Precise Subtyping for Asynchronous Multiparty Sessions

Silvia Ghilezan, Jovanka Pantović, Ivan Prokić
Univerzitet u Novom Sadu, Serbia

Alceste Scalas
Aston University, Birmingham, UK

Nobuko Yoshida
Imperial College London, UK

Abstract

Session subtyping is a cornerstone of refinement of communicating processes: a process implementing a session type (i.e., a communication protocol) T can be safely used whenever a process implementing one of its supertypes T' is expected, in any context, without introducing deadlocks nor other communication errors. This paper presents the first formalisation of the *precise* subtyping relation for *asynchronous multiparty* sessions: we show that the relation is *sound* (i.e., guarantees safe process replacement, as outlined above) and also *complete*: any extension of the relation is unsound. Previous work studies precise subtyping for *binary* sessions (with two participants), or multiparty sessions (with any number of participants) and *synchronous* interaction. Here, we cover *multiparty* sessions with *asynchronous* interaction, where messages are transmitted via FIFO queues (as in the TCP/IP protocol). In this setting, the subtyping relation becomes highly complex: under some conditions, participants can permute the order of their inputs and outputs, by sending some messages earlier, or receiving some later, without causing errors; the precise subtyping relation must capture *all* such valid permutations, and consequently, its formalisation and proofs become challenging. Our key discovery is a methdology to *decompose* session types into *single input/output session trees*, and then express the subtyping relation as a composition of refinement relations between such trees.

1 Introduction

Modern software systems are routinely designed and developed as ensembles of concurrent and distributed components, interacting via message-passing according to pre-determined *communication protocols*. In this setting, a key challenge is ensuring that each component abides by the desired protocol, thus avoiding run-time failures due to, e.g., communication errors and deadlocks. One of the most successful approaches to this problem are *session types* [35, 22, 23, 24], which allow to formalise *multiparty protocols as types*, and verify whether processes correctly implement them. Beyond their theoretical developments, session types have been implemented in many practical programming languages [1, 20].

One of the key features of session types is the notion of *subtyping*, which can be interpreted as *protocol refinement*: given two types/protocols T and T', if T' is a subtype (or refinement) of T, then a process that implements T' can be used whenever a process implementing T is needed. Subtyping allows to safely replace typed software components, and makes session types-based verification more flexible. For this reason, several papers have tackled the problem of finding the largest, ***precise*** subtyping relations [11, 21]. A subtyping relation $\leqslant$ is precise when it is both *sound* and *complete*: ***soundness*** means that, if we have a context C expecting some process P of type T, then $T' \leqslant T$ implies that any process P' of type T' can be placed into C without causing "bad behaviours" (e.g., communication errors or deadlocks); ***completeness*** means that $\leqslant$ cannot be extended without becoming unsound. More precisely: if $T' \not\leqslant T$, then we can find a process P' of type T', and a context C expecting a process of type T, such that if we place P' in C, it will cause "bad behaviours."

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The problem of finding the precise subtyping relation $\leqslant$ is greatly complicated when it combines *asynchrony* and *multiparty* interactions, i.e., the communication channels' capability to buffer messages after they are sent, and before they are received — as in the TCP/IP protocol. For example, suppose we are using an asynchronous message transport where the sending is non-blocking, and message order is preserved; consider a scenario where a participant r waits for the outcome of a distributed computation from p, and notifies q on whether to continue the calculation:

$$\mathsf{r} \triangleleft P_{\mathsf{r}} \;\mid\; \mathsf{p} \triangleleft P_{\mathsf{p}} \;\mid\; \mathsf{q} \triangleleft P_{\mathsf{q}} \;\mid\; \cdots$$

$$\text{where}\quad P_{\mathsf{r}} \;=\; \sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathsf{if}\ (x > 42)\ \mathsf{then}\ \mathsf{q}!\mathit{cont}\langle x\rangle.\mathbf{0}\ \mathsf{else}\ \mathsf{q}!\mathit{stop}\langle\rangle.\mathbf{0} \\ \mathsf{p}?\mathit{error}(\mathit{fatal}).\mathsf{if}\ (\neg\mathit{fatal})\ \mathsf{then}\ \mathsf{q}!\mathit{cont}\langle 43\rangle.\mathbf{0}\ \mathsf{else}\ \mathsf{q}!\mathit{stop}\langle\rangle.\mathbf{0} \end{Bmatrix}$$

Above, $\mathsf{r} \triangleleft P_{\mathsf{r}}$ denotes a process P_{r} executed by participant r, $\mathsf{p}?\ell(x)$ is an input of message ℓ with payload value x from participant p, and $\mathsf{q}!\ell\langle 5\rangle$ is an output of message ℓ with payload 5 to participant q. In the example, r waits to receive either *success* or *error* from p. In case of *success*, r checks whether the message payload x is greater than 42, and tells q to either *cont*inue (forwarding the payload x) or *stop*, then terminates ($\mathbf{0}$); in case of *error*, r checks whether the error is non-fatal, and then tells q to either *cont*inue (with a constant value 43), or *stop*. Note that r is blocked until a message is sent by p, and correspondingly, q is waiting for r, who is waiting for p. Yet, depending on the application, r might be *locally optimised*, by replacing P_{r} above with the following process:

$$P'_{\mathsf{r}} \;=\; \mathsf{if}\ (\ldots)\ \mathsf{then}\ \mathsf{q}!\mathit{cont}\langle 43\rangle.\sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathbf{0} \\ \mathsf{p}?\mathit{error}(y).\mathbf{0} \end{Bmatrix}\ \mathsf{else}\ \mathsf{q}!\mathit{stop}\langle\rangle.\sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathbf{0} \\ \mathsf{p}?\mathit{error}(y).\mathbf{0} \end{Bmatrix}$$

Process P'_{r} internally decides (with an omitted condition "…") whether to tell q to *cont*inue with a constant value 43, or *stop*. Then, r receives the *success*/*error* message from p, and does nothing with it. As a result, q can start its computation immediately, without waiting for p. Intuitively, this optimisation that swaps the order of inputs and outputs should not introduce any deadlock nor communication error in the system; consequently, it may seem that a subtyping relation should allow it: i.e., if T' is the type of P'_{r}, and T is the type of P_{r}, we should have $T' \leqslant T$ (we illustrate such types later on, in Example 3.5) — hence, the type system should let P'_{r} be used in place of P_{r}. Due to practical needs, similar program optimisations have been implemented for various programming languages, e.g., [32, 31, 25, 10]. Yet, this optimisation is *not* allowed by *synchronous* multiparty session subtyping [21]: in fact, under synchrony, there are cases where allowing $T' \leqslant T$ would introduce deadlocks. However, most real-world distributed and concurrent systems use *asynchronous multiparty* communication: is the optimisation above *always* safe in these settings, and should the subtyping allow for it? If we prove that such an optimisation (and others) are indeed sound, it would be possible to check them locally, at the type-level, for each participant.

Formulating the largest, *precise* subtyping relation is technically challenging, as it must consider type-level asynchrony, multiple participants, choices, and recursion. We solve this problem by introducing a novel methodology, a ***session decomposition***, from branching and selection types into single input / output trees, and then express the subtyping relation as a composition of refinement relations between such trees. For our development, we adopt a recent advancement of the multiparty session type theory [34]: this way, we achieve not only the largest, precise subtyping relation, but also a simpler formulation, and more general results, than [11, 29, 21], by typing a larger set of concurrent and distributed processes.

Outline. Section 2 formalises the asynchronous multiparty session calculus. Section 3 presents our asynchronous multiparty session subtyping relation, with its decomposition technique. Section 4 introduces the typing system, and Section 5 proves the preciseness of our subtyping. Related work is in Section 6. *Due to space limits, proofs are in appendices.*

$\mathcal{M}$	$::=$		**Sessions**
		$\mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h$	*individual participant*
	$\mid$	$\mathcal{M} \mid \mathcal{M}$	*parallel*
	$\mid$	error	*error*
h	$::=$		**Message queues**
		$\varnothing$	*empty queue*
	$\mid$	$(\mathsf{q}, \ell(\mathsf{v}))$	*message*
	$\mid$	$h \cdot h$	*concatenation*

P, Q	$::=$		**Processes**
		$\sum_{i \in I} \mathsf{p}?\ell_i(x_i).P_i$	*ext. choice*
	$\mid$	$\mathsf{p}!\ell\langle \mathsf{e} \rangle.P$	*output*
	$\mid$	$\mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q$	*conditional*
	$\mid$	X	*variable*
	$\mid$	$\mu X.P$	*recursion*
	$\mid$	$\mathbf{0}$	*inaction*

Table 1 Syntax of sessions, processes and queues. We assume that in recursive processes, recursion variables are guarded by external choices and/or outputs.

[R-SEND]	$\mathsf{p} \triangleleft \mathsf{q}!\ell\langle \mathsf{e}\rangle.P \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathcal{M} \longrightarrow \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \cdot (\mathsf{q}, \ell(\mathsf{v})) \mid \mathcal{M}$	$(\mathsf{e} \downarrow \mathsf{v})$
[R-RCV]	$\mathsf{p} \triangleleft \sum_{i \in I} \mathsf{q}?\ell_i(x_i).P_i \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft Q \mid \mathsf{q} \triangleleft (\mathsf{p}, \ell_k(\mathsf{v})) \cdot h \mid \mathcal{M}$ $\longrightarrow \mathsf{p} \triangleleft P_k\{\mathsf{v}/x_k\} \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft Q \mid \mathsf{q} \triangleleft h \mid \mathcal{M}$	$(k \in I)$
[R-COND-T]	$\mathsf{p} \triangleleft \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid \mathsf{p} \triangleleft h \mid \mathcal{M} \longrightarrow \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h \mid \mathcal{M}$	$(\mathsf{e} \downarrow \mathsf{true})$
[R-STRUCT]	$\mathcal{M}_1 \equiv \mathcal{M}_1', \quad \mathcal{M}_1' \longrightarrow \mathcal{M}_2', \quad \mathcal{M}_2' \equiv \mathcal{M}_2 \quad \Longrightarrow \quad \mathcal{M}_1 \longrightarrow \mathcal{M}_2$	
[ERR-MISM]	$\mathsf{p} \triangleleft \sum_{i \in I} \mathsf{q}?\ell_i(x_i).P_i \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft Q \mid \mathsf{q} \triangleleft (\mathsf{p}, \ell(\mathsf{v})) \cdot h \mid \mathcal{M} \longrightarrow \mathsf{error}$	$(\forall i \in I. \ell_i \neq \ell)$
[ERR-OPHN]	$\mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft Q \mid \mathsf{q} \triangleleft (\mathsf{p}, \ell(\mathsf{v})) \cdot h \mid \mathcal{M} \longrightarrow \mathsf{error}$	$(\mathsf{q}? \notin \mathsf{act}(P))$
[ERR-STRV]	$\mathsf{p} \triangleleft \sum_{i \in I} \mathsf{q}?\ell_i(x_i).P_i \mid \mathsf{q} \triangleleft Q \mid \mathsf{q} \triangleleft h_{\mathsf{q}} \mid \mathcal{M} \longrightarrow \mathsf{error}$	$(\mathsf{p}! \notin \mathsf{act}(Q), h_{\mathsf{q}} \not\equiv (\mathsf{p}, -(-)) \cdot h_{\mathsf{q}}')$
[ERR-EVAL]	$\mathsf{p} \triangleleft \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid \mathsf{p} \triangleleft h \mid \mathcal{M} \longrightarrow \mathsf{error}$	$(\nexists \mathsf{v} : \mathsf{e} \downarrow \mathsf{v})$
[ERR-EVAL2]	$\mathsf{p} \triangleleft \mathsf{q}!\ell\langle \mathsf{e}\rangle.P \mid \mathsf{p} \triangleleft h \mid \mathcal{M} \longrightarrow \mathsf{error}$	$(\nexists \mathsf{v} : \mathsf{e} \downarrow \mathsf{v})$
[ERR-DLOCK]	$\prod_{j \in J}(\mathsf{p}_j \triangleleft \sum_{i_j \in I_j} \mathsf{q}_j?\ell_{i_j}(x_{i_j}).P_{i_j} \mid \mathsf{p}_j \triangleleft h_{\mathsf{p}_j}) \longrightarrow \mathsf{error}$	$(\forall j \in J : h_{\mathsf{q}_j} \not\equiv (\mathsf{p}_j, -(-)) \cdot h_{\mathsf{q}_j}')$

Table 2 Reduction relation on sessions ([R-COND-F] is similar to [R-COND-T], and is omitted).

2 Asynchronous Multiparty Session Calculus

This section presents the syntax and operational semantics of an asynchronous multiparty session calculus. Our formulation extends the synchronous calculus in [21].

Syntax. An *asynchronous multiparty session* (ranged over by $\mathcal{M}, \mathcal{M}', \ldots$), defined in Table 1, is a parallel composition of individual participants (ranged over by $\mathsf{p}, \mathsf{q}, \ldots$) associated with their own *process* P and *queue* h (notation: $\mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h$), also defined in Table 1. In the processes syntax, the *external choice* $\sum_{i \in I} \mathsf{p}?\ell_i(x_i).P_i$ denotes the input from participant p of a message with label ℓ_i carrying value x_i, for any $i \in I$; instead, $\mathsf{p}!\ell\langle \mathsf{e}\rangle.P$ denotes the output towards participant p of a message with label ℓ carrying the value returned by expression e. The conditional $\mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q$ is standard. The term $\mathsf{p} \triangleleft h$ states that h is the message queue of participant p; if a message $(\mathsf{q}, \ell(\mathsf{v}))$ is in the queue of participant p, it means that p has sent $\ell(\mathsf{v})$ to q. Messages are consumed by their recipients on a FIFO (first in, first out) basis. The rest of the syntax is standard [21]. The set $\mathsf{act}(P)$ contains the *input and output actions* of P, and is defined as: $\mathsf{act}(\mathbf{0}) = \emptyset$; $\mathsf{act}(\mathsf{p}!\ell\langle \mathsf{e}\rangle.P) = \{\mathsf{p}!\} \cup \mathsf{act}(P)$; $\mathsf{act}(\sum_{i \in I} \mathsf{p}?\ell_i(x_i).P_i) = \{\mathsf{p}?\} \cup \bigcup_{i \in I} \mathsf{act}(P_i)$ (other cases are homomorphic).

Reductions and errors. The operational semantics is defined in Table 2. By [R-SEND], a participant p sends $\ell\langle \mathsf{e}\rangle$ to a participant q, enqueuing the message $(\mathsf{q}, \ell(\mathsf{v}))$, where "$\mathsf{e} \downarrow \mathsf{v}$" means that expression e evaluates to v. Rule [R-RCV] lets participant p receive a message from q: if one of the input labels ℓ_k matches a queued message $(\mathsf{p}, \ell_k(\mathsf{v}))$ previously sent by q (for some $k \in I$), the message is dequeued, and the continuation P_i proceeds with value v substituting x_k. The rules for conditionals are standard. Rule [R-STRUCT] defines the reduction modulo a standard *structural congruence* $\equiv$, with rules to unfold recursions ($\mu X.P \equiv P\{\mu X.P/X\}$), erase an inact participant ($\mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing \mid \mathcal{M} \equiv \mathcal{M}$) and rearrange queued messages. *(Full definition in Appendix A.2)*

$$h_1 \cdot (\mathsf{q}_1, \ell_1(\mathsf{v}_1)) \cdot (\mathsf{q}_2, \ell_2(\mathsf{v}_2)) \cdot h_2 \;\equiv\; h_1 \cdot (\mathsf{q}_2, \ell_2(\mathsf{v}_2)) \cdot (\mathsf{q}_1, \ell_1(\mathsf{v}_1)) \cdot h_2 \qquad (\text{if } \mathsf{q}_1 \neq \mathsf{q}_2)$$

Table 2 also formalises *error reductions*, modelling the following scenarios: in [ERR-MISM], a process tries to read a queued message with an unsupported label; in [ERR-ORPH], there is a buffered message from q to p, but p's process does not contain any input from q, hence the message is orphan; in [ERR-STRV], p is waiting for a message from q, but no such message is queued, and q's process does not contain any output for p, hence p will starve; in [ERR-EVAL] and [ERR-EVAL2], an expression like "succ(true)" cannot reduce to any value; and in [ERR-DLOCK], the session cannot reduce further, but there is some participant with a non-terminated process or a non-empty queue. *(Example A.2 shows the reduction rules in action.)*

3 Asynchronous Multiparty Session Subtyping

This section introduces our asynchronous session subtyping relation, in two phases:

1. we introduce a *refinement relation* for *session trees* (defined below) having only singleton choices in all branchings and selections, called *single-input-single-output (SISO) trees*;
2. then, we consider trees that have only singleton choices in branchings (called *single-input (SI) trees*), or in selections (*single-output (SO) trees*), and we define the session subtyping over all session types by considering their decomposition into SI, SO, and SISO trees.

This two-phases approach is crucial to capture all input/output reorderings needed by the *precise* subtyping relation, while taming the technical complexity of its formulation.

We begin with the standard definition of (local) session types.

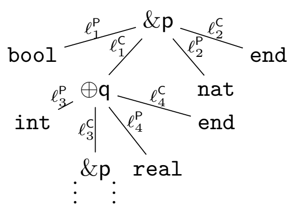
▶ **Definition 3.1.** *The* sorts S *and* session types $\mathbb{T}$ *defined as follows:*

$$\mathsf{S} ::= \mathtt{nat} \mid \mathtt{int} \mid \mathtt{bool} \qquad \mathbb{T} ::= \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathbb{T}_i \;\mid\; \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{T}_i \;\mid\; \mathtt{end} \;\mid\; \mu \mathbf{t}.\mathbb{T} \;\mid\; \mathbf{t}$$

where for all $i, j \in I$: $i \neq j \Rightarrow \ell_i \neq \ell_j$. We assume guarded recursion. We define $\equiv$ as the least congruence such that $\mu\mathbf{t}.\mathbb{T} \equiv \mathbb{T}\{^{\mu\mathbf{t}.\mathbb{T}}/_{\mathbf{t}}\}$. We define $\mathtt{pt}(\mathbb{T})$ as the set of participants in $\mathbb{T}$.

Sorts are the types of values (naturals, integers, booleans). A session type $\mathbb{T}$ describes the behaviour of a participant in a multiparty session. The *branching type* (or *external choice*) $\&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathbb{T}_i$ denotes waiting for a message from participant p, where (for some $i \in I$) the message has label ℓ_i and carries a payload value of sort S_i; then, the interaction continues by following $\mathbb{T}_i$. The *selection type* (or *internal choice*) $\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{T}_i$ denotes an output toward participant p of a message with label ℓ_i and payload of sort S_i, after which the interaction follows $\mathbb{T}_i$ (for some $i \in I$). Type $\mu\mathbf{t}.\mathbb{T}$ provides recursion, binding the *recursion variable* $\mathbf{t}$ in $\mathbb{T}$; the guarded recursion assumption means: in $\mu\mathbf{t}.\mathbb{T}$, we have $\mathbb{T} \neq \mathbf{t}'$ for any $\mathbf{t}'$ (which ensures *contractiveness*). Type end denotes that the participant has concluded its interactions. For brevity, we often omit branch/selection symbols in case of singleton inputs/outputs, unnecessary parentheses, and ends.

Session trees and their refinement



To define our subtyping relation, we use (finite or infinite) *session trees* with the standard formulation of [21, Appendix A.2], based on Pierce [33]. The diagram on the right depicts a session tree: its internal nodes represent branching (&p) or selection (⊕q) from/to a participant; leaf nodes are either payload sorts or end; edge annotations are either ℓ^{P}

or ℓ^{C}, respectively linking an internal node to the payload or continuation for message ℓ. A type $\mathbb{T}$ yields a tree $\mathcal{T}(\mathbb{T})$: the diagram above shows the (infinite) tree of $\mu\mathbf{t}.\&\{\mathsf{p}?\ell_1(\mathtt{bool}).\bigoplus\{\mathsf{q}!\ell_3(\mathtt{int}).\mathbf{t},\mathsf{q}!\ell_4(\mathtt{real}).\mathtt{end},\}\ ,\ \mathsf{p}?\ell_2(\mathtt{nat}).\mathtt{end}\}$. Notably, the trees of a recursive type $\mu\mathbf{t}.\mathbb{T}$ and its unfolding $\mathbb{T}\{\mu\mathbf{t}.\mathbb{T}/\mathbf{t}\}$ coincide. We will write T to denote a session tree, and we will represent it using the *coinductive* syntax:

$$\mathsf{T} \ ::= \ \mathtt{end} \ \mid \ \&_{i\in I}\,\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{T}_i \ \mid \ \bigoplus_{i\in I}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{T}_i$$

SISO trees. A SISO tree W only has singleton choices (i.e., one pair of payload+continuation edges) in all its branchings and selections. We represent W with the *coinductive* syntax:

$$\mathsf{W} \ ::= \ \mathtt{end} \ \mid \ \mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \ \mid \ \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}$$

We will write $\mathbb{W}$ to denote a SISO session type (i.e., having singleton choice in both branching and selection), such that $\mathcal{T}(\mathbb{W})$ yields a SISO tree. We coinductively define the set $\mathtt{act}(\mathsf{W})$ over a tree W as the set of participant names together with actions ? (input) or ! (output), as: $\mathtt{act}(\mathtt{end}) = \emptyset$; $\mathtt{act}(\mathsf{p}?\ell(\mathsf{S}).\mathsf{W}') = \{\mathsf{p}?\} \cup \{\mathtt{act}(\mathsf{W}')\}$; and $\mathtt{act}(\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}') = \{\mathsf{p}!\} \cup \{\mathtt{act}(\mathsf{W}')\}$. We also define $\mathtt{act}(\mathbb{W}) = \mathtt{act}(\mathcal{T}(\mathbb{W}))$.

SISO trees refinement. As discussed in Section 1, the asynchronous subtyping should allow to reorder the input/output actions of a session type: this is crucial to achieve the largest, most flexible, and *precise* subtyping. Such reorderings should allow anticipating a selection toward participant p before a finite number of branchings, and *also* before other selections which are *not* toward participant p. Dually, the subtyping should allow anticipating a branching from participant p before a finite number of branchings which are *not* from p.

To characterise such reordering of actions we define two kinds of finite sequences of inputs and outputs: $\mathcal{A}^{(\mathsf{p})}$ contains only inputs from participants distinct from p, while the sequence $\mathcal{B}^{(\mathsf{p})}$ contains inputs from any participant and/or outputs to participants distinct from p:

$$\mathcal{A}^{(\mathsf{p})} ::= \mathsf{q}?\ell(\mathsf{S}) \ \mid \ \mathsf{q}?\ell(\mathsf{S}).\mathcal{A}^{(\mathsf{p})} \qquad \mathcal{B}^{(\mathsf{p})} ::= \mathsf{r}?\ell(\mathsf{S}) \ \mid \ \mathsf{q}!\ell(\mathsf{S}) \ \mid \ \mathsf{r}?\ell(\mathsf{S}).\mathcal{B}^{(\mathsf{p})} \ \mid \ \mathsf{q}!\ell(\mathsf{S}).\mathcal{B}^{(\mathsf{p})} \quad (\mathsf{q} \neq \mathsf{p})$$

▶ **Definition 3.2.** *The* SISO tree refinement relation $\lesssim$ *is coinductively defined as:*

$$\frac{\mathsf{S}' \leq: \mathsf{S} \quad \mathsf{W} \lesssim \mathsf{W}'}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \lesssim \mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\ \text{[REF-IN]} \qquad \frac{\mathsf{S}' \leq: \mathsf{S} \quad \mathsf{W} \lesssim^{(2)} \mathcal{A}^{(\mathsf{p})}.\mathsf{W}' \quad \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{A}^{(\mathsf{p})}.\mathsf{W}')}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\ \text{[REF-}\mathcal{A}\text{]}$$

$$\frac{\mathsf{S} \leq: \mathsf{S}' \quad \mathsf{W} \lesssim \mathsf{W}'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \lesssim \mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\ \text{[REF-OUT]} \qquad \frac{\mathsf{S} \leq: \mathsf{S}' \quad \mathsf{W} \lesssim^{(2)} \mathcal{B}^{(\mathsf{p})}.\mathsf{W}' \quad \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{B}^{(\mathsf{p})}.\mathsf{W}')}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \lesssim \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\ \text{[REF-}\mathcal{B}\text{]} \qquad \frac{\text{[REF-END]}}{\mathtt{end} \lesssim \mathtt{end}}$$

Rule [REF-IN] relates inputs from a same participant with equal message labels; the subtyping between carried sorts must be contravariant, and the continuations must be related. Rule [REF-OUT] relates outputs from a same participant with equal message labels; the subtyping between carried sorts must be covariant, and the continuations must be related. Rule [REF-$\mathcal{A}$] allows anticipating an input from participant p before a finite number of inputs from any other participant; the two payload sorts and the rest of the trees satisfy the same conditions as in rule [REF-IN], while "$\mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{A}^{(\mathsf{p})}.\mathsf{W}')$" ensures soundness: without such a condition we could "forget" some inputs, and derive, e.g., $\mathcal{T}(\mu\mathbf{t}.\mathsf{p}?\ell(\mathsf{S}).\mathbf{t}) \lesssim \mathcal{T}(\mathsf{q}?\ell_1(\mathsf{S}_1).\mu\mathbf{t}.\mathsf{p}?\ell(\mathsf{S}).\mathbf{t})$ by taking $\mathcal{A}^{(\mathsf{p})} = \mathsf{q}?\ell_1(\mathsf{S}_1)$. Rule [REF-$\mathcal{B}$] enables anticipating an output to participant p before a finite number of inputs from any participant and/or outputs from any other participant; the payload types and the rest of the two trees are related similarly to rule [REF-OUT], while "$\mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{B}^{(\mathsf{p})}.\mathsf{W}')$" ensures that inputs or outputs are not "forgotten" (as rule [REF-$\mathcal{A}$].) The refinement $\lesssim$ is reflexive and transitive (Lemma B.6). *See also Example B.10.*

▶ Remark 3.3 (Prefixes vs. n-hole contexts). The binary asynchronous subtyping in [12, 11] uses the n-hole branching type context $\mathcal{A} ::= [\ ]^n \mid \&_{i\in I}?\ell_i(\mathsf{S}_i).\mathcal{A}_i$, which complicates the rules and reasoning (see Fig.2, Fig.3 in [11]). Our $\mathcal{A}^{(\mathsf{p})}$ and $\mathcal{B}^{(\mathsf{p})}$ have a similar purpose, but they are simpler (just sequences of inputs or outputs), and cater for multiple participants.

SO trees and SI trees. We need two more kinds of session trees: *single-output (SO) trees*, denoted U, have only singleton choices in their selections; dually, *single-input (SI) trees*, denoted V, have only singleton branchings. We represent them with a *coinductive* syntax:

$$\mathsf{U} ::= \mathtt{end} \mid \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i \mid \mathsf{p}!\ell(\mathsf{S}).\mathsf{U} \qquad \mathsf{V} ::= \mathtt{end} \mid \mathsf{p}?\ell(\mathsf{S}).\mathsf{V} \mid \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_i$$

We will write $\mathbb{U}$ (resp. $\mathbb{V}$) to denote a SO (resp. SI) session type, i.e., with only singleton selections (resp. branchings), such that $\mathcal{T}(\mathbb{U})$ (resp. $\mathcal{T}(\mathbb{V})$) yields a SO (resp. SI) tree.

We decompose session trees into their SO/SI subtrees, with the functions $[\![\cdot]\!]_{\mathsf{SO}}$ / $[\![\cdot]\!]_{\mathsf{SI}}$:

$$[\![\mathtt{end}]\!]_{\mathsf{SO}} = \{\mathtt{end}\} \qquad \begin{array}{rcl} [\![\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{T}_i]\!]_{\mathsf{SO}} &=& \{\mathsf{p}!\ell(\mathsf{S}_i).\mathsf{U} : \mathsf{U} \in [\![\mathsf{T}_i]\!]_{\mathsf{SO}}, i \in I\} \\ [\![\&_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{T}_i]\!]_{\mathsf{SO}} &=& \{\&_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U} : \mathsf{U} \in [\![\mathsf{T}_i]\!]_{\mathsf{SO}}\} \end{array}$$

$$[\![\mathtt{end}]\!]_{\mathsf{SI}} = \{\mathtt{end}\} \qquad \begin{array}{rcl} [\![\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{T}_i]\!]_{\mathsf{SI}} &=& \{\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V} : \mathsf{V} \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}\} \\ [\![\&_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{T}_i]\!]_{\mathsf{SI}} &=& \{\mathsf{p}?\ell(\mathsf{S}_i).\mathsf{V} : \mathsf{V} \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}, i \in I\} \end{array}$$

Hence, when $[\![\cdot]\!]_{\mathsf{SO}}$ is applied to a session tree T, it gives the set of all SO trees obtained by taking only a single choice from each selection in T (i.e., we take a single continuation edge and the corresponding payload edge starting in a selection node.) The function $[\![\cdot]\!]_{\mathsf{SI}}$ is dual. Notice that for any SO tree U, and SI tree V, both $[\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $[\![\mathsf{V}]\!]_{\mathsf{SO}}$ yield SISO trees.

