. Springer (2014). DOI 10.1007/978-3-662-44584-6_27
16. Bocchi, L., Lange, J., Yoshida, N.: Meeting deadlines together. In: CONCUR, *LIPIcs*, vol. 42, pp. 283–296. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015). DOI 10.4230/LIPIcs.CONCUR.2015.283
17. Bocchi, L., Yang, W., Yoshida, N.: Timed multiparty session types. In: CONCUR, *LNCS*, vol. 8704, pp. 419–434. Springer (2014). DOI 10.1007/978-3-662-44584-6_29
18. Bonelli, E., Compagnoni, A.: Multipoint session types for a distributed calculus. In: Trustworthy Global Computing, *LNCS*, vol. 4912, pp. 240–256. Springer (2008). DOI 10.1007/978-3-540-78663-4_17
19. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983). DOI 10.1145/322374.322380
20. Bravetti, M., Carbone, M., Zavattaro, G.: Undecidability of asynchronous session subtyping. CoRR **abs/1611.05026** (2016). URL http://arxiv.org/abs/1611.05026
21. Bravetti, M., Zavattaro, G.: Contract based multi-party service composition. In: FSEN, *LNCS*, vol. 4767, pp. 207–222. Springer (2007). DOI 10.1007/978-3-540-75698-9_14
22. Bravetti, M., Zavattaro, G.: Towards a unifying theory for choreography conformance and contract compliance. In: Software Composition, *LNCS*, vol. 4829, pp. 34–50. Springer (2007). DOI 10.1007/978-3-540-77351-1_4
23. Bravetti, M., Zavattaro, G.: Contract compliance and choreography conformance in the presence of message queues. In: WS-FM, *LNCS*, vol. 5387, pp. 37–54. Springer (2008). DOI 10.1007/978-3-642-01364-5_3
24. Brinksma, E., Rensink, A., Vogler, W.: Fair testing. In: CONCUR, *LNCS*, vol. 962, pp. 313–327. Springer (1995). DOI 10.1007/3-540-60218-6_23
25. Capecchi, S., Giachino, E., Yoshida, N.: Global Escape in Multiparty Sessions. In: FSTTCS, *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 8, pp. 338–351 (2010). DOI 10.4230/LIPIcs.FSTTCS.2010.338

26. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centred programming for web services. In: ESOP, *LNCS*, vol. 4421, pp. 2–17. Springer (2007). DOI 10.1007/978-3-540-71316-6_2
27. Carbone, M., Honda, K., Yoshida, N.: Structured communication-centered programming for web services. ACM Trans. Program. Lang. Syst. **34**(2), 8:1–8:78 (2012). DOI 10.1145/2220365.2220367
28. Carpineti, S., Castagna, G., Laneve, C., Padovani, L.: A formal account of contracts for Web Services. In: WS-FM (2006)
29. Castagna, G., Dezani-Ciancaglini, M., Giachino, E., Padovani, L.: Foundations of session types. In: Proceedings of the 11th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP '09, pp. 219–230. ACM (2009). DOI 10.1145/1599410.1599437
30. Castagna, G., Dezani-Ciancaglini, M., Padovani, L.: On global types and multi-party session. Logical Methods in Computer Science **8**(1) (2012)
31. Castagna, G., Gesbert, N., Padovani, L.: A theory of contracts for Web services. ACM TOPLAS **31**(5) (2009)
32. Castagna, G., Padovani, L.: Contracts for mobile processes. In: CONCUR, *LNCS*, vol. 5710. Springer (2009). DOI 10.1007/978-3-642-04081-8_15
33. Castellani, I., Hennessy, M.: Testing theories for asynchronous languages. In: V. Arvind, S. Ramanujam (eds.) FSTTCS, *LNCS*, vol. 1530, pp. 90–101. Springer (1998). DOI 10.1007/978-3-540-49382-2_9
34. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Inf. Comput. **202**(2), 166–190 (2005). DOI 10.1016/j.ic.2005.05.006
35. Chen, T.C., Dezani-Ciancaglini, M., Scalas, A., Yoshida, N.: On the preciseness of subtyping in session types. Logical Methods in Computer Science (2017). To appear
36. Chen, T.C., Dezani-Ciancaglini, M., Yoshida, N.: On the preciseness of subtyping in session types. In: PPDP, pp. 146–135. ACM Press (2014)
37. Coppo, M., Dezani-Ciancaglini, M., Yoshida, N.: Asynchronous session types and progress for object oriented languages. In: FMOODS, *LNCS*, vol. 4468. Springer (2007). DOI 10.1007/978-3-540-72952-5_1
38. De Nicola, R., Hennessy, M.: Testing equivalences for processes. Theor. Comput. Sci. **34**, 83–133 (1984). DOI 10.1016/0304-3975(84)90113-0
39. De Nicola, R., Hennessy, M.: CCS without tau's. In: TAPSOFT, Vol.1 (1987)
40. Demangeon, R., Honda, K.: Full abstraction in a subtyped pi-calculus with linear types. In: CONCUR, pp. 280–296 (2011). DOI 10.1007/978-3-642-23217-6_19
41. Demangeon, R., Yoshida, N.: On the Expressiveness of Multiparty Sessions. In: FSTTCS (2015). DOI 10.4230/LIPIcs.FSTTCS.2015.560
42. Deniélou, P.M., Yoshida, N.: Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In: ICALP (2013)
43. Ericsson Computer Science Laboratory: The Erlang programming language (2015). http://erlang.org/
44. Gay, S., Hole, M.: Subtyping for session types in the Pi calculus. Acta Inf. **42**(2) (2005). DOI 10.1007/s00236-005-0177-z
45. Gay, S., Vasconcelos, V.T.: Asynchronous functional session types. Tech. Rep. 2007–251, University of Glasgow (2007)
46. Gay, S.J.: Subtyping Supports Safe Session Substitution, pp. 95–108. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-30936-1_5
47. Gay, S.J., Hole, M.: Types and subtypes for client-server interactions. In: ESOP, pp. 74–90 (1999). DOI 10.1007/3-540-49099-X_6
48. Gouda, M., Manning, E., Yu, Y.: On the progress of communication between two finite state machines. Information and Control **63**(3), 200–216 (1984). DOI 10.1016/S0019-9958(84)80014-5
49. Honda, K.: Types for dyadic interaction. In: CONCUR (1993)
50. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type discipline for structured communication-based programming. In: ESOP (1998)
51. Honda, K., Yoshida, N., Carbone, M.: Multiparty asynchronous session types. In: POPL (2008). DOI 10.1145/1328438.1328472
52. Hu, R., Neykova, R., Yoshida, N., Demangeon, R., Honda, K.: Practical interruptible conversations - distributed dynamic verification with session types and python. In: RV, pp. 130–148 (2013). DOI 10.1007/978-3-642-40787-1_8
53. Hüttel, H., Lanese, I., Vasconcelos, V.T., Caires, L., Carbone, M., Deniélou, P., Mostrous, D., Padovani, L., Ravara, A., Tuosto, E., Vieira, H.T., Zavattaro, G.: Foundations of session types and behavioural contracts. ACM Comput. Surv. **49**(1), 3:1–3:36 (2016). DOI 10.1145/2873052
54. Kouzapas, D., Yoshida, N., Honda, K.: On asynchronous session semantics. In: FMOODS/FORTE (2011)
55. Laneve, C., Padovani, L.: The *Must* preorder revisited. In: Proc. CONCUR, pp. 212–225 (2007). DOI 10.1007/978-3-540-74407-8_15
56. Lange, J., Tuosto, E., Yoshida, N.: From communicating machines to graphical choreographies. In: Proc. POPL, pp. 221–232. ACM (2015). DOI 10.1145/2676726.2676964
57. Lange, J., Yoshida, N.: On the undecidability of asynchronous session subtyping. In: FoSSaCS (2017)
58. Madiot, J.M., Pous, D., Sangiorgi, D.: Bisimulations up-to: Beyond first-order transition systems. In: P. Baldan, D. Gorla (eds.) CONCUR 2014 – Concurrency Theory, *LNCS*, vol. 8704, pp. 93–108. Springer (2014). DOI 10.1007/978-3-662-44584-6_8
59. Milner, R.: Communication and concurrency. Prentice-Hall, Inc. (1989)
60. Milner, R., Parrow, J., Walker, D.: A Calculus of Mobile Processes, I and II. Information and Computation **100**(1) (1992)
61. Mostrous, D.: Session types in concurrent calculi: Higher-order processes and objects. Ph.D. thesis, Imperial College London (2009)
62. Mostrous, D., Vasconcelos, V.T.: Session typing for a featherweight Erlang. In: COORDINATION (2011)
63. Mostrous, D., Yoshida, N.: Session-based communication optimisation for higher-order mobile processes. In: TLCA, pp. 203–218 (2009). DOI 10.1007/978-3-642-02273-9_16
64. Mostrous, D., Yoshida, N., Honda, K.: Global principal typing in partially commutative asynchronous sessions. In: ESOP, pp. 316–332 (2009). DOI 10.1007/978-3-642-00590-9_23
65. Neubauer, M., Thiemann, P.: Session types for asynchronous communication. Universität Freiburg (2004)
66. Padovani, L.: Contract-based discovery of web services modulo simple orchestrators. Theor. Comput. Sci. **411**(37) (2010)
67. Padovani, L., Vasconcelos, V.T., Vieira, H.T.: Typing liveness in multiparty communicating systems. In: COORDINATION (2014)
68. Park, D.: Concurrency on automata and infinite sequences. In: Conf. on Theoretical Computer Science, *LNCS*, vol. 104, p. 167–183. Springer (1981)

69. Pierce, B.: Types and programming languages. MIT Press, Cambridge, Mass (2002)
70. Pierce, B.C., Sangiorgi, D.: Typing and subtyping for mobile processes. Mathematical Structures in Computer Science **6**(5), 409–453 (1996)
71. Sangiorgi, D., Walker, D.: The π-calculus: A Theory of Mobile Processes. Cambridge University Press (2001)
72. Takeuchi, K., Honda, K., Kubo, M.: An interaction-based language and its typing system. In: PARLE (1994)
73. Vasconcelos, V.: Fundamentals of session types. In: M. Bernardo, L. Padovani, G. Zavattaro (eds.) Formal Methods for Web Services, *LNCS*, vol. 5569, pp. 158–186. Springer (2009). DOI 10.1007/978-3-642-01918-0_4

A Proofs for Section 2

Definition A.1. $\equiv$ *is the largest relation between session types* coinductively *defined by the rules:*

$$\frac{U \equiv T[\mathrm{rec}_X\, T/X]}{U \equiv \mathrm{rec}_X\, T}\ (\text{Eq-RecR}) \qquad \frac{T[\mathrm{rec}_X\, T/X] \equiv U}{\mathrm{rec}_X\, T \equiv U}\ (\text{Eq-RecL})$$

$$\frac{\forall i \in I\,.\, T_i \equiv T_i'}{\&_{i\in I}?\mathsf{a}_i.T_i \equiv \&_{i\in I}?\mathsf{a}_i.T_i'}\ (\text{Eq-Ctx}\&) \qquad \frac{\forall i \in I\,.\, T_i \equiv T_i'}{\oplus_{i\in I}!\mathsf{a}_i.T_i \equiv \oplus_{i\in I}!\mathsf{a}_i.T_i'}\ (\text{Eq-Ctx}\oplus)$$

Proposition A.2. $\equiv$ *is an equivalence relation for session types.*

Proof. Reflexivity of $\equiv$ is proved by showing that each pair (T,T) in the identity relation between session types satisfies the rules in Definition A.1.

Symmetry of $\equiv$ is proved by showing that each pair (T,U) in the relation $\mathcal{R} = \{(T,U) \mid U \equiv T\}$ satisfies the rules in Definition A.1.

Transitivity of $\equiv$ is proved by showing that each pair (T,U) in the relation $\mathcal{R} = \{(T,U) \mid \exists T' :\ T \equiv T' \text{ and } T' \equiv U\}$ satisfies the rules in Definition A.1. □

Proposition A.3 below shows that the syntactic form of synchronous session types can be determined by observing their transitions.

Proposition A.3. *For all T:*

(i) $T \xrightarrow{?\mathsf{a}}$ *iff* $T \equiv ?\mathsf{a}.T' \,\&\, \&_{i\in I}?\mathsf{b}_i.T_i$;
(ii) $T \xrightarrow{\tau}$ *iff* $T \equiv \oplus_{i\in I}!\mathsf{b}_i.T_i$, *with* $|I| > 1$;
(iii) $T \xrightarrow{!\mathsf{a}}$ *iff* $T \equiv !\mathsf{a}.T'$.

Proof. For all items *(i)*–*(iii)*, the $\Longleftarrow$ direction follows from Definition 2.10.

For the $\Longrightarrow$ direction, we proceed by induction on the rules in Definition 2.10 generating the transitions:

- item *(i)*. The $?\mathsf{a}$-transition of T can be generated in two ways:
  - *a.* by rule (TExt) (base case). Then, $T = ?\mathsf{a}T' \& \&_{i\in I}?\mathsf{b}_i.T_i$. We conclude by reflexivity of $\equiv$ (Proposition A.2);
  - *b.* by rule (TRec) (inductive case). Then, $T = \mathrm{rec}_X\, T''$, with premise $T''[\mathrm{rec}_X\, T''/X] \xrightarrow{?\mathsf{a}}$. By the induction hypothesis, we have $T''[\mathrm{rec}_X\, T''/X] \equiv ?\mathsf{a}.T' \,\&\, \&_{i\in I}?\mathsf{b}_i.T_i$; by reflexivity of $\equiv$ (Proposition A.2) and (Eq-RecR), we have $T \equiv T''[\mathrm{rec}_X\, T''/X]$ — and we conclude by transitivity of $\equiv$ (Proposition A.2);
- item *(ii)*. The τ-transition of T can be generated in two ways:
  - *a.* by rule (TInt) (base case). Then, $T = \oplus_{i\in I}!\mathsf{b}_i.T_i$, with $|I| > 1$. We conclude by reflexivity of $\equiv$ (Proposition A.2);
  - *b.* by rule (TRec) (inductive case). Then, $T = \mathrm{rec}_X\, T''$, with premise $T''[\mathrm{rec}_X\, T''/X] \xrightarrow{\tau}$. By the induction hypothesis, we have $T''[\mathrm{rec}_X\, T''/X] \equiv \oplus_{i\in I}!\mathsf{b}_i.T_i$, with $|I| > 1$; by reflexivity of $\equiv$ (Proposition A.2) and (Eq-RecR), we have $T \equiv T''[\mathrm{rec}_X\, T''/X]$ — and we conclude by transitivity of $\equiv$ (Proposition A.2);
- item *(iii)*. The $!\mathsf{a}$-transition of T can be generated in two ways:
  - *a.* by rule (TOut) (base case). Then, $T = !\mathsf{a}.T'$. We conclude by reflexivity of $\equiv$ (Proposition A.2);
  - *b.* by rule (TRec) (inductive case). Then, $T = \mathrm{rec}_X\, T''$, with premise $T''[\mathrm{rec}_X\, T''/X] \xrightarrow{!\mathsf{a}}$. By the induction hypothesis, we have $T''[\mathrm{rec}_X\, T''/X] \equiv !\mathsf{a}.T'$; by reflexivity of $\equiv$ (Proposition A.2) and (Eq-RecR), we have $T \equiv T''[\mathrm{rec}_X\, T''/X]$ — and we conclude by transitivity of $\equiv$ (Proposition A.2). □

The main semantic differences between synchronous and asynchronous session types are formalised in Proposition A.4 below: roughly, asynchrony turns sequences of outputs into sequences of τs, but input transitions and τ-transitions are preserved. For example, we have:

(i) $?\mathsf{a}.\mathbf{0} \xrightarrow{?\mathsf{a}} \mathbf{0}$ becomes $?\mathsf{a}.\mathbf{0}[] \xrightarrow{?\mathsf{a}} \mathbf{0}[]$ (i.e., input transitions are preserved by buffering);

(ii) $!\mathsf{a}\mathbf{0} \oplus !\mathsf{b}\mathbf{0} \xrightarrow{\tau} !\mathsf{a}\mathbf{0}$ becomes $!\mathsf{a}\mathbf{0} \oplus !\mathsf{b}\mathbf{0}[] \xrightarrow{\tau} \mathbf{0}[!\mathsf{a}]$, and similarly for the $!\mathsf{b}$-branch (i.e., the τ-transition of a non-deterministic internal choice is preserved in the asynchronous behaviour — where it signals the buffering of the selected prefix);

(iii) $!\mathsf{a}.!\mathsf{b}.\mathbf{0} \xrightarrow{!\mathsf{a}} !\mathsf{b}.\mathbf{0} \xrightarrow{!\mathsf{b}} \mathbf{0}$ becomes $!\mathsf{a}.!\mathsf{b}.\mathbf{0}[] \xrightarrow{\tau} !\mathsf{b}.\mathbf{0}[!\mathsf{a}] \xrightarrow{\tau} \mathbf{0}[!\mathsf{a}.!\mathsf{b}]$ (i.e., deterministic output transitions are "replaced" with τs in the asynchronous behaviour).

By observing the transitions and the buffer of an asynchronous session behaviour, we can recover information about the corresponding synchronous behaviour: e.g., from $T[\sigma] \xrightarrow{?\mathsf{a}} T'[\sigma]$ we can infer $T \xrightarrow{?\mathsf{a}} T'$ (and therefore, by item *(i)* of Proposition A.3, T is an external choice).

Proposition A.4. *For all T, T' and σ,*

(i) $T \xrightarrow{?\mathsf{a}} T'$ *iff* $T[\sigma] \xrightarrow{?\mathsf{a}} T'[\sigma]$;
(ii) *for all* $\mathsf{a} \neq \mathsf{b}$, $T \xrightarrow{\tau} !\mathsf{a}.T'$ *and* $T \xrightarrow{\tau} !\mathsf{b}.T''$ *iff* $T[\sigma] \xrightarrow{\tau} T'[\sigma.!\mathsf{a}] \wedge T[\sigma] \xrightarrow{\tau} T''[\sigma.!\mathsf{b}]$;
(iii) $T \xrightarrow{!\mathsf{a}} T'$ *iff* $T[\sigma] \xrightarrow{\tau} T'[\sigma.!\mathsf{a}] \wedge \nexists \mathsf{b} \neq \mathsf{a}\,.\, T[\sigma] \xrightarrow{\tau} T'[\sigma.!\mathsf{b}]$.

Proof. $\Longrightarrow$ direction:

- item *(i)*: by Proposition A.3 (item *(i)*), T is equivalent to a (possibly recursive) external choice with a $?\mathsf{a}$-branch. From such T, by rules (ruleCExtA) and (TRecA), we obtain the thesis;
- item *(ii)*: by Proposition A.3 (item *(ii)*), T is equivalent to a (possibly recursive) internal choice with 2 or more branches. From such T, by rules (TIntA) and (TRecA), we obtain the thesis, with $!\mathsf{a}, !\mathsf{b}$ being the distinct guards of two of the branches of T;
- item *(iii)*: by Proposition A.3 (item *(iii)*), T is equivalent to a (possibly recursive) single-branch internal choice $!\mathsf{a}.T''$. From such T, by rules (TOutA) and (TRecA), we obtain the thesis.

For the $\Longleftarrow$ direction, we proceed by induction on the rules in Definition 2.11 generating the transitions. The development is similar to the $\Longrightarrow$ direction in the proof of Proposition A.3: we determine the possible syntactic form of T, which may be (up-to recursion) an external choice for item *(i)*, or an internal choice for items *(ii)*–*(iii)* (resp. with

multiple branches including $!\mathsf{a}$ and $!\mathsf{b}$, or with just one $!\mathsf{a}$-branch). Then, we conclude by the rules in Definition 2.10 (i.e., (TExt) for item *(i)*, (TInt) for item *(ii)*, and (TOut) for item *(iii)*). □

The following propositions state that, in an asynchronous session behaviour, an output is only visible when it is at the head of the buffer, and is persistent.

Proposition A.5. $T[\sigma]\xrightarrow{!\mathsf{a}}$ *iff* $\exists\sigma' \,.\, \sigma = !\mathsf{a}.\sigma'$. *If* $T \overset{!\mathsf{b}}{\Rightarrow} T'$, *then* $\exists \mathsf{a} \,.\, T[\sigma] \xrightarrow{\tau} T'[\sigma \,.\, !\mathsf{b}]\xrightarrow{!\mathsf{a}}$.