Asynchronous session subtyping

We can now define our asynchronous session subtyping relation: it relates two session types by decomposing them into their SI, SO, and SISO trees, and checking their refinements.

▶ **Definition 3.4.** *The* asynchronous subtyping relation $\leqslant$ *over session trees is defined as:*

$$\frac{\forall \mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}' \in [\![\mathsf{T}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \qquad \mathsf{W} \lesssim \mathsf{W}'}{\mathsf{T} \leqslant \mathsf{T}'}$$

The subtyping relation for session types is defined as $\mathbb{T} \leqslant \mathbb{T}'$ *iff* $\mathcal{T}(\mathbb{T}) \leqslant \mathcal{T}(\mathbb{T}')$.

Definition 3.4 says that a session tree T is subtype of T' if, for all SO decompositions of T and all SI decompositions of T', there are paths (i.e., SISO decompositions) related by $\lesssim$. The asynchronous subtyping relation $\leqslant$ over trees is reflexive and transitive *(Lemma B.9)*.

We now illustrate the relation with two examples: we reprise the scenario in the introduction, and we discuss a case from [3, 4]. *More examples are available in Appendix B.1*

▶ **Example 3.5.** Consider the opening example in Section 1. The following types describe the interactions of P'_{r} and P_{r}, respectively:

$$\mathbb{T}' = \bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\, \&\, \mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases} \\ \mathit{stop}(\mathtt{unit}).\, \&\, \mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases} \end{cases} \quad \mathbb{T} = \&\, \mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases} \\ \mathit{error}(\mathtt{bool}).\bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases} \end{cases}$$

In order to derive $\mathbb{T}' \leqslant \mathbb{T}$, we first show the two SO trees such that $[\![\mathcal{T}(\mathsf{T}')]\!]_{\mathsf{SO}} = \{\mathsf{U}_1, \mathsf{U}_2\}$:

$$\mathsf{U}_1 = \mathsf{q}!\mathit{cont}(\mathtt{int}).\,\&\,\mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases} \quad \mathsf{U}_2 = \mathsf{q}!\mathit{stop}(\mathtt{unit}).\,\&\,\mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases}$$

and these are the two SI trees such that $[\![\mathcal{T}(\mathsf{T})]\!]_{\mathsf{SI}} = \{\mathsf{V}_1, \mathsf{V}_2\}$:

$$\mathsf{V}_1 = \mathsf{p}?\mathit{success}(\mathtt{int}).\bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases} \quad \mathsf{V}_2 = \mathsf{p}?\mathit{error}(\mathtt{bool}).\bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases}$$

Therefore, for all $i, j \in \{1, 2\}$, we can find $\mathsf{W}' \in [\![\mathsf{U}_i]\!]_{\mathsf{SI}}$ and $\mathsf{W} \in [\![\mathsf{V}_j]\!]_{\mathsf{SO}}$ such that $\mathsf{W}' \lesssim \mathsf{W}$ can be derived using [REF-$\mathcal{B}$]. Hence, $\mathbb{T}' \leqslant \mathbb{T}$ holds.

$$\frac{}{\vdash \varnothing : \epsilon}\,[\text{T-NUL}] \qquad \frac{\vdash \mathsf{v} : \mathsf{S}}{\vdash (\mathsf{q}, \ell(\mathsf{v})) : \mathsf{q}!\ell(\mathsf{S})}\,[\text{T-ELM}] \qquad \frac{\vdash h_i : \sigma_i \ \ (i=1,2)}{\vdash h_1 \cdot h_2 : \sigma_1 \cdot \sigma_2}\,[\text{T-QUEUE}] \qquad \frac{}{\Theta \vdash \mathbf{0} : \mathsf{end}}\,[\text{T-}\mathbf{0}]$$

$$\frac{}{\Theta, X : \mathbb{T} \vdash X : \mathbb{T}}\,[\text{T-VAR}] \qquad \frac{\Theta, X : \mathbb{T} \vdash P : \mathbb{T}}{\Theta \vdash \mu X.P : \mathbb{T}}\,[\text{T-REC}] \qquad \frac{\Theta \vdash \mathsf{e} : \mathsf{S} \quad \Theta \vdash P : \mathbb{T}}{\Theta \vdash \mathsf{q}!\ell\langle\mathsf{e}\rangle.P : \mathsf{q}!\ell(\mathsf{S}).\mathbb{T}}\,[\text{T-OUT}]$$

$$\frac{\forall i \in I \quad \Theta, x_i : \mathsf{S}_i \vdash P_i : \mathbb{T}_i}{\Theta \vdash \sum_{i\in I} \mathsf{q}?\ell_i(x_i).P_i : \&_{i\in I}\, \mathsf{q}?\ell_i(\mathsf{S}_i).\mathbb{T}_i}\,[\text{T-EXT}] \qquad \frac{\Theta \vdash \mathsf{e} : \mathsf{bool} \quad \Theta \vdash P_i : \mathbb{T} \ \ (i=1,2)}{\Theta \vdash \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 : \mathbb{T}}\,[\text{T-COND}]$$

$$\frac{\Theta \vdash P : \mathbb{T} \quad \mathbb{T} \leqslant \mathbb{T}'}{\Theta \vdash P : \mathbb{T}'}\,[\text{T-SUB}] \qquad \frac{\Gamma = \{\mathsf{p}_i\colon (\sigma_i, \mathbb{T}_i) \mid i \in I\} \quad \forall i \in I \quad \vdash P_i : \mathbb{T}_i \quad \vdash h_i : \sigma_i}{\Gamma \vdash \prod_{i\in I}(\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i)}\,[\text{T-SESS}]$$

Table 3 Typing rules for queues, processes and a session.

▶ **Example 3.6.** For the following types $\mathbb{T}, \mathbb{T}'$, we have $\mathbb{T} \leqslant \mathbb{T}'$ *(proof: Appendix B.1)*. Notably, this relation cannot be proved by the binary asynchronous subtyping algorithm in [3, 4].

$$\mathbb{T} = \mu\mathbf{t}_1.\,\&\,\mathsf{p}?\begin{cases}\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_1\\ \ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2\end{cases} \qquad \mathbb{T}' = \mu\mathbf{t}_1.\,\&\,\mathsf{p}?\begin{cases}\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_1\\ \ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2\end{cases}$$

4 Typing System and Type Safety

Our multiparty session typing system blends [21] with [34, Section 7]: like the latter, we type multiparty sessions without need for global types, thus simplifying our formalism and generalising our results. The key differences are our asynchronous subtyping (Def. 3.4) and our choice of *typing environment liveness* (Def. 4.4): their interplay yields our preciseness results. Before proceeding, we need *queue types* for message queues, extending Def. 3.1:

$$\sigma \ ::= \ \epsilon \ \mid \ \mathsf{p}!\ell(\mathsf{S}) \ \mid \ \sigma \cdot \sigma$$

Type ϵ denotes an empty queue; $\mathsf{p}!\ell(\mathsf{S})$ denotes a queued message with recipient p, label ℓ, and payload of sort S; they are concatenated as $\sigma \cdot \sigma'$.

▶ **Definition 4.1** (Type system). *The type system uses 4 judgments: (1) for expressions,* $\Theta \vdash \mathsf{e} : \mathsf{S}$*; (2) for queues,* $\vdash h : \sigma$*; (3) for processes,* $\Theta \vdash P : \mathbb{T}$*; and (4) for sessions,* $\Gamma \vdash \mathcal{M}$*. They are inductively defined in Table 3, with typing environments defined as:*

$$\Gamma \ ::= \ \emptyset \ \mid \ \Gamma, \mathsf{p} : (\sigma, \mathbb{T}) \qquad \Theta \ ::= \ \emptyset \ \mid \ \Theta, X : \mathbb{T} \ \mid \ \Theta, x : \mathsf{S}$$

The judgment for expressions is standard *(see Appendix C, Table 7)*. The judgment for queues means that queue h has queue type σ. The judgment for processes states that, given the types of the variables in Θ, process P behaves as prescribed by $\mathbb{T}$. The judgment for sessions states that multiple participants and queues behave as prescribed by Γ, which maps each participant p to the pairing of a queue type (for p's message queue) and a session type (for p's process). If $\Theta = \emptyset$ we write $\vdash \mathsf{e} : \mathsf{S}$ and $\vdash P : \mathbb{T}$.

We now comment the rules for processes and sessions (other rules are self-explanatory). Rule [T-$\mathbf{0}$] types a terminated process. Rule [T-VAR] types a process variable with the assumption in the environment. By [T-REC], a recursive process is typed with $\mathbb{T}$ if the process variable body have the same type $\mathbb{T}$. By [T-OUT], an output process is typed with a singleton selection type, if the message being sent is of the correct sort, and the process continuation has the continuation type. By [T-EXT], a process external choice is typed as a branching type with matching participant p and labels ℓ_i (for all $i \in I$); in the rule premise, each continuation process P_i must be typed with the corresponding continuation type $\mathbb{T}_i$, assuming the bound variable x_i is of sort S_i (for all $i \in I$). By [T-COND], a conditional has type $\mathbb{T}$ if its expression has sort `bool`, and its "then" and "else" branches have type $\mathbb{T}$. Rule [T-SUB] is the *subsumption rule*: it states that a process of type $\mathbb{T}$ is also typed by any supertype of $\mathbb{T}$ (and since $\leqslant$ relates types up-to unfolding, this rule makes our type system *equi-recursive* [33]). By [T-SESS], a session $\mathcal{M}$ is typed by environment Γ if all the participants in $\mathcal{M}$ have processes and queues typed by the session and queue types pairs in Γ.

▶ **Example 4.2.** The processes P_{r} and its optimised version P'_{r} in Section 1 are typable using the rules in Def. 4.1, and the types $\mathbb{T}, \mathbb{T}'$ in Example 3.5. And since $\mathbb{T}' \leqslant \mathbb{T}$, by rule [T-SUB] our type system allows to use P'_{r} whenever a process of type $\mathbb{T}$ (such as P_{r}) is expected.

Typing environment evolutions. To formulate the soundness result for our type system, we define typing environment reductions. The reductions rely on a standard structural congruence relation over queue types, that is the least congruence satisfying:

$$\sigma \cdot \epsilon \;\equiv\; \epsilon \cdot \sigma \;\equiv\; \sigma \qquad \sigma_1 \cdot (\sigma_2 \cdot \sigma_3) \;\equiv\; (\sigma_1 \cdot \sigma_2) \cdot \sigma_3$$
$$\sigma \cdot \mathsf{p}_1!\ell_1(\mathsf{S}_1) \cdot \mathsf{p}_2!\ell_2(\mathsf{S}_2) \cdot \sigma' \;\equiv\; \sigma \cdot \mathsf{p}_2!\ell_2(\mathsf{S}_2) \cdot \mathsf{p}_1!\ell_1(\mathsf{S}_1) \cdot \sigma' \quad (\text{if } \mathsf{p}_1 \neq \mathsf{p}_2)$$

For pairs of queue/session types, we define structural congruence as $(\sigma_1, \mathbb{T}_1) \equiv (\sigma_2, \mathbb{T}_2)$ iff $\sigma_1 \equiv \sigma_2$ and $\mathbb{T}_1 \equiv \mathbb{T}_2$, and subtyping as $(\sigma_1, \mathbb{T}_1) \leqslant (\sigma_2, \mathbb{T}_2)$ iff $\sigma_1 \equiv \sigma_2$ and $\mathbb{T}_1 \leqslant \mathbb{T}_2$. By extension, we write $\Gamma \equiv \Gamma'$ (resp. $\Gamma \leqslant \Gamma'$) iff $\forall \mathsf{p} \in dom(\Gamma \cap \Gamma')$: $\Gamma(\mathsf{p}) \equiv \Gamma'(\mathsf{p})$ (resp. $\Gamma(\mathsf{p}) \leqslant \Gamma'(\mathsf{p})$) and $\forall \mathsf{p} \in dom(\Gamma \setminus \Gamma'), \mathsf{q} \in dom(\Gamma' \setminus \Gamma)$: $\Gamma(\mathsf{p}) \equiv (\epsilon, \mathsf{end}) \equiv \Gamma'(\mathsf{q})$.

▶ **Definition 4.3** (Typing environment reduction). *The reduction* $\xrightarrow{\alpha}$ *of asynchronous session typing environments* Γ *is inductively defined as follows:*

$$\text{[E-RCV]} \quad \mathsf{p}\colon (\mathsf{q}!\ell_k(\mathsf{S}'_k)\cdot\sigma, \mathbb{T}_{\mathsf{p}}), \mathsf{q}\colon (\sigma_{\mathsf{q}}, \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathbb{T}_i), \Gamma \xrightarrow{\mathsf{q}:\mathsf{p}?\ell_k} \mathsf{p}\colon (\sigma, \mathbb{T}_{\mathsf{p}}), \mathsf{q}\colon (\sigma_{\mathsf{q}}, \mathbb{T}_k), \Gamma \quad (k \in I, \mathsf{S}'_k \leq\colon \mathsf{S}_k)$$
$$\text{[E-SEND]} \quad \mathsf{p}\colon (\sigma, \textstyle\bigoplus_{i\in I} \mathsf{q}!\ell_i(\mathsf{S}_i).\mathbb{T}_i), \Gamma \xrightarrow{\mathsf{p}:\mathsf{q}!\ell_k} \mathsf{p}\colon (\sigma\cdot\mathsf{q}!\ell_k(\mathsf{S}_k)\cdot\epsilon, \mathbb{T}_k), \Gamma \quad (k \in I)$$
$$\text{[E-STRUCT]} \quad \Gamma \equiv \Gamma_1 \xrightarrow{\alpha} \Gamma'_1 \equiv \Gamma' \implies \Gamma \xrightarrow{\alpha} \Gamma'$$

We often write $\Gamma \longrightarrow \Gamma'$ *instead of* $\Gamma \xrightarrow{\alpha} \Gamma'$, *when* α *is not important.*

Rule [E-RCV] says that an environment can take a reduction step if participant p has a message toward q with label ℓ_k and payload sort S'_k at the head of its queue, while q's type is a branching from p including label ℓ_k and a corresponding sort S_k being supertype of S'_k; the environment evolves with a reduction labelled $\mathsf{q}{:}\mathsf{p}?\ell_k$, by consuming q's queued message and activating the continuation $\mathbb{T}_k$ in q's type. In rule [E-SEND] the environment evolves by letting participant p (having a selection type) send a message toward q; the reduction is labelled $\mathsf{p}{:}\mathsf{q}!\ell_k$ (with ℓ_k being a selection label), and it places the message at the end of p's queue. Rule [E-STRUCT] closes the reduction under structural congruence.

Similarly to [34], we define a behavioral property of typing environments (and their evolutions) called *liveness*[1]: we will use it as a precondition for typing, to ensure that typed processes cannot reduce to any error in Table 2.

▶ **Definition 4.4** (Live typing environment). *A* typing environment path *is a finite or infinite sequence of typing environments* $(\Gamma_i)_{i\in I}$, *where* $I = \{0, 1, 2, \ldots\}$ *is a set of consecutive natural numbers, and,* $\forall i \in I$, $\Gamma_i \longrightarrow \Gamma_{i+1}$. *We say that a path* $(\Gamma_i)_{i\in I}$ *is* fair *iff,* $\forall i \in I$:

(F1) *whenever* $\Gamma_i \xrightarrow{\mathsf{p}:\mathsf{q}!\ell} \Gamma'$, *then* $\exists k, \ell'$ *such that* $I \ni k{+}1 > i$, *and* $\Gamma_k \xrightarrow{\mathsf{p}:\mathsf{q}!\ell'} \Gamma_{k+1}$
(F2) *whenever* $\Gamma_i \xrightarrow{\mathsf{p}:\mathsf{q}?\ell} \Gamma'$, *then* $\exists k$ *such that* $I \ni k{+}1 > i$, *and* $\Gamma_k \xrightarrow{\mathsf{p}:\mathsf{q}?\ell} \Gamma_{k+1}$

We say that a path $(\Gamma_i)_{i\in I}$ *is* live *iff,* $\forall i \in I$:

[1] Notably, our definition of liveness is stronger than the "liveness" in [34, Fig. 5], and is closer to "liveness^{+}" therein: we adopt it because a weaker "liveness" would not allow to achieve Theorem 5.15 later on.

(L1) *if* $\Gamma_i(\mathsf{p}) \equiv (\mathsf{q}!\ell(\mathsf{S})\cdot\sigma\,,\,\mathbb{T})$, *then* $\exists k$*:* $I \ni k{+}1 > i$ *and* $\Gamma_k \xrightarrow{\mathsf{q}:\mathsf{p}?\ell} \Gamma_{k+1}$

(L2) *if* $\Gamma_i(\mathsf{p}) \equiv \left(\sigma_{\mathsf{p}}\,,\,\&_{j\in J}\,\mathsf{q}?\ell_j(\mathsf{S}_j).\mathbb{T}_j\right)$, *then* $\exists k, \ell'$*:* $I \ni k{+}1 > i$ *and* $\Gamma_k \xrightarrow{\mathsf{p}:\mathsf{q}?\ell'} \Gamma_{k+1}$

We say that a typing environment Γ *is* live *iff all fair paths beginning with* Γ *are live.*

By Def. 4.4, a path is a (possibly infinite) sequence of reductions of a typing environment. Intuitively, a fair path represents a "fair scheduling:" along its reductions, every pending internal choice eventually enqueues a message (**F1**), and every pending message reception is eventually performed (**F2**). A path is live if, along its reductions, every queued message is eventually consumed (**L1**), and every waiting external choice eventually consumes a queued message (**L2**). A typing environment is live if, under "fair scheduling," it yields a live path. Liveness is preserved by environment reductions and subtyping, i.e., if Γ is live and $\Gamma \longrightarrow \Gamma'$, then Γ' is live *(Proposition C.2)*; and if Γ is live and $\Gamma' \leqslant \Gamma$, then Γ' is live *(Lemma C.13)*.

Our Theorem 4.5 below says: if session $\mathcal{M}$ is typed by a live Γ, and $\mathcal{M} \longrightarrow \mathcal{M}'$, then $\mathcal{M}$ might anticipate some inputs/outputs prescribed by Γ, as allowed by subtyping $\leqslant$. Hence, $\mathcal{M}$ reduces by following some $\Gamma'' \leqslant \Gamma$, which evolves to Γ' that types $\mathcal{M}'$. *(Proofs: Appendix C.3)*.

▶ **Theorem 4.5** (Subject Reduction). *Assume* $\Gamma \vdash \mathcal{M}$ *with* Γ *live. If* $\mathcal{M} \longrightarrow \mathcal{M}'$, *then there are live type environments* Γ', Γ'' *such that* $\Gamma'' \leqslant \Gamma$, $\Gamma'' \longrightarrow^* \Gamma'$ *and* $\Gamma' \vdash \mathcal{M}'$.

▶ **Corollary 4.6** (Type Safety and Progress). *Let* $\Gamma \vdash \mathcal{M}$ *with* Γ *live. Then,* $\mathcal{M} \longrightarrow^* \mathcal{M}'$ *implies* $\mathcal{M}' \neq \mathsf{error}$*; also, either* $\mathcal{M}' \equiv \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$, *or* $\exists \mathcal{M}''$ *such that* $\mathcal{M}' \longrightarrow \mathcal{M}'' \neq \mathsf{error}$.

Notably, since our errors (Table 2) include orphan messages, deadlocks, and starvation, Corollary 4.6 implies *session liveness*: a typed session will never deadlock, all its external choices will be eventually activated, all its queued messages will be eventually consumed.

5 Preciseness of Asynchronous Multiparty Session Subtyping

We now present our main results. Our asynchronous multiparty subtyping $\leqslant$ (Def. 3.4) is precise, with two meanings: it is the largest sound subtyping for our type system (Theorem 5.14), and it is the largest liveness-preserving refinement (Theorem 5.15).

A subtyping relation $\leqslant$ is *sound* if it satisfies the Liskov and Wing's substitution principle [28]: if $\mathbb{T} \leqslant \mathbb{T}'$, then a process of type $\mathbb{T}'$ engaged in a well-typed session may be safely replaced with a process of type $\mathbb{T}$. If $\leqslant$ is the largest relation with such a property, then $\leqslant$ is *precise*; in this case, the implication in the soundness statement is also true when reversed — and the reversed implication is called *completeness.* This is formalised in Def. 5.1 below (where we use the contrapositive of the completeness implication).

▶ **Definition 5.1** (Preciseness). *Let* $\preccurlyeq$ *be a preorder over session types. We say that* $\preccurlyeq$ *is:*

(1) *a* sound subtyping *if* $\mathbb{T} \preccurlyeq \mathbb{T}'$ *implies that, for all* $\mathsf{r} \notin \mathtt{pt}(\mathbb{T}'), \mathcal{M}, P$*, the following holds:*
 a. *if* $\left(\forall Q : \vdash Q : \mathbb{T}' \implies \Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M} \text{ for some live } \Gamma\right)$ *then*
 $\left(\vdash P : \mathbb{T} \implies (\mathsf{r} \triangleleft P \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M} \longrightarrow^* \mathcal{M}' \text{ implies } \mathcal{M}' \neq \mathsf{error})\right)$

(2) *a* complete subtyping *if* $\mathbb{T} \not\preccurlyeq \mathbb{T}'$ *implies that there are* $\mathsf{r} \notin \mathtt{pt}(\mathbb{T}'), \mathcal{M}, P$ *such that:*
 a. $\left(\forall Q : \vdash Q : \mathbb{T}' \implies \Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M} \text{ for some live } \Gamma\right)$
 b. $\vdash P : \mathbb{T}$
 c. $\mathsf{r} \triangleleft P \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M} \longrightarrow^* \mathsf{error}$.

(3) *a* precise subtyping *if it is both sound and complete.*

As customary, our subtyping relation is embedded in the type system via a subsumption rule, giving soundness as an immediate consequence of the subject reduction property.

$$\frac{\mathsf{p}! \notin \mathtt{act}(\mathsf{W}')}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{W}'}\,[\text{N-OUT}] \quad \frac{\mathsf{p}? \notin \mathtt{act}(\mathsf{W}')}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{W}'}\,[\text{N-INP}] \quad \frac{\mathsf{p}! \notin \mathtt{act}(\mathsf{W})}{\mathsf{W} \not\lesssim \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}'}\,[\text{N-OUT-R}] \quad \frac{\mathsf{p}? \notin \mathtt{act}(\mathsf{W})}{\mathsf{W} \not\lesssim \mathsf{p}?\ell(\mathsf{S}).\mathsf{W}'}\,[\text{N-INP-R}]$$

$$\frac{\ell \neq \ell'}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}'}\,[\text{N-INP-}\ell] \quad \frac{\mathsf{S}' \not\leq: \mathsf{S}}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-INP-S}] \quad \frac{\mathsf{S}' \leq: \mathsf{S} \quad \mathsf{W} \not\lesssim \mathsf{W}'}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-INP-W}]$$

$$\frac{\ell \neq \ell'}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}'}\,[\text{N-}\mathcal{A}\text{-}\ell] \quad \frac{\mathsf{S}' \not\leq: \mathsf{S}}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-}\mathcal{A}\text{-S}]$$

$$\frac{\mathsf{S}' \leq: \mathsf{S} \quad \mathsf{W} \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{W}'}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-}\mathcal{A}\text{-W}] \quad \frac{}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{q}!\ell'(\mathsf{S}').\mathsf{W}'}\,[\text{N-I-O-1}] \quad \frac{}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{q}!\ell'(\mathsf{S}').\mathsf{W}'}\,[\text{N-I-O-2}]$$

$$\frac{\ell \neq \ell'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{p}!\ell'(\mathsf{S}').\mathsf{W}'}\,[\text{N-OUT-}\ell] \quad \frac{\mathsf{S} \not\leq: \mathsf{S}'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-OUT-S}] \quad \frac{\mathsf{S} \leq: \mathsf{S}' \quad \mathsf{W} \not\lesssim \mathsf{W}'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-OUT-W}]$$

$$\frac{\ell \neq \ell'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell'(\mathsf{S}').\mathsf{W}'}\,[\text{N-}\mathcal{B}\text{-}\ell] \quad \frac{\mathsf{S} \not\leq: \mathsf{S}'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-}\mathcal{B}\text{-S}] \quad \frac{\mathsf{S} \leq: \mathsf{S}' \quad \mathsf{W} \not\lesssim \mathcal{B}^{(\mathsf{p})}.\mathsf{W}'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \not\lesssim \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\,[\text{N-}\mathcal{B}\text{-W}]$$

Table 4 The relation $\not\lesssim$ between SISO trees.

▶ **Theorem 5.2** (Soundness). *The asynchronous multiparty session subtyping $\leqslant$ is sound.*

Proof. Take any $\mathbb{T}, \mathbb{T}'$ such that $\mathbb{T} \leqslant \mathbb{T}'$, and $\mathsf{r}, \mathcal{M}$ satisfying the following condition:

$$\forall Q: \ \vdash Q : \mathbb{T}' \implies \Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M} \ \text{ for some live } \Gamma \tag{1}$$

If $\vdash P : \mathbb{T}$, we derive by [T-SUB] that $\vdash P : \mathbb{T}'$ holds. By (1), $\Gamma \vdash \mathsf{r} \triangleleft P \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}$ for some live Γ. Hence, by Corollary 4.6, $\mathsf{r} \triangleleft P \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M} \longrightarrow^* \mathcal{M}'$ implies $\mathcal{M}' \neq \mathsf{error}$. ◀

To prove the completeness of $\leqslant$, we show that it satisfies item *(2)* of Def. 5.1, in 4 steps:

[Step 1] We define the *negation $\not\lesssim$ of the SISO trees refinement relation* by an inductive definition, thus getting a perspicuous characterisation of the complement $\not\leqslant$ of the subtyping relation. In addition, for every pair $\mathbb{T}, \mathbb{T}'$ with $\mathbb{T} \not\leqslant \mathbb{T}'$ we choose a pair $\mathbb{U}, \mathbb{V}'$ satisfying $\mathbb{U} \not\lesssim \mathbb{V}'$ and $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$.

[Step 2] We define for every $\mathbb{U}$ a *characteristic process* $\mathcal{P}(\mathbb{U})$. If $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$, we prove that $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{T}$.

[Step 3] For every $\mathbb{V}'$ with $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$, and for every participant $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}')$, we define a *characteristic session* $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$, which is typable if composed with a process Q of type $\mathbb{T}'$:

$$\forall Q: \ \vdash Q : \mathbb{T}' \implies \Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'} \ \text{ for some live } \Gamma.$$

[Step 4] Finally, we show that for all $\mathbb{U}, \mathbb{V}'$ such that $\mathbb{U} \not\lesssim \mathbb{V}'$, the characteristic session $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$ (Step 3) reduces to error if composed with the characteristic process of $\mathbb{U}$ (Step 2):

$$\mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'} \longrightarrow^* \mathsf{error}.$$

Hence, we prove the completeness of $\leqslant$ by showing that, for all $\mathbb{T}, \mathbb{T}'$ such that $\mathbb{T} \not\leqslant \mathbb{T}'$, we can find $\mathsf{r} \notin \mathtt{pt}(\mathbb{T}')$, $P = \mathcal{P}(\mathbb{U})$ (Steps 1,2), and $\mathcal{M} = \mathcal{M}_{\mathsf{r},\mathbb{V}'}$ (Step 3) satisfying Def. 5.1*(2)* (Step 4). We now illustrate each step in more detail.

Step 1: subtyping negation

In Table 4 we *inductively* define the relation $\not\lesssim$ over SISO trees. It contains all pairs of SISO trees that are *not* related by $\lesssim$, as stated in Lemma 5.3 below. *(Proof in Appendix D).*

The first category of rules checks a direct syntactic mismatch: whether their sets of actions are disjunctive ([N-OUT], [N-INP], [N-OUT-R], [N-INP-R]); the label of the LHS is not equal to the label of the RHS ([N-INP-ℓ], [N-OUT-ℓ]); or matching labels followed by mismatching sorts or continuations ([N-INP-S], [N-OUT-S], [N-INP-W], [N-OUT-W]).

The second category checks more subtle cases related to asynchronous permutations; rule [N-$\mathcal{A}$-ℓ] checks a label mismatch when the input on the RHS is preceded by a finite number of inputs from other participant; similarly rules [N-$\mathcal{A}$-S] and [N-$\mathcal{A}$-W] check mismatching sorts or continuations. Rules [N-I-O-1] and [N-I-O-2] formulate the cases such that the top prefix on the LHS is input and the top sequence of prefixes on the RHS consists of a finite number of inputs from other participants and/or outputs. Finally, rules [N-$\mathcal{B}$-ℓ], [N-$\mathcal{B}$-S] and [N-$\mathcal{B}$-W] check the cases of label mismatch, or matching labels followed by mismatching sorts or continuations of the two types with output prefixes targeting a same participant, where the RHS is prefixed by a finite number of outputs to other participants and/or inputs. *(Details: Appendix D)*.

▶ **Lemma 5.3.** *Let* W *and* W' *be SISO trees. If* $\neg(\mathsf{W} \lesssim \mathsf{W}')$ *then* $\mathsf{W} \not\lesssim \mathsf{W}'$ *is derivable.*

It is immediate from Def. 3.4 that $\mathbb{T}$ is *not* a subtype of $\mathbb{T}'$, written $\mathbb{T} \not\leqslant \mathbb{T}'$, if and only if:

$$\exists \mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}} \;\; \exists \mathsf{V}' \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}} \;\; \forall \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}} \;\; \forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \;\; \mathsf{W} \not\lesssim \mathsf{W}' \tag{2}$$

Moreover, we prove that whenever $\mathbb{T} \not\leqslant \mathbb{T}'$, we can find *regular, syntax-derived* SO/SI trees usable as the witnesses U, V' in (2) *(Appendix, page 50)*. Thus, $\mathbb{T} \not\leqslant \mathbb{T}'$ implies:

$$\exists \mathbb{U}, \mathbb{V}' : \; \mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}} \;\; \mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}} \;\; \forall \mathsf{W} \in [\![\mathcal{T}(\mathbb{U})]\!]_{\mathsf{SI}} \;\; \forall \mathsf{W}' \in [\![\mathcal{T}(\mathbb{V}')]\!]_{\mathsf{SO}} \;\; \mathsf{W} \not\lesssim \mathsf{W}' \tag{3}$$

▶ **Example 5.4.** Consider the example in Section 1, and its types $\mathbb{T}'$ and $\mathbb{T}$ in Example 3.5:

$$\mathbb{T}' = \bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\,\&\,\mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases} \\ \mathit{stop}(\mathtt{unit}).\,\&\,\mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases} \end{cases} \qquad \mathbb{T} = \&\,\mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases} \\ \mathit{error}(\mathtt{bool}).\bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases} \end{cases}$$

We have seen that $\mathbb{T}' \leqslant \mathbb{T}$ holds (Example 3.5), and thus, by subsumption, our type system allows to use the optimised process P'_{r} in place of P_{r} (Example 4.2). We now show that the inverse relation does *not* hold, i.e., $\mathbb{T} \not\leqslant \mathbb{T}'$, hence the inverse process replacement is disallowed. Take, e.g., $\mathbb{U}, \mathbb{V}'$ as follows, noticing that $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$:

$$\mathbb{U} = \&\,\mathsf{p}? \begin{cases} \mathit{success}(\mathtt{int}).\mathsf{q}!\mathit{cont}(\mathtt{int}).\mathtt{end} \\ \mathit{error}(\mathtt{bool}).\mathsf{q}!\mathit{stop}(\mathtt{unit}).\mathtt{end} \end{cases} \qquad \mathbb{V}' = \bigoplus \mathsf{q}! \begin{cases} \mathit{cont}(\mathtt{int}).\mathsf{p}?\mathit{success}(\mathtt{int}).\mathtt{end} \\ \mathit{stop}(\mathtt{int}).\mathsf{p}?\mathit{error}(\mathtt{bool}).\mathtt{end} \end{cases}$$

For all $\mathsf{W} \in [\![\mathcal{T}(\mathbb{U})]\!]_{\mathsf{SI}} = \{\mathsf{p}?\mathit{success}(\mathtt{int}).\mathsf{q}!\mathit{cont}(\mathtt{int}).\mathtt{end}\,,\, \mathsf{p}?\mathit{error}(\mathtt{bool}).\mathsf{q}!\mathit{stop}(\mathtt{unit}).\mathtt{end}\}$ and all $\mathsf{W}' \in [\![\mathcal{T}(\mathbb{V}')]\!]_{\mathsf{SO}} = \{\mathsf{q}!\mathit{cont}(\mathtt{int}).\mathsf{p}?\mathit{success}(\mathtt{int}).\mathtt{end}\,,\, \mathsf{q}!\mathit{stop}(\mathtt{unit}).\mathsf{p}?\mathit{error}(\mathtt{bool}).\mathtt{end}\}$ we get by [N-I-O-1] that $\mathsf{W} \not\lesssim \mathsf{W}'$. Therefore, we conclude $\mathbb{T} \not\leqslant \mathbb{T}'$.

Step 2: characteristic processes

For any SO type $\mathbb{U}$, we define a characteristic process $\mathcal{P}(\mathbb{U})$ (Def. 5.5): intuitively, it is a process constructed to communicate as prescribed by $\mathbb{U}$, and to be typable by $\mathbb{U}$.

▶ **Definition 5.5.** *The characteristic process* $\mathcal{P}(\mathbb{U})$ *of type* $\mathbb{U}$ *is defined inductively as follows:*

$$\mathcal{P}(\mathtt{end}) = \mathbf{0} \qquad \mathcal{P}(\boldsymbol{t}) = X_{\boldsymbol{t}} \qquad \mathcal{P}(\mu \boldsymbol{t}.\mathbb{U}) = \mu X_{\boldsymbol{t}}.\mathcal{P}(\mathbb{U}) \qquad \mathcal{P}(\mathsf{p}!\ell(\mathsf{S}).\mathbb{U}) = \mathsf{p}!\ell\langle \mathbf{val}(\!|\mathsf{S}|\!)\rangle.\mathcal{P}(\mathbb{U})$$

$$\mathcal{P}\Big(\&_{i\in I}\,\mathsf{p}?\ell_i(\mathsf{S}_i).\mathbb{U}_i\Big) \;=\; \textstyle\sum_{i\in I} \mathsf{p}?\ell_i(x_i).\mathsf{if}\ \mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i)$$

$$\textit{where:} \begin{cases} \mathbf{val}(\!|\mathtt{nat}|\!) = 1 \quad \mathbf{val}(\!|\mathtt{int}|\!) = -1 \quad \mathbf{val}(\!|\mathtt{bool}|\!) = \mathsf{true} \\ \mathbf{expr}(\!|\mathsf{e}, \mathtt{bool}|\!) = (\neg \mathsf{e}) \quad \mathbf{expr}(\!|\mathsf{e}, \mathtt{nat}|\!) = (\mathsf{succ}(\mathsf{e}) > 0) \quad \mathbf{expr}(\!|\mathsf{e}, \mathtt{int}|\!) = (\mathsf{inv}(\mathsf{e}) > 0) \end{cases}$$

By Def. 5.5, for every output in $\mathbb{U}$, $\mathcal{P}(\mathbb{U})$ sends a value $\mathbf{val}(\!|\mathsf{S}|\!)$ of the right sort S; and for every external choice in $\mathbb{U}$, $\mathcal{P}(\mathbb{U})$ performs a branching, and uses any received value x_i of sort S_i in a boolean expression $\mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)$.

Crucially, for all $\mathbb{T}$ and $\mathbb{U}$ such that $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ (e.g., from (3) above), we have $\mathbb{U} \leqslant \mathbb{T}$: therefore, $\mathcal{P}(\mathbb{U})$ is also typable by $\mathbb{T}$, as per Prop. 5.6 below. *(Proof: Appendix D)*

▶ **Proposition 5.6.** *For all closed types $\mathbb{T}$ and $\mathbb{U}$, if $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ then $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{T}$.*

Step 3: characteristic session

The next step to prove completeness is to define for each session type $\mathbb{V}'$ and participant $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}')$ a characteristic session $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$, that is well typed (with a live typing environment) when composed with participant r associated with a process of type $\mathbb{V}'$ and empty queue.

For a SI type $\mathbb{V}'$ and $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}') = \{\mathsf{p}_1, \ldots, \mathsf{p}_m\}$, we define m *characteristic SO session types* where participants $\mathsf{p}_1, \ldots, \mathsf{p}_m$ are engaged in a live multiparty interaction with r, and with each other. Def. 5.7 ensures that after each communication between r and some $\mathsf{p} \in \mathtt{pt}(\mathbb{V}')$, there is a cyclic sequence of communications starting with p, involving *all* other $\mathsf{q} \in \mathtt{pt}(\mathbb{V}')$, and ending with p — with each participant acting both as receiver, and as sender.

▶ **Definition 5.7.** *Let $\mathbb{V}'$ be a SI session type and $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}') = \{\mathsf{p}_1, ..., \mathsf{p}_m\}$. For every $k \in \{1, ..., m\}$, if $m \geq 2$ we define a* characteristic SO session type $\mathtt{cyclic}(\mathbb{V}', \mathsf{p}_k, \mathsf{r})$ *as follows:*

$$
\begin{array}{lll}
\mathtt{cyclic}(\mathtt{end}, \mathsf{p}_k, \mathsf{r}) & = \mathtt{end} & \\
\mathtt{cyclic}(\mathbf{t}, \mathsf{p}_k, \mathsf{r}) & = \mathbf{t} & \\
\mathtt{cyclic}(\mu\mathbf{t}.\mathbb{V}_1'', \mathsf{p}_k, \mathsf{r}) & = \mu\mathbf{t}.\mathtt{cyclic}(\mathbb{V}_1'', \mathsf{p}_k, \mathsf{r}) & \\
\mathtt{cyclic}(\mathsf{p}_k?\ell(\mathsf{S}).\mathbb{V}', \mathsf{p}_k, \mathsf{r}) & = \mathsf{r}!\ell(\mathsf{S}).\mathsf{p}_{k+1}!\ell(\mathtt{bool}).\mathsf{p}_{k-1}?\ell(\mathtt{bool}).\mathtt{cyclic}(\mathbb{V}', \mathsf{p}_k, \mathsf{r}) & \\
\mathtt{cyclic}(\mathsf{q}?\ell(\mathsf{S}).\mathbb{V}', \mathsf{p}_k, \mathsf{r}) & = \mathsf{p}_{k-1}?\ell(\mathtt{bool}).\mathsf{p}_{k+1}!\ell(\mathtt{bool}).\mathtt{cyclic}(\mathbb{V}', \mathsf{p}_k, \mathsf{r}) & \textit{(if } \mathsf{q} \neq \mathsf{p}_k\textit{)} \\
\mathtt{cyclic}(\bigoplus_{j \in J} \mathsf{p}_k!\ell_j(\mathsf{S}_j).\mathbb{V}_j', \mathsf{p}_k, \mathsf{r}) & = \&_{j \in J}\, \mathsf{r}?\ell_j(\mathsf{S}_j).\mathsf{p}_{k+1}!\ell_j(\mathtt{bool}).\mathsf{p}_{k-1}?\ell_j(\mathtt{bool}).\mathtt{cyclic}(\mathbb{V}_j', \mathsf{p}_k, \mathsf{r}) & \\
\mathtt{cyclic}(\bigoplus_{j \in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathbb{V}_j', \mathsf{p}_k, \mathsf{r}) & = \&_{j \in J}\, \mathsf{p}_{k-1}?\ell_j(\mathtt{bool}).\mathsf{p}_{k+1}!\ell_j(\mathtt{bool}).\mathtt{cyclic}(\mathbb{V}_j', \mathsf{p}_k, \mathsf{r}) & \textit{(if } \mathsf{q} \neq \mathsf{p}_k\textit{)}
\end{array}
$$

If $m = 1$ (i.e., if there is only one participant in $\mathbb{V}'$) we define $\mathtt{cyclic}(\mathbb{V}', \mathsf{p}_1, \mathsf{r})$ *as above, but we omit the (highlighted) cyclic communications, and the cases with $\mathsf{q} \neq \mathsf{p}_k$ do not apply.*

▶ **Example 5.8** (Characteristic session types). Consider the following SI type:

$$\mathbb{V}' \;=\; \mu\mathbf{t}.\bigoplus\{\mathsf{q}!\ell_2(\mathtt{nat}).\mathsf{p}?\ell_1(\mathtt{nat}).\mathbf{t}\,,\; \mathsf{q}!\ell_3(\mathtt{nat}).\mathsf{p}?\ell_4(\mathtt{nat}).\mathbf{t}\}$$

Let $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}')$. The characteristic session types for participants $\mathsf{p}, \mathsf{q} \in \mathtt{pt}(\mathbb{V}')$ are:

$$
\begin{aligned}
\mathtt{cyclic}(\mathbb{V}', \mathsf{p}, \mathsf{r}) \;&=\; \mu\mathbf{t}.\,\&\begin{cases}\mathsf{q}?\ell_2(\mathtt{bool}).\mathsf{q}!\ell_2(\mathtt{bool}).\mathsf{r}!\ell_1(\mathtt{nat}).\mathsf{q}!\ell_1(\mathtt{bool}).\mathsf{q}?\ell_1(\mathtt{bool}).\mathbf{t}\\ \mathsf{q}?\ell_3(\mathtt{bool}).\mathsf{q}!\ell_3(\mathtt{bool}).\mathsf{r}!\ell_4(\mathtt{nat}).\mathsf{q}!\ell_4(\mathtt{bool}).\mathsf{q}?\ell_4(\mathtt{bool}).\mathbf{t}\end{cases}\\
\mathtt{cyclic}(\mathbb{V}', \mathsf{q}, \mathsf{r}) \;&=\; \mu\mathbf{t}.\,\&\begin{cases}\mathsf{r}?\ell_2(\mathtt{nat}).\mathsf{p}!\ell_2(\mathtt{bool}).\mathsf{p}?\ell_2(\mathtt{bool}).\mathsf{p}?\ell_1(\mathtt{bool}).\mathsf{p}!\ell_1(\mathtt{bool}).\mathbf{t}\\ \mathsf{r}?\ell_3(\mathtt{nat}).\mathsf{p}!\ell_3(\mathtt{bool}).\mathsf{p}?\ell_3(\mathtt{bool}).\mathsf{p}?\ell_4(\mathtt{bool}).\mathsf{p}!\ell_4(\mathtt{bool}).\mathbf{t}\end{cases}
\end{aligned}
$$

Note that if r follows type $\mathbb{V}'$, then it must select and send to q one message between ℓ_2 and ℓ_3; correspondingly, the characteristic session type for q receives the message (with a branching), and propagates it to p, who sends it back to q (cyclic communication). Then, r waits for a message from p (either ℓ_1 or ℓ_4, depending on the previous selection): p will send such a message, and also propagate it to q with a cyclic communication. ■

Given an SI type $\mathbb{V}'$, we can use Def. 5.7 to construct the following typing environment:

$$\Gamma \;=\; \{\mathsf{r}\colon (\epsilon, \mathbb{V}')\} \cup \{\mathsf{p}\colon (\epsilon, \mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \in \mathtt{pt}(\mathbb{V}')\} \quad \text{where } \forall \mathsf{p} \in \mathtt{pt}(\mathbb{V}')\colon \mathbb{U}_{\mathsf{p}} = \mathtt{cyclic}(\mathbb{V}', \mathsf{p}, \mathsf{r}) \qquad (4)$$

i.e., we compose $\mathbb{V}'$ with the characteristic session types of all its participants. The cyclic communications of Def. 5.7 ensure that Γ is live. We can use Γ to type the composition of a process for r, of type $\mathbb{V}'$, together with the characteristic processes of the characteristic session types of each participants in $\mathbb{V}'$: we call such processes the *characteristic session* $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$. This is formalised in Def. 5.9 and Prop. 5.10 below.

▶ **Definition 5.9.** *For any SI type* $\mathbb{V}'$ *and* $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}')$*, we define the* characteristic session*:*

$$\mathcal{M}_{\mathsf{r},\mathbb{V}'} \;=\; \prod_{\mathsf{p}\in\mathtt{pt}(\mathbb{V}')} \Big(\mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \;\mid\; \mathsf{p} \triangleleft \varnothing\Big) \qquad \textit{where } \forall \mathsf{p}\in\mathtt{pt}(\mathbb{V}') : \; \mathbb{U}_{\mathsf{p}} = \mathtt{cyclic}(\mathbb{V}', \mathsf{p}, \mathsf{r})$$

▶ **Proposition 5.10.** *Let* $\mathbb{V}'$ *be a SI type and* $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}')$*. Let* Q *be a process such that* $\vdash Q : \mathbb{V}'$*. Then, there is a live typing environment* Γ *(see* (4)*) such that* $\Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$*.*

Crucially, for all $\mathbb{T}'$ and $\mathbb{V}'$ such that $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ (e.g., from (3) above), we have $\mathbb{T}' \leqslant \mathbb{V}'$. Thus, by subsumption, $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$ is also typable with a process of type $\mathbb{T}'$ (Prop. 5.11).

▶ **Proposition 5.11.** *Take any* $\mathbb{T}'$, $\mathsf{r} \notin \mathtt{pt}(\mathbb{T}')$*, SI type* $\mathbb{V}'$ *such that* $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$*, and* Q *such that* $\vdash Q : \mathbb{T}'$*. Then, there is a live* Γ *(see* (4)*) such that* $\Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$*.*

Step 4: completeness

This final step of our completeness proof encompasses all elements introduced thus far. *(Proofs in Appendix D)*

▶ **Proposition 5.12.** *Let* $\mathbb{T}$ *and* $\mathbb{T}'$ *be session types such that* $\mathbb{T} \not\leqslant \mathbb{T}'$*. Take any* $\mathsf{r} \notin \mathbb{T}'$*. Then, there are* $\mathbb{U}$ *and* $\mathbb{V}'$ *with* $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ *and* $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ *and* $\mathbb{U} \not\lesssim \mathbb{V}'$ *such that:*

1. $\forall Q : \;\vdash Q : \mathbb{T}' \implies \Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$ *for some live* Γ*;* *(by* (4) *and Prop. 5.11)*
2. $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{T}$*;* *(by Prop. 5.6)*
3. $\mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'} \longrightarrow^* \mathsf{error}$.

Intuitively, we obtain item 3 of Prop. 5.12 because the characteristic session $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$ expects to interact with a process of type $\mathbb{V}'$ (or a subtype, like $\mathbb{T}'$); however, when a process that behaves like $\mathbb{U}$ is inserted, the cyclic communications and/or the expressions of $\mathcal{M}_{\mathsf{r},\mathbb{V}'}$ (given by Def. 5.5 and 5.7) are disrupted: this is because $\mathbb{U} \not\lesssim \mathbb{V}'$, and the (incorrect) message reorderings and mutations allowed by $\not\lesssim$ (Table 4) cause the errors in Table 2.

We now conclude with our main results.

▶ **Theorem 5.13.** *The asynchronous multiparty session subtyping* $\leqslant$ *is complete.*

Proof. Direct consequence of Prop. 5.12: by taking r and letting $\mathcal{M} = \mathcal{M}_{\mathsf{r},\mathbb{V}'}$ and $P = \mathcal{P}(\mathbb{U})$ from its statement, we satisfy item *(2)* of Def. 5.1. ◀

▶ **Theorem 5.14.** *The asynchronous multiparty session subtyping* $\leqslant$ *is precise.*

Proof. Direct consequence of Theorems 5.2 and 5.13, which satisfy item *(3)* of Def. 5.1. ◀

Our results also provide a proof that *our multiparty asynchronous subtyping is precise wrt. liveness*, as formalised below — where $\Gamma\{\mathsf{p} \mapsto (\sigma, \mathbb{T})\}$ is the typing environment obtained from Γ by replacing the entry for p with $(\sigma, \mathbb{T})$.

▶ **Theorem 5.15.** *For all session types* $\mathbb{T}$ *and* $\mathbb{T}'$*, multiparty asynchronous subtyping* $\leqslant$ *is:*

sound wrt. liveness: *if* $\mathbb{T} \leqslant \mathbb{T}'$*, then* $\forall \Gamma$ *with* $\Gamma(\mathsf{r}) = (\epsilon, \mathbb{T}')$*, if* Γ *is live,* $\Gamma\{\mathsf{r} \mapsto (\epsilon, \mathbb{T})\}$ *is live;*

complete wrt. liveness: *if* $\mathbb{T} \not\leqslant \mathbb{T}'$*, then* $\exists \Gamma$ *live with* $\Gamma(\mathsf{r}) = (\epsilon, \mathbb{T}')$*, but* $\Gamma\{\mathsf{r} \mapsto (\epsilon, \mathbb{T})\}$ *is not live.*

Proof. Soundness of $\leqslant$ is a result used to prove subject reduction (*Lemma C.12*). Completeness, instead, descends from Prop. 5.12: for any $\mathbb{T}, \mathbb{T}'$ such that $\mathbb{T} \not\leqslant \mathbb{T}'$, it builds live a typing context Γ (see (4)) with $\Gamma(\mathsf{r}) = (\epsilon, \mathbb{T}')$ and cyclic communications (item 1). Observe that the environment $\Gamma\{\mathsf{r} \mapsto (\epsilon, \mathbb{T})\}$ types the session of Prop. 5.12(3), that reduces to error. Hence, by the contrapositive of Corollary 4.6, we conclude that $\Gamma\{\mathsf{r} \mapsto (\epsilon, \mathbb{T})\}$ is not live. ◀

6 Related Work and Conclusion

The preciseness of subtyping relations has been adopted to justify the canonicity of refinement, in the context of both functional and concurrent calculi. Operational preciseness of subtyping was first introduced in [2] (and later published in [27]), and applied to λ-calculus with iso-recursive types. Later, [17] adapts the idea of [2] to the setting of the concurrent λ-calculus with intersection and union types by [16]. In the context of the λ-calculus, a similar framework, *semantic subtyping*, is proposed in [9]: each type T is interpreted as the set of values having type T, and subtyping is defined as subset inclusion between type interpretations. This gives a precise subtyping as long as the calculus allows to operationally distinguish values of different types. Semantic subtyping is also studied in [7] (for a π-calculus with a patterned input and IO-types), and in [8] (for a π-calculus with binary session types); in both works, types are built using type constructors including union, intersection and negation. Semantic subtyping is precise for the calculi of [7, 8, 18]: this is due to the type case constructor in [18], and to the blocking of inputs for values of "wrong" types in [7, 8].

In the context of *binary* session types, the first general formulation of precise subtyping (synchronous and asynchronous) is given in [12, 11], for a π-calculus where processes are typed by giving session types to channels (as in [22]). The first result by [12, 11] is that the well-known branching-selection subtyping [19, 13] is sound and complete for the *synchronous* binary session π- calculus. [12, 11] also examine an *asynchronous* binary session π-calculus, and introduce a subtyping relation (restricting the subtyping for the higher-order π-calculus by [29]) that is also proved precise.

In the context of *multiparty* session types, an asynchronous subtyping relation was proposed in [30] (and claimed to be decidable — a claim later disproved in [6]). Following an approach similar to [12, 11], [21] shows that the *synchronous* multiparty extension of binary session subtyping [19] is sound and complete, hence precise.

Asynchronous session subtyping was shown to be undecidable, even for binary sessions, in [26, 5], using a link between session types and communicating automata theories [14, 15]. Various proposals of limited decidable subsets of *binary* session automata are in [3, 26, 6]. The aim of our paper is *not* finding a decidable approximation of asynchronous multiparty session subtyping, but defining a canonical, *precise* subtyping. Interestingly, our SISO decomposition technique leads to: *(1)* intuitive but general refinement rules (see Example 3.6, where $\leqslant$ proves an example not supported by the algorithm in [3]); and *(2)* preciseness of $\leqslant$ wrt. liveness (Theorem 5.15) which is directly usable to define the largest multiparty asynchronous refinement relation wrt. liveness in communicating session automata [14, 15].

Conclusion. Unlike this paper, no other published work addresses precise *asynchronous multiparty* session subtyping. A main challenge was the exact formalisation of the subtyping itself, which must satisfy many *desiderata*: it must capture a wide variety of input/output reorderings performed by different participants, without being too strict (otherwise, completeness is lost) nor too lax (otherwise, soundness is lost); moreover, its definition must not be overly complex to understand, and tractable in proofs. We achieved these *desiderata* with our novel approach, based on SISO tree decomposition and refinement, which yeilds a simpler subtyping relation than [12, 11] (see Remark 3.3). Moreover, our results are much more general than [21]: by using live typing environments (Def. 4.4), we are not limited to sessions that match some global type; our results are also stronger, as we prove soundness wrt. a wider range of errors (see Table 2). Our future work includes the study of precise subtyping for richer multiparty session π-calculi, with multiple session initiations and delegation.

References

1. D. Ancona, V. Bono, M. Bravetti, G. Castagna, J. Campos, S. J. Gay, E. Giachino, E. B. Johnsen, V. Mascardi, N. Ng, L. Padovani, P.-M. Deniélou, N. Gesbert, R. Hu, F. Martins, F. Montesi, R. Neykova, V. T. Vasconcelos, and N. Yoshida. Behavioral types in programming languages. 2016. *Foundations and Trends in Programming Languages.*
2. J. Blackburn, I. Hernandez, J. Ligatti, and M. Nachtigal. Completely subtyping iso-recursive types. Technical Report CSE-071012, University of South Florida, 2012.
3. M. Bravetti, M. Carbone, J. Lange, N. Yoshida, and G. Zavattaro. A sound algorithm for asynchronous session subtyping. In *CONCUR*, volume 140 of *LIPIcs*, pages 38:1–38:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
4. M. Bravetti, M. Carbone, J. Lange, N. Yoshida, and G. Zavattaro. A sound algorithm for asynchronous session subtyping and its implemenation. *CoRR*, abs/1907.00421, 2019.
5. M. Bravetti, M. Carbone, and G. Zavattaro. Undecidability of asynchronous session subtyping. *Inf. Comput.*, 256:300–320, 2017.
6. M. Bravetti, M. Carbone, and G. Zavattaro. On the boundary between decidability and undecidability of asynchronous session subtyping. *Theor. Comput. Sci.*, 722:19–51, 2018.
7. G. Castagna, R. De Nicola, and D. Varacca. Semantic subtyping for the pi-calculus. *Theoretical Computer Science*, 398(1-3):217–242, 2008.
8. G. Castagna, M. Dezani-Ciancaglini, E. Giachino, and L. Padovani. Foundations of session types. In *PPDP*, pages 219–230. ACM, 2009.
9. G. Castagna and A. Frisch. A gentle introduction to semantic subtyping. In *ICALP*, volume 3580 of *LNCS*, pages 30–34. Springer, 2005.
10. D. Castro-Perez and N. Yoshida. Compiling First-Order Functions to Session-Typed Parallel Code. In *29th International Conference on Compiler Construction*, CC 2020, pages 143–154. ACM, 2020.
11. T.-C. Chen, M. Dezani-Ciancaglini, A. Scalas, and N. Yoshida. On the Preciseness of Subtyping in Session Types. *LMCS*, 13:1–62, 2017.
12. T.-C. Chen, M. Dezani-Ciancaglini, and N. Yoshida. On the Preciseness of Subtyping in Session Types. In *PPDP*, pages 135–146. ACM Press, 2014.
13. R. Demangeon and K. Honda. Full abstraction in a subtyped pi-calculus with linear types. In *CONCUR*, volume 6901 of *LNCS*, pages 280–296. Springer, 2011.
14. P.-M. Deniélou and N. Yoshida. Multiparty session types meet communicating automata. In *ESOP*, volume 7211 of *LNCS*, pages 194–213, 2012.
15. P.-M. Deniélou and N. Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP*, volume 7966 of *LNCS*, pages 174–186. Springer, 2013.
16. M. Dezani-Ciancaglini, U. de'Liguoro, and A. Piperno. A filter model for concurrent lambda-calculus. *SIAM Journal on Computing*, 27(5):1376–1419, 1998.
17. M. Dezani-Ciancaglini and S. Ghilezan. Preciseness of subtyping on intersection and union types. In *RTATLCA*, volume 8560 of *LNCS*, pages 194–207. Springer, 2014.
18. A. Frisch, G. Castagna, and V. Benzaken. Semantic subtyping: dealing set-theoretically with function, union, intersection, and negation types. *Journal of ACM*, 55(4):1–64, 2008.
19. S. Gay and M. Hole. Subtyping for session types in the pi calculus. *Acta Informatica*, 42(2/3):191–225, 2005.
20. S. Gay and A. Ravara, editors. *Behavioural Types: from Theory to Tools*. River Publishers series in automation, control and robotics. River Publishers, June 2017.
21. S. Ghilezan, S. Jaksic, J. Pantovic, A. Scalas, and N. Yoshida. Precise subtyping for synchronous multiparty sessions. *J. Log. Algebr. Meth. Program.*, 104:127–173, 2019.
22. K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type disciplines for structured communication-based programming. In *ESOP*, volume 1381 of *LNCS*, pages 122–138. Springer, 1998.