Proof. The first part of the statement follows by Definition 2.11. For the second part, since $T \overset{!\mathsf{b}}{\Rightarrow} T'$, then by Definition 2.10 it must be $T \equiv !\mathsf{b}.T' \oplus \bigoplus_{i\in I} !\mathsf{c}_i.T_i$ (and in particular, $I = \emptyset$ iff $T \xrightarrow{!\mathsf{b}} T'$). We then conclude by Definition 2.11. □

Proposition A.6. $T[\sigma]\overset{!!\mathsf{a}}{\Longrightarrow}$ *iff* $\exists\sigma' \,.\, \sigma = !\mathsf{a}.\sigma'$, *or* $\sigma = \epsilon$ *and* $\exists T' \,.\, T \equiv !\mathsf{a}.T'$

Proof. See page 27. □

Proof. For the $\Longleftarrow$ direction, we have the following two cases:

- $\sigma = !\mathsf{a}.\sigma'$. By Proposition A.5, whenever $T[!\mathsf{a}.\sigma'] \Rightarrow T'[\sigma'']$, each τ-transition corresponds to an output of T being enqueued to the buffer. Therefore, $\sigma'' = !\mathsf{a}.\sigma'.\sigma'''$ (for some σ'''), and we have $T'[\sigma'']\xrightarrow{!\mathsf{a}}$, from which the thesis follows.
- $\sigma = \epsilon$ and $T \equiv !\mathsf{a}T'$. The thesis follows from Definition 2.11.

For the $\Longrightarrow$ direction, we proceed by cases on the structure of σ:

- $\sigma = !\mathsf{a}.\sigma'$. The thesis follows trivially;
- $\sigma = !\mathsf{b}.\sigma'$ (with $\mathsf{b} \neq \mathsf{a}$). In this case, by Definition 2.11 we have $T[\sigma]\overset{!\mathsf{a}}{\not\Rightarrow}$ — and therefore, $T[\sigma]\overset{!!\mathsf{a}}{\not\Longrightarrow}$ (contradiction);
- $\sigma = \epsilon$. Then either:
  - $T \equiv \&_{i\in I} ?\mathsf{b}_i.T_i$. Then, by Definition 2.11, $\forall \mathsf{c} \,.\, T[\sigma]\overset{!\mathsf{c}}{\not\Rightarrow}$ — and therefore, $T[\sigma]\overset{!!\mathsf{a}}{\not\Longrightarrow}$ (contradiction);
  - $T \equiv \bigoplus_{i\in I} !\mathsf{b}_i.T_i$. If $|I| \neq 1$, we have either:
    - $|I| = 0$. Then, $T = \mathbf{0}$ (since $I = \emptyset$), and therefore $T[\epsilon]\overset{!!\mathsf{a}}{\not\Longrightarrow}$ (contradiction);
    - $|I| > 1$. Then, since the guards of $\oplus$ are pairwise distinct, $\exists i \in I \,.\, \mathsf{b}_i \neq \mathsf{a}$. Hence, $T[\epsilon] \xrightarrow{\tau} T_i[!\mathsf{b}_i]\overset{!\mathsf{a}}{\not\Rightarrow}$ (by Definition 2.11), and therefore $T[\sigma]\overset{!!\mathsf{a}}{\not\Longrightarrow}$ (contradiction).

  We are left to examine the case $|I| = 1$: we have $T[\epsilon] \equiv !\mathsf{b}.T'[\epsilon] \xrightarrow{\tau} T'[!\mathsf{b}]$ for some $\mathsf{b} \in I = \{\mathsf{b}\}$ and T'. If $\mathsf{b} \neq \mathsf{a}$, by Definition 2.11 we have $T'[!\mathsf{b}]\overset{!\mathsf{a}}{\not\Rightarrow}$, and thus $T[\sigma]\overset{!!\mathsf{a}}{\not\Longrightarrow}$ (contradiction). Therefore, we conclude that $T \equiv !\mathsf{a}.T'$ (for some T'). □

B Proofs for Section 4

Proposition 4.3 (⊣/⊣⊢-induced shapes of session types)**.** *If* $T \dashv U$, *then exactly one of the following cases holds:*

a. $T = \mathbf{0}$;
b. $T \equiv \&_{i\in I} ?\mathsf{a}_i.T_i$ *and* $U \equiv \bigoplus_{j\in J} !\mathsf{a}_j.U_j$, *with* $\emptyset \neq J \subseteq I$, *and* $\forall j \in J \,.\, T_j \dashv U_j$;
c. $T \equiv \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ *and* $U \equiv \&_{j\in J} ?\mathsf{a}_j.U_j$, *with* $\emptyset \neq I \subseteq J$, *and* $\forall i \in I \,.\, T_i \dashv U_i$.

Furthermore, if $T \dashv\vdash U$, *we have either:*

a. $T = U = \mathbf{0}$;
b. $T \equiv \&_{i\in I} ?\mathsf{a}_i.T_i$ *and* $U \equiv \bigoplus_{j\in J} !\mathsf{a}_j.U_j$, *with* $\emptyset \neq J \subseteq I$, *and* $\forall j \in J \,.\, T_j \dashv\vdash U_j$;
c. $T \equiv \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ *and* $U \equiv \&_{j\in J} ?\mathsf{a}_j.U_j$, *with* $\emptyset \neq I \subseteq J$, *and* $\forall i \in I \,.\, T_i \dashv\vdash U_i$.

Proof. See [4]: the statement for $\dashv$ derives from the proof of Proposition 2.9 therein, while the statement for $\dashv\vdash$ derives from Lemma 3.3 in the same work (modulo two minor differences: in this paper, we postulate that $\mathbf{0}$ is the success state, and the unfolding of $\mathrm{rec}_X \cdots$ does not emit a τ-transition). □

Lemma 4.5. $\bowtie \;=\; \ddot{\triangleright} \cap \ddot{\triangleleft}$.

Proof. By Definition 4.4, we have $\bowtie \subseteq (\ddot{\triangleright} \cap \ddot{\triangleleft})$. To prove the inverse inclusion, let $\mathcal{R} = (\ddot{\triangleright} \cap \ddot{\triangleleft})$. Whenever $(p, q) \in \mathcal{R}$, we have $p \,\ddot{\triangleright}\, q$ and $q \,\ddot{\triangleleft}\, p$, and:

a. $\big(p{\Downarrow^!} \subseteq \mathrm{co}(q{\Downarrow^?})$
   and $\big(p{\Downarrow^!} = \emptyset \wedge p{\Downarrow^?} \neq \emptyset$ implies $\emptyset \neq q{\Downarrow^!} \subseteq \mathrm{co}(p{\Downarrow^?})\big)\big)$
   and
   $\big(q{\Downarrow^!} \subseteq \mathrm{co}(p{\Downarrow^?})$
   and $\big(q{\Downarrow^!} = \emptyset \wedge q{\Downarrow^?} \neq \emptyset$ implies $\emptyset \neq p{\Downarrow^!} \subseteq \mathrm{co}(q{\Downarrow^?})\big)\big)$;
b. $p \xrightarrow{\ell} p' \wedge q \xrightarrow{\mathrm{co}(\ell)} q'$ implies $p' \,\ddot{\triangleright}\, q'$
   and $q \xrightarrow{\ell'} q' \wedge p \xrightarrow{\mathrm{co}(\ell')} p'$ implies $q' \,\ddot{\triangleleft}\, p'$;
c. $p \xrightarrow{\tau} p'$ implies $p' \,\ddot{\triangleright}\, q$ and $q \xrightarrow{\tau} q'$ implies $q' \,\ddot{\triangleleft}\, p$;
d. $q \xrightarrow{\tau} q'$ implies $p \,\ddot{\triangleright}\, q'$ and $p \xrightarrow{\tau} p'$ implies $q \,\ddot{\triangleleft}\, p'$.

With some simplifications and rearrangements, for all $(p, q) \in \mathcal{R}$, we have:

a. $p{\Downarrow^!} \subseteq \mathrm{co}(q{\Downarrow^?})$ and $q{\Downarrow^!} \subseteq \mathrm{co}(p{\Downarrow^?})$
   and $\big(p{\Downarrow^!} = \emptyset \wedge p{\Downarrow^?} \neq \emptyset$ implies $\emptyset \neq q{\Downarrow^!} \subseteq \mathrm{co}(p{\Downarrow^?})\big)$
   and $\big(q{\Downarrow^!} = \emptyset \wedge p{\Downarrow^?} \neq \emptyset$ implies $\emptyset \neq p{\Downarrow^!} \subseteq \mathrm{co}(q{\Downarrow^?})\big)$;
b. $p \xrightarrow{\ell} p' \wedge q \xrightarrow{\mathrm{co}(\ell)} q'$ implies $p' \,\ddot{\triangleright}\, q' \wedge q' \,\ddot{\triangleleft}\, p'$, i.e. $p' \,\mathcal{R}\, q'$;
c. $p \xrightarrow{\tau} p'$ implies $p' \,\ddot{\triangleright}\, q \wedge q \,\ddot{\triangleleft}\, p'$, i.e. $p' \,\mathcal{R}\, q$;
d. $q \xrightarrow{\tau} q'$ implies $p \,\ddot{\triangleright}\, q' \wedge q' \,\ddot{\triangleleft}\, p$, i.e. $p \,\mathcal{R}\, q'$.

Therefore, each $(p, q) \in \mathcal{R}$ satisfies items *a*–*d* of Definition 4.4 — hence, $\mathcal{R}$ is an I/O compliance relation. Furthermore, $\mathcal{R}$ is symmetric. Thus, $\mathcal{R} = (\ddot{\triangleright} \cap \ddot{\triangleleft}) \subseteq \bowtie$. □

Proposition B.1. *Let* $p \circ q$, *with* $\circ \in \{\ddot{\triangleleft}, \ddot{\triangleright}, \bowtie\}$. *Then,* $p \Rightarrow p'$ *implies* $p' \circ q$.

Proof. We first prove the statement for $\circ = \ddot{\triangleleft}$. We proceed by induction on the length of the sequence of τ-transitions in $p \Rightarrow p'$. The base case ($n = 0$) follows from the hypothesis. For the inductive case, let $p \Rightarrow p^* \xrightarrow{\tau} p'$: by the induction hypothesis, we have $p^* \,\ddot{\triangleleft}\, q$ — and we conclude by item *c* of Definition 4.4.

The proof for $\circ = \ddot{\triangleright}$ is similar, except that we conclude by item *d* of Definition 4.4.

Finally, the thesis for $\circ = \bowtie$ follows from Lemma 4.5. □

Proposition B.2. *Let* $p \circ q$, *with* $\circ \in \{\ddot{\triangleleft}, \ddot{\triangleright}, \bowtie\}$. *Then,* $p \overset{!\mathsf{a}}{\Rightarrow} p' \wedge q \overset{?\mathsf{a}}{\Rightarrow} q'$ *implies* $p' \circ q'$.

Proof. Let:

$$p \Rightarrow p_0 \xrightarrow{!\mathsf{a}} p_0' \Rightarrow p'$$

$$q \Rightarrow q_0 \xrightarrow{?\mathsf{a}} q_0' \Rightarrow q'$$

We first consider the case $\circ = \ddot{\triangleleft}$.By Proposition B.1 on $p \Rightarrow p_0$, we have $p_0 \,\ddot{\triangleleft}\, q$; then, by applying Proposition B.1 on $q \Rightarrow q_0$, we have $p_0 \,\ddot{\triangleleft}\, q_0$. Now, by item *b* of Definition 4.4), we obtain

$p'_0 \ddot{\lhd} q'_0$; finally, again by applying Proposition B.1 on $p'_0 \Rightarrow p'$ and then on $q'_0 \Rightarrow q'$, we conclude $p' \ddot{\lhd} q'$.

The proof for $\circ = \ddot{\rhd}$ is similar.

Finally, the thesis for $\circ = \bowtie$ follows from Lemma 4.5. □

Proposition B.3. *Let $p \ddot{\lhd} q$ and $p \overset{!a}{\Rightarrow}$. Then, $q \overset{??a}{\Longrightarrow}$.*

Proof. Let q' be such that $q \Rightarrow q'$. By Proposition B.1 we have that $p \ddot{\lhd} q'$; thus, by item *a* of Definition 4.4, it follows that $p{\Downarrow^!} \subseteq \mathrm{co}(q'{\Downarrow^?})$. Therefore, $q' \overset{?a}{\Rightarrow}$ — from which we conclude that $q \overset{??a}{\Longrightarrow}$. □

Corollary B.4. *Let $w \in (\mathsf{A}^!)^*$, and let $p \overset{w}{\Rightarrow}$. Then, $p \ddot{\lhd} q$ implies $q \overset{\mathrm{co}(w)}{\Longrightarrow}$. Moreover, $\forall p', q' \,.\, p \overset{w}{\Rightarrow} p' \wedge q \overset{\mathrm{co}(w)}{\Longrightarrow} q'$ implies $p' \ddot{\lhd} q'$.*

Proof. The first part of the statemen is proved by induction on w, and follows from Proposition B.1, Proposition B.3 and Proposition B.2.

The *"moreover..."* part, again by induction on w, follows from Proposition B.3. □

Proposition B.5. *Let $p \ddot{\lhd} q$, with $p{\Downarrow^!} = \emptyset$, $p{\Downarrow^?} \neq \emptyset$ and $q \overset{!a}{\Rightarrow}$: then, $p \overset{??a}{\Longrightarrow}$.*

Proof. $\forall p' \,.\, p \Rightarrow p'$, by Proposition B.1 we have $p{\Downarrow^!} = \emptyset$ and $p' \ddot{\lhd} q$; hence, by item *a* of Definition 4.4), we have $\forall p' \,.\, p \Rightarrow p'$ implies $q{\Downarrow^!} \subseteq \mathrm{co}(p'{\Downarrow^?})$, i.e. $p' \overset{?a}{\Rightarrow}$. We conclude that $p \overset{??a}{\Longrightarrow}$. □

Theorem 4.9. *(a) If $p \ddot{\lhd} q$, then $p \dashv q$; (b) if $p, q \in \mathbb{U}_{ST}$ and $p \dashv q$, then $p \ddot{\lhd} q$.*

Proof. We first prove $\ddot{\lhd} \subseteq \dashv$. Let:

$$F(\mathcal{X}) = \left\{ (p,q) \;\middle|\; \begin{array}{l} p \parallel q \overset{\tau}{\not\rightarrow} \text{ implies } p \cong \mathbf{0} \quad \text{and} \\ p \parallel q \overset{\tau}{\rightarrow} p' \parallel q' \text{ implies } (p',q') \in \mathcal{X} \end{array} \right\}$$

It is easy to check that $\mathrm{gfp}(F) = \dashv$. By the coinduction proof principle, if $\ddot{\lhd} \subseteq F(\ddot{\lhd})$, we can deduce that $\ddot{\lhd} \subseteq \mathrm{gfp}(F) = \dashv$. So, we show that $\ddot{\lhd} \subseteq F(\ddot{\lhd})$. Let $p \ddot{\lhd} q$.

For the first clause of the definition of F, assume $p \parallel q \overset{\tau}{\not\rightarrow}$. We show that p has no outgoing transitions (i.e., $p \cong \mathbf{0}$) by showing the absurdity of the following cases:

- $p \overset{\tau}{\rightarrow}$. Not possible, because it would contradict the assumption $p \parallel q \overset{\tau}{\not\rightarrow}$.
- $p \overset{!a}{\rightarrow}$. By item *a* of Definition 4.4, we have that $p{\Downarrow^!} \subseteq \mathrm{co}(q{\Downarrow^?})$, hence $q \overset{?a}{\Rightarrow}$. Note that $q \overset{\tau}{\not\rightarrow}$, because otherwise we would contradict the assumption $p \parallel q \overset{\tau}{\not\rightarrow}$. Therefore, it must be $q \overset{?a}{\rightarrow}$, and so by definition of $\parallel$ we would obtain $p \parallel q \overset{\tau}{\rightarrow}$ — contradiction.
- $p \overset{?a}{\rightarrow}$ and $\nexists \mathsf{b} \,.\, p \overset{!b}{\rightarrow}$. Note that $p \overset{\tau}{\not\rightarrow}$ and $q \overset{\tau}{\not\rightarrow}$, because otherwise we would contradict the assumption $p \parallel q \overset{\tau}{\not\rightarrow}$. Since $p{\Downarrow^!} = \emptyset$ and $p{\Downarrow^?} \neq \emptyset$, by item *a* of Definition 4.4 we have that $\emptyset \neq q{\Downarrow^!} \subseteq \mathrm{co}(p{\Downarrow^?})$. Since $q \overset{\tau}{\not\rightarrow}$, it must be $q \overset{!a}{\rightarrow}$, and so by definition of $\parallel$ we would obtain $p \parallel q \overset{\tau}{\rightarrow}$ — contradiction.

Thus, since p cannot have outgoing transitions, we conclude that $p \cong \mathbf{0}$.

For the second clause of the definition of F, assume that $p \parallel q \overset{\tau}{\rightarrow}$. We have the following three cases:

- $p \overset{\tau}{\rightarrow} p'$. By definition of $\parallel$, this implies $p \parallel q \overset{\tau}{\rightarrow} p' \parallel q$. By item *c* of Definition 4.4, we conclude that $p' \ddot{\lhd} q$.
- $q \overset{\tau}{\rightarrow} q'$. Similar to the previous case.
- $p \overset{\ell}{\rightarrow} p'$ and $q \overset{\mathrm{co}(\ell)}{\longrightarrow} q'$, for some label ℓ. By definition of $\parallel$, this implies $p \parallel q \overset{\tau}{\rightarrow} p' \parallel q'$. By item *b* of Definition 4.4, we conclude that $p' \ddot{\lhd} q'$.

In all three cases, we have concluded that $p' \ddot{\lhd} q'$, thus satisfying the second clause of F.

We now prove that $T \dashv U$ implies $T \ddot{\lhd} U$, for all session types T, U. To do that, it suffices to show that $\dashv$ is an I/O compliance relation. Assume that $T \dashv U$. We show that all the clauses of Definition 4.4 are satisfied:

- recall the 3 possible forms of T and U from Proposition 4.3: modulo unfolding, either $T = \mathbf{0}$, or $T = \bigoplus_{i \in I} !\mathsf{a}_i.T_i$ (with $|I| \geq 1$) and $U = \&_{j \in J} ?\mathsf{a}_j.U_j$ with $I \subseteq J$, or *vice versa* (i.e., T is an external choice and U is an internal choice). Each of these three possible forms satisfies item *a* in Definition 4.4;
- when the premise of item *b* of Definition 4.4 holds, we have $T \parallel U \overset{\tau}{\rightarrow} T' \parallel U'$; by Definition 4.1, we conclude $T' \dashv U'$;
- when $T \overset{\tau}{\rightarrow} T'$ (premise of item *c* of Definition 4.4), by Definition 4.1, we conclude $T' \dashv U$;
- similarly, when $U \overset{\tau}{\rightarrow} U'$ (premise of item *d* of Definition 4.4), by Definition 4.1, we conclude $T \dashv U'$. □

B.1 I/O compliance and asynchrony

Notation B.6. *We write $\sigma.T'$ for the session type obtained by prefixing T' with the sequence of outputs σ.*

Lemma B.7 (Sync session behaviours and async I/O compliance (I))**.** *Let $T \circ U$, for some $\circ \in \{\vdash, \dashv, \dashv\vdash, \ddot{\rhd}, \ddot{\lhd}, \ddot{\bowtie}\}$. Then, $T[] \Rightarrow T'[\sigma]$ implies $\sigma.T' \circ U$.*

Proof. We first prove the statement for $\circ = \dashv$, and therefore we show that:

$$T \dashv U \;\wedge\; T[] \Rightarrow T'[\sigma] \quad \text{implies} \quad \sigma.T' \dashv U \tag{5}$$

If T is equivalent to a (possibly empty) *external* choice, we have $T[] \overset{\tau}{\not\rightarrow}$, and hence $T' = T$ and $\sigma = \epsilon$: therefore, the thesis coincides with the hypothesis.

Otherwise, if T is a non-empty *internal* choice, we proceed by induction on the length of σ, which (by the semantics in Definition 2.11) corresponds to the number of τ-transitions along $T[] \Rightarrow T'[\sigma]$:

- base case: $\sigma = \epsilon$. Then, $T' = T$ and $\sigma = \epsilon$, and therefore the thesis coincides with the hypothesis;
- inductive case: $\sigma = \sigma'.!\mathsf{a}$, with $\sigma' = !\mathsf{b}_1 \ldots !\mathsf{b}_n$. In this case, we have $T[] \Rightarrow T''[\sigma'] \overset{\tau}{\rightarrow} T'[\sigma'.!\mathsf{a}]$, and by applying the induction hypothesis, $\sigma'.T'' \dashv U$. Therefore, by applying Proposition 4.3 for n times along the synchronisations of $\sigma'.T''$ and U, we have:

$$\begin{array}{rrl} !\mathsf{b}_1.!\mathsf{b}_2.\ldots.!\mathsf{b}_n.T'' \dashv U = U_1 & = ?\mathsf{b}_1.U_2 \;\&\; \&_{i \in I_1} ?\mathsf{b}_i.U_i \\ !\mathsf{b}_2.\ldots.!\mathsf{b}_n.T'' \dashv \quad U_2 & = ?\mathsf{b}_2.U_3 \;\&\; \&_{i \in I_2} ?\mathsf{b}_i.U_i \\ & \ldots \\ !\mathsf{b}_n.T'' \dashv \quad U_n & = ?\mathsf{b}_n.U_{n+1} \;\&\; \&_{i \in I_n} ?\mathsf{b}_i.U_i \\ T'' \dashv U_{n+1} & \end{array}$$

At this point, since $T''[\sigma'] \overset{\tau}{\rightarrow} T'[\sigma'.!\mathsf{a}]$ (i.e., T'' enqueues an output in its buffer), by the semantics in Definition 2.11 we have:

$$T''[\sigma'] = \left(!\mathsf{a}.T' \oplus \bigoplus_{i \in I} !\mathsf{a}_i.T_i \right)[\sigma'] \;\overset{\tau}{\rightarrow}\; T'[\sigma'.!\mathsf{a}]$$

i.e., T'' must be a non-empty internal choice with a $!\mathsf{a}$-guarded branch; correspondingly, from $T'' \dashv U_{n+1}$ and Proposition 4.3, we have:

$$U_{n+1} = ?\mathsf{a}.U' \,\&\, \mathop{\&}_{k\in K} ?\mathsf{a}_k.U'_k \quad \text{with } I \subseteq K \text{ and } T' \dashv U'$$

But then, $\sigma'.!\mathsf{a}.T' = \sigma.T' \dashv U$. This concludes the proof of Equation (5).

With the proof above, the other possible values of $\circ$ can be verified in a straightforward way:

- case $\circ = \vdash$ can be proved in a similar way, noticing that in the inductive case, U may also become $\mathbf{0}$, i.e. it may terminate at any point without consuming T's output with an external choice;
- case $\circ = \dashv\!\vdash$ holds because (by Definition 4.1) $\dashv\!\vdash = \vdash \cap \dashv$;
- case $\circ = \ddot{\lhd}$ follows by Equation (5) and Theorem 4.9; case $\circ = \ddot{\rhd}$ is similar;
- finally, case $\circ = \ddot{\bowtie}$ follows by the previous item, because (by Lemma 4.5) $\ddot{\bowtie} = \ddot{\rhd} \cap \ddot{\lhd}$.

□

Notation B.8. *Given a sequence of outputs $\sigma = !\mathsf{a}_1 \ldots !\mathsf{a}_n$, we write $\overset{\sigma}{\Rightarrow}$ for $\overset{!\mathsf{a}_1}{\Longrightarrow} \cdots \overset{!\mathsf{a}_n}{\Longrightarrow}$.*

Lemma B.9 (Diamond lemma for asynchronous session types)**.** *Let:*

$$T_0[] \parallel U_0[] \;\; (\xrightarrow{\tau})^n \;\; T_n[\sigma_n] \parallel U_n[\rho_n]$$

with $\sigma_n \neq \epsilon$, $\rho_n = \epsilon$, and $\forall i \leq n : (\sigma_i = \epsilon$ or $\rho_i = \epsilon)$. Let m be the length of σ_n. Then, there exists some $k \leq n$ such that:

1. *$\sigma_k = \epsilon$ and $\forall i > k : \sigma_i \neq \epsilon$*
2. *$\forall i \geq k : \rho_i = \epsilon$*
3. *there exist T'_j, U'_j (for all $j \in k..n$) such that:*

$$\begin{aligned} T_k[\sigma_k] \parallel U_k[\rho_k] = T'_k[] \parallel U'_k[] \\ (\xrightarrow{\tau})^{n-k-m} \;\; T'_{n-m}[] \parallel U'_{n-m}[] \\ (\xrightarrow{\tau})^{m} \;\; T'_n[\sigma_n] \parallel U'_n[] \;\; = \;\; T_n[\sigma_n] \parallel U_n[] \end{aligned}$$

where $\forall i \in 0..m : U'_{(n-m)+i} = U_n$

4. *$T'_{n-m} \overset{\sigma_n}{\Longrightarrow} T_n$*

Proof. (*Sketch*) Let k be the greatest index such that both queues σ_k and ρ_k are empty. Since σ_n is not empty, then Items 1 and 2 follow. For Item 3, we observe that T_k performs some outputs (and no inputs, since the queues ρ_i are persistently empty for $i \geq k$). These outputs can be split in two parts: in the first one (called $\tilde{\sigma}$) the inputs are read later on by U_k, while in the second part (called $\tilde{\sigma'}$) the inputs remain enqueued (and so, $\tilde{\sigma'} = \sigma_n$). It is possible to reorder the moves without altering the length of the computation, and so that: (i) at the beginning, $\tilde{\sigma}$ is enqueued and read by U; (ii) then, the second part is enqueued, and never read. The terms T'_j, U'_j (for all $j \in k..n$) are defined according to this reordering. After the steps (i), both queues are empty, and the component U no longer moves, hence it is equal to U_n until the end of the computation.

Item 4 is a direct consequence of Item 3. □

Lemma 4.11. *Let $T \circ U$, with $\circ \in \{\dashv, \vdash, \dashv\!\vdash, \ddot{\lhd}, \ddot{\rhd}, \ddot{\bowtie}\}$. If $T[] \parallel U[] \Rightarrow T'[\sigma] \parallel U'[\rho]$, then:*

(i) *$\sigma = \epsilon$ or $\rho = \epsilon$.*
(ii) *if $\sigma = \rho = \epsilon$ then $T' \circ U'$.*

Proof. We first consider the case $\circ = \dashv$, showing that:

$$T \dashv U \;\wedge\; T[] \parallel U[] \Rightarrow T'[\sigma] \parallel U'[\rho] \;\text{ implies }\; \sigma = \epsilon \text{ or } \rho = \epsilon \quad (6)$$

$$\text{the premise above and } \sigma = \rho = \epsilon \;\text{ implies }\; T' \dashv U' \quad (7)$$

We proceed by induction on the length of the sequence of τ-transitions in $T[] \parallel U[] \Rightarrow T'[\sigma] \parallel U'[\rho]$. In the base case (i.e., when there are no transitions) the thesis coincides with the hypothesis, and both Equations (6) and (7) trivially hold.

In the inductive case, let:

$$T[] \parallel U[] \;\Rightarrow\; T''[\sigma''] \parallel U''[\rho''] \;\xrightarrow{\tau}\; T'[\sigma'] \parallel U'[\rho'] \quad (8)$$

and by the induction hypothesis,

$$(\sigma'' = \epsilon \;\text{ or }\; \rho'' = \epsilon) \;\text{ and }\; (\sigma'' = \rho'' = \epsilon \;\text{ implies }\; T'' \dashv U'') \quad (9)$$

We have the following cases:

- $\sigma'' \neq \epsilon$ and $\rho'' = \epsilon$. By Lemma B.9, the computation $T[] \parallel U[] \Rightarrow T''[\sigma''] \parallel U''[\rho'']$ can be rearranged as follows (while maintaining the original length):

  $$T[] \parallel U[] \;\Rightarrow\; T'''[] \parallel U''[] \;\Rightarrow\; T''[\sigma''] \parallel U''[]$$

  where $T''' \overset{\sigma''}{\Longrightarrow} T''$ (by item 4 of Lemma B.9). Since the length of the computation

  $$T[] \parallel U[] \Rightarrow T'''[] \parallel U''[]$$

  is shorter than the the length of the original computation, by the induction hypothesis we have $T''' \dashv U''$. Let $\sigma'' = !\mathsf{a}.\sigma'''$. Then, by Proposition 4.3, U'' must be (up-to unfolding) an *external* choice with a $?\mathsf{a}$-branch. Hence, the rightmost τ-transition in Equation (8) can only be generated in two ways:
  - via synchronisation, i.e.:

    $$T''[!\mathsf{a}.\sigma'''] \parallel U''[] \;\xrightarrow{\tau}\; T'[\sigma'''] \parallel U'[] \qquad (\text{with } \sigma''' = \sigma')$$

    where $T' = T''$. We can notice that item *(i)* of the thesis is already satisfied. For item *(ii)*, assume that $\sigma''' = \epsilon$. Then, $!\mathsf{a}$ is the only output transition from T''' to T'' — i.e., $T''' \overset{!\mathsf{a}}{\Rightarrow} T'' = T'$; and since $U'' \xrightarrow{?\mathsf{a}} U'$, from $T''' \dashv U''$, we deduce $T' \dashv U'$.
  - via buffering, i.e., for some b:

    $$T''[!\mathsf{a}.\sigma'''] \parallel U''[] \;\xrightarrow{\tau}\; T'[!\mathsf{a}.\sigma'''.!\mathsf{b}] \parallel U'[] \qquad (\text{where } U' = U'')$$

    Then, we satisfy item *(i)* and (vacuously) item *(ii)* of the thesis;
- $\sigma'' = \epsilon$ and $\rho'' \neq \epsilon$. The proof is similar to the previous case, by swapping the roles of T'' and U'', and the roles of σ and ρ.
- $\sigma'' = \rho'' = \epsilon$. Item *(i)* holds because in a single τ move, at most one of the two queues can become non-empty. Item *(ii)* holds vacuously, because the only way of reaching $\sigma' = \rho' = \epsilon$ would be through a synchronisation, which is not possible because $\sigma'' = \rho'' = \epsilon$.

This concludes the proof of Equations (6) and (7), in case $\circ = \dashv$. The other possible values of $\circ$ can be verified as follows:

- $\circ = \ddot{\lhd}$ follows by Equation (6) and $\dashv = \ddot{\lhd}$ (Theorem 4.9);
- $\circ \in \{\vdash, \ddot{\rhd}\}$ follow by symmetry;
- $\circ = \dashv\!\vdash$ holds because (by Definition 4.1) $\dashv\!\vdash = \vdash \cap \dashv$;
- finally, $\circ = \ddot{\bowtie}$ holds because $\ddot{\bowtie} = \ddot{\rhd} \cap \ddot{\lhd}$ (by Lemma 4.5).

□

Lemma 4.12. *If $T \ddot{\lhd} U$, then $T[] \ddot{\lhd} U[]$.*

Proof. Assume that $T \ddot{\lhd} U$, and let:

$$\mathcal{R} = \{(T'[\sigma], U'[\rho]) \mid T[] \parallel U[] \Rightarrow T'[\sigma] \parallel U'[\rho]\}$$

We prove that $\mathcal{R}$ is an I/O compliance relation, i.e. it satisfies all the items of Definition 4.4. The thesis will then follow by the fact that $(T[], U[]) \in \mathcal{R}$.

Let $(T'[\sigma], U'[\rho]) \in \mathcal{R}$. Note that items *b*–*d* of Definition 4.4 follow directly by definition of $\parallel$, so we only need to prove item *a*. By Lemma 4.11, we have $\sigma = \epsilon$ or $\rho = \epsilon$, and so we have the following three cases:

- $\sigma \neq \epsilon$ and $\rho = \epsilon$. Let $\sigma = !\mathsf{a}.\sigma'$. Then:

  $$T'[\sigma]\Downarrow^! = \{!\mathsf{a}\} \tag{10}$$

  By items *(ii)* and *(iii)* of Proposition A.4 we know that each output in σ has been generated by some T'' such that $T'' \stackrel{\sigma}{\Rightarrow} T'$; let us take such T'' so that:

  $$T[] \parallel U[] \Rightarrow T''[] \parallel U'[] \Rightarrow T'[\sigma] \parallel U'[]$$

  From $T \ddot{\lhd} U$, $T[] \parallel U[] \Rightarrow T''[] \parallel U'[]$ and Lemma 4.11 we get $T'' \ddot{\lhd} U'$. By items *(ii)* and *(iii)* of Proposition A.3, we know that T'' is an internal choice, hence by Theorem 4.9 and Proposition 4.3 we know that U' must be a (larger) external choice, i.e. $T''\Downarrow^! \subseteq \mathrm{co}(U'\Downarrow^?)$. Hence, from Equation (10) we obtain:

  $$\{!\mathsf{a}\} = T'[\sigma]\Downarrow^! \subseteq T''\Downarrow^! \subseteq \mathrm{co}(U'\Downarrow^?) = \mathrm{co}(U'[]\Downarrow^?)$$

  This proves the first part of item *a* in Definition 4.4. The second part holds vacuously, since $T'[\sigma]\Downarrow^! \neq \emptyset$.
- $\sigma = \epsilon$ and $\rho \neq \epsilon$. If $T' = \mathbf{0}$, then item *a* in Definition 4.4 is trivially satisfied. Otherwise, the reasoning is similar to the case above, by swapping the role of T' and U', and considering the queue ρ instead of σ. More in detail, we let $\rho = !\mathsf{a}.\rho'$, and so we obtain:

  $$U'[\rho]\Downarrow^! = \{!\mathsf{a}\} \tag{11}$$

  As in the previous item, via Proposition A.4 we take U'' such that $U'' \stackrel{\rho}{\Rightarrow} U'$, from which we have:

  $$T[] \parallel U[] \Rightarrow T'[] \parallel U''[] \Rightarrow T'[] \parallel U'[\rho]$$

  Since $T \ddot{\lhd} U$ and $T[] \parallel U[] \Rightarrow T'[] \parallel U''[]$, from Lemma 4.11 we obtain that $T' \ddot{\lhd} U''$.
  By Proposition A.3 we have that U'' is a non-empty internal choice, and by Theorem 4.9 and Proposition 4.3 we know that T' is a (larger) external choice, i.e. $\emptyset \neq U''\Downarrow^! \subseteq \mathrm{co}(T'\Downarrow^?)$. Since T' is an external choice, then $T'[]\Downarrow^! = \emptyset$, which vacuously satisfies the first part of item *a* in Definition 4.4. For the second part of the item, from Equation (11) we have:

  $$\emptyset \neq \{!\mathsf{a}\} = U'[\rho]\Downarrow^! \subseteq U''\Downarrow^! \subseteq \mathrm{co}(T'\Downarrow^?) = \mathrm{co}(T'[]\Downarrow^?)$$

- $\sigma = \rho = \epsilon$. By Lemma 4.11, it follows that $T' \ddot{\lhd} U'$. Hence, by Theorem 4.9 and Proposition 4.3, we can proceed by cases on the form of T' and U':
  - $T' = \mathbf{0}$. Then, item *a* of Definition 4.4 is trivially satisfied.
  - $T' = \&_{i\in I} ?\mathsf{a}_i.T_i$ and $U' = \bigoplus_{j\in J} !\mathsf{a}_j.U_j$, with $\emptyset \neq J \subseteq I$. Then, $T'[]\Downarrow^! = \emptyset$, which satisfies the first part of item *a*. Further, $\emptyset \neq U'[]\Downarrow^! \subseteq \mathrm{co}(T'[]\Downarrow^?)$, thus satisfying the second part of the item.
  - $T' = \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ and $U' = \&_{j\in J} ?\mathsf{a}_j.U_j$, with $\emptyset \neq I \subseteq J$. Then, $T'[]\Downarrow^! \subseteq \mathrm{co}(U'[]\Downarrow^?)$, which satisfies the first part of item *a*; the second part of the item is vacuously true, since $T'[]\Downarrow^! \neq \emptyset$. □

C Proofs for Section 5

Lemma 5.4. *Let $T[\sigma] \parallel U[\rho]$ be an orphan message configuration, with $!\mathsf{a}$ orphan message of $T[\sigma]$. Then, $T[\sigma] \parallel U[\rho] \Rightarrow T'[\sigma'] \parallel U'[\rho']$ implies that $T'[\sigma'] \parallel U'[\rho']$ is an orphan message configuration, with $!\mathsf{a}$ orphan message of $T'[\sigma']$. Furthermore, $\sigma' = \sigma.\sigma''$ (for some σ'').*

Proof. By hypothesis and Definition 5.1 we have $T[\sigma] \stackrel{!!\mathsf{a}}{\Longrightarrow}$ and $U[\rho] \not\stackrel{?\mathsf{a}}{\Longrightarrow}$. Moreover, by Lemma 5.3 we have either $\sigma = !\mathsf{a}.\sigma''$ (for some σ''), or $\sigma = \epsilon$ and $T = !\mathsf{a}.T'$: therefore, after at most one τ-step (when $\sigma = \epsilon$), $!\mathsf{a}$ is at the head of T's buffer and becomes *only* output (weakly) reachable from $T[\sigma]$ (by Proposition A.6). At that point, since $U[\rho] \not\stackrel{?\mathsf{a}}{\Longrightarrow}$, the two behaviours cannot synchronise on such $!\mathsf{a}$, and thus it cannot be removed from the head of the buffer. Hence, each τ-move along the trace $T[\sigma] \parallel U[\rho] \Rightarrow T'[\sigma'] \parallel U'[\rho']$ may be generated in only two ways, by Definition 2.11:

a. T or U add new outputs to their respective buffers;
b. T reaches an external choice which synchronises with the head of U's queue, consuming it.

(Note that case *b.* prevents us from resorting to Proposition 5.2, since the latter only considers the *internal* τ-transitions of the behaviours, without synchronisations). Therefore, $\sigma' = \sigma.\sigma''$ for some σ'' deriving from case *a.* above (which proves the *"furthermore..."* part of the statement), and $U'[\rho'] \not\stackrel{?\mathsf{a}}{\Longrightarrow}$: by Definition 5.1, we conclude that $T'[\sigma'] \parallel U'[\rho']$ is still an orphan message configuration, with $!\mathsf{a}$ orphan message of $T'[\sigma']$. □

Proposition C.1 below states that an input behaviour cannot stop interacting after some τ-moves.

Proposition C.1. *If p is an input behaviour, then $p \Rightarrow p'$ implies $p' \not\approx \mathbf{0}$.*

Proof. From Definition 5.7, we have that $\exists p'', \mathsf{a} \,.\, p' \Rightarrow p'' \xrightarrow{?\mathsf{a}}$: hence, $p'' \not\approx \mathbf{0}$, which in turn implies $p' \not\approx \mathbf{0}$. □

D Proofs for Section 6

Lemma 6.2 ($\sqsubseteq_{\mathbb{UST}}$-induced shapes of session types)**.** *$T \sqsubseteq_{\mathbb{UST}} U$ implies either:*

a. $T = U = \mathbf{0}$;
b. $T \equiv \&_{k\in K} ?\mathsf{a}_k.T_k$ *and* $U \equiv \&_{i\in I} ?\mathsf{a}_i.T_i$, *with* $\emptyset \neq I \subseteq K$ *and* $\forall i \in I \,.\, T_i \sqsubseteq_{\mathbb{UST}} U_i$;
c. $T \equiv \bigoplus_{k\in K} !\mathsf{a}_k.T_k$ *and* $U \equiv \bigoplus_{i\in I} !\mathsf{a}_i.T_i$, *with* $\emptyset \neq K \subseteq I$, *and* $\forall k \in K \,.\, T_k \sqsubseteq_{\mathbb{UST}} U_k$.

Proof. In the following, when no ambiguity arises, we will write $\sqsubseteq$ instead of $\sqsubseteq_{\mathbb{UST}}$.