23 K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, pages 273–284. ACM, 2008.
24 K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. *J. ACM*, 63(1):9:1–9:67, 2016.
25 R. Hu. Distributed Programming Using Java APIs Generated from Session Types. *Behavioural Types: from Theory to Tools*, pages 287–308, 2017.
26 J. Lange and N. Yoshida. On the undecidability of asynchronous session subtyping. In *FoSSaCS*, volume 10203 of *LNCS*, pages 441–457, 2017.
27 J. Ligatti, J. Blackburn, and M. Nachtigal. On subtyping-relation completeness, with an application to iso-recursive types. *ACM Trans. Program. Lang. Syst.*, 39(1):4:1–4:36, Mar. 2017.
28 B. H. Liskov and J. M. Wing. A behavioral notion of subtyping. *ACM Trans. Program. Lang. Syst.*, 16(6):1811–1841, Nov. 1994.
29 D. Mostrous and N. Yoshida. Session Typing and Asynchronous Subtying for Higher-Order π-Calculus. *Info.& Comp.*, 241:227–263, 2015.
30 D. Mostrous, N. Yoshida, and K. Honda. Global principal typing in partially commutative asynchronous sessions. In *ESOP*, volume 5502 of *LNCS*, pages 316–332. Springer, 2009.
31 N. Ng, J. G. Coutinho, and N. Yoshida. Protocols by default: Safe mpi code generation based on session types. In *CC 2015*, volume 9031 of *LNCS*, pages 212–232. Springer, 2015.
32 N. Ng, N. Yoshida, and K. Honda. Multiparty Session C: Safe parallel programming with message optimisation. In *TOOLS*, volume 7304 of *LNCS*, pages 202–218. Springer, 2012.
33 B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.
34 A. Scalas and N. Yoshida. Less is more: multiparty session types revisited. *PACMPL*, 3(POPL):30:1–30:29, 2019.
35 K. Takeuchi, K. Honda, and M. Kubo. An Interaction-based Language and its Typing System. In *PARLE'94*, volume 817 of *LNCS*, pages 398–413, 1994.

A Appendix of Section 2

We list an example of processes from Section 1 and omitted definitions from Section 2.

A.1 Syntax

Multiparty sessions are ranged over by $\mathcal{M}, \mathcal{M}', \ldots$; *session participants* by $\mathsf{p}, \mathsf{q}, \ldots$; *processes* by $P, Q, \ldots$; *queues* by $h, h', \ldots$; *message labels* by $\ell, \ell', \ldots$; *values*, by $\mathsf{v}, \mathsf{v}', \ldots$; *expressions* by $\mathsf{e}, \mathsf{e}', \ldots$; *expression variables* by $x, y, z \ldots$; and *process variables* by $X, Y, Z, \ldots$.

We give a full definition of $\mathtt{act}(P)$ below:

$$\mathtt{act}(P) = \begin{cases} \{\mathsf{p}?\} \cup \{\mathtt{act}(P_i) : i \in I\} & P = \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \\ \{\mathsf{p}!\} \cup \{\mathtt{act}(P')\} & P = \mathsf{p}!\ell\langle \mathsf{e}\rangle.P' \\ \mathtt{act}(P_1) \cup \mathtt{act}(P_2) & P = \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \\ \mathtt{act}(P') & P = \mu X.P' \\ \emptyset & P = \mathbf{0} \end{cases}$$

A.2 Operational semantics

We give the operational semantics of expressions. A *value* v can be a natural number n, an integer i, or a boolean true / false. An expression e can be a variable, a value, or a term built from expressions by applying the operators $\mathtt{succ}, \mathtt{inv}, \neg$, or the relation $=$. An *evaluation context* $\mathcal{E}$ is an expression with exactly one hole. *The value* v *of expression* e (notation $\mathsf{e} \downarrow \mathsf{v}$) is computed as defined in Table 5. The successor operation $\mathtt{succ}$ is defined only on natural numbers, the inverse operation $\mathtt{inv}$ is defined on integers, and negation $\neg$ is defined only on boolean values.

$$\mathsf{succ}(\mathsf{n}) \downarrow (\mathsf{n}+1) \qquad \mathsf{inv}(\mathsf{i}) \downarrow (-\mathsf{i}) \qquad \neg\mathsf{true} \downarrow \mathsf{false} \qquad \neg\mathsf{false} \downarrow \mathsf{true} \qquad \mathsf{v} \downarrow \mathsf{v}$$

$$\frac{\mathsf{e}_1 \downarrow \mathsf{v}_1 \quad \mathsf{e}_2 \downarrow \mathsf{v}_2 \quad \mathsf{v}_1 = \mathsf{v}_2}{(\mathsf{e}_1 = \mathsf{e}_2) \downarrow \mathsf{true}} \qquad \frac{\mathsf{e}_1 \downarrow \mathsf{v}_1 \quad \mathsf{e}_2 \downarrow \mathsf{v}_2 \quad \mathsf{v}_1 \neq \mathsf{v}_2}{(\mathsf{e}_1 = \mathsf{e}_2) \downarrow \mathsf{false}} \qquad \frac{\mathsf{e} \downarrow \mathsf{v} \quad \mathcal{E}(\mathsf{v}) \downarrow \mathsf{v}'}{\mathcal{E}(\mathsf{e}) \downarrow \mathsf{v}'}$$

Table 5 Expression evaluation

Table 6 defines structural congruence rules.

▶ Remark A.1. Some error rules in Table 2 partially overlap: e.g., a deadlocked session may reduce to error via [ERR-DEADLOCK], and possibly also via [ERR-ORPH-MSG] and [ERR-STARV].

▶ **Example A.2** (Reduction relations). We now exemplify the operational semantics using the running example from the Introduction. The session

$$\mathsf{r} \triangleleft \sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathsf{if}\ (x > 42)\ \mathsf{then}\ \mathsf{q}!\mathit{cont}\langle x\rangle.\mathbf{0}\ \mathsf{else}\ \mathsf{q}!\mathit{stop}\langle\rangle.\mathbf{0} \\ \mathsf{p}?\mathit{error}(\mathit{fatal}).\mathsf{if}\ (\neg\mathit{fatal})\ \mathsf{then}\ \mathsf{q}!\mathit{cont}\langle 43\rangle.\mathbf{0}\ \mathsf{else}\ \mathsf{q}!\mathit{stop}\langle\rangle.\mathbf{0} \end{Bmatrix} \ \mid\ \mathsf{p} \triangleleft \cdots \ \mid\ \mathsf{q} \triangleleft \cdots \ \mid\ \ldots$$

cannot reduce further since r is blocked until a message is sent by p to r. If this does not happen, the session will reduce to error by [ERR-STARV]. However, the optimised session

$$\mathsf{r} \triangleleft \mathsf{if}\ (\ldots)\ \mathsf{then}\ \mathsf{q}!\mathit{cont}\langle 43\rangle.\sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathbf{0} \\ \mathsf{p}?\mathit{error}(y).\mathbf{0} \end{Bmatrix}\ \mathsf{else}\ \mathsf{q}!\mathit{stop}\langle\rangle.\sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathbf{0} \\ \mathsf{p}?\mathit{error}(y).\mathbf{0} \end{Bmatrix}$$

$$h_1 \cdot (\mathsf{q}_1, \ell_1(\mathsf{v}_1)) \cdot (\mathsf{q}_2, \ell_2(\mathsf{v}_2)) \cdot h_2 \equiv h_1 \cdot (\mathsf{q}_2, \ell_2(\mathsf{v}_2)) \cdot (\mathsf{q}_1, \ell_1(\mathsf{v}_1)) \cdot h_2 \text{ if } \mathsf{q}_1 \neq \mathsf{q}_2$$
$$\varnothing \cdot h \equiv h \qquad h \cdot \varnothing \equiv h \qquad h_1 \cdot (h_2 \cdot h_3) \equiv (h_1 \cdot h_2) \cdot h_3$$

$$\mu X.P \equiv P\{\mu X.P/X\}$$

$$\mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing \mid \mathcal{M} \equiv \mathcal{M} \quad \mathcal{M}_1 \mid \mathcal{M}_2 \equiv \mathcal{M}_2 \mid \mathcal{M}_1 \quad (\mathcal{M}_1 \mid \mathcal{M}_2) \mid \mathcal{M}_3 \equiv \mathcal{M}_1 \mid (\mathcal{M}_2 \mid \mathcal{M}_3)$$
$$P \equiv Q \text{ and } h_1 \equiv h_2 \Rightarrow \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h_1 \mid \mathcal{M} \equiv \mathsf{p} \triangleleft Q \mid \mathsf{p} \triangleleft h_2 \mid \mathcal{M}$$

Table 6 Structural congruence rules for queues, processes and sessions

can reduce since r internally decides whether to reduce, either to

$$\mathsf{r} \triangleleft \sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathbf{0} \\ \mathsf{p}?\mathit{error}(y).\mathbf{0} \end{Bmatrix} \mid \mathsf{r} \triangleleft (\mathsf{q}, \mathit{cont}(43)) \;\mid\; \mathsf{p} \triangleleft \cdots \;\mid\; \mathsf{q} \triangleleft \cdots \;\mid\; \ldots$$

or to

$$\mathsf{r} \triangleleft \sum \begin{Bmatrix} \mathsf{p}?\mathit{success}(x).\mathbf{0} \\ \mathsf{p}?\mathit{error}(y).\mathbf{0} \end{Bmatrix} \mid \mathsf{r} \triangleleft (\mathsf{q}, \mathit{stop}()) \;\mid\; \mathsf{p} \triangleleft \cdots \;\mid\; \mathsf{q} \triangleleft \cdots \;\mid\; \ldots$$

by the rule [R-SEND] and then q to continue the reduction with the rule [R-RCV]. Further, r reduces to $\mathbf{0}$ if it receives the *success*/*error* message from p. This kind of desirable optimisation will be achieved by means of subtyping in the following section.

B Appendix of Section 3

We say that a binary relation $\mathcal{R}$ over single-input-single-output trees is a type simulation if it complies with the rules given in Definition 3.2, i.e., if for every $(\mathsf{W}_1, \mathsf{W}_2) \in \mathcal{R}$ there is a rule with $\mathsf{W}_1 \lesssim \mathsf{W}_2$ in its conclusion and it holds $(\mathsf{W}'_1, \mathsf{W}'_2) \in \mathcal{R}$ if $\mathsf{W}'_1 \lesssim \mathsf{W}'_2$ is in the premise of the rule. It is required that all other premises hold as well.

▶ **Example B.1.** In this example, we present some basic cases when inputs and/or outputs can be swapped. We give three simple examples and prove that $\mathbb{W}_1 \lesssim \mathbb{W}_2$ by constructing in each case a suitable tree simulation for $\mathcal{T}(\mathbb{W}_1) \lesssim \mathcal{T}(\mathbb{W}_2)$ (denoted as $\mathsf{W}_1 \lesssim \mathsf{W}_2$):

- if $\mathbb{W}_1 = \mu\mathbf{t}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{q}?\ell'(\mathsf{S}').\mathbf{t}$, $\mathbb{W}_2 = \mu\mathbf{t}.\mathsf{q}?\ell'(\mathsf{S}').\mathsf{p}!\ell(\mathsf{S}).\mathbf{t}$, the tree simulation is $\mathcal{R} = \{(\mathsf{W}_1, \mathsf{W}_2), (\mathsf{q}?\ell'(\mathsf{S}').\mathsf{W}_1, \mathsf{q}?\ell'(\mathsf{S}').\mathsf{W}_2)\}$
- if $\mathbb{W}_1 = \mu\mathbf{t}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{p}?\ell'(\mathsf{S}').\mathbf{t}$, $\mathbb{W}_2 = \mu\mathbf{t}.\mathsf{p}?\ell'(\mathsf{S}').\mathsf{p}!\ell(\mathsf{S}).\mathbf{t}$, the tree simulation is $\mathcal{R} = \{(\mathsf{W}_1, \mathsf{W}_2), (\mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}_1, \mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}_2)\}$

▶ **Lemma B.2.** *If $\mathcal{R} \subseteq \lesssim$ is a tree simulation and $(\mathsf{W}, \mathsf{W}') \in \mathcal{R}$ then $\mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathsf{W}')$.*

▶ **Lemma B.3.** **1.** *If $\mathcal{B}^{(\mathsf{p})} \neq \mathcal{B}^{(\mathsf{q})}$ and there are W_1 and W_2 such that $\mathsf{W} = \mathcal{B}^{(\mathsf{p})}.\mathsf{W}_1$ and $\mathsf{W} = \mathcal{B}^{(\mathsf{q})}.\mathsf{W}_2$, then one of the following holds:*

a. $\mathsf{W} = \mathcal{B}^{(\mathsf{p})}.\mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}_2$*, where* $\mathcal{B}^{(\mathsf{q})} = \mathcal{B}^{(\mathsf{p})}.\mathcal{B}_1^{(\mathsf{q})}$ *and* $\mathsf{W}_1 = \mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}_2$*;*

b. $\mathsf{W} = \mathcal{B}^{(\mathsf{q})}.\mathcal{B}_1^{(\mathsf{p})}.\mathsf{W}_1$*, where* $\mathcal{B}^{(\mathsf{p})} = \mathcal{B}^{(\mathsf{q})}.\mathcal{B}_1^{(\mathsf{p})}$ *and* $\mathsf{W}_2 = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{W}_1$*.*

2. *If $\mathcal{A}^{(\mathsf{p})} \neq \mathcal{A}^{(\mathsf{q})}$ and there are W_1 and W_2 such that $\mathsf{W} = \mathcal{A}^{(\mathsf{p})}.\mathsf{W}_1$ and $\mathsf{W} = \mathcal{A}^{(\mathsf{q})}.\mathsf{W}_2$, then one of the following holds:*

a. $\mathsf{W} = \mathcal{A}^{(\mathsf{p})}.\mathcal{A}_1^{(\mathsf{q})}.\mathsf{W}_2$*, where* $\mathcal{A}^{(\mathsf{q})} = \mathcal{A}^{(\mathsf{p})}.\mathcal{A}_1^{(\mathsf{q})}$ *and* $\mathsf{W}_1 = \mathcal{A}_1^{(\mathsf{q})}.\mathsf{W}_2$*;*

b. $\mathsf{W} = \mathcal{A}^{(\mathsf{q})}.\mathcal{A}_1^{(\mathsf{p})}.\mathsf{W}_1$*, where* $\mathcal{A}^{(\mathsf{p})} = \mathcal{A}^{(\mathsf{q})}.\mathcal{A}_1^{(\mathsf{p})}$ *and* $\mathsf{W}_2 = \mathcal{A}_1^{(\mathsf{p})}.\mathsf{W}_1$*.*

3. *If* $\mathsf{W} = \mathcal{B}^{(\mathsf{p})}.\mathsf{W}_1$ *and* $\mathsf{W} = \mathcal{A}^{(\mathsf{q})}.\mathsf{W}_2$, *for some* W_1 *and* W_2, *where* $\mathcal{B}^{(\mathsf{p})}$ *is not an I-sequence, then* $\mathsf{W} = \mathcal{A}^{(\mathsf{q})}.\mathcal{B}_1^{(\mathsf{p})}.\mathsf{W}_1$, *where* $\mathcal{B}^{(\mathsf{p})} = \mathcal{A}^{(\mathsf{q})}.\mathcal{B}_1^{(\mathsf{p})}$ *and* $\mathsf{W}_2 = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{W}_1$.

▶ **Lemma B.4.** *Let* $\mathcal{R} \subseteq \lesssim$ *be a tree simulation.*

1. *If* $(\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}') \in \mathcal{R}$ *then*

$$\mathsf{W}' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 \text{ or } \mathsf{W}' = \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1, \text{ where } \mathsf{S} \leq: \mathsf{S}_1.$$

2. *If* $(\mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}') \in \mathcal{R}$ *then*

$$\mathsf{W}' = \mathcal{A}_1^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}_1).\mathsf{W}_1 \text{ or } \mathsf{W}' = \mathsf{p}?\ell(\mathsf{S}_1).\mathsf{W}_1, \text{ where } \mathsf{S}_1 \leq: \mathsf{S}.$$

Proof. **1.** The proof is by induction on the structure of context $\mathcal{B}^{(\mathsf{p})}$. The basis step is included in the induction step if we notice that the lemma holds by definition of $\lesssim$ for $(\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}') \in \mathcal{R}$. For the inductive step, we distinguish two cases:

a. Let $\mathcal{B}^{(\mathsf{p})} = \mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}$ and $\mathsf{p} \neq \mathsf{q}$.
Then, $(\mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}') \in \mathcal{R}$ could be derived by rules [ref-out] or [ref-$\mathcal{B}$].

i. If the rule applied is [ref-out] then we have

$$\mathsf{W}' = \mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}'' \text{ and } \mathsf{S}' \leq: \mathsf{S}'' \text{ and } (\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}'') \in \mathcal{R}.$$

By induction hypothesis

$$\mathsf{W}'' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 \text{ or } \mathsf{W}'' = \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1, \text{ where } \mathsf{S} \leq: \mathsf{S}_1.$$

Both cases follow directly.

ii. If the rule applied is [ref-$\mathcal{B}$] then we have

$$\mathsf{W}' = \mathcal{B}_1^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}'' \text{ and } \mathsf{S}' \leq: \mathsf{S}'' \text{ and } (\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}'') \in \mathcal{R}.$$

By induction hypothesis one of the following holds:

A. If $\mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$ then,

- either $\mathcal{B}_1^{(\mathsf{q})} = \mathcal{B}_1^{(\mathsf{p})}$ and $\mathsf{W}'' = \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$ and
  $\mathsf{W}' = \mathcal{B}_1^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$,
- or, if $\mathcal{B}_1^{(\mathsf{q})} \neq \mathcal{B}_1^{(\mathsf{p})}$, by Lemma B.3
  $\mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathcal{B}_2^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$, where $\mathcal{B}_2^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})} = \mathcal{B}_1^{(\mathsf{p})}$ and
  $\mathsf{W}' = \mathcal{B}_2^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$, or
  $\mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_3^{(\mathsf{q})}.\mathsf{W}''$, where $\mathcal{B}_1^{(\mathsf{q})} = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_3^{(\mathsf{q})}$ and
  $\mathsf{W}' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_3^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}''$.

B. If $\mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$, where $\mathsf{S} \leq: \mathsf{S}_1$, then, either $\mathcal{B}_1^{(\mathsf{q})} = \mathsf{p}!\ell(\mathsf{S}_1)$ and $\mathsf{W}'' = \mathsf{W}_1$, or by Lemma B.3

$$\mathcal{B}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_2^{(\mathsf{q})}.\mathsf{W}'', \text{ where } \mathcal{B}_1^{(\mathsf{q})} = \mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_2^{(\mathsf{q})}.$$

In both cases the proof follows directly.

b. Let $\mathcal{B}^{(\mathsf{p})} = \mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}$. Then, $(\mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}') \in \mathcal{R}$ could be derived by rules [ref-in] or [ref-$\mathcal{A}$].

i. If [ref-in] is applied then we get this case by the same reasoning as in the first part of the proof.

ii. If the rule applied is [REF-$\mathcal{A}$] then

$$\mathsf{W}' = \mathcal{A}_1^{(\mathsf{q})}.\mathsf{q}?\ell'(\mathsf{S}'').\mathsf{W}'' \text{ and } \mathsf{S}'' \leq: \mathsf{S}' \text{ and } (\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{A}_1^{(\mathsf{q})}.\mathsf{W}'') \in \mathcal{R}.$$

By induction hypothesis we have only the case $\mathcal{A}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$, where $\mathsf{S} \leq: \mathsf{S}_1$, since the case $\mathcal{A}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$ is not possible. By Lemma B.3 we have $\mathcal{A}_1^{(\mathsf{q})}.\mathsf{W}'' = \mathcal{A}_1^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1$, where $\mathcal{A}_1^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})} = \mathcal{B}_1^{(\mathsf{p})}$.

2. The proof is by induction on the structure of context $\mathcal{A}^{(\mathsf{p})}$ and follows by similar reasoning. ◀

By definition of $\lesssim$ we consider related pairs by peeling left-hand side trees from left to right, i.e. by matching and eliminating always the leftmost tree. The proof of transitivity requires also to consider actions that are somewhere in the middle of the left-hand side tree. For that purpose, we associate each binary relation $\mathcal{R}$ over SISO trees with its extension $\mathcal{R}^+$ as follows:

$$\begin{array}{rcll}
\mathcal{R}^+ & = & \mathcal{R} & \cup \ \{(\mathcal{B}^{(\mathsf{p})}.\mathsf{W}, \mathcal{B}_1^{(\mathsf{p})}.\mathsf{W}_1) : (\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}\} \\
& & & \cup \ \{(\mathcal{A}^{(\mathsf{p})}.\mathsf{W}, \mathcal{A}_1^{(\mathsf{p})}.\mathsf{W}_1) : (\mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathsf{W}, \mathcal{A}_1^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}\} \\
& & & \cup \ \{(\mathcal{B}^{(\mathsf{p})}.\mathsf{W}, \mathsf{W}_1) : (\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}\} \\
& & & \cup \ \{(\mathcal{A}^{(\mathsf{p})}.\mathsf{W}, \mathsf{W}_1) : (\mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathsf{W}, \mathsf{p}?\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}\}.
\end{array}$$

▶ **Lemma B.5.** *If $\mathcal{R} \subseteq \lesssim$ is a tree simulation then $\mathcal{R}^+$ is a tree simulation.*

Proof. We discuss some interesting cases. Let $(\mathcal{B}^{(\mathsf{p})}.\mathsf{W}, \mathcal{B}_1^{(\mathsf{p})}.\mathsf{W}_1) \in \mathcal{R}^+ \setminus \mathcal{R}$.

1. If $\mathcal{B}^{(\mathsf{p})} = \mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}$, then,

$$(\mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}$$

could be derived by two rules:

a. if the rule applied is [REF-OUT] then

$$\mathcal{B}_1^{(\mathsf{p})} = \mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_3^{(\mathsf{p})} \text{ and } (\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}, \text{ where } \mathsf{S}' \leq: \mathsf{S}''.$$

By definition of $\mathcal{R}^+$, we get $(\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}, \mathcal{B}_3^{(\mathsf{p})}.\mathsf{W}_1) \in \mathcal{R}^+$.

b. if the rule applied is [REF-$\mathcal{B}$] then $\mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 = \mathcal{B}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}''$ and by Lemma B.3 we distinguish two cases

i.

$$\mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 = \mathcal{B}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1, \text{ where } \mathcal{B}_1^{(\mathsf{p})} = \mathcal{B}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_3^{(\mathsf{p})}$$

Then, by [REF-$\mathcal{B}$] we get $(\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{B}^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}$ and $\mathsf{S}' \leq: \mathsf{S}''$. By definition of $\mathcal{R}^+$, we have $(\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}, \mathcal{B}^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{W}_1) \in \mathcal{R}^+$.

ii.

$$\mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_2^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}'', \text{ where } \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathcal{B}_2^{(\mathsf{q})} = \mathcal{B}^{(\mathsf{q})}.$$

Then, we apply similar reasoning as in the first case.

2. if $\mathcal{B}^{(\mathsf{p})} = \mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}$ and $(\mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}$, then by Lemma B.4 we have $\mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 = \mathcal{A}_1^{(\mathsf{q})}.\mathsf{q}?\ell'(\mathsf{S}'').\mathsf{W}''$ and $\mathsf{S}'' \leq: \mathsf{S}'$. By Lemma B.3

$$\mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1 = \mathcal{A}_1^{(\mathsf{q})}.\mathsf{q}?\ell'(\mathsf{S}'').\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1, \text{ where } \mathcal{B}_1^{(\mathsf{p})} = \mathcal{A}_1^{(\mathsf{q})}.\mathsf{q}?\ell'(\mathsf{S}'').\mathcal{B}_3^{(\mathsf{p})}.$$

Then, by [REF-$\mathcal{A}$] we get $(\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathcal{A}_1^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}_1) \in \mathcal{R}$, and consequently $(\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}, \mathcal{A}_1^{(\mathsf{q})}.\mathcal{B}_3^{(\mathsf{p})}.\mathsf{W}_1) \in \mathcal{R}^+$.

◀

▶ **Lemma B.6.** *The refinement relation $\lesssim$ over SISO trees is reflexive and transitive.*

Proof. Reflexivity is straightforward: any SISO tree W is related to itself by a coinductive derivation which only uses rules [REF-IN], [REF-OUT], and [REF-END] in Def. 3.2.

We now focus on the proof of transitivity. If $\mathsf{W}_1 \lesssim \mathsf{W}_2$ and $\mathsf{W}_2 \lesssim \mathsf{W}_3$ then there are tree simulations $\mathcal{R}_1$ and $\mathcal{R}_2$ such that $(\mathsf{W}_1, \mathsf{W}_2) \in \mathcal{R}_1$ and $(\mathsf{W}_2, \mathsf{W}_3) \in \mathcal{R}_2$. We shall prove that relation

$$\mathcal{R} = \mathcal{R}_1 \circ \mathcal{R}_2^+$$

is a tree simulation that contains $(\mathsf{W}_1, \mathsf{W}_3)$. It follows directly from definition of $\mathcal{R}^+$ that $(\mathsf{W}_1, \mathsf{W}_3) \in \mathcal{R}$ and it remains to prove that $\mathcal{R}$ is a tree simulation. Assuming that $(\mathsf{W}_1', \mathsf{W}_3') \in \mathcal{R}$ we consider the following possible cases for W_1' :

1. $\mathsf{W}_1' = \mathsf{end}$: By definition of $\mathcal{R}$ there is W_2' such that $(\mathsf{end}, \mathsf{W}_2') \in \mathcal{R}_1$ and $(\mathsf{W}_2', \mathsf{W}_3') \in \mathcal{R}_2^+$. Since $\mathcal{R}_1$ and $\mathcal{R}_2^+$ are tree simulations, it holds by [REF-END] that $\mathsf{W}_2' = \mathsf{end}$ and also $\mathsf{W}_3' = \mathsf{end}$.