When $T \sqsubseteq U$, by Definition 6.1 we know that $\forall V \,.\, U \bowtie V \implies T \bowtie V$. By Theorem 4.9, this is equivalent to saying that $\forall V \,.\, U \dashv\vdash V \implies T \dashv\vdash V$. Hence, the possible forms of the pairs U, V and T, V (up-to unfolding) are given by Proposition 4.3 (case for $\dashv\vdash$). Therefore, by cases on U, we have:

- $U = \mathbf{0}$. Then, we have $V = \mathbf{0}$, and $T = \mathbf{0}$.
- $U = \&_{i\in I} ?\mathsf{a}_i.U_i$, with $I \neq \emptyset$. Then, we have $V = \bigoplus_{j\in J} !\mathsf{a}_j.V_j$, with $\emptyset \neq J \subseteq I$, and $\forall j \in J . U_j \dashv\vdash V_j$ (i.e., $U_j \bowtie V_j$). Since $T \dashv\vdash V$, we also have $T = \&_{k\in K} ?\mathsf{a}_k.T_k$, with $\emptyset \neq J \subseteq K$, and $\forall j \in J \,.\, T_j \dashv\vdash V_j$ (i.e., $T_j \bowtie V_j$). Furthermore, we have $I \subseteq K$: otherwise, $\exists i \in I \,.\, i \notin K$, and if we take a V such that $J = I$, we would have the contradiction $J \not\subseteq K$. Finally, since (as seen above) $\forall j \in J$ we have $U_j \bowtie V_j$ and $T_j \bowtie V_j$, we conclude $\forall j \in J \,.\, T_j \sqsubseteq U_j$;

$U = \bigoplus_{i\in I} ?\mathsf{a}_i.U_i$, with $I \neq \emptyset$. Then, we have $V = \&_{j\in J} !\mathsf{a}_j.V_j$, with $I \subseteq J$, and $\forall i \in I \,.\, U_i \dashv\vdash V_i$ (i.e., $U_i \bowtie V_i$). Since $T \dashv\vdash V$, we also have $T = \bigoplus_{k\in K} ?\mathsf{a}_k.T_k$, with $\emptyset \neq K \subseteq J$, and $\forall k \in K \,.\, T_k \dashv\vdash V_k$ (i.e., $T_k \bowtie V_k$). Furthermore, we have $K \subseteq I$: otherwise, $\exists k \in K \,.\, k \notin I$, and if we take a V such that $J = I$, we would have the contradiction $K \not\subseteq J$. Finally, since (as seen above) $\forall k \in K$ we have $U_k \bowtie V_k$ and $T_k \bowtie V_k$, we conclude $\forall k \in K \,.\, T_k \sqsubseteq U_k$.

□

Lemma 6.3. $T \sqsubseteq U \iff T \sqsubseteq_{\mathbb{U}_{ST}} U$.

Proof. ($\Longrightarrow$). Follows from Definition 6.1, since $\mathbb{U}_{\mathrm{ST}} \subseteq \mathbb{U}$.
($\Longleftarrow$). Let:

$$\mathcal{R} \;=\; \left\{(T,r) \;\middle|\; \exists U \,.\, T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U \wedge U \bowtie r\right\} \;\cup\; \text{symmetric}$$

We show that $\mathcal{R}$ is a symmetric I/O compliance relation. For each $(T,r) \in \mathcal{R}$, there exists U such that $T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U$ and $U \bowtie r$. For all such U, we have (from the proof of Lemma 4.5):

a. $U\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?)$ and $r\Downarrow^! \subseteq \mathrm{co}(U\Downarrow^?)$ and $\big(U\Downarrow^! = \emptyset \wedge U\Downarrow^? \neq \emptyset \implies r\Downarrow^! \neq \emptyset\big)$ and $\big(r\Downarrow^! = \emptyset \wedge r\Downarrow^? \neq \emptyset \implies U\Downarrow^! \neq \emptyset\big)$;
b. $U \xrightarrow{\ell} U' \wedge r \xrightarrow{\mathrm{co}(\ell)} r' \implies U' \bowtie r'$;
c. $U \xrightarrow{\tau} U' \implies U' \bowtie r$;
d. $r \xrightarrow{\tau} r' \implies U \bowtie r'$.

Before proceeding, we also observe that, since $T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U$ by hypothesis, by Lemma 6.2 and $U \bowtie r$ we have the following possibilities (up-to unfolding):

1. $U = T = \mathbf{0}$. Therefore, in this case, $r\Downarrow = r\Downarrow^? = r\Downarrow^! = \emptyset$;
2. $U = \&_{i\in I} ?\mathsf{a}_i.T_i$ and $T = \&_{k\in K} ?\mathsf{a}_k.T_k$, with $\emptyset \neq I \subseteq K$ and $\forall i \in I \,.\, (T_i, U_i) \in \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}}$. Therefore, in this case, $\emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(U\Downarrow^?) \subseteq \mathrm{co}(T\Downarrow^?)$. Furthermore, $\forall i \in I \,.\, T \xrightarrow{?\mathsf{a}_i} T_i$ and $r \overset{!\mathsf{a}_i}{\Longrightarrow} r'$ implies $U_i \bowtie r'$ (from $U \overset{?\mathsf{a}_i}{\Longrightarrow} U_i$ and Proposition B.2), while $\forall j \in K \setminus I \,.\, r \overset{!\mathsf{a}_j}{\not\Longrightarrow}$ (from $U \bowtie r$ and Proposition B.3);
3. $U = \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ and $T = \bigoplus_{k\in K} !\mathsf{a}_k.T_k$, with $\emptyset \neq K \subseteq I$, and $\forall k \in K \,.\, (T_k, U_k) \in \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}}$. Therefore, in this case, $T\Downarrow^! \subseteq U\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?) \neq \emptyset$. Furthermore, $\forall k \in K \,.\, T \overset{!\mathsf{a}_k}{\Longrightarrow} T_k$ implies $r \overset{?\mathsf{a}}{\Longrightarrow}$ (from $U \overset{!\mathsf{a}_k}{\Longrightarrow} U_k$ and Proposition B.3) and $\forall r' \,.\, r \overset{?\mathsf{a}_k}{\Longrightarrow} r'$ implies $U_k \bowtie r'$ (from $U \bowtie r$ and Proposition B.2). Finally, by Proposition B.3, we have $r\Downarrow^! = \emptyset$.

We can now prove that $\mathcal{R}$ is a I/O compliance relation, examining the clauses of Definition 4.4:

a: we need to show that $T\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?)$ and $r\Downarrow^! \subseteq \mathrm{co}(T\Downarrow^?)$ and $\big(T\Downarrow^! = \emptyset \wedge T\Downarrow^? \neq \emptyset \implies r\Downarrow^! \neq \emptyset\big)$ and $\big(r\Downarrow^! = \emptyset \wedge r\Downarrow^? \neq \emptyset \implies T\Downarrow^! \neq \emptyset\big)$: these requirements are satisfied in all cases *1.*–*3.* above;
b: assume that $T \xrightarrow{\ell} T'$ and $r \xrightarrow{\mathrm{co}(\ell)} r'$. From cases *2.* and *3.* above, it follows that $\exists U' \,.\, U \overset{\ell}{\Rightarrow} U'$ and $T' \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U' \bowtie r'$. Therefore, we conclude $(T', r') \in \mathcal{R}$ and, by symmetry of $\mathcal{R}$, $(r', T') \in \mathcal{R}$;
c: assume that $T \xrightarrow{\tau} T'$. By the semantics in Definition 2.10 we are in case *3.* above (i.e., both T and U are internal choices with multiple branches). Hence, $\exists U' \,.\, U \xrightarrow{\tau} U'$ and $T' \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U' \bowtie r$ (by Proposition B.1). Therefore, we conclude $(T', r) \in \mathcal{R}$ and, by symmetry of $\mathcal{R}$, $(r, T') \in \mathcal{R}$;
d: assume that $r \xrightarrow{\tau} r'$. By Proposition B.1 we have $T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U \bowtie r'$. Therefore, we conclude $(T, r') \in \mathcal{R}$ and, by symmetry of $\mathcal{R}$, $(r', T) \in \mathcal{R}$;

Hence, $\mathcal{R}$ is a symmetric I/O compliance relation. Now, we observe that for all T, U and $r \in \mathbb{U}$, $T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U$ and $U \bowtie r$ imply $(T, r) \in \mathcal{R} \ni (r, T)$, and so $T \bowtie r$. By Definition 6.1, we conclude $T \sqsubseteq U$. □

Lemma 6.7. *Let $\ddot{\mathbb{R}}$ be a set of I/O simulations. Then, $\bigcup\ddot{\mathbb{R}}$ is an I/O simulation.*

Proof. We show that $\forall (p,q) \in (\bigcup\ddot{\mathbb{R}})$, all conditions in Definition 6.4 hold. We simply notice that when $p\ (\bigcup\ddot{\mathbb{R}})\ q$, then there exists an I/O simulation $\ddot{\mathcal{R}} \subseteq (\bigcup\ddot{\mathbb{R}})$ such that $p\ \ddot{\mathcal{R}}\ q$, for some predictive set $\mathbb{Q}$; using such a predictive set, clauses *a*–*b* of Definition 6.4 hold — and moreover, the pairs of reducts $(p', q') \in \ddot{\mathcal{R}}$ from clauses *c*–*e* also belong to $(\bigcup\ddot{\mathbb{R}})$. □

D.1 On $\ddot{\leqslant}$ as a preorder for $\mathbb{U}$

Lemma D.1. *$\ddot{\leqslant}$ is reflexive.*

Proof. Consider the identity relation: $\ddot{\mathcal{R}} = \{(p,p) \,|\, p \in \mathbb{U}\}$. We can easily verify that that $\ddot{\mathcal{R}}$ is an I/O simulation, with $\{p\}$ being the predictive supporting each pair $p\ \ddot{\mathcal{R}}\ p$. □

Lemma D.2 ($\ddot{\leqslant}$ and τ-moves (I)). *Let $p \ddot{\leqslant} q$, with predictive set $\mathbb{Q}$. Then, $p \Rightarrow p'$ implies $p' \ddot{\leqslant} q$, with predictive set $\mathbb{Q}'$ such that $\mathbb{Q} \Rightarrow \mathbb{Q}'$.*

Proof. We proceed by induction on the length of the sequence of τ-transitions in $p \Rightarrow p'$. The base case ($n = 0$) is trivial. For the inductive case, let $p \Rightarrow p^* \xrightarrow{\tau} p'$. By the induction hypothesis, we have $p^* \ddot{\leqslant} q$ with predictive set $\mathbb{Q}^*$ such that $\mathbb{Q} \Rightarrow \mathbb{Q}^*$. To show that $p' \ddot{\leqslant} q$, we observe that, (by item *c* of Definition 6.4) $p^* \xrightarrow{\tau} p'$ implies that $\exists q' \,.\, \mathbb{Q}^* \Rightarrow q' \wedge p' \ddot{\leqslant} q'$; furthermore, $p' \ddot{\leqslant} q'$ is supported by some predictive set $\mathbb{Q}'$ such that $q' \Rightarrow \mathbb{Q}'$. Let now $\ddot{\mathcal{R}} = \ddot{\leqslant} \cup \{(p', q)\}$: we show that $\ddot{\mathcal{R}}$ is an I/O simulation. This claim trivially holds for the subset $\ddot{\leqslant} \subseteq \ddot{\mathcal{R}}$, and thus we only need to check whether the clauses of Definition 6.4 hold for the additional pair (p', q). Since $q \Rightarrow q'$ and $q' \Rightarrow \mathbb{Q}'$, we have $q \Rightarrow \mathbb{Q}'$. Then, using $\mathbb{Q}'$ as a predictive set for (p', q), we have:

item *a*: $p'\Downarrow^! = \emptyset \implies \mathbb{Q}'\Downarrow^! = \emptyset$ (since $\mathbb{Q}'$ is a predictive set for $p' \ddot{\leqslant} q'$);
item *b*: $\mathbb{Q}'\Downarrow^{??} \subseteq p'\Downarrow^?$ and $\mathbb{Q}'\Downarrow^? = \emptyset \implies p'\Downarrow^? = \emptyset$ (since $\mathbb{Q}'$ is a predictive set for $p' \ddot{\leqslant} q'$);
item *c* let $p' \xrightarrow{\tau} p''$. Since $p' \ddot{\leqslant} q'$, we have that $\exists q'' \,.\, \mathbb{Q}' \Rightarrow q'' \wedge p'' \ddot{\leqslant} q''$, which is the thesis;
item *d* let $p' \xrightarrow{!\mathsf{a}} p''$. Since $p' \ddot{\leqslant} q'$, we have that $\exists q'' \,.\, \mathbb{Q}' \overset{!\mathsf{a}}{\Rightarrow} q'' \wedge p'' \ddot{\leqslant} q''$, which is the thesis;
item *e* let $p' \xrightarrow{?\mathsf{a}} p'' \wedge \mathbb{Q}' \overset{??\mathsf{a}}{\Longrightarrow}$. Since $p' \ddot{\leqslant} q'$, we have that $\exists q'' \,.\, \mathbb{Q}' \overset{?\mathsf{a}}{\Rightarrow} q'' \wedge p'' \ddot{\leqslant} q''$, which is the thesis.

Therefore, $\ddot{\mathcal{R}}$ is an I/O simulation, and $(p', q) \in \ddot{\mathcal{R}}$.

We conclude by observing that, from $\mathbb{Q} \Rightarrow \mathbb{Q}^*$, $\mathbb{Q}^* \Rightarrow q'$ and $q' \Rightarrow \mathbb{Q}'$, we have $\mathbb{Q} \Rightarrow \mathbb{Q}'$. □

Lemma D.3 ($\ddot{\leqslant}$ and τ-moves (II)). *Let $p \ddot{\leqslant} q$, with predictive set $\mathbb{Q}$. Then, $q' \Rightarrow q$ implies $p \ddot{\leqslant} q'$, again with predictive set $\mathbb{Q}$.*

Proof. We proceed by induction on the length of the sequence of τ-transitions in $q' \Rightarrow q$. The base case ($n = 0$) is trivial. For the inductive case, let $q' \xrightarrow{\tau} q^* \Rightarrow q$. By the induction hypothesis, we have $p \ddot{\leqslant} q^*$, with predictive set $\mathbb{Q}$. Let now $\ddot{\mathcal{R}} = \ddot{\leqslant} \cup \{(p, q')\}$: we show that $\ddot{\mathcal{R}}$ is an I/O simulation. This claim trivially holds for the subset $\ddot{\leqslant} \subseteq \ddot{\mathcal{R}}$, and thus we only

need to check whether the clauses of Definition 6.4 hold for the additional pair (p, q'). Since $q' \xrightarrow{\tau} q^*$ and $q^* \Rrightarrow \mathbb{Q}$, we have $q' \Rrightarrow \mathbb{Q}$. Then, using $\mathbb{Q}$ as a predictive set for (p, q'), we have:

item *a*: $p\Downarrow^! = \emptyset \implies \mathbb{Q}\Downarrow^! = \emptyset$ (since $\mathbb{Q}$ is a predictive set for $p \mathrel{\ddot\lesssim} q^*$);

item *b*: $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^?$ and $\mathbb{Q}\Downarrow^? = \emptyset \implies p\Downarrow^? = \emptyset$ (since $\mathbb{Q}$ is a predictive set for $p \mathrel{\ddot\lesssim} q^*$);

item *c* let $p \xrightarrow{\tau} p'$. Since $p \mathrel{\ddot\lesssim} q^*$, we have that $\exists q'' \,.\, \mathbb{Q} \Rrightarrow q'' \wedge p' \mathrel{\ddot\lesssim} q''$, which is the thesis;

item *d* let $p \xrightarrow{!\mathsf{a}} p'$. Since $p \mathrel{\ddot\lesssim} q^*$, we have that $\exists q'' \,.\, \mathbb{Q} \overset{!\mathsf{a}}{\Rightarrow} q'' \wedge p' \mathrel{\ddot\lesssim} q''$, which is the thesis;

item *e* let $p \xrightarrow{?\mathsf{a}} p' \wedge \mathbb{Q}\overset{??\mathsf{a}}{\Longrightarrow}$. Since $p \mathrel{\ddot\lesssim} q^*$, we have that $\exists q'' \,.\, \mathbb{Q} \overset{?\mathsf{a}}{\Rightarrow} q'' \wedge p' \mathrel{\ddot\lesssim} q''$, which is the thesis.

Therefore, $\ddot{\mathcal{R}}$ is an I/O simulation; we conclude by observing that $(p, q') \in \ddot{\mathcal{R}}$. □

Lemma 6.8. *If $p \mathrel{\ddot\lesssim} q$, $p \Rightarrow p'$ and $q' \Rightarrow q$, then $p' \mathrel{\ddot\lesssim} q'$.*

Proof. By Lemma D.2 we have that, whenever $p \mathrel{\ddot\lesssim} q$ holds, then $p \Rightarrow p'$ implies $p' \mathrel{\ddot\lesssim} q$; therefore, by Lemma D.3, $q' \Rightarrow q$ implies $p' \mathrel{\ddot\lesssim} q'$. □

D.2 Properties of predictive sets

Proposition D.4. *For all $\mathbb{Q} \subseteq \mathbb{U}$, if $?\mathsf{a} \in \mathbb{Q}\Downarrow^{??}$ then:*

(i) $\forall q \in \mathbb{Q} \,.\, q\overset{??\mathsf{a}}{\Longrightarrow}$

(ii) $\forall q' \,.\, \mathbb{Q} \Rrightarrow q'$ *implies* $q'\overset{??\mathsf{a}}{\Longrightarrow}$.

Proof. Direct consequences of Definition 2.5. □

Proposition D.5. *If $p \mathrel{\ddot\lesssim} q$ with pred. set $\mathbb{Q}$, then $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^{??}$.*

Proof. Assume $\mathbb{Q}\overset{??\mathsf{a}}{\Longrightarrow}$, and let p' such that $p \Rightarrow p'$. By Lemma D.2, we have $p' \mathrel{\ddot\lesssim} q$, with a predictive set $\mathbb{Q}'$ such that $\mathbb{Q} \Rrightarrow \mathbb{Q}'$. This, by Proposition D.4, means $\mathbb{Q}'\overset{??\mathsf{a}}{\Longrightarrow}$, and therefore (by item *b* of Definition 6.4), $p' \overset{?\mathsf{a}}{\Rightarrow}$. Hence, by Definition 2.5, we conclude $p\overset{??\mathsf{a}}{\Longrightarrow}$. □

Proposition D.6. *Let $p \mathrel{\ddot\lesssim} q$ with predictive set $\mathbb{Q}$. Then, $q\overset{??\mathsf{a}}{\Longrightarrow}$ implies $\mathbb{Q}\overset{??\mathsf{a}}{\Longrightarrow}$.*

Proof. By Definition 6.4, $\forall q' \in \mathbb{Q} \,.\, q \Rightarrow q'$. Therefore, by Definition 2.5, we have $q'\overset{??\mathsf{a}}{\Longrightarrow}$, and we conclude $\mathbb{Q}\overset{??\mathsf{a}}{\Longrightarrow}$. □

Lemma D.7 ("Neutral elements" in a predictive set)**.** *Let $p \mathrel{\ddot\lesssim} q$, for some predictive set $\mathbb{Q}$. Then, for all q_0 such that $q \Rightarrow q_0$ and $q_0\Downarrow^! \subseteq \mathbb{Q}\Downarrow^!$, we have that $\mathbb{Q}_1 = \mathbb{Q} \cup \{q_0\}$ is still a predictive set for $p \mathrel{\ddot\lesssim} q$.*

Proof. Immediate, by noticing that for all weak barbs of q_0 allowed by the hypotheses, we have:

- $\mathbb{Q}\Downarrow^! = \mathbb{Q}_1\Downarrow^!$, and therefore $p\Downarrow^! = \emptyset \implies \mathbb{Q}_1\Downarrow^! = \emptyset$, thus satisfying clause *a* of Definition 6.4.
- $\mathbb{Q}_1\Downarrow^{??} \subseteq \mathbb{Q}\Downarrow^{??}$ — and so $\mathbb{Q}_1$ does not require more inputs to p w.r.t. $\mathbb{Q}$, thus satisfying the first part of clause *b* of Definition 6.4. For the second part, we notice that $\mathbb{Q}_1\Downarrow^? = \emptyset$ implies $\mathbb{Q}\Downarrow^? = \emptyset$, which implies $p\Downarrow^? = \emptyset$;
- all states reachable from $\mathbb{Q}$ are still reachable from $\mathbb{Q}_1$ with the same transitions, thus satisfying clauses *c*–*d* of Definition 6.4;

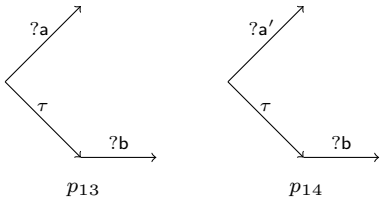


Figure 9: $p_{13} \mathrel{\ddot\approx} p_{14}$ but $p_{13} \not\mathrel{\ddot\approx} p_{14}$.