2. $\mathsf{W}_1' = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}$: By definition of $\mathcal{R}$ there is W_2' such that $(\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}, \mathsf{W}_2') \in \mathcal{R}_1$ and $(\mathsf{W}_2', \mathsf{W}_3') \in \mathcal{R}_2^+$. Since $\mathcal{R}_1$ is tree simulation, using definition of $\mathcal{B}$-sequence and applying [REF-OUT] or [REF-$\mathcal{B}$], we get three possibilities for W_2':

a. $\mathsf{W}_2' = \mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}'$ with $\mathsf{S} \leq: \mathsf{S}_1$ and $(\mathsf{W}, \mathsf{W}') \in \mathcal{R}_1$: Since $\mathcal{R}_2^+$ is tree simulation and $(\mathsf{W}_2', \mathsf{W}_3') \in \mathcal{R}_2^+$, there are two possibilities for W_3' (by [REF-OUT] or [REF-$\mathcal{B}$]):

i. $\mathsf{W}_3' = \mathsf{p}!\ell(\mathsf{S}_2).\mathsf{W}''$ and $\mathsf{S}_1 \leq: \mathsf{S}_2$ and $(\mathsf{W}', \mathsf{W}'') \in \mathcal{R}_2^+$: Then, by transitivity of $\leq:$ and definition of $\mathcal{R}$ we get

$$\mathsf{S} \leq: \mathsf{S}_2 \text{ and } (\mathsf{W}, \mathsf{W}'') \in \mathcal{R}_1 \circ \mathcal{R}_2^+ = \mathcal{R}.$$

ii. $\mathsf{W}_3' = \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_2).\mathsf{W}''$ and $\mathsf{S}_1 \leq: \mathsf{S}_2$ and $(\mathsf{W}', \mathcal{B}^{(\mathsf{p})}.\mathsf{W}'') \in \mathcal{R}_2$: Then, by transitivity of $\leq:$ and definition of $\mathcal{R}$ we get

$$\mathsf{S} \leq: \mathsf{S}_2 \text{ and } (\mathsf{W}, \mathcal{B}^{(\mathsf{p})}.\mathsf{W}'') \in \mathcal{R}_1 \circ \mathcal{R}_2^+ = \mathcal{R}.$$

b. $\mathsf{W}_2' = \mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}'$ with $\mathsf{S} \leq: \mathsf{S}_1$ and

$$(\mathsf{W}, \mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{W}') \in \mathcal{R}_1 \text{ and } \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{W}') : \tag{5}$$

Since $(\mathsf{W}_2', \mathsf{W}_3') \in \mathcal{R}_2^+$, we have two cases (by [REF-OUT] or [REF-$\mathcal{B}$]):

i. $\mathsf{W}_3' = \mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}''$ with $\mathsf{S}' \leq: \mathsf{S}''$ and $(\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}', \mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{W}'') \in \mathcal{R}_2^+$. By Lemma B.4, we have that

$$\mathsf{W}_3' = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_2).\mathsf{W}''' \text{ and } \mathsf{S}_1 \leq: \mathsf{S}_2.$$

By Lemma B.3, there are two possibilities:

A. $\mathsf{W}_3' = \mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_2).\mathsf{W}'''$ and $\mathsf{p}! \notin \mathtt{act}(\mathcal{B}_{11}^{(\mathsf{q})})$:

$$\begin{aligned}&(\mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{W}', \mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}''') \in \mathcal{R}_2^+ \text{ and}\\ &\mathtt{act}(\mathsf{q}!\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{W}') = \mathtt{act}(\mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}''')\end{aligned} \tag{6}$$

Hence, we conclude from (5) and (6) that

$$(\mathsf{W}, \mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}''') \in \mathcal{R}_1 \circ \mathcal{R}_2^+ \text{ and } \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathcal{B}_{11}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathcal{B}_2^{(\mathsf{p})}.\mathsf{W}''').$$

B. $\mathsf{W}'_3 = \mathcal{B}_1^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_2).\mathcal{B}_{12}^{(\mathsf{q})}.\mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}''$ and $\mathsf{q}! \notin \mathtt{act}(\mathcal{B}_1^{(\mathsf{p})})$: The proof is similar to the previous case.

ii. $\mathsf{W}'_3 = \mathsf{q}!\ell'(\mathsf{S}'').\mathsf{W}''$ with $\mathsf{S}' \leq: \mathsf{S}''$ and $(\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}', \mathsf{W}'') \in \mathcal{R}_2^+$.

c. $\mathsf{W}'_2 = \mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}'$ with $\mathsf{S} \leq: \mathsf{S}_1$ and

$$(\mathsf{W}, \mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{W}') \in \mathcal{R}_1 \text{ and } \mathtt{act}(\mathsf{W}) = \mathtt{act}(\mathsf{q}?\ell'(\mathsf{S}').\mathcal{B}^{(\mathsf{p})}.\mathsf{W}') : \tag{7}$$

Since $(\mathsf{W}'_2, \mathsf{W}'_3) \in \mathcal{R}_2^+$, we have two cases (by [REF-IN] and [REF-$\mathcal{B}$]):

i. $\mathsf{W}'_3 = \mathsf{q}?\ell'(\mathsf{S}'').\mathsf{W}''$ and $\mathsf{S}'' \leq: \mathsf{S}'$ and $(\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}', \mathsf{W}'') \in \mathcal{R}_2^+$):

ii. $\mathsf{W}'_3 = \mathcal{A}^{(\mathsf{q})}.\mathsf{q}?\ell'(\mathsf{S}'').\mathsf{W}''$ and $\mathsf{S}'' \leq: \mathsf{S}'$ and $(\mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell(\mathsf{S}_1).\mathsf{W}', \mathcal{A}^{(\mathsf{q})}.\mathsf{W}'') \in \mathcal{R}_2^+$):

3. $\mathsf{W}'_1 = \mathsf{p}?\ell(\mathsf{S}).\mathsf{W}$: The proof in this case follows similarly.

◀

▶ **Lemma B.7.** *Let* $[\![\mathsf{T}]\!]_{\mathsf{SI}} = \{\mathsf{V}_j : j \in J\}$ *and let* $Y = \{\mathsf{W}_j : j \in J\}$, *where* $\mathsf{W}_j \in [\![V_j]\!]_{\mathsf{SO}}$ *for* $j \in J$. *Then, there is* $\mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}}$ *such that* $[\![\mathsf{U}]\!]_{\mathsf{SI}} \subseteq Y$.

Proof. In this proof we consider coinductive interpretation of the definition of tree T.

1. $\mathsf{T} = \mathtt{end}$: Since $[\![\mathsf{T}]\!]_{\mathsf{SI}} = \{\mathtt{end}\}$, by selecting $U = Y = \mathtt{end}$ the case follows.
2. $\mathsf{T} = \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{T}_i$: Assume $[\![\mathsf{T}]\!]_{\mathsf{SI}} = \{\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{V}'_{j_i} : \mathsf{V}'_{j_i} \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}, j_i \in J_i, i \in I\}$. Then,
$$Y = \{\mathsf{W}_j \in [\![\mathsf{V}_j]\!]_{\mathsf{SO}} : j \in J\} = \{W_{j_i} = \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{W}'_{j_i} : \mathsf{W}'_{j_i} \in [\![\mathsf{V}_i]\!]_{\mathsf{SO}}, j_i \in J_i, i \in I\}.$$
Since for $i \in I$ we have $[\![\mathsf{T}_i]\!]_{\mathsf{SI}} = \{\mathsf{V}'_{j_i} : j_i \in J_i\}$ and $Y_i = \{\mathsf{W}'_{j_i} \in [\![\mathsf{V}'_{j_i}]\!]_{\mathsf{SI}} : j_i \in J_i\}$, we can apply coinductive hypothesis and obtain $\mathsf{U}_i \in [\![\mathsf{T}_i]\!]_{\mathsf{SO}}$, such that $[\![\mathsf{U}_i]\!]_{\mathsf{SI}} \subseteq Y_i$. Hence, for $\mathsf{U} = \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).U_i$ we have $\mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}}$ and $[\![\mathsf{U}]\!]_{\mathsf{SI}} \subseteq Y$.
3. $\mathsf{T} = \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{T}_i$: Assume $[\![\mathsf{T}]\!]_{\mathsf{SI}} = \{\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}'_{j_i} : \mathsf{V}'_{j_i} \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}, j_i \in J_i, i \in I\}$. Then,
$$Y = \{\mathsf{W}_j \in [\![\mathsf{V}_j]\!]_{\mathsf{SO}} : j \in J\} = \{\mathsf{W}_{k_i} = \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}'_{k_i} : \mathsf{W}'_{k_i} \in [\![\mathsf{V}_i]\!]_{\mathsf{SO}}, j_i \in K_i \subseteq J_i, i \in N \subseteq I\}$$
We claim that there is $i \in I$ such that for all $j_i \in J_i$ holds $\{\mathsf{p}!\ell_i(\mathsf{S}_i).[\![\mathsf{V}_{j_i}]\!]_{\mathsf{SO}}\} \cap Y \neq \emptyset$. To prove the claim let us assume the opposite: for all $i \in I$ there is $j_i \in J_i$ such that $\{\mathsf{p}!\ell_i(\mathsf{S}_i).[\![\mathsf{V}_{j_i}]\!]_{\mathsf{SO}}\} \cap Y = \emptyset$. For such j_i's, let us consider $\mathsf{V} = \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_{j_i}$. Since $\mathsf{V} \in [\![\mathsf{T}]\!]_{\mathsf{SI}}$ and $[\![\mathsf{V}]\!]_{\mathsf{SO}} \cap Y = \emptyset$ we obtain a contradiction with the definition of Y.
Let us now fix $i \in I$ for which the above claim holds. Let $Y' = \{\mathsf{W}'_{k_i} : \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}'_{k_i} \in Y\}$. For T_i we have $[\![\mathsf{T}_i]\!]_{\mathsf{SI}} = \{\mathsf{V}'_{j_i} : j_i \in J_i\}$ and for each $j_i \in J_i$ there is $\mathsf{W}'_{j_i} \in [\![\mathsf{V}'_{j_i}]\!]_{\mathsf{SO}}$ such that $\mathsf{W}'_{i_j} \in Y'$. Hence, we can apply coinductive hypothesis and get $\mathsf{U}' \in [\![\mathsf{T}_i]\!]_{\mathsf{SO}}$ for which $[\![\mathsf{U}']\!]_{\mathsf{SI}} \subseteq Y'$ holds. By taking $\mathsf{U} = \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{U}'$ we can conclude $\mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}}$ and $[\![\mathsf{U}]\!]_{\mathsf{SI}} \subseteq \{\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}'_{j_i} : \mathsf{W}'_{j_i} \in Y'\} \subseteq Y$.

◀

▶ **Lemma B.8.** *For any tree* T *we have*

$$\forall \mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}}\ \forall \mathsf{V} \in [\![\mathsf{T}]\!]_{\mathsf{SI}}\ \exists \mathsf{W} \quad \textit{such that} \quad \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}} \cap [\![\mathsf{V}]\!]_{\mathsf{SO}}.$$

Proof. By coinduction on the definition of tree T.

1. $\mathsf{T} = \mathtt{end}$: Since $[\![\mathsf{T}]\!]_{\mathsf{SO}} = [\![\mathsf{T}]\!]_{\mathsf{SI}} = \{\mathtt{end}\}$, the proof follows directly.

2. $\mathsf{T} = \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{T}_i$: Then,

$$\mathsf{U} \in \{\&_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}' : \mathsf{U}' \in [\![\mathsf{T}_i]\!]_{\mathsf{SO}}\}$$

$$\mathsf{V} \in \{\mathsf{p}?\ell(\mathsf{S}_i).\mathsf{V}' : \mathsf{V}' \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}, i \in I\}$$

By coinductive hypothesis for all $i \in I$ we have

$$\forall \mathsf{U}' \in [\![\mathsf{T}_i]\!]_{\mathsf{SO}}\ \forall \mathsf{V}' \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}\ \exists \mathsf{W}' \quad \text{such that} \quad \mathsf{W}' \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \cap [\![\mathsf{V}']\!]_{\mathsf{SO}}.$$

Thus, for all $\mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}}$, $i \in I$ and $\mathsf{V} \in \{\mathsf{p}?\ell(\mathsf{S}_i).\mathsf{V}' : \mathsf{V}' \in [\![\mathsf{T}_i]\!]_{\mathsf{SI}}\}$ we obtain there exists $\mathsf{W} = \mathsf{p}?\ell(\mathsf{S}_i).\mathsf{W}'$, such that

$$\mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}} \cap [\![\mathsf{V}]\!]_{\mathsf{SO}}$$

3. $\mathsf{T} = \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{T}_i$: Follows by a similar reasoning.

◀

▶ **Lemma B.9.** *The asynchronous subtyping relation $\leqslant$ over trees is reflexive and transitive.*

Proof. Reflexivity is straightforward from Lemma B.8 and reflexivity of $\lesssim$.

We now focus on the proof of transitivity. Assume that $\mathsf{T}_1 \leqslant \mathsf{T}_2$ and $\mathsf{T}_2 \leqslant \mathsf{T}_3$. From $\mathsf{T}_1 \leqslant \mathsf{T}_2$, by Definition 3.4, we have

$$\forall \mathsf{U}_1 \in [\![\mathsf{T}_1]\!]_{\mathsf{SO}}\ \forall \mathsf{V}_2 \in [\![\mathsf{T}_2]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_1 \in [\![\mathsf{U}_1]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_2 \in [\![\mathsf{V}_2]\!]_{\mathsf{SO}} \quad \mathsf{W}_1 \lesssim \mathsf{W}_2 \tag{8}$$

From $\mathsf{T}_2 \leqslant \mathsf{T}_3$, by Definition 3.4,

$$\forall \mathsf{U}_2 \in [\![\mathsf{T}_2]\!]_{\mathsf{SO}}\ \forall \mathsf{V}_3 \in [\![\mathsf{T}_3]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_2' \in [\![\mathsf{U}_2]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_3 \in [\![\mathsf{V}_3]\!]_{\mathsf{SO}} \quad \mathsf{W}_2' \lesssim \mathsf{W}_3 \tag{9}$$

Let us now fix one $\mathsf{U}_1 \in [\![\mathsf{T}_1]\!]_{\mathsf{SO}}$. By (8) we have that

$$\forall \mathsf{V}_2 \in [\![\mathsf{T}_2]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_2 \in [\![\mathsf{V}_2]\!]_{\mathsf{SO}}\ \exists \mathsf{W}_1 \in [\![\mathsf{U}_1]\!]_{\mathsf{SI}} \text{ such that } \mathsf{W}_1 \lesssim \mathsf{W}_2 \tag{10}$$

and let Y be the set of all such W_2's. By Lemma B.7, there exist $\mathsf{U} \in [\![\mathsf{T}_2]\!]_{\mathsf{SO}}$ such that $[\![\mathsf{U}]\!]_{\mathsf{SI}} \subseteq Y$. Now from (9) we have $\forall \mathsf{V}_3 \in [\![\mathsf{T}_3]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_2 \in [\![\mathsf{U}]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_3 \in [\![\mathsf{V}_3]\!]_{\mathsf{SO}}$ such that $\mathsf{W}_2 \lesssim \mathsf{W}_3$. Then, we conclude by transitivity of $\lesssim$ that

$$\forall \mathsf{U}_1 \in [\![\mathsf{T}_1]\!]_{\mathsf{SO}}\ \forall \mathsf{V}_3 \in [\![\mathsf{T}_3]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_1 \in [\![\mathsf{U}_1]\!]_{\mathsf{SI}}\ \exists \mathsf{W}_3 \in [\![\mathsf{V}_3]\!]_{\mathsf{SO}} \quad \mathsf{W}_1 \lesssim \mathsf{W}_3.$$

◀

B.1 Further Examples

We illustrate the refinement relation with an example.

▶ **Example B.10.** Consider $\mathbb{W}_1 = \mu\mathbf{t}.\mathsf{p}!\ell(\mathsf{S}).\mathsf{q}?\ell'(\mathsf{S}').\mathbf{t}$ and $\mathbb{W}_2 = \mu\mathbf{t}.\mathsf{q}?\ell'(\mathsf{S}').\mathsf{p}!\ell(\mathsf{S}).\mathbf{t}$. Their trees are related by the following coinductive derivation:

$$\cfrac{\cfrac{\mathcal{T}(\mathbb{W}_1) \lesssim \mathcal{T}(\mathbb{W}_2)}{\mathsf{q}?\ell'(\mathsf{S}').\mathcal{T}(\mathbb{W}_1) \lesssim \mathsf{q}?\ell'(\mathsf{S}').\mathcal{T}(\mathbb{W}_2)}\ [\text{REF-IN}]}{\mathcal{T}(\mathbb{W}_1) \lesssim \mathcal{T}(\mathbb{W}_2)}\ [\text{REF-}\mathcal{B}] \text{ with } \mathcal{B}^{(\mathsf{p})} = \mathsf{q}?\ell'(\mathsf{S}')$$

Next we give a simple asynchronous subtyping example.

▶ **Example B.11** (Asynchronous subtyping). Let $\mathbb{T} = \mathsf{p}!\ell_1(\mathsf{S}_1).\,\&\,\mathsf{q}?\begin{cases}\ell_3(\mathsf{S}_3).\mathtt{end}\\ \ell_4(\mathsf{S}_4).\mathtt{end}\end{cases}$ and $\mathbb{T}' = \bigoplus \mathsf{p}!\begin{cases}\ell_1(\mathsf{S}_1).\mathsf{q}?\ell_3(\mathsf{S}_3).\mathtt{end}\\ \ell_2(\mathsf{S}_2).\mathtt{end}\end{cases}$ We show that $\mathbb{T} \leqslant \mathbb{T}'$. Notice that $\mathbb{T}$ is a SO type and $\mathbb{T}'$ is a SI type: hence, by Def. 3.4, we only need to show that there are $\mathsf{W} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SI}}$ and $\mathsf{W}' \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SO}}$ such that $\mathsf{W} \lesssim \mathsf{W}'$. Since:

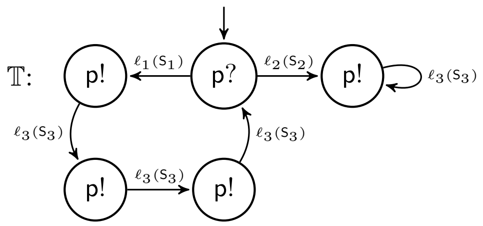
$$\begin{array}{rcl}[\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SI}} & = & \{\,\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{q}?\ell_3(\mathsf{S}_3).\mathtt{end},\ \mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{q}?\ell_4(\mathsf{S}_4).\mathtt{end}\,\}\\ [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SO}} & = & \{\,\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{q}?\ell_3(\mathsf{S}_3).\mathtt{end},\ \mathsf{p}!\ell_2(\mathsf{S}_2).\mathtt{end}\,\}\end{array}$$

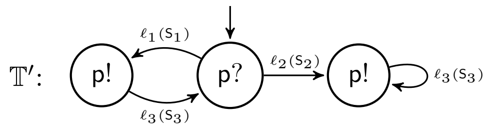
we have that $\mathsf{W} \lesssim \mathsf{W}'$ holds for $\mathsf{W} = \mathsf{W}' = \mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{q}?\ell_3(\mathsf{S}_3).\mathtt{end}$, by reflexivity of $\lesssim$.

Next we consider a complex example of asynchronous subtyping which contains branching, selection and recursion. This example is undecided by the algorithm in [3, 4] (it returns "unknown") but we can reason by our rules using the SISO decomposition method as demonstrated below.

▶ **Example B.12.** Consider the examples of session types M_1 and M_2 from [4, Example 3.21], here denoted by $\mathbb{T}$ and $\mathbb{T}'$, respectively, where

$$\mathbb{T} = \mu\mathbf{t}_1.\,\&\,\mathsf{p}?\begin{cases}\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_1\\ \ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2\end{cases} \qquad \mathbb{T}' = \mu\mathbf{t}_1.\,\&\,\mathsf{p}?\begin{cases}\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_1\\ \ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2\end{cases}$$





In [3, 4], if the algorithm returns "true" ("false"), then the considered types are (are not) in the subtyping relation. The algorithm can return "unknown", meaning that the algorithm cannot check whether the types are in the subtyping relation or not.

For the considered types, the algorithm in [3, 4] returns "unknown" and thus it can*not* check that $\mathbb{T} \leqslant \mathbb{T}'$, which is, according to the authors of [3, 4], due to the complex accumulation patterns of these types which cannot be recognised by their theory.

Here our approach comes into the picture and we demonstrate that the two decomposition functions into SO and SI trees are sufficiently fine-grained to recognise the complex structure of these types and to prove that $\mathbb{T} \leqslant \mathbb{T}'$. We show that types $\mathbb{T}$ and $\mathbb{T}'$ are in the subtyping relation $\mathbb{T} \leqslant \mathbb{T}'$, i.e., the corresponding session trees are in the subtyping relation $\mathcal{T}(\mathbb{T}) \leqslant \mathcal{T}(\mathbb{T}')$ by showing that

$$\forall \mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}' \in [\![\mathsf{T}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \qquad \mathsf{W} \lesssim \mathsf{W}'$$

where $\mathsf{T} = \mathcal{T}(\mathbb{T})$ and $\mathsf{T}' = \mathcal{T}(\mathbb{T}')$. For the sake of simplicity, let us use the following notations and abbreviations:

$$\begin{array}{ll}\mathsf{W}_1 = \mathcal{T}(\mu\mathbf{t}.\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}) & \mathsf{W}_2 = \mathsf{p}?\ell_2(\mathsf{S}_2).\,\mathcal{T}(\mu\mathbf{t}.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t})\\ \mathsf{W}_3 = \mathcal{T}(\mu\mathbf{t}.\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}) & \end{array}$$

$$\pi_1 \equiv \mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3) \qquad \pi_1^n \equiv \underbrace{\pi_1.\ldots.\pi_1.}_{\mathrm{n}}$$

$$\pi_3 \equiv \mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{p}!\ell_3(\mathsf{S}_3) \quad \pi_3^n \equiv \underbrace{\pi_3.\ldots.\pi_3.}_{\mathrm{n}}$$

Then $[\![\mathsf{T}]\!]_{\mathsf{SO}} = \{\mathsf{T}\}$ since T is a SO tree, whereas

$$[\![\mathsf{T}']\!]_{\mathsf{SI}} \;=\; \{\mathsf{W}_1, \mathsf{W}_2, \pi_1.\mathsf{W}_2, \pi_1^2.\mathsf{W}_2, \ldots, \pi_1^n.\mathsf{W}_2, \ldots\}$$

then $\mathsf{U} = \mathsf{T}$ and

$$[\![\mathsf{U}]\!]_{\mathsf{SI}} \;=\; \{\mathsf{W}_3, \mathsf{W}_2, \pi_3.\mathsf{W}_2, \pi_3^2.\mathsf{W}_2, \ldots, \pi_3^n.\mathsf{W}_2, \ldots\}$$

Notice that all $\mathsf{V}' \in [\![\mathsf{T}']\!]_{\mathsf{SI}}$ are SISO trees hence $[\![\mathsf{V}']\!]_{\mathsf{SO}} = \{\mathsf{V}'\}$ and $\mathsf{W}' = \mathsf{V}'$.

We show now that for all $\mathsf{W}' \in [\![\mathsf{T}']\!]_{\mathsf{SI}}$ there is a $\mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ such that $\mathsf{W} \lesssim \mathsf{W}'$ by showing:

1. $\mathsf{W}_3 \lesssim \mathsf{W}_1$
2. $\pi_3^n.\mathsf{W}_2 \lesssim \pi_1^n.\mathsf{W}_2$, $n \geq 0$

1. The simulation tree for $\mathsf{W}_3 \lesssim \mathsf{W}_1$ is

$$\begin{array}{rcl}
\mathcal{R} & = & \{(\mathsf{W}_3, \mathsf{W}_1), \\
 & & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\mathsf{W}_3, \mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_1), \\
 & & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\mathsf{W}_3, \mathsf{W}_1), \\
 & & (\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_3, \mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{W}_1), \\
 & & (\mathsf{W}_3, (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{W}_1), \\
 & & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\mathsf{W}_3, (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_1), \\
 & & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\mathsf{W}_3, (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{W}_1), \\
 & & (\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_3, (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{W}_1) \mid n \geq 1\}
\end{array}$$

where $(\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n \equiv \underbrace{\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1).\ldots\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1).}_{2n}$ $n \in \mathbb{N}$ and $(\mathsf{p}!\ell_3(\mathsf{S}_3))^i \equiv \underbrace{\mathsf{p}!\ell_3(\mathsf{S}_3).\ldots\mathsf{p}!\ell_3(\mathsf{S}_3)}_{\mathrm{i}}$ $i = 1, 2, 3$.

Proof. The trees W_3 and W_1 are related by the following coinductive derivations: for $n \geq 1$

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{W}_3 \lesssim (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^{n+1}.\mathsf{W}_1}{\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_3 \lesssim (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{W}_1}\;{\scriptstyle [\text{REF-}\mathcal{B}],\; \mathcal{B}^{(\mathsf{p})} = (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^{n+1}}}{(\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\mathsf{W}_3 \lesssim (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{W}_1}\;{\scriptstyle [\text{REF-}\mathcal{B}],\; \mathcal{B}^{(\mathsf{p})} = (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n\mathsf{p}?\ell_1(\mathsf{S}_1)}}{(\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\mathsf{W}_3 \lesssim (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_1}\;{\scriptstyle [\text{REF-}\mathcal{B}],\; \mathcal{B}^{(\mathsf{p})} = (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n}}{\mathsf{W}_3 \lesssim (\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1))^n.\mathsf{W}_1}\;{\scriptstyle [\text{REF-IN}]}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathsf{W}_3 \lesssim \mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{W}_1}{\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_3 \lesssim \mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{W}_1}\;{\scriptstyle [\text{REF-}\mathcal{B}],\; \mathcal{B}^{(\mathsf{p})} = \mathsf{p}?\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_1(\mathsf{S}_1)}}{(\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\mathsf{W}_3 \lesssim \mathsf{W}_1}\;{\scriptstyle [\text{REF-}\mathcal{B}],\; \mathcal{B}^{(\mathsf{p})} = \mathsf{p}?\ell_1(\mathsf{S}_1)}}{(\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\mathsf{W}_3 \lesssim \mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_1}\;{\scriptstyle [\text{REF-OUT}]}}{\mathsf{W}_3 \lesssim \mathsf{W}_1}\;{\scriptstyle [\text{REF-IN}]}$$

2. Case $\pi_3^n.\mathsf{W}_2 \lesssim \pi_1^n.\mathsf{W}_2$, $n \geq 0$.
   If $n = 0$, then $\mathsf{W}_2 \lesssim \mathsf{W}_2$ holds by reflexivity of $\lesssim$, Lemma B.6.
   In case $n > 0$, we first show that

$$\pi_3^n.\mathsf{W}_2 \lesssim \pi_1.\pi_3^{n-1}.\mathsf{W}_2 \tag{11}$$

The tree simulation is

$$
\begin{array}{rcl}
\mathcal{R} & = & \{(\pi_3^n.\mathsf{W}_2, \pi_1.\pi_3^{n-1}.\mathsf{W}_2), \\
& & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\pi_3^{n-1}.\mathsf{W}_2, \mathsf{p}!\ell_3(\mathsf{S}_3).\pi_3^{n-1}.\mathsf{W}_2), \\
& & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\pi_3^{n-1}.\mathsf{W}_2, \pi_3^{n-1}.\mathsf{W}_2), \\
& & (\mathsf{p}!\ell_3(\mathsf{S}_3).\pi_3^{n-1}.\mathsf{W}_2, \pi_1.\mathsf{p}!\ell_3(\mathsf{S}_3).\pi_3^{n-2}.\mathsf{W}_2), \\
& & (\pi_3^{n-1}.\mathsf{W}_2, \pi_1.\pi_3^{n-2}.\mathsf{W}_2), \\
& & \vdots \\
& & (\pi_3.\mathsf{W}_2, \pi_1.\mathsf{W}_2), \\
& & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\mathsf{W}_2, \mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_2), \\
& & ((\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\mathsf{W}_2, \mathsf{W}_2), \\
& & (\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_2, \mathsf{W}_2), \\
& & (\mathsf{W}_2, \mathsf{W}_2)\}
\end{array}
$$

The refinement $\pi_3^n.\mathsf{W}_2 \lesssim \pi_1.\pi_3^{n-1}.\mathsf{W}_2$ is derived by the following coinductive derivation

$$
\begin{array}{cl}
\mathsf{W}_2 \lesssim \mathsf{W}_2 & \\ \hline\hline
\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_2 \lesssim \mathsf{W}_2 & [\text{REF-}\mathcal{B}],\ \mathcal{B}^{(\mathsf{p})} = \mathsf{p}?\ell_1(\mathsf{S}_1) \\ \hline\hline
(\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\mathsf{W}_2 \lesssim \mathsf{W}_2 & [\text{REF-}\mathcal{B}],\ \mathcal{B}^{(\mathsf{p})} = \mathsf{p}?\ell_1(\mathsf{S}_1) \\ \hline\hline
(\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\mathsf{W}_2 \lesssim \mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_2 & [\text{REF-OUT}] \\ \hline\hline
\pi_3.\mathsf{W}_2 \lesssim \pi_1.\mathsf{W}_2 & [\text{REF-IN}] \\ \hline\hline
\mathsf{p}!\ell_3(\mathsf{S}_3).\pi_3.\mathsf{W}_2 \lesssim \pi_1.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathsf{W}_2 & [\text{REF-}\mathcal{B}],\ \mathcal{B}^{(\mathsf{p})} = \mathsf{p}?\ell_1(\mathsf{S}_1) \\ \hline\hline
(\mathsf{p}!\ell_3(\mathsf{S}_3))^2.\pi_3.\mathsf{W}_2 \lesssim \pi_3.\mathsf{W}_2 & [\text{REF-}\mathcal{B}],\ \mathcal{B}^{(\mathsf{p})} = \mathsf{p}?\ell_1(\mathsf{S}_1) \\ \hline\hline
(\mathsf{p}!\ell_3(\mathsf{S}_3))^3.\pi_3.\mathsf{W}_2 \lesssim \mathsf{p}!\ell_3(\mathsf{S}_3).\pi_3.\mathsf{W}_2 & [\text{REF-OUT}] \\ \hline\hline
\pi_3^2.\mathsf{W}_2 \lesssim \pi_1.\pi_3.\mathsf{W}_2 & [\text{REF-IN}] \\ \hline\hline
\vdots & \\ \hline\hline
\pi_3^n.\mathsf{W}_2 \lesssim \pi_1.\pi_3^{n-1}.\mathsf{W}_2 & [\text{REF-IN}]
\end{array}
$$

This coinductive derivation proves all refinements

$$\pi_3^{n-k}.\mathsf{W}_2 \lesssim \pi_1.\pi_3^{n-k-1}.\mathsf{W}_2, \quad k = 0, \dots, n-1.$$

By k consecutive application first of [REF-OUT] and then [REF-IN], it holds that

$$\pi_1^k \pi_3^{n-k}.\mathsf{W}_2 \lesssim \pi_1^{k+1}.\pi_3^{n-k-1}.\mathsf{W}_2, \quad k = 0, \dots, n-1.$$

which means that

$$
\begin{array}{l}
\pi_3^n.\mathsf{W}_2 \lesssim \pi_1.\pi_3^{n-1}.\mathsf{W}_2, \\
\pi_1.\pi_3^{n-1}.\mathsf{W}_2 \lesssim \pi_1^2.\pi_3^{n-2}.\mathsf{W}_2, \\
\vdots \\
\pi_1^k \pi_3^{n-k}.\mathsf{W}_2 \lesssim \pi_1^{k+1}.\pi_3^{n-k-1}.\mathsf{W}_2, \\
\vdots \\
\pi_1^{n-1}\pi_3.\mathsf{W}_2 \lesssim \pi_1^n.\mathsf{W}_2
\end{array}
$$

then $\pi_3^n.\mathsf{W}_2 \lesssim \pi_1^n.\mathsf{W}_2$ follows by transitivity of $\lesssim$, Lemma B.6.

This concludes the proof that $\mathbb{T} \leqslant \mathbb{T}'$, which could not be given by the answer (Yes or No) by the algorithm in [4, Example 3.21].

$$\Theta \vdash \mathsf{n} : \mathtt{nat} \qquad \Theta \vdash \mathsf{i} : \mathtt{int} \qquad \Theta \vdash \mathsf{true} : \mathtt{bool} \qquad \Theta \vdash \mathsf{false} : \mathtt{bool} \qquad \Theta, x : \mathsf{S} \vdash x : \mathsf{S}$$

$$\frac{\Theta \vdash \mathsf{e} : \mathtt{nat}}{\Theta \vdash \mathsf{succ(e)} : \mathtt{nat}} \qquad \frac{\Theta \vdash \mathsf{e} : \mathtt{int}}{\Theta \vdash \mathsf{inv(e)} : \mathtt{int}} \qquad \frac{\Theta \vdash \mathsf{e} : \mathtt{bool}}{\Theta \vdash \neg\mathsf{e} : \mathtt{bool}} \qquad \frac{\Theta \vdash \mathsf{e}_1 : \mathtt{int} \quad \Theta \vdash \mathsf{e}_2 : \mathtt{int}}{\Theta \vdash \mathsf{e}_1 = \mathsf{e}_2 : \mathtt{bool}}$$

$$\frac{\Theta \vdash \mathsf{e} : \mathsf{S} \quad \mathsf{S} \leq: \mathsf{S}'}{\Theta \vdash \mathsf{e} : \mathsf{S}'}$$

Table 7 Typing rules for expressions.

C Appendix of Section 4

C.1 Typing Rules for Expression

Table 7 lists the typing rules for expressions.

C.2 Proof of Theorem C.13

The main aim of this section is to prove Theorem C.13.

▶ **Lemma C.1.** *If Γ is live and $\Gamma \equiv \Gamma'$ then Γ' is also live.*

Proof. From the the definition of $\Gamma \equiv \Gamma'$, Γ and Γ' perform exactly the same reductions (i.e., they are strongly bisimilar). Therefore, the result follows by Definition 4.4. ◀

▶ **Proposition C.2.** *If Γ is live and $\Gamma \longrightarrow \Gamma'$, then, Γ' is live.*

Proof. Direct consequence of Def. 4.4. ◀

In the rest of this section, we will use the following alternative formulation of Def. 4.4, that is more handy to construct proofs by coinduction.

▶ **Definition C.3** (Coinductive liveness)**.** φ is a p-liveness property *iff, whenever $\varphi(\Gamma)$:*

- [LP&] $\Gamma(\mathsf{p}) = (\sigma_{\mathsf{p}}, \mathsf{T}_{\mathsf{p}})$ *with* $\mathsf{T}_{\mathsf{p}} = \&_{i\in I}\, \mathsf{q}?\ell_i(\mathsf{S}_i).\mathsf{T}_i$ *implies that, for all fair paths* $(\Gamma_j)_{j\in J}$ *such that* $\Gamma_0 = \Gamma$, $\exists h \in J, k \in I$ *such that* $\Gamma \longrightarrow^* \Gamma_{h-1} \longrightarrow \Gamma_h$, *with:*
  1. $\Gamma_{h-1}(\mathsf{p}) = (\sigma_{\mathsf{p}}, \mathsf{T}_{\mathsf{p}})$ *and* $\Gamma_{h-1}(\mathsf{q}) = (\mathsf{p}!\ell_k(\mathsf{S}) \cdot \sigma', \mathsf{T}_{\mathsf{q}})$*;*
  2. $\Gamma_h(\mathsf{p}) = (\sigma_{\mathsf{p}}, \mathsf{T}_k)$ *and* $\Gamma_h(\mathsf{q}) = (\sigma', \mathsf{T}_{\mathsf{q}})$*;*
- [LP⊕] $\Gamma(\mathsf{p}) = (\mathsf{q}!\ell(\mathsf{S}) \cdot \sigma, \mathsf{T}_{\mathsf{p}})$ *implies that, for all fair paths* $(\Gamma_j)_{j\in J}$ *such that* $\Gamma_0 = \Gamma$, $\exists h \in J, k \in I$ *such that* $\Gamma \longrightarrow^* \Gamma_{h-1} \longrightarrow \Gamma_h$, *with:*
  1. $\Gamma_{h-1}(\mathsf{p}) = (\mathsf{q}!\ell(\mathsf{S}) \cdot \sigma'_{\mathsf{p}}, \mathsf{T}'_{\mathsf{p}})$ *and* $\Gamma_{h-1}(\mathsf{q}) = (\sigma_{\mathsf{q}}, \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{T}_i)$*;*
  2. $\Gamma_h(\mathsf{p}) = (\sigma'_{\mathsf{p}}, \mathsf{T}'_{\mathsf{p}})$ *and* $\Gamma_h(\mathsf{q}) = (\sigma_{\mathsf{q}}, \mathsf{T}_k)$*;*
- [LP⟶] $\Gamma \longrightarrow \Gamma'$ *implies* $\varphi(\Gamma')$.

We say that Γ is p-live *iff $\varphi(\Gamma)$ for some p-liveness property φ. We say that Γ is* live *iff Γ is p-live for all $\mathsf{p} \in dom(\Gamma)$.*

▶ **Definition C.4** (SISO tree projections). *The* projections *of a SISO tree* W *are SISO trees coinductively defined as follows:*

$$
\begin{array}{ll}
\mathsf{end} \upharpoonright^{!} \mathsf{p} = \mathsf{end} & \mathsf{end} \upharpoonright^{?} \mathsf{p} = \mathsf{end} \\
(\mathsf{p}!\ell(\mathsf{S}).\mathsf{W}') \upharpoonright^{!} \mathsf{p} = \mathsf{p}!\ell(\mathsf{S}).(\mathsf{W}' \upharpoonright^{!} \mathsf{p}) & (\mathsf{p}?\ell(\mathsf{S}).\mathsf{W}') \upharpoonright^{?} \mathsf{p} = \mathsf{p}?\ell(\mathsf{S}).(\mathsf{W}' \upharpoonright^{?} \mathsf{p}) \\
(\mathsf{q}!\ell(\mathsf{S}).\mathsf{W}') \upharpoonright^{!} \mathsf{p} = \begin{cases} \mathsf{W}' \upharpoonright^{!} \mathsf{p} & \text{if } \mathsf{q} \neq \mathsf{p} \wedge \mathsf{p} \in \mathtt{pt}(\mathsf{W}') \\ \mathsf{end} & \text{if } \mathsf{q} \neq \mathsf{p} \wedge \mathsf{p} \notin \mathtt{pt}(\mathsf{W}') \end{cases} & (\mathsf{q}?\ell(\mathsf{S}).\mathsf{W}') \upharpoonright^{?} \mathsf{p} = \begin{cases} \mathsf{W}' \upharpoonright^{?} \mathsf{p} & \text{if } \mathsf{q} \neq \mathsf{p} \wedge \mathsf{p} \in \mathtt{pt}(\mathsf{W}') \\ \mathsf{end} & \text{if } \mathsf{q} \neq \mathsf{p} \wedge \mathsf{p} \notin \mathtt{pt}(\mathsf{W}') \end{cases} \\
(\mathsf{q}?\ell(\mathsf{S}).\mathsf{W}') \upharpoonright^{!} \mathsf{p} = \begin{cases} \mathsf{W}' \upharpoonright^{!} \mathsf{p} & \text{if } \mathsf{p} \in \mathtt{pt}(\mathsf{W}') \\ \mathsf{end} & \text{if } \mathsf{p} \notin \mathtt{pt}(\mathsf{W}') \end{cases} & (\mathsf{q}!\ell(\mathsf{S}).\mathsf{W}') \upharpoonright^{?} \mathsf{p} = \begin{cases} \mathsf{W}' \upharpoonright^{?} \mathsf{p} & \text{if } \mathsf{p} \in \mathtt{pt}(\mathsf{W}') \\ \mathsf{end} & \text{if } \mathsf{p} \notin \mathtt{pt}(\mathsf{W}') \end{cases}
\end{array}
$$

▶ **Definition C.5.** *The* strict refinement $\sqsubseteq$ *between SISO trees is coinductively defined as:*

$$
\frac{\mathsf{S}' \leq: \mathsf{S} \quad \mathsf{W} \sqsubseteq \mathsf{W}'}{\mathsf{p}?\ell(\mathsf{S}).\mathsf{W} \sqsubseteq \mathsf{p}?\ell(\mathsf{S}').\mathsf{W}'}\ [\text{STR-IN}] \qquad \frac{\mathsf{S} \leq: \mathsf{S}' \quad \mathsf{W} \sqsubseteq \mathsf{W}'}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{W} \sqsubseteq \mathsf{p}!\ell(\mathsf{S}').\mathsf{W}'}\ [\text{STR-OUT}] \qquad \frac{}{\mathsf{end} \sqsubseteq \mathsf{end}}\ [\text{STR-END}]
$$

By Def. C.5, $\sqsubseteq$ is a sub-relation of $\lesssim$ that does not allow to change the order of inputs nor outputs. And by Prop. C.6 below, the refinement $\lesssim$ does not alter the order of inputs nor outputs from/to a given participant p: the message reorderings allowed by $\lesssim$ can only alter interactions targeting different participants, or outputs wrt. inputs to a same participant.

▶ **Proposition C.6.** *For all* W *and* W' *and* p*, if* $\mathsf{W} \lesssim \mathsf{W}'$*, then* $(\mathsf{W} \upharpoonright^{!} \mathsf{p}) \sqsubseteq (\mathsf{W}' \upharpoonright^{!} \mathsf{p})$ *and* $(\mathsf{W} \upharpoonright^{?} \mathsf{p}) \sqsubseteq (\mathsf{W}' \upharpoonright^{?} \mathsf{p})$.

Proof. By coinduction on the derivation of $\mathsf{W} \lesssim \mathsf{W}'$. ◀

▶ **Proposition C.7.** *Take any* p*-live* Γ *with* $\Gamma(\mathsf{p}) = (\sigma, \mathsf{T})$*. Take* $\Gamma' = \Gamma\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\}$ *with* $\sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T}$*. Then, for any fair path* $(\Gamma'_j)_{j \in J'}$ *with* $\Gamma'_0 = \Gamma'$*:*

1. *for all* n*, the first* n *inputs/outputs of* p *along* $(\Gamma'_j)_{j \in J'}$ *match the first* n *input/output actions of some* $\mathsf{W}' \in [\![\mathsf{U}']\!]_{\mathsf{SI}}$*, with* $\mathsf{U}' \in [\![\mathsf{T}']\!]_{\mathsf{SO}}$*;*
2. *there is a fair path* $(\Gamma_j)_{j \in J}$ *with* $\Gamma_0 = \Gamma$ *such that, for all* n*, the first* n *inputs/outputs of* p *along* $(\Gamma_j)_{j \in J}$ *match the first* n *input/output actions of some* $\mathsf{W} \in [\![\mathsf{V}]\!]_{\mathsf{SO}}$*, with* $\mathsf{V} \in [\![\mathsf{T}]\!]_{\mathsf{SI}}$*; and*
3. $\sigma' \cdot \mathsf{W}' \lesssim \sigma \cdot \mathsf{W}$.

Proof. Before proceeding, with a slight abuse of notation, we "rewind" Γ and Γ', i.e., we consider Γ and Γ' in the statement to be defined such that:

$$
\Gamma(\mathsf{p}) = (\epsilon, \sigma \cdot \mathsf{T}) \qquad \Gamma'(\mathsf{p}) = (\epsilon, \sigma' \cdot \mathsf{T}') \tag{12}
$$

i.e., the outputs in σ and σ' are not yet queued; instead, the queues of $\Gamma(\mathsf{p})$ and $\Gamma'(\mathsf{p})$ are empty, and the outputs σ and σ' are prefixes of the respective types, and are about to be sent. We will "undo" this rewinding at the end of the proof, to obtain the final result.

Since $\sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T}$ (by hypothesis), by Def. 3.4 we have:

$$
\forall \mathsf{U}' \in [\![\sigma' \cdot \mathsf{T}']\!]_{\mathsf{SO}} \quad \forall \mathsf{V} \in [\![\sigma \cdot \mathsf{T}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_2 \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_1 \in [\![\mathsf{V}]\!]_{\mathsf{SO}} \qquad \mathsf{W}_2 \lesssim \mathsf{W}_1 \tag{13}
$$

Observe that U' and V in (13) are quantified over sets of session trees beginning with a same sequence of singleton selections (σ' and σ, respectively). Therefore, such sequences of selections appear at the beginning of all SISO trees extracted from any such $\mathbb{U}'$ and $\mathbb{V}'$, which means:

$$
\forall \mathsf{U}' \in [\![\mathsf{T}']\!]_{\mathsf{SO}} \quad \forall \mathsf{V} \in [\![\mathsf{T}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_a \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_b \in [\![\mathsf{V}]\!]_{\mathsf{SO}} \qquad \sigma' \cdot \mathsf{W}_a \lesssim \sigma \cdot \mathsf{W}_b \tag{14}
$$

We now define a procedure that, using the first m steps of a fair path $(\Gamma'_j)_{j\in J'}$, constructs the beginning of a fair path $(\Gamma_j)_{j\in J}$; also, the procedure ensures that: *(1)* in the first m steps of $(\Gamma'_j)_{j\in J'}$, participant p follows σ' and then a prefix of some W_a; *(2)* the path $(\Gamma_j)_{j\in J}$ is constructed so that p follows σ and then some W_b such that $\sigma'{\cdot}\mathsf{W}_a \lesssim \sigma{\cdot}\mathsf{W}_b$; and *(3)* such W_a and W_b are quantified in (14). In particular, the procedure lets each participant in $\Gamma' \setminus \mathsf{p} = \Gamma \setminus \mathsf{p}$ fire the same sequences of actions along $(\Gamma'_j)_{j\in J'}$ and $(\Gamma_j)_{j\in J}$. The difference is that some actions may be delayed in $(\Gamma_j)_{j\in J}$, because p in Γ' may anticipate some outputs and inputs (thanks to subtyping) wrt. Γ, and unlock some participants in Γ' earlier than Γ: the procedure remembers any delayed actions (and their order), and fires them as soon as they become enabled, thus ensuring fairness.

For the procedure, we use:

- $\mathsf{p}\text{-actions}'(i)$: sequence of input/output actions performed by p in Γ' when it reaches Γ'_i. We begin with $\mathsf{p}\text{-actions}'(0) = \epsilon$;
- $\gamma(i)$: number of reduction steps constructed along $(\Gamma_j)_{j\in J}$ when Γ' has reached Γ'_i. We begin with $\gamma(0) = 0$;
- $\mathsf{p}\text{-actions}(i)$: sequence of input/output actions performed by p in Γ when Γ' reaches Γ'_i (note that, at this stage, Γ has reached $\Gamma_{\gamma(i)}$). We begin with $\mathsf{p}\text{-actions}(0) = \epsilon$;
- $\mathcal{W}(i)$: set of SISO tree pairs $(\mathsf{W}_a, \mathsf{W}_b)$ such that, when Γ' has reached Γ'_i, and Γ has reached $\Gamma_{\gamma(i)}$, the sequence $\mathsf{p}\text{-actions}'(i)$ matches a prefix of $\sigma'{\cdot}\mathsf{W}_a$, and the sequence $\mathsf{p}\text{-actions}(i)$ matches a prefix of $\sigma{\cdot}\mathsf{W}_b$, and $\sigma{\cdot}\mathsf{W}_a \lesssim \sigma'{\cdot}\mathsf{W}_b$. We begin with $\mathcal{W}(0)$ containing all pairs W_a and W_b quantified in (14);
- $\mathit{delayed}(i)$: sequence of reduction labels that have been fired by Γ' when it reaches Γ'_i, but have not (yet) been fired by Γ when it reaches $\Gamma_{\gamma(i)}$. Labels in this sequence will be fired with the highest priority. We begin with $\mathit{delayed}(\mathit{0}) = \epsilon$.

We also use the following function:

$$\text{tryFire}(d, \Gamma, d', f, s) = \begin{cases} \text{if } d = \epsilon \text{ then } (d', f, s) \\ \text{else if } \Gamma \xrightarrow{\mathit{head}(d)} \Gamma' \text{ then tryFire}(\mathit{tail}(d), \Gamma', d', f{\cdot}\mathit{head}(d), s{\cdot}\Gamma) \\ \text{else tryFire}(\mathit{tail}(d), \Gamma, d'{\cdot}\mathit{head}(d), f, s) \end{cases}$$

The function $\text{tryFire}(d, \Gamma, d', f, s)$ tries to fire the environment reduction labels in the sequence d from Γ. The other parameters are used along recursive calls, to build the triplet that is returned by the function:

- d': a sequence of labels that have *not* been fired. It is extended each time the topmost label in d cannot be fired;
- f: a sequence of labels that *have* been fired. It is extended each time the topmost label in d is fired; and
- s: a sequence of typing environments reducing from one into another through the sequence of labels f. It is extended each time f is extended (see above).

When $(\Gamma'_j)_{j\in J'}$ performs a step $m+1$, with a label α such that $\Gamma'_m \xrightarrow{\alpha} \Gamma'_{m+1}$, we proceed as follows:

1. if α does *not* involve an input/output by p, i.e., $\alpha = \mathsf{q}{:}\mathsf{r}?\ell$ or $\alpha = \mathsf{q}{:}\mathsf{r}!\ell$ for some $\mathsf{q}, \mathsf{r} \neq \mathsf{p}$:
   a. $\mathsf{p}\text{-actions}'(m+1) = \mathsf{p}\text{-actions}'(m)$
2. otherwise (i.e., if $\alpha = \mathsf{p}{:}\mathsf{q}!\ell$ or $\alpha = \mathsf{p}{:}\mathsf{q}?\ell$):

   a. $\mathsf{p}\text{-actions}'(m+1) = \mathsf{p}\text{-actions}'(m) \cdot \alpha$;

3. $(d', f, s) = \text{tryFire}\big(\mathit{delayed}(m), \Gamma_{\gamma(m)}, \epsilon, \epsilon, \epsilon\big)$ (i.e., try to fire each delayed action)
4. $\Gamma^* = \text{if } |s| > 0 \text{ then } \mathit{last}(s) \text{ else } \Gamma_{\gamma(m)}$ (i.e., Γ^* is the latest env. reached from Γ)
5. $\forall i = 1..|s| : \Gamma_{\gamma(m)+i} = s(i)$ (i.e., we extend the path of Γ to reach Γ^*)
6. if $\Gamma^* \xrightarrow{\alpha} \Gamma''$ (for some Γ''), we add Γ'' to the path of Γ, as follows:
   a. $\Gamma_{\gamma(m)+|s|+1} = \Gamma''$
   b. $\gamma(m+1) = \gamma(m) + |s| + 1$
   c. $\mathit{delayed}(m+1) = d'$
   d. if α does *not* involve an input/output by p, i.e., $\alpha = \mathsf{q}{:}\mathsf{r}?\ell$ or $\alpha = \mathsf{q}{:}\mathsf{r}!\ell$ for some $\mathsf{q}, \mathsf{r} \neq \mathsf{p}$:
      i. $\mathsf{p}\text{-actions}(m+1) = \mathsf{p}\text{-actions}(m)$ extended with all labels in f involving p;
   e. otherwise (i.e., if $\alpha = \mathsf{p}{:}\mathsf{q}!\ell$ or $\alpha = \mathsf{p}{:}\mathsf{q}?\ell$):
      i. $\mathsf{p}\text{-actions}(m+1) = \mathsf{p}\text{-actions}(m)$ extended with all labels in f involving p, followed by α;
7. otherwise (i.e., if there is no Γ'' such that $\Gamma^* \xrightarrow{\alpha} \Gamma''$), we add α to the delayed actions, as follows:
   a. $\mathsf{p}\text{-actions}(m+1) = \mathsf{p}\text{-actions}(m)$ extended with all labels in f involving p
   b. $\gamma(m+1) = \gamma(m) + |s|$
   c. $\mathit{delayed}(m+1) = d' \cdot \alpha$
8. $\mathcal{W}(m+1) = \left\{ (\mathsf{W}_a, \mathsf{W}_b) \in \mathcal{W}(m) \;\middle|\; \begin{array}{l} \mathsf{W}_a \text{ matches } \mathsf{p}\text{-actions}'(m+1) \\ \mathsf{W}_b \text{ matches } \mathsf{p}\text{-actions}(m+1) \end{array} \right\}$

The procedure has the following invariants, for all $i \geq 0$:

(i1) $\mathsf{p}\text{-actions}'(i)$ is the sequence if inputs/outputs of p fired along the transitions from $\Gamma'_0 = \Gamma'$ to Γ'_i;
(i2) $\mathsf{p}\text{-actions}(i)$ is the sequence if inputs/outputs of p fired along the transitions from $\Gamma_0 = \Gamma$ to $\Gamma_{\gamma(i)}$;
(i3) $\mathcal{W}(i) \neq \emptyset$ and $\forall (\mathsf{W}_a, \mathsf{W}_b) \in \mathcal{W}(i)$: W_a matches $\mathsf{p}\text{-actions}'(i)$ and W_b matches $\mathsf{p}\text{-actions}(i)$.

We now obtain our thesis, by invariants *(i1)*–*(i3)* above: by taking any fair path $(\Gamma'_j)_{j \in J'}$ with $\Gamma'_0 = \Gamma'$, and applying the procedure above for any $n = |\,\mathsf{p}\text{-actions}'(m)|$ for some $m \geq n$ such that $m \in J'$ (i.e., for any number n of reductions of p that are performed along the path, within m steps), we find some W' such that $\sigma' \cdot \mathsf{W}'$ matches $\mathsf{p}\text{-actions}'(m)$, and we construct the beginning of a fair path $(\Gamma_j)_{j \in J}$ where p behaves according to some W such that $\sigma \cdot \mathsf{W}$ matches $\mathsf{p}\text{-actions}(m)$, and such that $\sigma' \cdot \mathsf{W}' \lesssim \sigma \cdot \mathsf{W}$; and by increasing n and m, we correspondingly extend the sequences $\mathsf{p}\text{-actions}'(m)$ and $\mathsf{p}\text{-actions}(m)$.

To conclude the proof, we need to undo the "rewinding" in (12). Consider any path $(\Gamma'_j)_{j \in J'}$, and the corresponding $(\Gamma_j)_{j \in J}$ obtained with Γ', Γ rewinded as in (12): we can undo the rewinding of σ' and σ by:

1. choosing a path $(\Gamma'_j)_{j \in J'}$ that fires the outputs in σ' in its first reductions, thus reaching the "original" Γ' from the statement;
2. then, for such a path of Γ', the procedure gives us the beginning of a corresponding live path $(\Gamma_j)_{j \in J}$ that fires all outputs in σ, within k steps (for some k). By induction on k and σ, we can reorder the first k actions of $(\Gamma_j)_{j \in J}$ so that the outputs in σ are fired first, thus reaching the "original" Γ from the statement;
3. after the outputs in σ and σ' are fired along such paths of Γ' and Γ, we have that p follows SISO trees W_a and W_b extracted respectively from T' and T, and we have $\sigma \cdot \mathsf{W}_a \lesssim \sigma' \cdot \mathsf{W}_b$: therefore, such paths satisfy the statement.