- when $\mathbb{Q}\Downarrow^{??} \ni ?\mathsf{a} \notin \mathbb{Q}_1\Downarrow^{??}$, then the premise of clause *e* of Definition 6.4 becomes false; otherwise, when $?\mathsf{a} \in \mathbb{Q}_1\Downarrow^{??} \cap \mathbb{Q}\Downarrow^{??}$, we notice (as in the previous point) that all states reachable from $\mathbb{Q}$ are still reachable from $\mathbb{Q}_1$ with the same transitions. Therefore, in both cases, clause *e* of Definition 6.4 is satisfied. □

Proposition D.8 ($\mathrel{\ddot\lesssim}$ vs. $\mathrel{\ddot\precsim}$)**.** $\mathrel{\ddot\lesssim} \not\subseteq \mathrel{\ddot\precsim} \not\subseteq \mathrel{\ddot\lesssim}$.

Proof. Consider the following CCS processes, where $\mathsf{a} \neq \mathsf{b}$:

- let $P = ?\mathsf{a} + ?\mathsf{b}$ and $Q = ?\mathsf{a}$. We have $P \mathrel{\ddot\lesssim} Q$ and $P \not\mathrel{\ddot\precsim} Q$;
- let $P = \tau.?\mathsf{a} + \tau.?\mathsf{b}$ and $Q = ?\mathsf{a} + ?\mathsf{b}$. We have $P \mathrel{\ddot\precsim} Q$ and $P \not\mathrel{\ddot\lesssim} Q$. □

Theorem 6.10. *$(\mathbb{U}, \mathrel{\ddot\lesssim})$ is a preorder.*

Proof. Reflexivity follows by Lemma D.1. For transitivity, let:

$$\ddot{\mathcal{R}} \;=\; \{(p, r) \mid \exists q \,.\, p \mathrel{\ddot\lesssim} q \wedge q \mathrel{\ddot\lesssim} r\}$$

We show that $\ddot{\mathcal{R}}$ is an I/O simulation. Let $(p, r) \in \ddot{\mathcal{R}}$, and let q be such that $p \mathrel{\ddot\lesssim} q$ and $q \mathrel{\ddot\lesssim} r$. Then, let:

(i) $\ddot{\mathcal{R}}_1$ be an I/O simulation such that $p \mathrel{\ddot{\mathcal{R}}_1} q$, with $\mathbb{Q}$ as predictive set. So, we have $q \Rrightarrow \mathbb{Q}$, and:

a. $p\Downarrow^! = \emptyset \implies \mathbb{Q}\Downarrow^! = \emptyset$;

b. $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^? \;\wedge\; \mathbb{Q}\Downarrow^? = \emptyset \implies p\Downarrow^? = \emptyset$;

c. $p \xrightarrow{\tau} p' \implies \exists q' \,.\, \mathbb{Q} \Rightarrow q' \wedge p' \mathrel{\ddot{\mathcal{R}}_1} q'$;

d. $p \xrightarrow{!\mathsf{a}} p' \implies \exists q' \,.\, \mathbb{Q} \overset{!\mathsf{a}}{\Rightarrow} q' \wedge p' \mathrel{\ddot{\mathcal{R}}_1} q'$;

e. $p \xrightarrow{?\mathsf{a}} p' \wedge \mathbb{Q}\overset{??\mathsf{a}}{\Longrightarrow} \implies \exists q' \,.\, \mathbb{Q} \overset{?\mathsf{a}}{\Rightarrow} q' \wedge p' \mathrel{\ddot{\mathcal{R}}_1} q'$;

(ii) $\ddot{\mathcal{R}}_2$ be an I/O simulation such that $q \mathrel{\ddot{\mathcal{R}}_2} r$.

In the following, let I index the elements of $\mathbb{Q}$ — i.e., $\mathbb{Q} = \{q_i\}_{i \in I}$. Before proceeding, we highlight some results and definitions that we will reuse throughout this proof.

Proposition D.9. *$\forall i \in I$, we have $q_i \mathrel{\ddot\lesssim} r$.*

Proof. Since $\forall i \in I \,.\, q \Rightarrow q_i$, the statement follows from Lemma D.2. □

Definition D.10. *$\forall i \in I$, we fix $\mathbb{R}_i$ as a predictive set supporting $q_i \mathrel{\ddot\lesssim} r$. Furthermore, we define:*

$$\mathbb{R} \;=\; \bigcup_{i \in I} \mathbb{R}_i$$

Corollary D.11. *$\mathbb{R}\Downarrow^{??} \subseteq \mathbb{Q}\Downarrow^{??}$.*

Proof. Let $?\mathsf{a} \in \mathbb{R}\Downarrow^{??}$. Then, $?\mathsf{a} \in \tilde{r}\Downarrow^{??}$, for all $\tilde{r} \in \mathbb{R}$, and so $?\mathsf{a} \in \mathbb{R}_i\Downarrow^{??}$. Since $q_i \mathrel{\ddot\lesssim} r$ (by Proposition D.9) with predictive set $\mathbb{R}_i$ (by Definition D.10), then by Proposition D.5 it follows that $?\mathsf{a} \in q_i\Downarrow^{??}$. Since the above holds for all $i \in I$, we conclude that $?\mathsf{a} \in \mathbb{Q}\Downarrow^{??}$. □

We now show that $\mathbb{R}$ is a predictive set for the pair $(p, r) \in \ddot{\mathcal{R}}$, according to Definition 6.4:

item *a*: assume that $p\Downarrow^! = \emptyset$. Then, by item *(i)a.*, we have $\mathbb{Q}\Downarrow^! = \emptyset$ — and therefore, $q_i\Downarrow^! = \emptyset$ for all $i \in I$. Since $q_i \mathrel{\ddot{\lesssim}} r$, by item *a* of Definition 6.4, this implies that $\mathbb{R}_i\Downarrow^! = \emptyset$, for all $i \in I$. Therefore $\mathbb{R}\Downarrow^! = \emptyset$.

item *b*: for the first part of the item, from Corollary D.11, we have $\mathbb{R}\Downarrow^{??} \subseteq \mathbb{Q}\Downarrow^{??}$. Furthermore, by item *(i)b.*, we have $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^?$. So, we conclude $\mathbb{R}\Downarrow^{??} \subseteq p\Downarrow^?$.
For the second part of item *b*, assume $\mathbb{R}\Downarrow^? = \emptyset$: for all $i \in I$, this implies $\mathbb{R}_i\Downarrow^? = \emptyset$; by item *b* of Definition 6.4, we have $\forall i \in I \,.\, q_i\Downarrow^? = \emptyset$, which in turn implies $\mathbb{Q}\Downarrow^? = \emptyset$; and from item *(i)b.*, we have $p\Downarrow^? = \emptyset$. Therefore, we conclude $p\Downarrow^? = \emptyset$.

item *c*: we have to show that, whenever $p \xrightarrow{\tau} p'$, then $\exists r' \,.\, \mathbb{R} \Rightarrow r' \wedge (p', r') \in \ddot{\mathcal{R}}$. From item *(i)c.* we know that whenever $p \xrightarrow{\tau} p'$, $\exists q' \,.\, \mathbb{Q} \Rightarrow q' \wedge (p', q') \in \ddot{\mathcal{R}}_1$; therefore, for some $i \in I$, $q_i \Rightarrow q'$. Now, from Lemma D.3, we have $p' \mathrel{\ddot{\lesssim}} q_i$. Since (by Proposition D.9) we also have $q_i \mathrel{\ddot{\lesssim}} r$, we choose $r' = r$, and we conclude $(p', r') \in \ddot{\mathcal{R}}$.

item *d*: we have to show that, whenever $p \xrightarrow{!\mathsf{a}} p'$, then $\exists r' \,.\, \mathbb{R} \overset{!\mathsf{a}}{\Rightarrow} r'$ and $(p', r') \in \ddot{\mathcal{R}}$. By item *(i)d.* above, we know that whenever $p \xrightarrow{!\mathsf{a}} p'$, then $\exists i \in I, q_*, q'_*, q'$ such that $\mathbb{Q} \ni q_i \Rightarrow q_* \xrightarrow{!\mathsf{a}} q'_* \Rightarrow q'$ and $p' \mathrel{\ddot{\mathcal{R}}_1} q'$. Since $p' \mathrel{\ddot{\lesssim}} q'$ and $q'_* \Rightarrow q'$, then by Lemma D.3 it follows that $p' \mathrel{\ddot{\lesssim}} q'_*$. By Proposition D.9 and Definition D.10, we have $q_i \mathrel{\ddot{\lesssim}} r$ with some predictive set $\mathbb{R}_i \subseteq \mathbb{R}$; and by Lemma D.2, we also have $q_* \mathrel{\ddot{\lesssim}} r$ with some predictive set $\mathbb{R}_*$ such that $\mathbb{R}_i \Rightarrow \mathbb{R}_*$ — and therefore (by item *d* of Definition 6.4):

$$\exists r'' \,.\, \mathbb{R}_i \Rightarrow \mathbb{R}_* \overset{!\mathsf{a}}{\Rightarrow} r'' \wedge q'_* \mathrel{\ddot{\lesssim}} r'' \tag{12}$$

Combining $\mathbb{R} \Rightarrow \mathbb{R}_i$ and Equation (12) above, we obtain:

$$\exists r'' \,.\, \mathbb{R} \overset{!\mathsf{a}}{\Rightarrow} r'' \wedge q'_* \mathrel{\ddot{\lesssim}} r''$$

Let $r' = r''$. Since $p' \mathrel{\ddot{\lesssim}} q'_*$ and $q'_* \mathrel{\ddot{\lesssim}} r'$, we conclude that $(p', r') \in \ddot{\mathcal{R}}$.

item *e*: we have to show that, whenever $p \xrightarrow{?\mathsf{a}} p'$ and $\mathbb{R} \overset{??\mathsf{a}}{\Longrightarrow}$, then $\exists r' \,.\, \mathbb{R} \overset{?\mathsf{a}}{\Rightarrow} r'$ and $(p', r') \in \ddot{\mathcal{R}}$. Assume that $p \xrightarrow{?\mathsf{a}} p'$ and $\mathbb{R} \overset{??\mathsf{a}}{\Longrightarrow}$. By Corollary D.11, we have $\mathbb{Q} \overset{??\mathsf{a}}{\Longrightarrow}$. Therefore, by item *(i)e.* above, $\exists i \in I, q_*, q'_*, q'$ such that $\mathbb{Q} \ni q_i \Rightarrow q_* \xrightarrow{?\mathsf{a}} q'_* \Rightarrow q'$ and $p' \mathrel{\ddot{\mathcal{R}}_1} q'$. By Proposition D.9 and Definition D.10, we have $q_i \mathrel{\ddot{\lesssim}} r$ with some predictive set $\mathbb{R}_i \subseteq \mathbb{R}$; and by Lemma D.2, we also have $q_* \mathrel{\ddot{\lesssim}} r$ with some predictive set $\mathbb{R}_*$ such that $\mathbb{R}_i \Rightarrow \mathbb{R}_*$. Therefore, by item *e* of Definition 6.4:

$$\mathbb{R}_* \overset{??\mathsf{a}}{\Longrightarrow} \text{ implies } \exists r'' \,.\, \mathbb{R}_i \Rightarrow \mathbb{R}_* \overset{?\mathsf{a}}{\Rightarrow} r'' \wedge q'_* \mathrel{\ddot{\lesssim}} r'' \tag{13}$$

Finally, we notice that since $\mathbb{R} \overset{??\mathsf{a}}{\Longrightarrow}$ implies $\mathbb{R}_i \overset{??\mathsf{a}}{\Longrightarrow}$, we have $\mathbb{R}_* \overset{??\mathsf{a}}{\Longrightarrow}$; this, combined with $\mathbb{R} \Rightarrow \mathbb{R}_i$ and Equation (13) above, gives us:

$$\exists r'' \,.\, \mathbb{R} \overset{?\mathsf{a}}{\Rightarrow} r'' \wedge q'_* \mathrel{\ddot{\lesssim}} r''$$

By Lemma D.2, we have $q' \mathrel{\ddot{\lesssim}} r''$. Let $r' = r''$. Since $p' \mathrel{\ddot{\lesssim}} q'$ and $q' \mathrel{\ddot{\lesssim}} r'$, we conclude that $(p', r') \in \ddot{\mathcal{R}}$. □

Theorem 6.11. $\approx \subsetneq \mathrel{\ddot{\approx}}$

Proof. To show that $\approx \neq \mathrel{\ddot{\approx}}$, consider Figure 9: we have that $p_{13} \mathrel{\ddot{\approx}} p_{14}$, but $p_{13} \not\approx p_{14}$. To show that $\approx \subset \mathrel{\ddot{\approx}}$, since the weak bisimulation relation is symmetric, it is enough to prove that it is an I/O simulation. Let $p \approx q$. We have that $p \xrightarrow{\ell} p'$ implies $\exists q' \,.\, q \overset{\ell}{\Rightarrow} q' \wedge p' \approx q'$, and the converse holds for the ℓ-transitions emanating from q. From this, it is immediate to see that $p\Downarrow^? = q\Downarrow^?$ and $p\Downarrow^! = q\Downarrow^!$, and using the former in an inductive argument we also obtain $p\Downarrow^{??} = q\Downarrow^{??}$. Hence, for all $(p, q) \in \approx$, we can satisfy clauses *a*–*e* in Definition 6.4 by taking as predictive set $\mathbb{Q} = \{q\}$. □

Theorem 6.12. $p \mathrel{\ddot{\lesssim}} q \circ r \implies p \circ r$, *for* $\circ \in \{\ddot{\triangleright}, \ddot{\bowtie}\}$.

Proof. We first prove the statement for $\circ = \ddot{\triangleright}$. Let:

$$\mathcal{R} = \{(p, r) \mid \exists q \,.\, p \mathrel{\ddot{\lesssim}} q \wedge q \mathrel{\ddot{\triangleright}} r\}$$

We will show that $\mathcal{R}$ is an I/O compliance relation between p and r (in the inverse order). Let $(p, r) \in \mathcal{R}$, via some q such that $p \mathrel{\ddot{\lesssim}} q$ and $r \mathrel{\ddot{\triangleleft}} q$. Now, let $\mathbb{Q}$ be the predictive set supporting the pair $(p, q) \in \mathrel{\ddot{\lesssim}}$, and let us examine the clauses of Definition 4.4.

item *a* From $p \mathrel{\ddot{\lesssim}} q$ (item *b* of Definition 6.4) we have $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^?$; furthermore, from $r \mathrel{\ddot{\triangleleft}} q$ (by Proposition B.3) we have $r\Downarrow^! \subseteq \mathrm{co}(q\Downarrow^{??})$, and from Proposition D.6 we obtain:

$$r\Downarrow^! \subseteq \mathrm{co}(\mathbb{Q}\Downarrow^{??}) \tag{14}$$

Furthermore, since $\forall q' \in \mathbb{Q} \,.\, q \Rightarrow q'$, from $r \mathrel{\ddot{\triangleleft}} q$ and Proposition B.1 we have $\forall q' \in \mathbb{Q} \,.\, r \mathrel{\ddot{\triangleleft}} q'$. Hence, by item *b* of Definition 6.4:

$$\forall q' \in \mathbb{Q} \;.\; \left(r\Downarrow^! = \emptyset \wedge r\Downarrow^? \neq \emptyset \implies \emptyset \neq q'\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?)\right)$$

and therefore:

$$r\Downarrow^! = \emptyset \wedge r\Downarrow^? \neq \emptyset \implies \emptyset \neq \mathbb{Q}\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?) \tag{15}$$

From $p \mathrel{\ddot{\lesssim}} q$, by Definition 6.4 we have:
- $p\Downarrow^! = \emptyset \implies \mathbb{Q}\Downarrow^! = \emptyset$ (item *a* of Definition 6.4), and therefore $\mathbb{Q}\Downarrow^! \neq \emptyset \implies p\Downarrow^! \neq \emptyset$;
- $p\Downarrow^! \subseteq \mathbb{Q}\Downarrow^!$ (item *d*)
- $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^?$ (item *b*).

Summing up:

$$\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^? \quad \text{and} \quad \mathbb{Q}\Downarrow^! \neq \emptyset \implies \emptyset \neq p\Downarrow^! \subseteq \mathbb{Q}\Downarrow^! \tag{16}$$

and combining Equation (16) with Equations (14) and (15), we conclude:

$$\left(r\Downarrow^! \subseteq \mathrm{co}(p\Downarrow^?)\right) \wedge \left(r\Downarrow^! = \emptyset \wedge r\Downarrow^? \neq \emptyset \text{ implies } \emptyset \neq p\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?)\right)$$

item *b* We have to show that $p \xrightarrow{\ell} p'$ and $r \xrightarrow{\mathrm{co}(\ell)}$ implies $(p', r') \in \mathcal{R}$. Assuming the premise, we have two cases, depending on whether ℓ is an input or an output action:
- $\ell = !\mathsf{a}$. Then, by item *d* of Definition 6.4, $\exists q' \,.\, \mathbb{Q} \overset{!\mathsf{a}}{\Rightarrow} q' \wedge p' \mathrel{\ddot{\lesssim}} q'$. Now, from $q \mathrel{\ddot{\triangleright}} r$ and Proposition B.2 we have $q' \mathrel{\ddot{\triangleright}} r'$: we conclude $(p', r') \in \mathcal{R}$;
- $\ell = ?\mathsf{a}$. Then, $r \xrightarrow{!\mathsf{a}} r'$; hence, from $q \mathrel{\ddot{\triangleright}} r$ and Proposition B.3 we have $q \overset{??\mathsf{a}}{\Longrightarrow}$, and by Proposition D.6 we obtain $\mathbb{Q} \overset{??\mathsf{a}}{\Longrightarrow}$. Thus, by item *e* of Definition 6.4, $\exists q' \,.\, \mathbb{Q} \overset{?\mathsf{a}}{\Rightarrow} q' \wedge p' \mathrel{\ddot{\lesssim}} q'$. Now, from $q \mathrel{\ddot{\triangleright}} r$ and Proposition B.2 we have $q' \mathrel{\ddot{\triangleright}} r'$: we conclude $(p', r') \in \mathcal{R}$;

item *c* From $p \mathrel{\ddot{\lesssim}} q$ and Lemma D.2, we have that $p \xrightarrow{\tau} p'$ implies $p' \mathrel{\ddot{\lesssim}} q$: we conclude $(p', r) \in \mathcal{R}$;

item *d* From $q \mathrel{\ddot{\triangleright}} r$, by item *d* of Definition 4.4, we know that $r \xrightarrow{\tau} r'$ implies $q \mathrel{\ddot{\triangleright}} r'$: we conclude $(p, r') \in \mathcal{R}$.

This concludes the proof for $\circ = \ddot{\triangleright}$.