◀

▶ **Proposition C.8.** *Take any* p*-live* Γ *with* $\Gamma(\mathsf{p}) = (\sigma, \mathsf{T})$*. Take* $\Gamma' = \Gamma\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\}$ *with* $\sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T}$*. Assume that there is a fair path* $(\Gamma'_j)_{j \in J'}$ *with* $\Gamma'_0 = \Gamma'$ *such that, for some* $\mathsf{q}, \ell, \mathsf{S}_\mathsf{r}$*:*

$$\forall j \in J', \sigma_\mathsf{r}, \mathsf{T}_\mathsf{r} : \ \Gamma'_j(\mathsf{r}) \neq (\mathsf{q}!\ell(\mathsf{S}_\mathsf{r}) \cdot \sigma_\mathsf{r}, \mathsf{T}_\mathsf{r}) \tag{15}$$

Then, there is a fair path $(\Gamma_j)_{j \in J}$ *with* $\Gamma_0 = \Gamma$ *such that,*

$$\forall j \in J, \sigma_\mathsf{r}, \mathsf{T}_\mathsf{r} : \ \Gamma_j(\mathsf{r}) \neq (\mathsf{q}!\ell(\mathsf{S}_\mathsf{r}) \cdot \sigma_\mathsf{r}, \mathsf{T}_\mathsf{r}) \tag{16}$$

Proof. Take the path $(\Gamma'_j)_{j \in J'}$. By Prop. C.7 there is a SISO tree W' describing the first n reductions (for any n) of p in Γ', and a corresponding SISO tree W such that $\sigma' \cdot \mathsf{W}' \lesssim \sigma \cdot \mathsf{W}$ describing the reductions of p in Γ along a fair path $(\Gamma_j)_{j \in J}$. By Prop. C.6, $\sigma' \cdot \mathsf{W}'$ contains the same sequences of per-participant inputs and outputs of $\sigma \cdot \mathsf{W}$; moreover, by Def. 3.2, W' can perform the outputs appearing in W, possibly earlier. And by the path construction in Prop. C.7, each participant $\mathsf{q} \in \mathit{dom}(\Gamma) = \mathit{dom}(\Gamma')$ (with $\mathsf{q} \neq \mathsf{p}$) can fire along $(\Gamma'_j)_{j \in J'}$ at least the same outputs and the same inputs (in the same respective order) that it fires along $(\Gamma_j)_{j \in J}$. Now, observe that by hypothesis (15), along the fair path $(\Gamma'_j)_{j \in J'}$, participant r never produces an output $\mathsf{q}!\ell(\mathsf{S}_\mathsf{r})$; but then, along $(\Gamma_j)_{j \in J}$, participant r never produces the output $\mathsf{q}!\ell(\mathsf{S}_\mathsf{r})$, either. Therefore, we obtain (16). ◀

▶ **Proposition C.9.** *Take any* p*-live* Γ *with* $\Gamma(\mathsf{p}) = (\sigma, \mathsf{T})$*. Take* $\Gamma' = \Gamma\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\}$ *with* $\sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T}$*. Assume that there is a fair path* $(\Gamma'_j)_{j \in J'}$ *with* $\Gamma'_0 = \Gamma$ *such that, for some* $\mathsf{q}, I, \ell_i, \mathsf{S}_{\mathsf{r},i}, \mathsf{T}_{\mathsf{r},i}$ $(i \in I)$*:*

$$\forall j \in J', \sigma_\mathsf{r} : \ \Gamma'_j(\mathsf{r}) \neq \left(\sigma_\mathsf{r}, \mathop{\&}_{i \in I} \mathsf{q}?\ell_i(\mathsf{S}_{\mathsf{r},i}).\mathsf{T}_{\mathsf{r},i}\right) \tag{17}$$

Then, there is a fair path $(\Gamma_j)_{j \in J}$ *with* $\Gamma_0 = \Gamma$ *such that:*

$$\forall j \in J, \sigma_\mathsf{r} : \ \Gamma_j(\mathsf{r}) \neq \left(\sigma_\mathsf{r}, \mathop{\&}_{i \in I} \mathsf{q}?\ell_i(\mathsf{S}_{\mathsf{r},i}).\mathsf{T}_{\mathsf{r},i}\right) \tag{18}$$

Proof. Similar to Prop. C.8. ◀

▶ **Definition C.10** (Queue output prefixing)**.** *We write* $\sigma \cdot \mathsf{T}$ *for the session tree obtained by prefixing* T *with the sequence of singleton internal choices matching the sequence of outputs in* σ*.*

▶ **Lemma C.11.** *If* $\Gamma, \mathsf{p}{:}(\sigma, \mathsf{T})$ *is* p*-live and* $\sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T}$*, then* $\Gamma, \mathsf{p}{:}(\sigma', \mathsf{T}')$ *is* p*-live.*

Proof. Let $\mathcal{L}$ be the set of all p-live typing contexts, i.e., the largest p-liveness property by Def. C.3. Consider the following property:

$$\mathcal{P} = \mathcal{L} \cup \mathcal{L}' \quad \text{where } \mathcal{L}' = \left\{ \Gamma\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\} \;\middle|\; \begin{array}{l} \Gamma \in \mathcal{L} \\ \Gamma(\mathsf{p}) = (\sigma, \mathsf{T}) \\ \sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T} \end{array} \right\} \tag{19}$$

We now prove that $\mathcal{P}$ is a p-liveness property — i.e., we prove that each element of $\mathcal{P}$ satisfies the clauses of Def. C.3. Since all elements of $\mathcal{L}$ trivially satisfy the clauses, we only need to examine the elements of $\mathcal{L}'$: to this purpose, we consider each $\Gamma' \in \mathcal{L}'$, and observe that, by (19), there exist $\Gamma, \sigma, \mathsf{T}, \sigma', \mathsf{T}'$ such that:

$$\Gamma' = \Gamma\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\} \tag{20}$$

$$\Gamma(\mathsf{p}) = (\sigma, \mathsf{T}) \tag{21}$$

$$\sigma' \cdot \mathsf{T}' \leqslant \sigma \cdot \mathsf{T} \tag{22}$$

By (22) and Def. 3.4, we also have:

$$\forall \mathsf{U}' \in [\![\sigma' \cdot \mathsf{T}']\!]_{\mathsf{SO}} \quad \forall \mathsf{V} \in [\![\sigma \cdot \mathsf{T}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_1 \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_2 \in [\![\mathsf{V}]\!]_{\mathsf{SO}} \qquad \mathsf{W}_1 \lesssim \mathsf{W}_2 \tag{23}$$

Observe that by the definitions of $[\![\cdot]\!]_{\mathsf{SO}}$ and $[\![\cdot]\!]_{\mathsf{SI}}$ on page 6, and by Def. 3.2, all relations $\mathsf{W}_1 \lesssim \mathsf{W}_2$ in (23) are yielded by a same refinement rule. We proceed by cases on such a rule.

- Case [REF-END]. In this case, we have $\sigma' \cdot \mathsf{T}' \lesssim \sigma \cdot \mathsf{T} = \mathtt{end}$, which means $\sigma' = \sigma = \epsilon$ and $\mathsf{T}' = \mathsf{T} = \mathtt{end}$. Therefore, by (21) and (20), $\Gamma' = \Gamma \in \mathcal{L}$, and thus, we conclude that Γ' satisfies the clauses of Def. C.3.
- Case [REF-IN]. In this case, we have:

$$\sigma = \sigma' = \epsilon \tag{24}$$

$\exists \mathsf{q}, I, I', \ell_i, \mathsf{S}_i, \mathsf{S}'_i, \mathsf{T}_i, \mathsf{T}'_i$ such that:

$$\mathsf{T}' = \mathop{\&}_{i \in I \cup I'} \mathsf{q}?\ell_i(\mathsf{S}'_i).\mathsf{T}'_i \quad \text{and} \quad \mathsf{T} = \mathop{\&}_{i \in I} \mathsf{q}?\ell_i(\mathsf{S}_i).\mathsf{T}_i \tag{25}$$

$$\forall i \in I : \mathsf{S}_i \leq: \mathsf{S}'_i \quad \text{and} \quad \mathsf{T}'_i \lesssim \mathsf{T}_i \tag{26}$$

We now show that Γ' satisfies all clauses of Def. C.3:

  - clause [LP&]. Since Γ is p-live, we know that, for all fair paths $(\Gamma_j)_{j \in J}$ such that $\Gamma_0 = \Gamma$, $\exists h \in J, k \in I$ such that $\Gamma \longrightarrow^* \Gamma_{h-1} \longrightarrow \Gamma_h$, with:
    1. $\Gamma_{h-1}(\mathsf{p}) = (\sigma, \mathsf{T})$ and $\Gamma_{h-1}(\mathsf{q}) = (\mathsf{p}!\ell(\mathsf{S}_\mathsf{q}) \cdot \sigma_\mathsf{q}, \mathsf{T}_\mathsf{q})$ with $\ell = \ell_i$ and $\mathsf{S}_\mathsf{q} \leqslant \mathsf{S}_i$ for some $i \in I$
    2. $\Gamma_h(\mathsf{p}) = (\sigma, \mathsf{T}_i)$ and $\Gamma_h(\mathsf{q}) = (\sigma_\mathsf{q}, \mathsf{T}_\mathsf{q})$.

    Now, for all such $(\Gamma_j)_{j \in J}$, we can construct a path corresponding $(\Gamma'_j)_{j \in J'}$ such that:
    * $\Gamma'_0 = \Gamma'$
    * $\forall n \in 0..h-1 : \Gamma'_n = \Gamma_n\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\}$;
    * $\Gamma'_h = \Gamma_h\{\mathsf{p} \mapsto (\sigma', \mathsf{T}'_i)\}$ (i.e., the queue of $\Gamma_h(\mathsf{p})$ is preserved in $\Gamma'_h(\mathsf{p})$);
    * the rest of the path after the h-th reduction is arbitrary (but fair).

    Observe that $(\Gamma'_j)_{j \in J'}$ is fair, reproduces the first h steps of $(\Gamma_j)_{j \in J}$, and triggers the top-level input of $\Gamma'(\mathsf{p})$. Also, observe that *all* fair paths from Γ' eventually trigger the top-level input of $\Gamma'(\mathsf{p})$: in fact, if (by contradiction) we assume that there is a fair path from Γ' that never triggers $\Gamma'(\mathsf{p})$'s input, then (by inverting the construction above) we would find a corresponding fair path of Γ that never triggers $\Gamma(\mathsf{p})$'s input — i.e., we would conclude that Γ is *not* p-live (contradiction). Thus, we conclude that Γ' satisfies clause [LP&] of Def. C.3;
  - clause [LP⊕]. The clause is vacuously satisfied;
  - clause [LP⟶]. Assume $\Gamma' \longrightarrow \Gamma''$. We have two possibilities:

    (a) the reduction does *not* involve p. Then, there is a corresponding reduction $\Gamma \longrightarrow \Gamma'''$ with $\Gamma'' = \Gamma'''\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\}$. Observe that Γ''' is p-live, and thus, $\Gamma''' \in \mathcal{L}$; therefore, by (19), we have $\Gamma'' \in \mathcal{L}' \subseteq \mathcal{P}$. Thus, we conclude that Γ' satisfies clause [LP⟶] of Def. C.3;

(b) the reduction *does* involve p. There are three sub-cases:

(i) p is enqueuing an output toward some participant r. This case is impossible, by (25);

(ii) p is receiving an input from q, i.e., $\Gamma'(\mathsf{q}) = \Gamma(\mathsf{q}) = (\mathsf{p}!\ell_i(\mathsf{S_q}) \cdot \sigma_\mathsf{q}, \mathsf{T_q})$ with $\mathsf{S_q} \leq: \mathsf{S}'_i$ and $\Gamma''(\mathsf{p}) = (\sigma', \mathsf{T}'_i)$ (for some $i \in I$). Notice that, since Γ is p-live, the same output from q can be received by p in Γ, and thus, there is a corresponding reduction $\Gamma \longrightarrow \Gamma'''$ where $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}_i)$ and $\Gamma'' = \Gamma'''\{\mathsf{p} \mapsto (\sigma', \mathsf{T}'_i)\}$. Observe that Γ''' is also p-live, and thus, $\Gamma''' \in \mathcal{L}$; therefore, by (26) and (19), we have $\Gamma'' \in \mathcal{L}' \subseteq \mathcal{P}$. Hence, we conclude that Γ' satisfies clause [LP$\longrightarrow$] of Def. C.3;

(iii) one of p's queued outputs in σ' is received by another participant. This case is impossible, by (24).

- Case [REF-$\mathcal{A}$]. In this case, we have:

$$\sigma = \sigma' = \epsilon \tag{27}$$

$$\exists I, I', \ell_i, \mathsf{S}'_i, \mathsf{T}'_i : \ \mathsf{T}' = \mathop{\&}_{i \in I \cup I'} \mathsf{q}?\ell_i(\mathsf{S}'_i).\mathsf{T}'_i \tag{28}$$

for all $\mathsf{W}_1, \mathsf{W}_2$ in (23), $\exists i \in I, \mathcal{A}^{(\mathsf{q})}, \ell_i, \mathsf{S}_i, \mathsf{S}'_i, \mathsf{W}, \mathsf{W}'$, such that:

$$\mathsf{W}_1 = \mathsf{q}?\ell_i(\mathsf{S}'_i).\mathsf{W}' \quad \text{and} \quad \mathsf{W}_2 = \mathcal{A}^{(\mathsf{q})}.\mathsf{q}?\ell_i(\mathsf{S}_i).\mathsf{W} \tag{29}$$

$$\mathsf{S}_i \leq: \mathsf{S}'_i \quad \text{and} \quad \mathsf{W}' \lesssim \mathcal{A}^{(\mathsf{q})}.\mathsf{W} \quad \text{and} \quad \mathtt{act}(\mathsf{W}') = \mathtt{act}(\mathcal{A}^{(\mathsf{q})}.\mathsf{W}) \tag{30}$$

We now show that Γ' satisfies all clauses of Def. C.3:

- clause [LP&]. We proceed by contradiction: we show that if there is a fair path $(\Gamma'_j)_{j \in J'}$ (with $\Gamma'_0 = \Gamma'$) that violates clause [LP&], then there is a corresponding fair path $(\Gamma_j)_{j \in J}$ (with $\Gamma_0 = \Gamma$) that violates the same clause, which would lead to the absurd conclusion that Γ is *not* p-live. Such a hypothetical path $(\Gamma'_j)_{j \in J'}$ consists of a series of transitions $\Gamma'_{j-1} \xrightarrow{\alpha_j} \Gamma'_j$ (for $j \in J'$) where, $\forall j \in J'$, α_j does *not* involve p (by (27)). This means that:

$$\forall j \in J' : \nexists \mathsf{T_q}, \sigma_\mathsf{q}, \mathsf{S_q} : \Gamma'_j(\mathsf{q}) = (\mathsf{p}!\ell_i(\mathsf{S_q}) \cdot \sigma_\mathsf{q}, \mathsf{T_q}) \text{ with } \mathsf{S_q} \leq: \mathsf{S}'_i \text{ (for any } i \in I) \tag{31}$$

But then, by Prop. C.8, there is a fair path $(\Gamma_j)_{j \in J}$ with $\Gamma_0 = \Gamma$ where p reduces according to some W_2 in (29), and such that:

$$\forall j \in J : \nexists \mathsf{T_q}, \sigma_\mathsf{q}, \mathsf{S_q} : \Gamma_j(\mathsf{q}) = (\mathsf{p}!\ell_i(\mathsf{S_q}) \cdot \sigma_\mathsf{q}, \mathsf{T_q}) \text{ with } \mathsf{S_q} \leq: \mathsf{S}_i \text{ (for any } i \in I) \tag{32}$$

Now, observe that the fair path $(\Gamma_j)_{j \in J}$ is constructed using Prop. C.7, and therefore, will eventually attempt to fire the input $\mathsf{q}?\ell_i(\mathsf{S}_i)$ of W_2 in (29) — but no suitable output will be available, by (32): thus, we obtain that Γ is not p-live — contradiction. Therefore, we conclude that Γ' satisfies clause [LP&] of Def. C.3;
- clause [LP$\oplus$]. The clause is vacuously satisfied;
- clause [LP$\longrightarrow$]. Assume $\Gamma' \longrightarrow \Gamma''$. We have two possibilities:

(a) the reduction does *not* involve p. The proof is similar to case [REF-IN][LP$\longrightarrow$](a) above;

(b) the reduction *does* involve p. There are three sub-cases:

(i) p is enqueuing an output toward some participant r. This case is impossible, by (28);

(ii) p is receiving an input from q, i.e., for some $i \in I$:

$$\Gamma'(\mathsf{q}) = \Gamma(\mathsf{q}) = (\mathsf{p}!\ell_i(\mathsf{S_q}) \cdot \sigma_\mathsf{q}, \mathsf{T_q}) \ \text{ with } \mathsf{S_q} \leq: \mathsf{S}'_i \tag{33}$$

$$\Gamma' \xrightarrow{\mathsf{p}:\mathsf{q}?\ell_i} \Gamma'' \quad \text{with } \Gamma''(\mathsf{p}) = (\sigma', \mathsf{T}'_i) \tag{34}$$

By Prop. C.7, the fair paths of Γ that match some witness W_2 in (29) eventually perform the reduction α above. But then, consider the session tree T^* that only has SISO trees similar to W_2 in (23), except that each one performs the input $\mathsf{q}?\ell_j(\mathsf{S})$ immediately — i.e.:

$$\begin{array}{l} \forall \mathsf{U}' \in [\![\sigma' \cdot \mathsf{T}']\!]_{\mathsf{SO}} \quad \forall \mathsf{V}^* \in [\![\sigma \cdot \mathsf{T}^*]\!]_{\mathsf{SI}} \\ \exists \mathsf{W}_1 = \mathsf{q}?\ell_i(\mathsf{S}'_i).\mathsf{W}' \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}^* = \mathsf{q}?\ell_i(\mathsf{S}_i).\mathcal{A}^{(\mathsf{q})}.\mathsf{W} \in [\![\mathsf{V}^*]\!]_{\mathsf{SO}} \quad \mathsf{W}_1 \lesssim \mathsf{W}^* \end{array} \tag{35}$$

And now, consider the typing environment $\Gamma^* = \Gamma\{\mathsf{p} \mapsto (\sigma, \mathsf{T}^*)\}$. Such Γ^* is p-live: it realises (part of) the fair paths of Γ, except that it consumes q's queued output earlier — and thus, we have $\Gamma^* \in \mathcal{L}$. Now, consider Γ''' such that $\Gamma^* \xrightarrow{\mathsf{p}:\mathsf{q}?\ell_i} \Gamma'''$: by clause [LP⟶] of Def. C.3, we have $\Gamma''' \in \mathcal{L}$. Also, we have $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}^{**})$ such that:

$$\begin{array}{l} \forall \mathsf{U}' \in [\![\sigma' \cdot \mathsf{T}'_i]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}^{**} \in [\![\sigma \cdot \mathsf{T}^{**}]\!]_{\mathsf{SI}} \\ \exists \mathsf{W}_1 = \mathsf{W}' \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}^* = \mathcal{A}^{(\mathsf{q})}.\mathsf{W} \in [\![\mathsf{V}^{**}]\!]_{\mathsf{SO}} \quad \mathsf{W}_1 \lesssim \mathsf{W}^* \end{array} \quad \text{(by Def. 4.3 and (30))}$$

Therefore, by Def. 3.4, we have $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}^{**})$, and $\Gamma'' = \Gamma'''\{\mathsf{p} \mapsto (\sigma', \mathsf{T}'_i)\}$, with $\sigma' \cdot \mathsf{T}'_i \leqslant \sigma \cdot \mathsf{T}^{**}$ — hence, by (30) and (19), we also have $\Gamma'' \in \mathcal{L}' \subseteq \mathcal{P}$. Thus, we conclude that Γ' satisfies clause [LP⟶] of Def. C.3;

(iii) one of p's queued outputs in σ' is received by another participant. This case is impossible, by (27).

- Case [REF-OUT]. In this case, we have:

$\exists \mathsf{q}, \ell, \sigma, \sigma', \mathsf{S}, \mathsf{S}', \mathsf{T}_1, \mathsf{T}_2$ such that:

$$\sigma' \cdot \mathsf{T}' = \mathsf{q}!\ell(\mathsf{S}').\mathsf{T}_1 \quad \text{and} \quad \sigma \cdot \mathsf{T} = \mathsf{q}!\ell(\mathsf{S}).\mathsf{T}_2 \tag{36}$$

$$\mathsf{S}' \leq: \mathsf{S} \quad \text{and} \quad \mathsf{T}_1 \leqslant \mathsf{T}_2 \tag{37}$$

We now show that Γ' satisfies all clauses of Def. C.3:

- clause [LP&]. If T' does *not* begin with an external choice $\mathsf{r}?\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}})$, the clause is vacuously satisfied. In the case where $\mathsf{T}' = \&_{i \in I}\, \mathsf{r}?\ell_{\mathsf{r},i}(\mathsf{S}'_{\mathsf{r},i}).\mathsf{T}_i$, the proof is similar to case [REF-$\mathcal{A}$][LP&] above; Thus, we conclude that Γ' satisfies clause [LP&] of Def. C.3;
- clause [LP⊕]. We proceed by contradiction: we show that if there is a fair path $(\Gamma'_j)_{j \in J'}$ (with $\Gamma'_0 = \Gamma'$) that violates clause [LP⊕], then there is a corresponding fair path $(\Gamma_j)_{j \in J}$ (with $\Gamma_0 = \Gamma$) that violates the same clause, which would lead to the absurd conclusion that Γ is *not* p-live. We need to consider any output at the head of the queue of $\Gamma'(\mathsf{p})$ up-to reordering via $\equiv$: let such an output be $\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}})$ (for some r). Consider the hypothetical non-p-live path $(\Gamma'_j)_{j \in J'}$: it must consist of a series of transitions $\Gamma'_{j-1} \xrightarrow{\alpha_j} \Gamma'_j$ (for $j \in J'$) where, $\forall j \in J'$, $\alpha_j \neq \mathsf{r}:\mathsf{p}?\ell$. Hence:

$$\forall j \in J' : \nexists \sigma_{\mathsf{r}}, I, \ell_i, \mathsf{S}_{\mathsf{r},i}, \mathsf{T}_{\mathsf{r},i} (i \in I) : \; \Gamma'_j(\mathsf{r}) = \left(\sigma_{\mathsf{r}}, \mathop{\&}_{i \in I} \mathsf{p}?\ell_i(\mathsf{S}_{\mathsf{r},i}).\mathsf{T}_{\mathsf{r},i}\right) \tag{38}$$

$$\text{with } \ell_i = \ell_{\mathsf{r}} \text{ and } \mathsf{S}'_{\mathsf{r}} \leq: \mathsf{S}_{\mathsf{r},i} \text{ (for some } i \in I)$$

But then, by (38) and Prop. C.9, there is a fair path $(\Gamma_j)_{j \in J}$ with $\Gamma_0 = \Gamma$ such that:

$$\forall j \in J : \nexists \sigma_{\mathsf{r}}, I, \ell_i, \mathsf{S}_{\mathsf{r},i}, \mathsf{T}_{\mathsf{r},i} (i \in I) : \; \Gamma_j(\mathsf{r}) = \left(\sigma_{\mathsf{r}}, \mathop{\&}_{i \in I} \mathsf{p}?\ell_i(\mathsf{S}_{\mathsf{r},i}).\mathsf{T}_{\mathsf{r},i}\right) \tag{39}$$

$$\text{with } \ell_i = \ell_{\mathsf{r}} \text{ and } \mathsf{S}'_{\mathsf{r}} \leq: \mathsf{S}_{\mathsf{r},i} \text{ (for some } i \in I)$$

Now, observe that the path $(\Gamma_j)_{j\in J}$ is constructed using Prop. C.7; therefore, by (20), (21), (22), and Prop. C.6, an output $\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}_{\mathsf{r}})$ (with $\mathsf{S}'_{\mathsf{r}} \leqslant \mathsf{S}_{\mathsf{r}}$) occurs along $(\Gamma_j)_{j\in J}$; and by (39), such an output will never be consumed: this means that Γ is not p-live — contradiction. Thus, we conclude that Γ' satisfies clause [LP⊕] of Def. C.3;

- clause [LP⟶]. Assume $\Gamma' \longrightarrow \Gamma''$. We have two possibilities:

(a) the reduction does *not* involve p. The proof is similar to case [REF-IN][LP⟶](a) above;

(b) the reduction *does* involve p. There are three sub-cases:

(i) p is enqueuing an output toward some participant r. In this case, we have:

$$\mathsf{T}' = \bigoplus_{i\in I} \mathsf{r}!\ell_i(\mathsf{S}'_i).\mathsf{T}'_i \tag{40}$$

$$\Gamma' \xrightarrow{\mathsf{p}:\mathsf{r}!\ell_i} \Gamma'' = \Gamma'\{\mathsf{p} \mapsto (\sigma'\cdot\mathsf{r}!\ell_i(\mathsf{S}'_i), \mathsf{T}'_i)\} \quad \text{(for some } i \in I) \quad \text{(by Def. 4.3)} \tag{41}$$

$$\Gamma'' = \Gamma''\{\mathsf{p} \mapsto (\sigma'\cdot\mathsf{r}!\ell_i(\mathsf{S}'_i), \mathsf{T}'_i)\} \qquad \text{(by (41) and (20))} \tag{42}$$

$$\sigma'\cdot\mathsf{r}!\ell_i(\mathsf{S}'_i)\cdot\mathsf{T}'_i \leqslant \sigma'\cdot\mathsf{T}' \qquad \text{(by induction on } \sigma'\text{, using (40) and Def. 3.4)} \tag{43}$$

$$\sigma'\cdot\mathsf{r}!\ell_i(\mathsf{S}'_i)\cdot\mathsf{T}'_i \leqslant \sigma\cdot\mathsf{T} \qquad \text{(by (43), (22), and Lemma B.9)} \tag{44}$$

$$\Gamma'' \in \mathcal{L}' \subseteq \mathcal{P} \qquad \text{(by (44) and (42))} \tag{45}$$

Therefore, by (45), we conclude that Γ' satisfies clause [LP⟶] of Def. C.3;

(ii) p is receiving an input from some r. In this case, we have:

$$\exists I, \ell_i, \mathsf{S}'_i, \mathsf{T}'_i : \ \mathsf{T}' = \&_{i\in I} \mathsf{r}?\ell_i(\mathsf{S}'_i).\mathsf{T}'_i \tag{46}$$

$$\Gamma' \xrightarrow{\alpha} \Gamma'' \ \text{ with } \alpha = \mathsf{p}:\mathsf{r}?\ell_i \text{ (for some } i \in I) \tag{47}$$

$$\sigma = \epsilon \ \text{ or } \ \mathit{head}(\sigma) = \mathsf{q}!\ell(\mathsf{S}') \qquad \text{(by (36))} \tag{48}$$

Now, from (48), (36) and (37),

$$\text{letting } \sigma'_0 \text{ such that } \sigma' = \mathsf{q}!\ell(\mathsf{S}').\sigma'_0 \text{ and } \sigma_0 = \begin{cases} \epsilon & \text{if } \sigma = \epsilon \\ \mathit{tail}(\sigma) & \text{otherwise} \end{cases}$$

for all $\mathsf{W}_1, \mathsf{W}_2$ in (23), $\exists i \in I, \mathcal{D}^{(\mathsf{r})}, \ell_i, \mathsf{S}_i, \mathsf{S}'_i, \mathsf{W}, \mathsf{W}'$, such that:

$$\mathsf{W}_1 = \mathsf{q}!\ell(\mathsf{S}').\sigma'_0.\mathsf{r}?\ell_i(\mathsf{S}'_i).\mathsf{W}' \quad \text{and} \quad \mathsf{W}_2 = \mathsf{q}!\ell(\mathsf{S}).\sigma_0.\mathcal{D}^{(\mathsf{r})}.\mathsf{r}?\ell_i(\mathsf{S}_i).\mathsf{W} \tag{49}$$

$$\mathsf{S}_i \leq: \mathsf{S}'_i \quad \text{and} \quad \sigma'_0.\mathsf{r}?\ell_i(\mathsf{S}'_i).\mathsf{W}' \lesssim \sigma_0.\mathcal{D}^{(\mathsf{r})}.\mathsf{r}?\ell_i(\mathsf{S}_i).\mathsf{W} \tag{50}$$

where $\mathcal{D}^{(\mathsf{r})}$ is a sequence of outputs to any participant, or inputs from any participant except r, for which (50) can be derived with 0 or more instances of [REF-$\mathcal{A}$] or [REF-$\mathcal{B}$]. Notice that, by induction on σ'_0 and $\sigma_0.\mathcal{D}^{(\mathsf{r})}$, for each pair of SISO trees related in (50) we prove that:

$$\sigma'_0.\mathsf{r}?\ell_i(\mathsf{S}'_i).\mathsf{W}' \lesssim \sigma_0.\mathsf{r}?\ell_i(\mathsf{S}_i).\mathcal{D}^{(\mathsf{r})}.\mathsf{W} \tag{51}$$

Now, by Prop. C.7, the fair paths of Γ that match some witness W_2 in (49) eventually perform the reduction α in (47). But then, consider the session tree T^* that only has SISO trees like the RHS of (51): such trees are similar to W_2 in (49), except that each one performs the input $\mathsf{r}?\ell_i(\mathsf{S}_i)$ earlier. And now, consider the typing environment $\Gamma^* = \Gamma\{\mathsf{p} \mapsto (\sigma, \mathsf{T}^*)\}$. Such Γ^* is p-live: it realises (part of) the fair paths of Γ, except that it consumes r's queued output earlier — and thus, we have $\Gamma^* \in \mathcal{L}$. Now, consider Γ''' such that $\Gamma^* \xrightarrow{\mathsf{p}:\mathsf{r}?\ell_i} \Gamma'''$: by clause [LP⟶] of Def. C.3, we have $\Gamma''' \in \mathcal{L}$. Also, we have $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}^{**})$ such that:

$$\forall \mathsf{U}' \in [\![\sigma'{\cdot}\mathsf{T}'_i]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}^{**} \in [\![\sigma{\cdot}\mathsf{T}^{**}]\!]_{\mathsf{SI}} \\ \exists \mathsf{W}_1 = \sigma'_0.\mathsf{W}' \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}^* = \sigma_0.\mathcal{D}^{(\mathsf{r})}.\mathsf{W} \in [\![\mathsf{V}^{**}]\!]_{\mathsf{SO}} \quad \mathsf{W}_1 \lesssim \mathsf{W}^* \qquad \text{(by Def. 4.3 and (51))}$$