The proof for $\circ = \ddot{\bowtie}$ follows a similar approach, but this time we let:

$$\mathcal{R} = \{(p, r) \mid \exists q \,.\, p \mathrel{\ddot{\lesssim}} q \wedge q \mathrel{\ddot{\bowtie}} r\} \tag{17}$$

Now, we prove that $\mathcal{R}$ is a symmetric I/O compliance relation. Since $\bowtie = \bar{\triangleright} \cap \ddot{\triangleleft}$ (by Lemma 4.5), most of the proof is already developed above, when $\circ = \bar{\triangleright}$. The only difference is that, in the *symmetric* version of item *a* of Definition 4.4, we have the following *additional* clause, introduced by $\ddot{\triangleleft}$, that needs to be satisfied:

$$(p\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?) \wedge (p\Downarrow^! = \emptyset \wedge p\Downarrow^? \neq \emptyset \text{ implies } \emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(p\Downarrow^?))) \quad (18)$$

We proceed by proving such a clause, for each $(p, r) \in \mathcal{R}$ and corresponding q from Equation (17). From $p \ddot{\leqslant} q$ (item *d* of Definition 6.4) we have $p\Downarrow^! \subseteq \mathbb{Q}\Downarrow^! \subseteq q\Downarrow^!$, and from $q \ddot{\triangleleft} r$ (item *a* of Definition 4.4) we have $q\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?)$; therefore,

$$p\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?) \quad (19)$$

Furthermore, from $q \ddot{\triangleleft} r$ (item *a* of Definition 4.4) we have:

$$q\Downarrow^! = \emptyset \wedge q\Downarrow^? \neq \emptyset \implies \emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(q\Downarrow^?) \quad (20)$$

If the premise holds, from $q \ddot{\triangleleft} r$ and Proposition B.5, we have $r\Downarrow^! \subseteq \mathrm{co}(q\Downarrow^{??})$, and from Proposition D.6 we have $q\Downarrow^{??} \subseteq \mathbb{Q}\Downarrow^{??}$ — and therefore $r\Downarrow^! \subseteq \mathrm{co}(\mathbb{Q}\Downarrow^{??})$. Combining these observations with Equation (20), we obtain:

$$q\Downarrow^! = \emptyset \wedge q\Downarrow^? \neq \emptyset \implies \emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(\mathbb{Q}\Downarrow^{??}) \quad (21)$$

Since $\forall q' \in \mathbb{Q} \,.\, q \Rightarrow q'$, by Proposition B.1 we have $\forall q' \in \mathbb{Q} \,.\, q' \ddot{\triangleleft} r$. Then, from Equation (21) we have:

$$\forall q' \in \mathbb{Q} \,.\, q'\Downarrow^! = \emptyset \wedge q'\Downarrow^? \neq \emptyset \implies \emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(\mathbb{Q}\Downarrow^{??}) \quad (22)$$

and therefore:

$$\mathbb{Q}\Downarrow^! = \emptyset \wedge \mathbb{Q}\Downarrow^? \neq \emptyset \implies \emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(\mathbb{Q}\Downarrow^{??}) \quad (23)$$

From $p \ddot{\leqslant} q$, by Definition 6.4 we have:

- $p\Downarrow^! = \emptyset \implies \mathbb{Q}\Downarrow^! = \emptyset$ (item *a*);
- $\mathbb{Q}\Downarrow^{??} \subseteq p\Downarrow^?$ (item *b*), and therefore $\mathrm{co}(\mathbb{Q}\Downarrow^{??}) \subseteq \mathrm{co}(p\Downarrow^?)$;
- $\mathbb{Q}\Downarrow^? = \emptyset \implies p\Downarrow^? = \emptyset$ (item *b*), and therefore $p\Downarrow^? \neq \emptyset \implies \mathbb{Q}\Downarrow^? \neq \emptyset$.

Summing up:

$$p\Downarrow^! = \emptyset \wedge p\Downarrow^? \neq \emptyset \implies \mathbb{Q}\Downarrow^! = \emptyset \wedge \mathbb{Q}\Downarrow^? \neq \emptyset \wedge \mathrm{co}(\mathbb{Q}\Downarrow^{??}) \subseteq \mathrm{co}(p\Downarrow^?) \quad (24)$$

and combining Equation (24) with Equations (16) and (19), we conclude:

$$(p\Downarrow^! \subseteq \mathrm{co}(r\Downarrow^?)) \wedge (p\Downarrow^! = \emptyset \wedge p\Downarrow^? \neq \emptyset \text{ implies } \emptyset \neq r\Downarrow^! \subseteq \mathrm{co}(p\Downarrow^?))$$

which corresponds to Equation (18).

The proofs for the remaining items *b*–*d* can be obtained from the ones provided for $\circ = \bar{\triangleright}$, simply by replacing all occurrences of $\bar{\triangleright}$ with $\bowtie$. We conclude that the statement also holds for $\circ = \bowtie$. □

Theorem 6.13. *(a)* $\ddot{\leqslant} \subsetneq \sqsubseteq$. *(b)* $T \ddot{\leqslant} U$ *iff* $T \sqsubseteq U$.

Proof. For item (a), if $p \ddot{\leqslant} q$, then by Theorem 6.12 we have that $\forall r \,.\, q \bowtie r \implies p \bowtie r$, i.e. $p \sqsubseteq q$ according to Definition 6.1. To prove that the inclusion is strict, recall the behaviour p_5 from Example 4.7. Since p_5 does not admit I/O compliant behaviours, it vacuously follows that $r \sqsubseteq p_5$, for all r. In particular, let r be the behaviour which loops onto itself via a $!\mathsf{a}$-transition, and assume, by contradiction, that $r \ddot{\leqslant} p_5$, (say, via predictive set $\mathbb{Q}$). By clause *d* of Definition 6.4, there exists p' such that $\mathbb{Q} \overset{!\mathsf{a}}{\Rightarrow} p'$ — contradiction, since $\mathbb{Q}\Downarrow^! \subseteq p_5\Downarrow^! = \emptyset$.

For item (b), direction $\Longleftarrow$, let $T \sqsubseteq U$. By Lemma 6.3, we have $T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U$, hence by Lemma 6.2 we can reason (up-to unfolding) by cases on the syntax T and U, verifying that (T, U) satisfies all clauses in Definition 6.4, via predictive set $\mathbb{Q} = \{U\}$. The $\implies$ direction is already proved by item (a). □

Lemma 6.15. $T \ddot{\leqslant} U \ddot{\triangleleft} V$ *implies* $T \ddot{\triangleleft} V$.

Proof. We adapt the proof of Theorem 6.12 as follows:

- for clause *a* of Definition 4.4, we can proceed as in the proof of Equation (18) on page 34, which only relies on $\ddot{\triangleleft}$;
- for clause *b*, the proof of Equation (18) for the sub-case $\ell = ?\mathsf{a}$ (page 33) does not hold for $\ddot{\triangleleft}$, because Proposition B.3 cannot be applied. We then deal with such case as follows:
  - Since $V \xrightarrow{!\mathsf{a}} V'$, by Proposition A.3 (item *(iii)*) then V is equivalent to an internal choice. Then, by Theorem 4.9, $U \ddot{\triangleleft} V$ implies $U \dashv V$, and so Proposition 4.3 we have that U is a external choice. By Theorem 6.13 and Lemma 6.3, $T \ddot{\leqslant} U$ implies $T \sqsubseteq_{\mathbb{U}_{\mathrm{ST}}} U$, and so by Lemma 6.2 we have that T is a larger (possibly empty) external choice. Since $T \xrightarrow{?\mathsf{a}}$, such choice is not empty, and so by Lemma 6.2 we have that also U is not empty. Since $V \xrightarrow{!\mathsf{a}}$, then by the second part of clause *a* of Definition 4.4 it follows that $U \xrightarrow{?\mathsf{a}}$. From Definition 2.10, this implies $U \overset{??\mathsf{a}}{\Longrightarrow}$ — which in turn implies $\mathbb{Q} \overset{??\mathsf{a}}{\Longrightarrow}$. Thus, by item *e* of Definition 6.4, $\exists U' \,.\, \mathbb{Q} \overset{?\mathsf{a}}{\Rightarrow} U' \wedge T' \ddot{\leqslant} U'$. Now, from $U \ddot{\triangleleft} V$ and Proposition B.2 we have $U' \ddot{\triangleleft} V'$: we conclude $(T', V') \in \mathcal{R}$.
- the proofs of clauses *c* and *d* are unmodified. □

D.3 I/O simulation and asychrony

Proposition D.12. *Let* $p \ddot{\leqslant} q$. *Then, for all* $w \in (\mathsf{A}^!)^*$, $p \overset{w}{\Rightarrow} p'$ *implies* $\exists q' \,.\, q \overset{w}{\Rightarrow} q'$ *and* $p' \ddot{\leqslant} q'$.

Proof. By induction on w. In the base case (when w is empty) we need to prove that $p \Rightarrow p'$ implies $\exists q' \,.\, q \Rightarrow q'$ and $p' \ddot{\leqslant} q'$: this follows from Lemma D.2, letting $q' = q$.

For the inductive step, let $w = w'!\mathsf{a}$ (where w' is a sequence of outputs). We have $p \overset{w'}{\Rightarrow} p''$ and $q \overset{w'}{\Rightarrow} q''$, with $p'' \ddot{\leqslant} q''$ (by the induction hypothesis); we need to show that:

$$\forall p_0, p_1 \,.\, p'' \Rightarrow p_0 \xrightarrow{!\mathsf{a}} p_1 \Rightarrow p' \quad \text{implies} \quad \exists q' . q'' \overset{!\mathsf{a}}{\Rightarrow} q' \text{ and } p' \ddot{\leqslant} q'$$

Now, by Lemma D.2, we have $p_0 \ddot{\leqslant} q''$; by clause *d* of Definition 6.4, we have that for some predictive set $\mathbb{Q}$, $p_0 \xrightarrow{!\mathsf{a}} p_1$ implies $\exists q''' \,.\, \mathbb{Q} \overset{!\mathsf{a}}{\Rightarrow} q'''$ and $p_1 \ddot{\leqslant} q'''$; since (by Definition 6.4) $q'' \Rrightarrow \mathbb{Q}$, we have $q'' \overset{!\mathsf{a}}{\Rightarrow} q'''$; finally, by Lemma D.2, we obtain $p' \ddot{\leqslant} q'''$. We conclude by letting $q' = q'''$. □

Proposition D.13. *Let* $p \preceq_? q$. *Then,* $\forall w \,.\, p \overset{w}{\Rightarrow} p'$ *and* $q \overset{w}{\Rightarrow} q'$ *implies* $p' \preceq_? q'$.

Proof. We prove the contrapositive. We have that $\exists w \,.\, p \overset{w}{\Rightarrow} p'$ and $q \overset{w}{\Rightarrow} q'$ but $p' \not\preceq_? q'$. This means that $\exists w', p''', q''', \mathsf{a}$ such that:

- $p' \overset{w'}{\Longrightarrow} p'''$ and $q' \overset{w'}{\Longrightarrow} q'''$ (item 2a of Definition 6.18), and
- $\forall q'' . q''' \overset{!}{\Rightarrow}^* q''$ implies $q'' \overset{!}{\Rightarrow}^* \xrightarrow{?\mathsf{a}}$ (item 2b of Definition 6.18),

and $p''' \overset{!}{\Rightarrow}^* \xrightarrow{?\mathsf{a}}$. is false. Then, under the same existential quantifications, we also have:

- $p \overset{ww'}{\Longrightarrow} p'''$ and $q \overset{ww'}{\Longrightarrow} q'''$ (item 2a of Definition 6.18), and
- $\forall q'' . q''' \overset{!}{\Rightarrow}^* q''$ implies $q'' \overset{!}{\Rightarrow}^* \xrightarrow{?\mathsf{a}}$ (item 2b of Definition 6.18),

and $p''' \overset{!}{\Rightarrow}^* \xrightarrow{?\mathsf{a}}$. is false. Therefore, we conclude $p \not\preceq_? q$. □

Proposition D.14. *For all* σ, T, T', $w \in (\mathsf{A}^!)^*$: $T \overset{w}{\Rightarrow} T'$ *implies* $\exists \sigma' \,.\, T[\sigma] \Rightarrow T'[\sigma']$.

Proof. We proceed by induction on the length of the sequence of transitions in $T \overset{w}{\Rightarrow} T'$. The base case is trivial: since we have no transitions, then $T' = T$ and we conclude by letting $\sigma' = \sigma$. For the inductive cases, we have:

- $T \overset{w}{\Rightarrow} T'' \xrightarrow{\tau} T'$. By the induction hypothesis, we have $\exists \sigma'' \,.\, T[\sigma] \Rightarrow T''[\sigma'']$. By item *(ii)* of Proposition A.4, $\exists \mathsf{b} . T''[\sigma''] \xrightarrow{\tau} T'[\sigma''.!\mathsf{b}]$. We conclude by letting $\sigma' = \sigma''.!\mathsf{b}$;
- $T \overset{w'}{\Rightarrow} T'' \xrightarrow{!\mathsf{b}} T'$. By the induction hypothesis, we have $\exists \sigma'' \,.\, T[\sigma] \Rightarrow T''[\sigma'']$. By item *(iii)* of Proposition A.4, $T''[\sigma''] \xrightarrow{\tau} T'[\sigma''.!\mathsf{b}]$. Choosing $\sigma' = \sigma''.!\mathsf{b}$ concludes. □

Proposition D.15. *If* $T[\sigma] \Rightarrow T'[\sigma']$, *then* $\exists w \in (\mathsf{A}^!)^* . T \overset{w}{\Rightarrow} T'$.

Proof. We proceed by induction on the length of the sequence of transitions in $T[\sigma] \Rightarrow T'[\sigma']$. The base case is trivial: since we have no transitions, then $T' = T$ and $\sigma' = \sigma$ — and we conclude by letting w be the empty sequence. For the inductive case, let $T[\sigma] \Rightarrow T''[\sigma''] \xrightarrow{\tau} T'[\sigma']$. By the induction hypothesis, there exists a sequence of outputs w' such that $T \overset{w'}{\Rightarrow} T''$. Now, from $T''[\sigma''] \xrightarrow{\tau} T'[\sigma']$, by Definition 2.11 we have that T'' is equivalent to a non-empty internal choice with some $!\mathsf{b}$-guarded branch, and $\sigma' = \sigma''.!\mathsf{b}$; moreover, by Proposition A.4 (items *(ii)* and *(iii)*), we have either $T'' \xrightarrow{\tau} \xrightarrow{!\mathsf{b}} T'$ or $T \xrightarrow{!\mathsf{b}} T'$ — and therefore, $T'' \overset{!\mathsf{b}}{\Rightarrow} T'$. Hence, from $T \overset{w'}{\Rightarrow} T'' \overset{!\mathsf{b}}{\Rightarrow} T'$, we conclude by letting $w = w'!\mathsf{b}$. □

Theorem 6.20. *If* $T \,(\circ \cap \preceq_?)\, U$ *then* $T[] \circ U[]$, *for* $\circ \in \{\lessdot, \sqsubseteq\}$.

Proof. Let us define:

$$\ddot{\mathcal{R}} = \left\{ (T[\sigma] \,,\, U[\sigma]) \;\middle|\; T \,(\lessdot \cap \preceq_?)\, U \right\}$$

By the coinduction proof principle, we show that $\ddot{\mathcal{R}}$ is an I/O simulation. For each $(T[\sigma] \,,\, U[\sigma]) \in \ddot{\mathcal{R}}$, we define $\mathbb{U} = \{U[\sigma]\}$ as predictive set, and verify the clauses of Definition 6.4:

item *a* assume $T[\sigma]\Downarrow^! = \emptyset$. Then, by Definition 2.11 we have $\sigma = \epsilon$, and T can only be a (possibly empty) external choice. Since $T \lessdot U$, we have that U is also an external choice (by Theorem 6.13, Lemma 6.3 and Lemma 6.2) — and thus, since $\mathbb{U} = \{U[\sigma]\}$, we conclude $\mathbb{U}\Downarrow^! = \emptyset$;

item *b* since $T \lessdot U$, from Theorem 6.13, Lemma 6.3 and Lemma 6.2 we can determine that (up-to unfolding) T and U can be either:

- both empty choices, i.e. $T = U = \mathbf{0}$. Then, since $\mathbb{U} = \{U[\sigma]\}$, we have $\emptyset = \mathbb{U}\Downarrow^{??} \subseteq T[\sigma]\Downarrow^? = \emptyset$ and $\left(\mathbb{U}\Downarrow^? = \emptyset \implies T[\sigma]\Downarrow^? = \emptyset\right)$;
- both *non*-empty external choices, with all branches of U included in T. We notice that, by Proposition A.4 (item *(i)*), we have:
  - $\forall \mathsf{a} \,.\, T \xrightarrow{?\mathsf{a}}$ iff $T[\sigma] \xrightarrow{?\mathsf{a}}$, and
  - $\forall \mathsf{a} \,.\, U \xrightarrow{?\mathsf{a}}$ iff $U[\sigma] \xrightarrow{?\mathsf{a}}$.

  Therefore, $\forall \mathsf{a} \,.\, U[\sigma] \xrightarrow{?\mathsf{a}}$ implies $T[\sigma] \xrightarrow{?\mathsf{a}}$. We also notice that, by Definition 2.11, we have:
  - $\forall \mathsf{a} \,.\, U[\sigma] \xrightarrow{?\mathsf{a}}$ iff $U[\sigma] \overset{??\mathsf{a}}{\Longrightarrow}$ (i.e., all inputs of external choices are persistent);
  - $\forall \mathsf{a} \,.\, T[\sigma] \overset{?\mathsf{a}}{\Rightarrow}$ iff $T[\sigma] \xrightarrow{?\mathsf{a}}$ (i.e., all weakly reachable inputs are also immediately enabled).

Summing up, $\forall \mathsf{a}$ we have:

$$\begin{array}{ccccc} U[\sigma] \overset{??\mathsf{a}}{\Longrightarrow} & \iff & U[\sigma] \xrightarrow{?\mathsf{a}} & \iff & U \xrightarrow{?\mathsf{a}} \\ & & & & \Downarrow \\ T[\sigma] \overset{?\mathsf{a}}{\Rightarrow} & \iff & T[\sigma] \xrightarrow{?\mathsf{a}} & \iff & T \xrightarrow{?\mathsf{a}} \end{array}$$

and therefore, since $\mathbb{U} = \{U[\sigma]\}$,

$$U[\sigma]\Downarrow^{??} = \mathbb{U}\Downarrow^{??} \subseteq T[\sigma]\Downarrow^?$$

Furthermore, we also have that $\mathbb{U}\Downarrow^? = \emptyset$ implies $T[\sigma]\Downarrow^? = \emptyset$ (vacuously);

- both *non*-empty internal choices. We first prove $\mathbb{U}\Downarrow^{??} \subseteq T[\sigma]\Downarrow^?$. If $\mathbb{U}\Downarrow^{??} = \emptyset$, the thesis is immediate. Otherwise, assume $\exists \mathsf{a} \,.\, \mathbb{U} \overset{??\mathsf{a}}{\Longrightarrow}$; since $\mathbb{U} = \{U[\sigma]\}$, this holds iff $U[\sigma] \overset{??\mathsf{a}}{\Longrightarrow}$ — and by Definition 2.5, this means:

$$\begin{array}{l} \forall U', \sigma' \;.\; U[\sigma] \Rightarrow U'[\sigma'] \\ \qquad \text{implies } \exists U'', \sigma'' \,.\, U'[\sigma'] \Rightarrow U''[\sigma''] \xrightarrow{?\mathsf{a}} \end{array}$$

By Proposition D.15, each τ-sequence between $U[\sigma]$ and $U'[\sigma']$ above corresponds to some sequence of outputs w such that $U \overset{w}{\Rightarrow} U'$; similarly, the τ-sequence between $U'[\sigma']$ and $U''[\sigma'']$ above corresponds to some sequence of outputs w' such that $U' \overset{w'}{\Rightarrow} U''$. Thus,

$$\begin{array}{l} \forall U', w = !\mathsf{b}_1, \ldots, !\mathsf{b}_n \;.\; U \overset{w}{\Rightarrow} U' \\ \qquad \text{implies } \exists U'', w' = !\mathsf{c}_1, \ldots, !\mathsf{c}_{n'} \,.\, U' \overset{w'}{\Rightarrow} U'' \xrightarrow{?\mathsf{a}} \end{array}$$

and therefore, by Notation 2.1,

$$\forall U' \;.\; U \overset{!}{\Rightarrow}^* U' \text{ implies } \exists U'' \,.\, U' \overset{!}{\Rightarrow}^* U'' \xrightarrow{?\mathsf{a}} \qquad (25)$$

Now, since $T \preceq_? U$, we have:

- an *empty* sequence of actions w_0 matches item 2a of Definition 6.18 for $T \overset{w_0}{\Rightarrow} T$ and $U \overset{w_0}{\Rightarrow} U$;
- from Equation (25), we have that $?\mathsf{a}$ is "persistent" in U by item 2b of Definition 6.18;
- so, by item 2c of Definition 6.18, $\exists T' \,.\, T \overset{!}{\Rightarrow}^* T' \xrightarrow{?\mathsf{a}}$.

Hence, $\exists w'' = !\mathsf{d}_1, \ldots, !\mathsf{d}_m$ such that $T \overset{w''}{\Rightarrow} T' \xrightarrow{?\mathsf{a}}$; and then, by Proposition D.14, we have $\exists \sigma' \,.\, T[\sigma] \Rightarrow T'[\sigma'] \xrightarrow{?\mathsf{a}}$. Summing up, we have shown that $\forall \mathsf{a}$, $\mathbb{U} \overset{??\mathsf{a}}{\Longrightarrow}$ implies $T[\sigma] \overset{?\mathsf{a}}{\Rightarrow}$: we conclude $\mathbb{U}\Downarrow^{??} \subseteq T[\sigma]\Downarrow^?$.
We are left to prove that $\mathbb{U}\Downarrow^? = \emptyset$ implies $T[\sigma]\Downarrow^? = \emptyset$. We first observe that since $\mathbb{U} = \{U[\sigma]\}$, then $\mathbb{U}\Downarrow^? = \emptyset$ implies $U[\sigma]\Downarrow^? = \emptyset$, which in turn gives:

$$\forall U', \sigma'' \,.\, U[\sigma] \Rightarrow U'[\sigma''] \text{ implies } U'[\sigma'']\Downarrow^? = \emptyset \qquad (26)$$

Let us now examine the possible weak transitions of $T[\sigma]$. From Proposition D.15 we know that $\forall T', \sigma' \,.\, T[\sigma] \Rightarrow T'[\sigma']$ implies $\exists w = !\mathsf{b}_1, \ldots, !\mathsf{b}_n . T \overset{w}{\Rightarrow} T'$. Moreover, since $T \lessdot U$, by Proposition D.12 we have that for all such w, $\exists U' \,.\, U \overset{w}{\Rightarrow} U'$ and $T' \lessdot U'$; and by Proposition D.14, $U \overset{w}{\Rightarrow} U'$ implies $\exists \sigma'' \,.\, U[\sigma] \Rightarrow U'[\sigma'']$. From this, Equation (26), gives $U'[\sigma'']\Downarrow^? = \emptyset$ — i.e., by item *(i)* of Proposition A.4, $U'\Downarrow^? = \emptyset$. But then, $T' \lessdot U'$ is supported by some predictive set $\mathbb{U}'$ such that $\mathbb{U}'\Downarrow^? = \emptyset$; and so, by item *b* of Definition 6.4, we have $T'\Downarrow^? = \emptyset$ — and by item *(i)* of Proposition A.4, $T'[\sigma']\Downarrow^? = \emptyset$. Thus, we conclude $T[\sigma]\Downarrow^? = \emptyset$.

item *c* assume $T[\sigma] \xrightarrow{\tau} T'[\sigma']$. Notice that, by Definition 2.11, $T[\sigma]$ can only generate a τ-transition when T is equivalent to an internal choice with some $!\mathsf{a}$-branch, and $!\mathsf{a}$ is appended to σ: hence, $\sigma' = \sigma\,.!\mathsf{a}$. Then, by Definition 2.10, we have $T \Rightarrow \xrightarrow{!\mathsf{a}} T'$. Since $T \mathrel{\ddot{\lesssim}} U$, by Proposition D.12 we have $\exists U'', U', U''' . U \Rightarrow U'' \xrightarrow{!\mathsf{a}} U' \Rightarrow U'''$ and $T' \mathrel{\ddot{\lesssim}} U'''$. By Proposition A.3, U'' is a single-branch internal choice, and by item *(iii)* of Proposition A.4, $U[\sigma] \xrightarrow{\tau} U'[\sigma\,.!\mathsf{a}] = U'[\sigma']$. Moreover, by Lemma D.3 we have $T' \mathrel{\ddot{\lesssim}} U'$, and by Proposition D.13 we have $T' \preceq_? U'$. Therefore, from the definition of $\mathbb{U}$ above, we conclude $\exists U' . \mathbb{U} \Rightarrow U'[\sigma'] \wedge T'[\sigma'] \mathrel{\ddot{\mathcal{R}}} U'[\sigma']$;

item *d* assume $T[\sigma] \xrightarrow{!\mathsf{a}} T'[\sigma']$. Notice that, by the semantics in Definition 2.11, $T[\sigma]$ can only generate a $!\mathsf{a}$-transition when an output is removed from the head of σ, turning it into σ', without changing T: so, $T' = T$. The same observation holds for $U[\sigma]$. Since $\mathbb{U} = \{U\}$, we have $\exists U' . \mathbb{U} \overset{!\mathsf{a}}{\Rightarrow} U'[\sigma']$ with $U' = U$, from which we obtain $T = T' \mathrel{\ddot{\lesssim}} U'$; moreover, by Proposition D.13 we have $T' \preceq_? U'$: we conclude $T'[\sigma'] \mathrel{\ddot{\mathcal{R}}} U'[\sigma']$;

item *e* assume $T[\sigma] \xrightarrow{?\mathsf{a}} T'[\sigma'] \wedge \mathbb{U} \overset{??\mathsf{a}}{\Longrightarrow}$. Notice that, by the semantics in Definition 2.11, $T[\sigma]$ can only generate a $?\mathsf{a}$-transition when T is an external choice, and σ is unchanged by the reduction — i.e., $\sigma' = \sigma$. Now, since $T \mathrel{\ddot{\lesssim}} U$, by Theorem 6.13, Lemma 6.3 and Lemma 6.2 we have that U is an external choice, too — and by Definition 2.10, we can verify that $U \xrightarrow{?\mathsf{a}}$ iff $U \overset{??\mathsf{a}}{\Longrightarrow}$, and since U is an external choice, by Definition 2.11 we have $U[\sigma] \xrightarrow{?\mathsf{a}}$ implies $U[\sigma] \overset{??\mathsf{a}}{\Longrightarrow}$. Moreover, since $T \mathrel{\ddot{\lesssim}} U$, again by Theorem 6.13, Lemma 6.3 and Lemma 6.2 we also have $\forall U' . T \xrightarrow{?\mathsf{a}} T' \wedge U \xrightarrow{?\mathsf{a}} U'$ implies $T' \mathrel{\ddot{\lesssim}} U'$. Finally, we observe that by Proposition D.13, $\forall U' . T \xrightarrow{?\mathsf{a}} T' \wedge U \xrightarrow{?\mathsf{a}} U'$ implies $T' \preceq_? U'$. Therefore, by definition of $\mathcal{R}$, we conclude $\exists U' . \mathbb{U} \overset{?\mathsf{a}}{\Rightarrow} U'[\sigma'] \wedge T'[\sigma'] \mathrel{\ddot{\mathcal{R}}} U'[\sigma']$.

Hence, $\ddot{\mathcal{R}}$ is an I/O simulation. Thus, $\forall \sigma, T\ (\ddot{\lesssim} \cap \preceq_?)\ U$ implies $T[\sigma] \mathrel{\ddot{\mathcal{R}}} U[\sigma]$, and so $T[\sigma] \mathrel{\ddot{\lesssim}} U[\sigma]$. Then, $T[] \mathrel{\ddot{\lesssim}} U[]$. □

E Proofs for Section 7

Lemma 7.1. *The relations in Table 4 hold in $\mathbb{U}_{aCCS}$.*

Proof. The transition diagrams of the LHS and RHS of each relation are isomorphic, by Definition 2.16. Then, they are also (strongly/weakly) bisimilar: we conclude by Theorem 6.11. □

Remark E.1. *In the following, we will only consider* restricted *I/O simulation relations such that for each pair of elements $(P[\sigma], Q[\rho])$, we have $\sigma = \rho$. This property is assumed for the I/O simulations given by hypothesis, and guaranteed when new I/O simulations are produced as part of a thesis.*

Notation E.2. *We write* $\{P[\sigma], Q[\sigma]\}_{\forall\sigma}$ *for:* $\bigcup_\sigma \{P[\sigma], Q[\sigma]\}$

Lemma E.3. *For all σ, let $P[\sigma] \mathrel{\ddot{\lesssim}} Q[\sigma]$ and $R[\sigma] \mathrel{\ddot{\lesssim}} S[\sigma]$, with $\mathrm{ins}(P) \cap \mathrm{ins}(R) = \mathrm{ins}(Q) \cap \mathrm{ins}(S) = \emptyset$. Then, $P \mid R[\sigma] \mathrel{\ddot{\lesssim}} Q \mid S[\sigma]$.*

Proof. From the hypothesis, let $\ddot{\mathcal{R}}_1, \ddot{\mathcal{R}}_2$ be I/O simulations such that for all σ, $P[\sigma] \mathrel{\ddot{\mathcal{R}}_1} Q[\sigma]$ and $R[\sigma] \mathrel{\ddot{\mathcal{R}}_2} S[\sigma]$. Let:

$$\ddot{\mathcal{R}} = \left\{ (P' \mid R'[\rho]\ ,\ Q' \mid S'[\rho]) \;\middle|\; \begin{array}{l} (P'[\rho'], Q'[\rho']) \in \ddot{\mathcal{R}}_1 \\ \wedge\ (R'[\rho''], S'[\rho'']) \in \ddot{\mathcal{R}}_2 \end{array} \right\}_{\forall\rho}$$

Note that $\ddot{\mathcal{R}}$ is an I/O simulation, where the predictive set for each pair $(P' \mid R'[\rho]\ ,\ Q' \mid S'[\rho])$ is:

$$\{ Q'' \mid S''[\rho] \;\mid\; Q''[\rho] \in \mathbb{Q} \ \wedge\ S''[\rho] \in \mathbb{S} \}$$

where $\mathbb{Q}, \mathbb{S}$ are respectively the predictive sets supporting $P'[\rho'] \mathrel{\ddot{\mathcal{R}}_1} Q'[\rho']$ and $R'[\rho'] \mathrel{\ddot{\mathcal{R}}_2} S'[\rho']$. The key observation is that by Proposition D.12, P''s (resp. R''s) τ and output moves are matched by Q' (resp. S') — and the corresponding pairs of reducts, being contained in $\ddot{\mathcal{R}}_1$ (resp. $\ddot{\mathcal{R}}_2$), are also contained in $\ddot{\mathcal{R}}$. The same matching also holds for input moves shared by P', Q' (resp. R', S') under clause *e* of Definition 6.4. The only problematic case is the following:

- $P'[\rho] \xrightarrow{?\mathsf{a}}$ and $Q'[\rho] \overset{?\mathsf{a}}{\not\Rightarrow}$;
- $R'[\rho] \overset{?\mathsf{a}}{\Rightarrow}$ and $S'[\rho] \overset{??\mathsf{a}}{\Longrightarrow}$;
- hence, $P' \mid R'[\rho] \xrightarrow{?\mathsf{a}}$;
- $Q' \mid S'[\rho] \overset{??\mathsf{a}}{\Longrightarrow}$.

In this case, by clause *e* of Definition 6.4, the $?\mathsf{a}$-transition of $P' \mid R'[\rho]$ (arising from P') would need to be I/O simulated by some weak $?\mathsf{a}$-transition of $Q' \mid S'[\rho]$ (arising from S') — albeit the $?\mathsf{a}$-transition of $P'[\sigma]$ was *not* I/O simulated by $Q'[\sigma]$! However, by hypothesis, we have that the inputs of P', R' and Q', S' cannot overlap; therefore, a persistent input of S' (resp. Q'), which is weakly reachable in R' (resp. P'), cannot be enabled in P' (resp. R'): we conclude that the case is impossible. □

Lemma 7.3. *The rules in Table 5 hold in $\mathbb{U}_{aCCS}$.*

Proof. Recall that all related behaviours in the statement are paired with a generic buffer $[\sigma]$, and that $P \mathrel{\ddot{\lesssim}} Q$ in $\mathbb{U}_{\mathrm{aCCS}}$ means: $\forall\sigma\ .\ P[\sigma] \mathrel{\ddot{\lesssim}} Q[\sigma]$.

For rule (+Ctx), we first prove that, when $P[\sigma] \mathrel{\ddot{\lesssim}} Q[\sigma]$ (for all σ), then $\ell_\tau.P[\sigma] \mathrel{\ddot{\lesssim}} \ell_\tau.Q[\sigma]$ (for all σ). We have three cases:

- $\ell_\tau = \tau$. Let $\sigma \in (\mathsf{A}^!)^*$. Then, $\ell_\tau.P[\sigma] \mathrel{\ddot{\lesssim}} \ell_\tau.Q[\sigma]$ is proved by the I/O simulation $\{(\ell_\tau.P[\sigma], \ell_\tau.Q[\sigma])\} \cup \ddot{\mathcal{R}}$, where $\ddot{\mathcal{R}}$ is an I/O simulation such that $P[\rho] \mathrel{\ddot{\mathcal{R}}} Q[\rho]$ (for all ρ), and $\{\ell_\tau.Q[\sigma]\}$ is the predictive set for the additional pair;
- $\ell_\tau = ?\mathsf{a}$. The proof is similar to the previous case;
- $\ell_\tau = !\mathsf{a}$. Let $\sigma \in (\mathsf{A}^!)^*$. Then, $\ell_\tau.P[\sigma] \mathrel{\ddot{\lesssim}} \ell_\tau.Q[\sigma]$ is proved by the I/O simulation $\{(\ell_\tau.P[\sigma], \ell_\tau.Q[\sigma])\} \cup \ddot{\mathcal{R}}$, where $\ddot{\mathcal{R}}$ is an I/O simulation such that $P[\rho.!\mathsf{a}] \mathrel{\ddot{\mathcal{R}}} Q[\rho.!\mathsf{a}]$ (for all ρ), and $\{\ell_\tau.Q[\sigma]\}$ is the predictive set for the additional pair.

Therefore, when for all σ we have $\forall i \in I\ .\ P_i[\sigma] \mathrel{\ddot{\lesssim}} Q_i[\sigma]$ for some I/O simulation $\ddot{\mathcal{R}}_i$ (by the premise of rule (+Ctx)), we have that $\sum_{i\in I} \ell_{\tau i}.P_i[\sigma] \mathrel{\ddot{\lesssim}} \sum_{i\in I} \ell_{\tau i}.Q_i[\sigma]$ (in the conclusion of rule (+Ctx)) holds by the following I/O simulation:

$$\left\{ \left(\sum_{i\in I} \ell_{\tau i}.P_i[\sigma], \sum_{i\in I} \ell_{\tau i}.Q_i[\sigma] \right) \right\}_{\forall\sigma} \cup \bigcup_{i\in I} \ddot{\mathcal{R}}_i$$

where each additional pair is supported by the predictive set $\{\sum_{i\in I} \ell_{\tau i}.Q_i[\sigma]\}$.

To prove rule $(+\tau)$, we simply observe that, for all σ, when the premise holds, $\{Q[\sigma]\}$ is a predictive set supporting the conclusion.

For rule (|L), by hypothesis we have $Q \equiv \sum_{i\in I} !\mathsf{c}_i.Q_i$, we have two I/O simulations $\ddot{\mathcal{R}}_1, \ddot{\mathcal{R}}_2$ such that $\forall\sigma'\ .\ P'[\sigma'] \mathrel{\ddot{\mathcal{R}}_1} !\mathsf{a}[\sigma']$ and $\forall\sigma'\ .\ P''[\sigma'] \mathrel{\ddot{\mathcal{R}}_2} ?\mathsf{b}[\sigma']$. Then, for all σ, we have that $P' \mid P''[\sigma] \mathrel{\ddot{\lesssim}} !\mathsf{a}.?\mathsf{b} + Q[\sigma]$ is proved by the following I/O simulation:

$\{(P'''[\sigma], !a?b+Q[\sigma])\}_{P'''\in\{P_1|P_2 \;|\; P'\Rightarrow P_1\Rightarrow !a+\cdots \wedge P''\Rightarrow P_2\Rightarrow ?b+\cdots\}} \cup \ddot{\mathcal{R}}_1 \cup \ddot{\mathcal{R}}_2$

Intuitively, P''' ranges over all the intermediate CCS$^-$ terms that $P' \mid P''[\sigma]$ may reach via τ transitions, before reaching respectively an !a-summation and ?b-summation (which must exist — otherwise, one of the two relations in the hypotheses would be false); then, all such intermediate terms are paired with the RHS of the relation we are proving — and the predictive set for all these additional pairs is $\{?b[\sigma.!a]\}$.

For rule (|R), when $P[\sigma] \ddot{\lesssim} ?b.Q[\sigma]$ (for all σ) and $R \equiv \sum_{i\in I} !c_i.R_i$, then we have that $!a.P[\sigma] \ddot{\lesssim} !a + R \mid ?b.Q[\sigma]$ is proved by the following I/O simulation:

$$\{(!a.P[\sigma], !a + R \mid ?b.Q[\sigma])\}_{\forall\sigma} \cup \ddot{\mathcal{R}}$$

where $\ddot{\mathcal{R}}$ is an I/O simulation such that $P[\sigma.!a] \;\ddot{\mathcal{R}}\; ?b.Q[\sigma.!a]$ (which exists by hypothesis). The predictive set for each additional pair is $\{!a + R \mid ?b.Q[\sigma]\}$.

For rule (|LR), when $P \ddot{\lesssim} Q$ and $R \ddot{\lesssim} S$ with $\mathrm{ins}(P) \cap \mathrm{ins}(R) = \mathrm{ins}(Q) \cap \mathrm{ins}(S) = \emptyset$, then $P \mid R[\sigma] \ddot{\lesssim} Q \mid S[\sigma]$ holds by Lemma E.3.

For rule (+Int), where Q is an *output*-guarded choice, let:

$$\ddot{\mathcal{R}} \;=\; \{(P[\sigma], Q[\sigma])\}_{\forall\sigma} \cup \bigcup_{i\in I} \ddot{\mathcal{R}}_i$$

where $\ddot{\mathcal{R}}_i$ is an I/O simulation such that $\forall\sigma \,.\, P_i[\sigma] \ddot{\lesssim} Q_i[\sigma]$ (which exists by the rule premises), and for a given σ, $\mathbb{Q} = \{Q[\sigma]\}$ is the predictive set for the additional pair $(P[\sigma], Q[\sigma])$. We can verify that $\ddot{\mathcal{R}}$ is an I/O simulation, which ignores the new branch $!b.Q'$, since its initial τ-move (for enqueuing !b) is not matched in the LHS.

For rule (+Ext), Q is an *input*-guarded choice. Note that whenever $j = i$, since (by premises) there exists an I/O simulation such that $P_j[\sigma] \ddot{\lesssim} Q_i[\sigma]$, by rule (+Ctx) there is also an I/O simulation such that $?a_j.P_j[\sigma] \ddot{\lesssim} ?a_i.Q_i[\sigma]$: therefore, let $\ddot{\mathcal{R}}_1$ be the union of all such I/O simulations. Finally, let us consider, for each $k \in K$, an I/O simulation such that $R_k \ddot{\lesssim} Q$, and let $\ddot{\mathcal{R}}_2$ be the union of these I/O simulations. Then, the relation in the conclusion is proved by the following I/O simulation:

$$\ddot{\mathcal{R}} \;=\; \left\{ \left(\sum_{j\in J} (?a_j.P_j) + \sum_{k\in K} \tau.P_k[\sigma],\; Q[\sigma] \right) \right\}_{\forall\sigma} \cup \ddot{\mathcal{R}}_1 \cup \ddot{\mathcal{R}}_2$$

where the predictive set for each additional pair is $\{Q[\sigma]\}$. □

Proposition E.4. *Let P be a CCS$^-$ term with some occurrence of X that is output-guarded, and* not *input-guarded. Let Q be a sequential CCS$^-$ term, with $Q[\sigma]\Downarrow^{??} \neq \emptyset$ (for any σ). Then, $P[Q/X] \not\ddot{\lesssim} Q$.*

Proof. We prove the statement by contradiction, assuming $P[Q/X] \ddot{\lesssim} Q$. This, in particular, implies $P[Q/X][\epsilon] \ddot{\lesssim} Q[\epsilon]$. Let $Q_0 = Q$. From the hypotheses on P, there exists a sequence of outputs $!a_1, \ldots, !a_n$ and some P' such that $P[Q_0/X][\epsilon] \overset{!a_1}{\Longrightarrow} \cdots \overset{!a_n}{\Longrightarrow} Q_0 \mid P'[\epsilon]$. Since $P[Q_0/X][\epsilon] \ddot{\lesssim} Q_0[\epsilon]$, by clauses *c* and *d* of Definition 6.4, $Q_0[\epsilon]$ must simulate such transitions: hence, reminding Remark E.1, there exists Q_1 such that $Q_0[\epsilon] \overset{!a_1}{\Longrightarrow} \cdots \overset{!a_n}{\Longrightarrow} Q_1[\epsilon]$ and $Q_0 \mid P'[\epsilon] \ddot{\lesssim} Q_1[\epsilon]$. But then, the same sequence of transitions can be fired by the LHS of the latter relation, and the RHS must simulate it: hence, there exists Q_2 such that $Q_1[\epsilon] \overset{!a_1}{\Longrightarrow} \cdots \overset{!a_n}{\Longrightarrow} Q_2[\epsilon]$ and $Q_1 \mid P'[\epsilon] \ddot{\lesssim} Q_2[\epsilon]$. By iterating this reasoning, we can see that there exists an infinite sequence of processes $(Q_m)_{m\geq 0}$ such that $Q_m[\epsilon] \overset{!a_1}{\Longrightarrow} \cdots \overset{!a_n}{\Longrightarrow} Q_{m+1}[\epsilon]$ and $Q_m \mid P'[\epsilon] \ddot{\lesssim} Q_{m+1}[\epsilon]$: this implies that $Q = Q_0$ is recursive, and for some $k \geq 0$, the outputs being fired by $(Q_m)_{m\geq k}$ occur under some recursive subterms of Q. Notice that the persistent inputs of Q (that exist by the hypothesis $Q[\sigma]\Downarrow^{??} \neq \emptyset$) are never fired throughout the transitions above — i.e., $\forall m \geq 0$, $Q_m[\epsilon]\Downarrow^{??} \supseteq Q[\epsilon]\Downarrow^{??} \neq \emptyset$. But then, from the sequentiality hypothesis, Q must contain a recursive subterm $\mu_Y Q'$, which in turn must contain a subterm $\sum_{i\in I} Q'_i$ where, for some $j, j' \in I$, Q'_j contains some inputs, and $Q'_{j'}$ has an occurrence of Y that is *not* input-guarded. Hence, we must conclude that Q does *not* respect clause *b* of Definition 7.6, i.e., Q is *not* a CCS$^-$ term: contradiction. □

Lemma 7.7. *Let $\mu_X P$ and Q be sequential CCS$^-$ terms. If $P[Q/X] \ddot{\lesssim} Q$, then $\mu_X P \ddot{\lesssim} Q$.*

Proof. If $X \notin \mathrm{fv}(P)$, the thesis trivially holds. Otherwise, when $X \in \mathrm{fv}(P)$, we need to produce an I/O simulation containing the pairs $\{(\mu_X P[\sigma], Q[\sigma])\}_{\forall\sigma}$. Before proceeding, we introduce some technical machinery:

1. we write $\underline{Q}$ (i.e., Q underlined) to "tag" the occurrences of term Q arising in $P[Q/X]$ due to variable substitution: e.g., if $P = ?a + !b.X$ and $Q = ?a$, we have $P[Q/X] = ?a + !b.\underline{?a}$;
2. $P[R/\underline{Q}]$ is the term substitution which replaces each occurrence of $\underline{Q}$ in P with the tagged term $\underline{R}$: for instance, if $P = ?a + !b.\underline{?a}$ we have $P[R/\underline{?a}] = ?a + !b.\underline{R}$ (notice that the first, untagged occurrence of ?a is unchanged);
3. we naturally extend the semantics of async CCS so that such syntactic tags are preserved along transitions, without altering the labels. E.g., taking P and Q from item 1. above, we have $P[Q/X][] \xrightarrow{\tau} \underline{?a}[!b] \xrightarrow{?a} \mathbf{0}[!b]$ (note that the tag is discarded when the occurrence of $\underline{Q} = \underline{?a.\mathbf{0}}$ is reduced).

Now, let $\ddot{\mathcal{R}} = \ddot{\mathcal{R}}' \cup \ddot{\mathcal{R}}''$, where:

$$\ddot{\mathcal{R}}' = \left\{ (P'[\mu_X P/\underline{Q}][\sigma'],\, Q'[\sigma']) \;\middle|\; \begin{array}{r} \exists \sigma_0, \sigma' \,.\, P[Q/X][\sigma_0] \to^* P'[\sigma'] \\ \wedge\; P'[\sigma'] \ddot{\lesssim} Q'[\sigma'] \end{array} \right\}$$

$$\ddot{\mathcal{R}}'' = \left\{ (P'[\sigma'],\, Q'[\sigma']) \;\middle|\; \begin{array}{l} \exists P'', Q'' \,.\; P'' \neq \underline{\mu_X P} \text{ and} \\ P'[\sigma'] \ddot{\lesssim} P''[\sigma'] \;\ddot{\mathcal{R}}'\; Q''[\sigma'] \ddot{\lesssim} Q'[\sigma'] \end{array} \right\}$$

Intuitively, $\ddot{\mathcal{R}}$ contains:

- from $\ddot{\mathcal{R}}'$: for all σ', all pairs $(P'[\sigma'], Q'[\sigma'])$, given by the following procedure:
  1. take $P''[\sigma']$, $Q'[\sigma']$ such that $P[Q/X][\sigma] \to^* P''[\sigma']$ and $P''[\sigma'] \ddot{\lesssim} Q'[\sigma']$
  2. obtain P' by replacing all occurrences of $\underline{Q}$ in P'' with $\underline{\mu_X P}$ (note that Q' is taken as-is);
- from $\ddot{\mathcal{R}}''$: the closure of $\ddot{\mathcal{R}}'$ w.r.t. $\ddot{\lesssim}$, for all pairs in $\ddot{\mathcal{R}}'$ such that the LHS is *not* yielded by an instance of $\underline{Q}$ replaced by $\mu_X P$.

We prove that $\ddot{\mathcal{R}}$ is an I/O simulation. Let:

$$F(\mathcal{X}) = \left\{ \begin{pmatrix} P''[\sigma], \\ Q''[\sigma] \end{pmatrix} \;\middle|\; \begin{array}{l} \exists \mathbb{Q}''. \\ Q''[\sigma] \Rrightarrow \mathbb{Q}'' \\ \text{and} \end{array} \left\{ \begin{array}{ll} a. & P''[\sigma]\Downarrow^{!} = \emptyset \text{ implies} \\ & \mathbb{Q}''\Downarrow^{!} = \emptyset; \\ b. & \mathbb{Q}''\Downarrow^{??} \subseteq P''[\sigma]\Downarrow^{?} \;\wedge \\ & (\mathbb{Q}''\Downarrow^{?} = \emptyset \\ & \quad \text{implies } P''[\sigma]\Downarrow^{?} = \emptyset); \\ c. & P''[\sigma] \xrightarrow{\tau} P'''[\sigma] \text{ implies} \\ & \exists Q'''[\sigma] \,.\; \mathbb{Q}'' \Rrightarrow Q'''[\sigma] \\ & \quad \wedge\; (P'''[\sigma], Q'''[\sigma]) \in \mathcal{X}; \\ d. & P''[\sigma] \xrightarrow{!a} P'''[\sigma'] \text{ implies} \\ & \exists Q'''[\sigma'] \,.\; \mathbb{Q}'' \overset{!a}{\Rightarrow} Q'''[\sigma'] \\ & \quad \wedge (P'''[\sigma'], Q'''[\sigma']) \in \mathcal{X}; \\ e. & P''[\sigma] \xrightarrow{?a} P'''[\sigma] \wedge \mathbb{Q}'' \overset{??a}{\Longrightarrow} \\ & \exists Q'''[\sigma] \,.\; \mathbb{Q}'' \overset{?a}{\Rightarrow} Q'''[\sigma] \\ & \quad \wedge\; (P'''[\sigma], Q'''[\sigma]) \in \mathcal{X}. \end{array} \right\}$$

By the coinduction proof principle, we have to show that $\ddot{\mathcal{R}} \subseteq F(\ddot{\mathcal{R}})$. When $(P''[\sigma], Q''[\sigma]) \in \ddot{\mathcal{R}}$, we have two cases:

A. if $(P''[\sigma], Q''[\sigma]) \in \ddot{\mathcal{R}}'$, then $P'' = P'[\mu_X P/\underline{Q}]$, for some $\sigma_0, P'[\sigma]$ such that $P[Q/X][\sigma_0] \to^* P'[\sigma] \ddot{\lesssim} Q''[\sigma]$. Regarding such P', we have two possibilities:

(i) $P' \neq \underline{Q}$. By clause *a* of Definition 7.6, X is always *guarded* in some subterm $\ell.X$ of P, and thus, all occurrences of $\underline{Q}$ in P' are guarded by some input or output ℓ, and the same holds for all occurrences of $\underline{\mu_X P}$ in $P'[\mu_X P/\underline{Q}] = P''$. Now, let $\mathbb{Q}''$ be a predictive set supporting $P'[\sigma] \ddot{\lesssim} Q''[\sigma]$: we show that $\mathbb{Q}''$ is also a predictive set for the pair $(P'[\mu_X P/\underline{Q}][\sigma], Q''[\sigma]) = (P''[\sigma], Q''[\sigma])$. In detail:

- clause *a*. Assume $P''[\sigma]\Downarrow^{!} = P'[\mu_X P[\sigma]/\underline{Q}]\Downarrow^{!} = \emptyset$. It suffices to show that this implies $P'[\sigma]\Downarrow^{!} = \emptyset$, and therefore $\mathbb{Q}''\Downarrow^{!} \neq \emptyset$ (since $P'[\sigma] \ddot{\lesssim} Q''[\sigma]$ with predictive set $\mathbb{Q}''$). Note that $P''[\sigma]\Downarrow^{!} = \emptyset$ implies $\sigma = \epsilon$. By contradiction, assume $P'[\epsilon]\Downarrow^{!} \neq \emptyset$: this means that the weakly reachable outputs of $P'[\epsilon]$ must occur in $\underline{Q}$, but are removed when replacing it with $\underline{\mu_X P}$ in P''. For such outputs to be part of $P'[\epsilon]\Downarrow^{!}$, we have two possibilities:
  - some $\underline{Q}$ in P' is unguarded. Then, since $P' \neq \underline{Q}$, we must have $P' \equiv \underline{Q} \mid P'_0$ (for some P'_0), which implies that $P[Q/X][\sigma_0]$ can reduce to a parallel term, i.e., $\mu_X P$ is *not* sequential: contradiction;
  - some $\underline{Q}$ in P' is guarded by τs only. Then, some $\underline{X}$ is P is *not* guarded in some subterm $\ell.X$, thus violating clause *a* of Definition 7.6. Hence, $\mu_X P$ is *not* a CCS$^-$ term: contradiction;
- clause *b*, first part. By Proposition D.5 we have $\mathbb{Q}''\Downarrow^{??} \subseteq P'[\sigma]\Downarrow^{??}$. By contradiction, let $\mathbb{Q}''\Downarrow^{??} \not\subseteq P''[\sigma]\Downarrow^{?}$, and therefore, $\mathbb{Q}''\Downarrow^{??} \not\subseteq P'[\mu_X P/\underline{Q}][\sigma]\Downarrow^{?}$. It means that some persistent input of $\mathbb{Q}''$ that is also persistent in $P'[\sigma]$ is not weakly reachable in $P'[\mu_X P/\underline{Q}][\sigma]$. Since P' is sequential, it follows that such an input occurs in $\underline{Q}$ as persistent (i.e., it belongs to $Q[\sigma]\Downarrow^{??}$), but it is not weakly reachable in P. Hence, we have $\emptyset \neq \underline{Q}[\sigma]\Downarrow^{??} \not\subseteq P[\sigma]\Downarrow^{?}$. Then, considering $P[Q/X][\sigma]$ and the sequentiality of P, we have two possibilities:
  - all X in P are input-guarded. Then, we have $P[Q/X][\sigma]\Downarrow^{?} = P[\sigma]\Downarrow^{?}$. Now, note that for any set $\mathbb{Q}$ such that $Q[\sigma] \Rrightarrow \mathbb{Q}$, we also have $\underline{Q}[\sigma]\Downarrow^{??} \subseteq \mathbb{Q}\Downarrow^{??} \not\subseteq P[\sigma]\Downarrow^{?} = P[Q/X][\sigma]\Downarrow^{?}$. Thus, we obtain the contradiction $P[Q/X][\sigma] \not\ddot{\lesssim} Q[\sigma]$, because no predictive set could satisfy the first part of clause *b* of Definition 6.4;
  - some X in P is output-guarded and *not* input-guarded. Since $Q[\sigma]\Downarrow^{??} \neq \emptyset$, by Proposition E.4 we get the contradiction $P[Q/X][\sigma] \not\ddot{\lesssim} Q[\sigma]$;
- clause *b*, second part. Assume $\mathbb{Q}''\Downarrow^{?} = \emptyset$: this implies $P'[\sigma]\Downarrow^{?} = \emptyset$. By contradiction, assume $P''[\sigma]\Downarrow^{?} = P'[\mu_X P/\underline{Q}][\sigma]\Downarrow^{?} \neq \emptyset$. This implies that $\underline{Q}[\sigma]\Downarrow^{?} = \emptyset$, while $\underline{\mu_X P}[\sigma]\Downarrow^{?} = P[\sigma]\Downarrow^{?} \neq \emptyset$. This means $P[Q/X][\sigma]\Downarrow^{?} \neq \emptyset$. Moreover, since $\underline{Q}[\sigma]\Downarrow^{?} = \emptyset$, for any set $\mathbb{Q}$ such that $Q[\sigma] \Rrightarrow \mathbb{Q}$ we have $\mathbb{Q}\Downarrow^{?} = \emptyset$. But then, we obtain the contradiction $P[Q/X][\sigma] \not\ddot{\lesssim} Q[\sigma]$, since no predictive set could satisfy the second part of clause *b* of Definition 6.4.

For the remaining clauses of F, reminding that $P'[\sigma] \ddot{\lesssim} Q''[\sigma]$, we take all pairs $P'''[\sigma'] \ddot{\lesssim} Q'''[\sigma']$ that satisfy Definition 6.4 after a transition of $P'[\sigma]$, apply the substitution $P'''[Q/X][\sigma']$, and show that the resulting pair satisfies clauses *c*–*e* of F. More precisely, take all $\ell_\tau, P'''[\sigma']$ such that $P'[\sigma] \xrightarrow{\ell_\tau} P'''[\sigma']$, and all $Q'''[\sigma']$ such that:

$$P'[\sigma] \xrightarrow{\ell_\tau} P'''[\sigma'] \quad \text{and} \quad Q''[\sigma] \Rrightarrow \mathbb{Q}'' \overset{\ell_\tau}{\Longrightarrow} Q'''[\sigma'] \quad \text{and} \quad P'''[\sigma'] \ddot{\lesssim} Q'''[\sigma']$$

Observe that, since

$$P[Q/X][\sigma_0] \to^* P'[\sigma] \xrightarrow{\ell_\tau} P'''[\sigma'] \text{ and } P'''[\sigma'] \ddot{\lesssim} Q'''[\sigma']$$

$\ddot{\mathcal{R}}'$ contains a pair $(P'''[\mu_X P/\underline{Q}][\sigma'], Q'''[\sigma'])$, and thus:

$$P''[\sigma] = P'[\mu_X P/\underline{Q}][\sigma] \xrightarrow{\ell_\tau} P'''[\mu_X P/\underline{Q}][\sigma'] \quad \text{and} \quad (P'''[\mu_X P/\underline{Q}][\sigma'], Q'''[\sigma']) \in \ddot{\mathcal{R}}' \subseteq \ddot{\mathcal{R}}$$

We can verify that such pairs $(P'''[\mu_X P/\underline{Q}][\sigma'], Q'''[\sigma'])$ satisfy the existentials in clauses *c*–*e* of F (depending on whether ℓ_τ is a τ, an output, or an input). Therefore, we conclude $(P''[\sigma], Q''[\sigma]) \in F(\ddot{\mathcal{R}})$;

(ii) $P' = \underline{Q}$. Then, we have $P'' = \underline{\mu_X P}$ and $P'[\sigma] = \underline{Q}[\sigma] \ddot{\lesssim} Q''[\sigma]$. We observe:

1. since by hypothesis $P[Q/X] \ddot{\lesssim} Q$, then from the pair $(P[Q/X][\sigma], Q[\sigma]) \in \ddot{\lesssim}$, the set $\ddot{\mathcal{R}}'$ contains:

$$(P[Q/X][\mu_X P/\underline{Q}][\sigma], Q[\sigma]) = (P[\mu_X P/X][\sigma], Q[\sigma]) \in \ddot{\mathcal{R}}'$$

2. since $P''[\sigma]$ and $P[\mu_X P/X][\sigma]$ are bisimilar, then by Theorem 6.11 we have $P''[\sigma] \ddot{\lesssim} P[\mu_X P/X][\sigma]$.

Summing up:

$$P''[\sigma] = \underline{\mu_X P}[\sigma] \;\ddot{\lesssim}\; P[\mu_X P/X][\sigma] \;\ddot{\mathcal{R}}'\; \underline{Q}[\sigma] \;\ddot{\lesssim}\; Q''[\sigma]$$

Hence, noticing that $P[\mu_X P/X] \neq \underline{\mu_X P}$ (since X is guarded in P), we also have $(P''[\sigma], Q''[\sigma]) \in \ddot{\mathcal{R}}''$: this is studied in case *B.* below;

B. if $(P''[\sigma], Q''[\sigma]) \in \ddot{\mathcal{R}}''$, then there exist $P' \neq \underline{\mu_X P}$ and Q' such that:

- $P''[\sigma] \ddot{\lesssim} P'[\sigma']$, for some predictive set $\mathbb{P}'$;
- $P'[\sigma'] \;\ddot{\mathcal{R}}'\; Q'[\sigma']$, for some predictive set $\mathbb{Q}'$, determined as per case *A.(i)* above (since $P' \neq \underline{\mu_X P}$);
- $Q'[\sigma'] \ddot{\lesssim} Q''[\sigma']$, for some predictive set $\mathbb{Q}''$.

Now, applying the technique in the proof of Theorem 6.10 (transitivity, page 32), we can:

- use $\mathbb{P}'$ and $\mathbb{Q}'$ to build a predictive set $\mathbb{Q}'''$ such that the pair $(P''[\sigma'], Q'[\sigma'])$ satisfies clauses *a*–*e* of F (note that the substitutions $[\mu_X P/\underline{Q}]$ in P'' are handled similarly to case *A.(i)* above);

– use $\mathbb{Q}'''$ and $\mathbb{Q}''$ to build a predictive set $\mathbb{Q}^*$ such that the pair $(P''[\sigma'], Q''[\sigma'])$ satisfies clauses a–e of F.

E.g., for clause e of F, we have:

$$P''[\sigma] \xrightarrow{?\mathsf{a}} P'''[\sigma] \quad \text{and} \quad \mathbb{Q}^* \overset{??\mathsf{a}}{\Longrightarrow} \quad \text{implies} \quad \exists P_0''', Q_0''', Q'''.$$

$$\mathbb{P}' \overset{?\mathsf{a}}{\Longrightarrow} P_0''' \text{ and } \mathbb{Q}' \overset{?\mathsf{a}}{\Longrightarrow} Q_0''' \text{ and } \mathbb{Q}^* \overset{?\mathsf{a}}{\Longrightarrow} Q'''[\sigma] \quad \text{and}$$

$$P'''[\sigma] \mathrel{\ddot{\leqslant}} P_0'''[\sigma] \mathrel{\ddot{\mathfrak{R}}_1} Q_0'''[\sigma] \mathrel{\ddot{\leqslant}} Q'''[\sigma]$$

and if $P''' \neq \underline{\mu_X P}$, then $(P'''[\sigma], Q'''[\sigma]) \in \ddot{\mathfrak{R}}'' \subseteq \ddot{\mathfrak{R}}$; otherwise, $P''' = \underline{\mu_X P}$ is bisimilar to $P[\mu_X P/X]$, and we have:

$$P'''[\sigma] \mathrel{\ddot{\leqslant}} P[\mu_X P/X][\sigma] \mathrel{\ddot{\mathfrak{R}}'} Q_0'''[\sigma] \mathrel{\ddot{\leqslant}} Q'''[\sigma]$$

and thus, since $P[\mu_X P/X] \neq \underline{\mu_X P}$, we get $(P'''[\sigma], Q'''[\sigma]) \in \ddot{\mathfrak{R}}'' \subseteq \ddot{\mathfrak{R}}$. Hence, we conclude $(P''[\sigma], Q''[\sigma]) \in F(\ddot{\mathfrak{R}})$.

Then, $\ddot{\mathfrak{R}}$ is an I/O simulation; further, since $\ddot{\leqslant}$ is the *largest* I/O simulation, we have $\ddot{\mathfrak{R}} \subseteq \ddot{\leqslant}$. Finally, we are left to prove that $\forall\sigma.\, \mu_X P[\sigma] \mathrel{\ddot{\leqslant}} Q[\sigma]$: we simply observe that $\forall\sigma.(\mu_X P[\sigma], Q[\sigma]) \in \ddot{\mathfrak{R}}' \subseteq \ddot{\mathfrak{R}} \subseteq \ddot{\leqslant}$. □