Therefore, by Def. 3.4, we have $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}^{**})$, and $\Gamma'' = \Gamma'''\{\mathsf{p} \mapsto (\sigma', \mathsf{T}'_i)\}$, with $\sigma'{\cdot}\mathsf{T}'_i \leqslant \sigma{\cdot}\mathsf{T}^{**}$ — hence, by (30) and (19), we also have $\Gamma'' \in \mathcal{L}' \subseteq \mathcal{P}$. Thus, we conclude that Γ' satisfies clause [LP⟶] of Def. C.3;

(iii) one of p's queued outputs in σ' is received by another participant. We have two possibilities:

(c) the received output is $\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}})$ (for some $\mathsf{r} \neq \mathsf{q}$). In this case, we have:

$$\exists I, \ell_{\mathsf{r}}, \mathsf{S}'_{\mathsf{r}}\sigma'' : \sigma' \equiv \mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}){\cdot}\sigma'' \tag{52}$$

$$\exists I, \ell_{\mathsf{r}}, \mathsf{S}'_{\mathsf{r}}, \sigma'_0, \sigma'_1 : \sigma' = \sigma'_0{\cdot}\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}){\cdot}\sigma'_1 \text{ with } \mathsf{r}! \notin \mathtt{act}(\sigma'_0) \qquad \text{(by (52) and queue congruence)} \tag{53}$$

$$\Gamma' \xrightarrow{\alpha} \Gamma'' \quad \text{with } \alpha = \mathsf{r}{:}\mathsf{p}?\ell_{\mathsf{r}} \tag{54}$$

for all $\mathsf{W}_1, \mathsf{W}_2$ in (23), $\mathcal{B}^{(\mathsf{r})}, \ell_{\mathsf{r}}, \mathsf{S}_{\mathsf{r}}, \mathsf{S}'_{\mathsf{r}}, \mathsf{W}, \mathsf{W}''$, such that:

$$\mathsf{W}_1 = \sigma'_0.\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}).\sigma'_1.\mathsf{W}' \quad \text{and} \quad \mathsf{W}_2 = \mathcal{B}^{(\mathsf{r})}.\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}_{\mathsf{r}}).\mathsf{W}'' \tag{55}$$

$$\mathsf{S}'_{\mathsf{r}} \leq: \mathsf{S}_{\mathsf{r}} \quad \text{and} \quad \sigma'_0.\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}).\sigma'_1.\mathsf{W}' \lesssim \mathcal{B}^{(\mathsf{r})}.\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}_{\mathsf{r}}).\mathsf{W}'' \tag{56}$$

Therefore, by (56), and by induction on σ'_0 using (53),

$$\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}).\sigma'_0.\sigma'_1.\mathsf{W}' \lesssim \sigma'_0.\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}).\sigma'_1.\mathsf{W}' \tag{57}$$

Notice that, by induction on $\mathcal{B}^{(\mathsf{r})}$, for each pair of SISO trees related in (56) we prove that:

$$\sigma'_0.\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}).\sigma'_1.\mathsf{W}' \lesssim \mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}_{\mathsf{r}}).\mathcal{B}^{(\mathsf{r})}.\mathsf{W}'' \tag{58}$$

From which we get:

$$\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}'_{\mathsf{r}}).\sigma'_0.\sigma'_1.\mathsf{W}' \lesssim \mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}_{\mathsf{r}}).\mathcal{B}^{(\mathsf{r})}.\mathsf{W}'' \text{ (by (57), (58), and Lemma B.9)} \tag{59}$$

$$\sigma'_0.\sigma'_1.\mathsf{W}' \lesssim \mathcal{B}^{(\mathsf{r})}.\mathsf{W}'' \qquad \text{(by (59) and [REF-OUT])} \tag{60}$$

Now, by Prop. C.7, the fair paths of Γ that match some witness W_2 in (55) eventually perform the reduction α in (54). But then, consider the session tree T^* that only has SISO trees like the RHS of (58): such trees are similar to (55), except that each one performs the output $\mathsf{r}!\ell_{\mathsf{r}}(\mathsf{S}_{\mathsf{r}})$ earlier. And now, consider the typing environment $\Gamma^* = \Gamma\{\mathsf{p} \mapsto (\sigma, \mathsf{T}^*)\}$. Such Γ^* is p-live: it realises (part of) the fair paths of Γ, except that performs the output to r earlier — and thus, we have $\Gamma^* \in \mathcal{L}$. Now, consider Γ''' such that $\Gamma^* \xrightarrow{\mathsf{r}:\mathsf{p}?\ell_{\mathsf{r}}} \Gamma'''$: by clause [LP⟶] of Def. C.3, we have $\Gamma''' \in \mathcal{L}$. Also, we have $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}^{**})$ such that:

$$\forall \mathsf{U}' \in [\![\sigma'{\cdot}\mathsf{T}'_i]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}^{**} \in [\![\sigma{\cdot}\mathsf{T}^{**}]\!]_{\mathsf{SI}} \\ \exists \mathsf{W}_1 = \sigma'_0.\sigma'_1.\mathsf{W}' \in [\![\mathsf{U}']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}^* = \mathcal{B}^{(\mathsf{r})}.\mathsf{W}'' \in [\![\mathsf{V}^{**}]\!]_{\mathsf{SO}} \quad \mathsf{W}_1 \lesssim \mathsf{W}^* \qquad \text{(by Def. 4.3 and (60))}$$

Therefore, by Def. 3.4, we have $\Gamma'''(\mathsf{p}) = (\sigma, \mathsf{T}^{**})$, and $\Gamma'' = \Gamma'''\{\mathsf{p} \mapsto (\sigma', \mathsf{T}'_i)\}$, with $\sigma'{\cdot}\mathsf{T}'_i \leqslant \sigma{\cdot}\mathsf{T}^{**}$ — hence, by (30) and (19), we also have $\Gamma'' \in \mathcal{L}' \subseteq \mathcal{P}$. Thus, we conclude that Γ' satisfies clause [LP⟶] of Def. C.3;

(d) the received output is $\mathsf{q}!\ell(\mathsf{S}')$ from (36). The proof is similar to case *(c)* above, letting $\mathsf{r} = \mathsf{q}$ — but the development is simpler, since we have $\sigma'_0 = \epsilon$, and $\mathcal{B}^{(\mathsf{q})}$ is empty;

- case [REF-$\mathcal{B}$]. In this case, we have:

$$\exists I, \ell_i, \mathsf{S}'_i, \mathsf{T}'_i :\ \sigma'{\cdot}\mathsf{T}' = \bigoplus_{i \in I} \mathsf{q}!\ell_i(\mathsf{S}'_i).\mathsf{T}'_i \tag{61}$$

for all $\mathsf{W}_1, \mathsf{W}_2$ in (23), $\exists i \in I, \mathcal{B}^{(\mathsf{q})}, \ell_i, \mathsf{S}_i, \mathsf{S}'_i, \mathsf{W}, \mathsf{W}'$, such that:

$$\mathsf{W}_1 = \mathsf{q}!\ell_i(\mathsf{S}'_i).\mathsf{W}' \quad \text{and} \quad \mathsf{W}_2 = \mathcal{B}^{(\mathsf{q})}.\mathsf{q}!\ell_i(\mathsf{S}_i).\mathsf{W} \tag{62}$$

$$\mathsf{S}'_i \leq: \mathsf{S}_i \quad \text{and} \quad \mathsf{W}' \lesssim \mathcal{B}^{(\mathsf{q})}.\mathsf{W} \quad \text{and} \quad \mathtt{act}(\mathsf{W}') = \mathtt{act}(\mathcal{B}^{(\mathsf{q})}.\mathsf{W}) \tag{63}$$

We now show that Γ' satisfies all clauses of Def. C.3:

- clause [LP&]. The proof is similar to case [REF-OUT][LP&] above;
- clause [LP$\oplus$]. The proof is similar to case [REF-OUT][LP$\oplus$] above;
- clause [LP$\longrightarrow$]. Assume $\Gamma' \longrightarrow \Gamma''$. We have two possibilities:

(a) the reduction does *not* involve p. The proof is similar to case [REF-IN][LP$\longrightarrow$](a) above;

(b) the reduction *does* involve p. There are three sub-cases:

(i) p is enqueuing an output toward some participant r. In this case, we have:

$$\mathsf{T}' \ = \ \bigoplus_{i \in J} \mathsf{r}!\ell_j(\mathsf{S}'_j).\mathsf{T}''_j$$

and the proof is similar to case [REF-OUT][LP$\longrightarrow$](b)(i) above;

(ii) p is receiving an input from some r. The proof is similar to case [REF-OUT][LP$\longrightarrow$](b)(ii) above;

(iii) one of p's queued outputs in σ' is received by another participant. The proof is similar to case [REF-OUT][LP$\longrightarrow$](b)(iii) above.

Summing up: we have proved that each element of $\mathcal{P}$ in (19) satisfies the clauses of Def. C.3 for participant p, which implies that $\mathcal{P}$ is a p-liveness property. Moreover, by (19), we have that for any Γ, if $\Gamma, \mathsf{p}{:}(\sigma, \mathsf{T})$ is p-live and $\mathsf{T}' \leqslant \mathsf{T}$, then $\Gamma, \mathsf{p}{:}(\sigma, \mathsf{T}') \in \mathcal{P}' \subseteq \mathcal{P}$. Therefore, we conclude that $\Gamma, \mathsf{p}{:}(\sigma, \mathsf{T}')$ is p-live. ◀

▶ **Proposition C.12.** *Assume that Γ is live, and that $\Gamma(\mathsf{p}) = (\sigma, \mathsf{T})$. If $\sigma'{\cdot}\mathsf{T}' \leqslant \sigma{\cdot}\mathsf{T}$, then $\Gamma\{\mathsf{p} \mapsto (\sigma', \mathsf{T}')\}$ is live.*

Proof. By Def. C.3, we need to show that Γ' is q-live for all participants $\mathsf{q} \in \mathit{dom}(\Gamma') = \mathit{dom}(\Gamma)$. By hypothesis, we know that Γ is q-live for all participants $\mathsf{q} \in \mathit{dom}(\Gamma)$. By Lemma C.11, we know that Γ' is p-live — and in particular, Γ' is in the p-liveness property $\mathcal{P}$ defined in (19). We are left to prove that Γ' is q-live for all other participants $\mathsf{q} \neq \mathsf{p}$.

By contradiction, assume that Γ' is *not* q-live for some $\mathsf{q} \in \mathit{dom}(\Gamma')$. This means that we can find some Γ''' such that $\Gamma' \longrightarrow^* \Gamma'''$ and Γ''' is not q-live because it violates clause [LP&] or [LP$\oplus$] of Def. C.3. Now, consider the p-liveness property $\mathcal{P}$ in (19): since $\mathcal{P}$ contains Γ, it also contains Γ''' (by iteration of clause [LP$\longrightarrow$] of Def. C.3), and by (19), there exists a corresponding p-live Γ'' such that:

$$\Gamma''' \setminus \mathsf{p} = \Gamma'' \setminus \mathsf{p} \tag{64}$$

$$\Gamma'''(\mathsf{p}) = (\sigma''', \mathsf{T}''') \ \text{ and } \ \Gamma''(\mathsf{p}) = (\sigma'', \mathsf{T}'') \ \text{ such that } \ \sigma'''{\cdot}\mathsf{T}''' \leqslant \sigma''{\cdot}\mathsf{T}'' \tag{65}$$

Let us examine the two (non-mutually exclusive) cases that can make Γ''' not q-live:

- Γ''' violates clause [LP&] of Def. C.3. This means that in Γ''' there is a participant r with a top-level external choice from some participant q, but some fair path of Γ''' never enqueues a corresponding output by q. In this case, similarly to the proof of Prop. C.11 (case [REF-$\mathcal{A}$][LP&]), we conclude that Γ'' is *not* r-live, hence not live — contradiction;
- Γ''' violates clause [LP$\oplus$] of Def. C.3. This means that in Γ''' there is a participant r with a top-level queued output toward participant q, but some fair path of Γ''' where q never consumes the message. In this case, similarly to the proof of Prop. C.11 (case [REF-OUT][LP$\oplus$]), we conclude that Γ'' is *not* r-live, hence not live — contradiction.

Summing up: if we assume that Γ' is *not* q-live for some $\mathsf{q} \in \mathit{dom}(\Gamma')$, then we derive a contradiction. Therefore, we obtain that Γ' is q-live for all $\mathsf{q} \in \mathit{dom}(\Gamma')$. Thus, by Def. C.3, we conclude that Γ' is live. ◀

▶ **Lemma C.13.** *If Γ is live and $\Gamma' \leqslant \Gamma$, then Γ' is live.*

Proof. Assume $\mathit{dom}(\Gamma) \cap \mathit{dom}(\Gamma') = \{\mathsf{p}_1, \mathsf{p}_2, \ldots, \mathsf{p}_n\}$. We first first show that:

$$\Gamma_i \;=\; \Gamma\{\mathsf{p}_1 \mapsto \Gamma'(\mathsf{p}_1)\}\ldots\{\mathsf{p}_2 \mapsto \Gamma'(\mathsf{p}_2)\}\ldots\{\mathsf{p}_i \mapsto \Gamma'(\mathsf{p}_i)\} \quad \text{is live, for all } i \in 0..n$$

We proceed by induction on $i \in 0..n$. The base case $i = 0$ is trivial: we apply no updates to Γ, which is live by hypothesis.

In the inductive case $i = m + 1$, we have (by the induction hypothesis) that Γ_m is live. By Definition of $\Gamma' \leqslant \Gamma$ (see page 8), we know that $\Gamma'(\mathsf{p}_i) \leqslant \Gamma(\mathsf{p}_i)$. Therefore, by Prop. C.12, we obtain that Γ_i is live.

To conclude the proof, consider the set $\mathit{dom}(\Gamma') \setminus \mathit{dom}(\Gamma_n) = \mathit{dom}(\Gamma') \setminus \mathit{dom}(\Gamma) = \{\mathsf{q}_1, \mathsf{q}_2, \ldots, \mathsf{q}_k\}$: it contains all participants that are in Γ', but not in Γ. By Definition of $\Gamma' \leqslant \Gamma$ (see page 8) we know that $\forall i \in 1..k : \Gamma'(\mathsf{q}_i) \equiv (\epsilon, \mathtt{end})$. Therefore, if we extend Γ_n by adding an entry $\mathsf{q}_i : (\epsilon, \mathtt{end})$ for each $i \in 1..k$, we obtain an environment Γ'' such that $\Gamma'' \equiv \Gamma_n$ and $\Gamma'' \equiv \Gamma'$ — hence, $\Gamma_n \equiv \Gamma'$. Therefore, since Γ_n is live, by Lemma C.1 we conclude that Γ' is live. ◀

C.3 Proofs of Subject Reduction and Type Safety

▶ **Lemma C.14** (Typing Inversion). *Let $\Theta \vdash P : \mathsf{T}$: Then,*

1. $P = \mu X.P_1$ *implies* $\Theta, X : \mathsf{T}_1 \vdash P_1 : \mathsf{T}_1$ *and* $\mathsf{T}_1 \leqslant \mathsf{T}$ *for some* T_1;
2. $P = \sum_{i\in I} \mathsf{q}?\ell_i(x_i).P_i$ *implies*
   a. $\&_{i\in I} \mathsf{q}?\ell_i(\mathsf{S}_i).\mathsf{T}_i \leqslant \mathsf{T}$ *and*
   b. $\forall i \in I\ \Theta, x_i : \mathsf{S}_i \vdash P_i : \mathsf{T}_i$;
3. $P = \mathsf{q}!\ell\langle \mathsf{e}\rangle.P_1$ *implies*
   a. $\mathsf{q}!\ell(\mathsf{S}_1).\mathsf{T}_1 \leqslant \mathsf{T}$ *and*
   b. $\Theta \vdash \mathsf{e} : \mathsf{S}$ *and* $\mathsf{S}_1 \leq: \mathsf{S}$ *and*
   c. $\Theta \vdash P_1 : \mathsf{T}_1$;
4. $P = \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$ *implies*
   a. $\Theta \vdash \mathsf{e} : \mathtt{bool}$ *and*
   b. $\Theta \vdash P_1 : \mathsf{T}$
   c. $\Theta \vdash P_2 : \mathsf{T}$

Let $\vdash h : \sigma$. *Then:*

5. $h = \varnothing$ *implies* $\sigma = \epsilon$

6. $h = (\mathsf{q}, \ell(\mathsf{v}))\cdot h_1$ *implies* $\vdash \mathsf{v} : \mathsf{S}$ *and* $\sigma \equiv \mathsf{q}!\ell(\mathsf{S})\cdot\sigma'$ *and* $\vdash h_1 : \sigma'$.

Let $\Gamma \vdash \mathcal{M}$. *Then:*

7. *If* $\Gamma \vdash \prod_{i\in I}(\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i)$ *then*

a. $\forall i \in I :\ \vdash P_i : \mathsf{T}_i$ *and*

b. $\forall i \in I :\ \vdash h_i : \sigma_i$ *and*

c. $\Gamma = \{\mathsf{p}_i : (M_i, \mathsf{T}_i) : i \in I\}$

Proof. The proof is by induction on type derivations. ◀

▶ **Lemma C.15** (Typing congruence). **1.** *If* $\Theta \vdash P : \mathsf{T}$ *and* $P \equiv Q$ *then* $\Theta \vdash Q : \mathsf{T}$.

2. *If* $\vdash h_1 : \sigma_1$ *and* $h_1 \equiv h_2$ *then there is* σ_2 *such that* $\sigma_1 \equiv \sigma_2$ *and* $\vdash h_2 : \sigma_2$.

3. *If* $\Gamma' \vdash \mathcal{M}'$ *and* $\mathcal{M} \equiv \mathcal{M}'$, *then there is* Γ *such that* $\Gamma \equiv \Gamma'$ *and* $\Gamma \vdash \mathcal{M}$.

Proof. The proof is by case analysis. ◀

▶ **Lemma C.16** (Substitution). *If* $\Theta, x : \mathsf{S} \vdash P : \mathsf{T}$ *and* $\Theta \vdash \mathsf{v} : \mathsf{S}$, *then* $\Theta \vdash P\{\mathsf{v}/x\} : \mathsf{T}$.

Proof. By structural induction on P. ◀

▶ **Lemma C.17.** *If* $\emptyset \vdash \mathsf{e} : \mathsf{S}$ *then there is* v *such that* $\mathsf{e} \downarrow \mathsf{v}$.

Proof. The proof is by induction on the derivation of $\emptyset \vdash \mathsf{e} : \mathsf{S}$. ◀

▶ **Lemma C.18.** *Let* $\vdash h : \sigma$. *If* $h \not\equiv (\mathsf{p}, -(-)) \cdot h'$ *then* $\sigma \not\equiv \mathsf{p}! - (-) \cdot \sigma'$.

Proof. The proof is by contrapositive: assume $\sigma \equiv \mathsf{p}! - (-) \cdot \sigma'$. Then, by induction on the derivation of $\sigma \equiv \mathsf{p}! - (-) \cdot \sigma'$, we may show that $h \equiv (\mathsf{p}, -(-)) \cdot h'$. ◀

▶ **Lemma C.19.** *If* $\Theta \vdash P : \mathsf{T}$ *then there is* T' *such that* $\mathsf{T}' \leqslant \mathsf{T}$ *and* $\Theta \vdash P : \mathsf{T}'$ *and* $\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(P)$.

Proof. By induction on $\Theta \vdash P : \mathsf{T}$. The only interesting case is [T-COND]. In this case, we have that $\Theta \vdash \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 : \mathsf{T}$ is derived from $\Theta \vdash \mathsf{e} : \mathtt{bool}$, $\Theta \vdash P_1 : \mathsf{T}$ and $\Theta \vdash P_2 : \mathsf{T}$. By induction hypothesis we derive that there exist T_1 and T_2 such that

$$\Theta \vdash P_1 : \mathsf{T}_1 \quad\text{and}\quad \mathsf{T}_1 \leqslant \mathsf{T} \quad\text{and}\quad \mathtt{act}(\mathsf{T}_1) \subseteq \mathtt{act}(P_1) \tag{66}$$

$$\Theta \vdash P_2 : \mathsf{T}_2 \quad\text{and}\quad \mathsf{T}_2 \leqslant \mathsf{T} \quad\text{and}\quad \mathtt{act}(\mathsf{T}_2) \subseteq \mathtt{act}(P_2) \tag{67}$$

We will show that there is T' such that $\mathsf{T}_1 \leqslant \mathsf{T}' \leqslant \mathsf{T}$ and $\mathsf{T}_2 \leqslant \mathsf{T}' \leqslant \mathsf{T}$ and

$$\mathtt{act}(\mathsf{T}') \ \subseteq\ \mathtt{act}(\mathsf{T}_1) \cup \mathtt{act}(\mathsf{T}_2)\ (\subseteq\ \mathtt{act}(P_1) \cup \mathtt{act}(P_1)\ =\ \mathtt{act}(P))$$

Let us now expand the derivations of $\mathsf{T}_1 \leqslant \mathsf{T}$ and $\mathsf{T}_2 \leqslant \mathsf{T}$ given in (66) and (67):

$$\forall \mathsf{U}_1 \in [\![\mathsf{T}_1]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}' \in [\![\mathsf{T}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_1 \in [\![\mathsf{U}_1]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_1' \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \qquad \mathsf{W}_1 \lesssim \mathsf{W}_1' \tag{68}$$

$$\forall \mathsf{U}_2 \in [\![\mathsf{T}_2]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}' \in [\![\mathsf{T}]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_2 \in [\![\mathsf{U}_2]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_2' \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \qquad \mathsf{W}_1 \lesssim \mathsf{W}_2' \tag{69}$$

Let us consider the sets:

- A_1 as the set of all actions occurring in any U_1 in (68);
- A_2 as the set of all actions occurring in any U_2 in (69);
- A' as the set of all actions occurring in any V' in (68) (or equivalently in (69)).

Notice that $\mathtt{act}(\mathsf{T}_1) = A_1, \mathtt{act}(\mathsf{T}_2) = A_2$ and $\mathtt{act}(\mathsf{T}) = A'$.

Furthermore, let us also consider the sets:

- B_1 as the set of all actions occurring in any W_1 selected in (68);
- B_2 as the set of all actions occurring in any W_2 selected in (69);
- B'_1 as the set of all actions occurring in any W'_1 selected in (68);
- B'_2 as the set of all actions occurring in any W'_2 selected in (69).

Now we have:

$$B_1 \subseteq A_1 \tag{70}$$
$$B_2 \subseteq A_2 \tag{71}$$
$$B'_1 \cup B'_2 \subseteq A' \tag{72}$$
$$B_1 = B'_1 \quad (\text{from } \mathsf{W}_1 \lesssim \mathsf{W}'_1) \tag{73}$$
$$B_2 = B'_2 \quad (\text{from } \mathsf{W}_2 \lesssim \mathsf{W}'_2) \tag{74}$$

Therefore, by (73) and (74) we have

$$B_1 \cup B_2 = B'_1 \cup B'_2 \tag{75}$$

and thus, by (70), (71), (66) and (67), we obtain

$$B'_1 \cup B'_2 \;\subseteq\; A_1 \cup A_2 \;\subseteq\; \mathtt{act}(P_1) \cup \mathtt{act}(P_2) \;=\; \mathtt{act}(P) \tag{76}$$

Now, consider the set $D = A' \setminus (A_1 \cup A_2)$: it contains all actions that occur in T, but not in T_1 nor T_2. Consider any action $\alpha \in D$: it must belong to some SISO tree W'' which was not selected neither as W'_1 in (68), nor as W'_2 in (69). Therefore, it must belong to some action of some SO tree in $[\![\mathsf{V}']\!]_{\mathsf{SO}}$ that is never selected by T_1 nor T_2. This means that α belongs to some internal choice branches of T that are never selected by T_1 nor T_2. Therefore, if we prune $\mathbb{T}$ (i.e., the syntactic type with tree T) by removing all such internal choice branches, we get a session type $\mathbb{T}''$ with tree $\mathsf{T}'' \leqslant \mathsf{T}$ such that:

$$\forall \mathsf{U}_1 \in [\![\mathsf{T}_1]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}' \in [\![\mathsf{T}'']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_1 \in [\![\mathsf{U}_1]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}'_1 \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \qquad \mathsf{W}_1 \lesssim \mathsf{W}'_1 \tag{77}$$
$$\forall \mathsf{U}_2 \in [\![\mathsf{T}_2]\!]_{\mathsf{SO}} \quad \forall \mathsf{V}' \in [\![\mathsf{T}'']\!]_{\mathsf{SI}} \quad \exists \mathsf{W}_2 \in [\![\mathsf{U}_2]\!]_{\mathsf{SI}} \quad \exists \mathsf{W}'_2 \in [\![\mathsf{V}']\!]_{\mathsf{SO}} \qquad \mathsf{W}_1 \lesssim \mathsf{W}'_2 \tag{78}$$

Hence, $\mathsf{T}_1 \leqslant \mathsf{T}''$ and $\mathsf{T}_2 \leqslant \mathsf{T}''$.

Now let $A'' = \mathtt{act}(\mathsf{T}'')$. If we compute $D' = A'' \setminus (A_1 \cup A_2)$ (i.e., the set of all actions that occur in T'', but not in T_1 nor T_2) using (77) and (78), we obtain $D' \subset D$, because α (and possibly some other actions in D) have been removed by the pruning. By iterating the procedure (i.e., by induction on the number of actions in D), noticing that D is finite (because T is syntax-derived from some $\mathbb{T}$), we will eventually find some $\mathbb{T}'$ with tree T' such that:

$$\mathsf{T}' \leqslant \mathsf{T} \;\text{ and }\; \mathsf{T}_1 \leqslant \mathsf{T}' \;\text{ and }\; \mathsf{T}_2 \leqslant \mathsf{T}'$$
$$\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(\mathsf{T}_1) \cup \mathtt{act}(\mathsf{T}_2)$$
$$\Theta \vdash P : \mathbb{T}'$$
$$\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(P)$$

◀

▶ **Lemma C.20** (Error subject reduction). *If $\mathcal{M} \longrightarrow \mathsf{error}$ then there is no type environment Γ such that Γ is live and $\Gamma \vdash \mathcal{M}$.*

Proof. By induction on derivation $\mathcal{M} \longrightarrow \mathsf{error}$.

Base cases:

[ERR-MISM]: We have

$$\mathcal{M} = \mathsf{p} \triangleleft \sum_{j\in J} \mathsf{q}?\ell_j(x_j).P_j \mid \mathsf{p} \triangleleft h_\mathsf{p} \mid \mathsf{q} \triangleleft P_\mathsf{q} \mid \mathsf{q} \triangleleft (\mathsf{p}, \ell(\mathsf{v}))\cdot h \mid \mathcal{M}_1 \tag{79}$$

$$\forall j \in J : \ell \neq \ell_j \tag{80}$$

$$\mathcal{M}_1 = \prod_{i\in I}(\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{81}$$

Assume to the contrary that there exists Γ such that:

$$\Gamma \vdash \mathcal{M} \tag{82}$$

$$\Gamma \text{ is live} \tag{83}$$

By Lemma C.14.7,

$$\vdash \sum_{j\in J} \mathsf{q}?\ell_j(x_j).P_j : \mathsf{T} \tag{84}$$

$$\vdash h_\mathsf{p} : \sigma_\mathsf{p} \tag{85}$$

$$\vdash P_\mathsf{q} : \mathsf{T}_\mathsf{q} \tag{86}$$

$$\vdash (\mathsf{p}, \ell(\mathsf{v}))\cdot h : \sigma \tag{87}$$

$$\forall i \in I : \ \vdash P_i : \mathsf{T}_i \tag{88}$$

$$\forall i \in I : \ \vdash h_i : \sigma_i \tag{89}$$

$$\Gamma = \{\mathsf{p} : (\sigma_\mathsf{p}, \mathsf{T}), \mathsf{q} : (\sigma, \mathsf{T}_\mathsf{q})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{90}$$

By Lemma C.14.2, there are $\mathsf{T}'_j, \mathsf{S}'_j$ (for $j \in J$) such that:

$$\mathop{\&}_{j\in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j \leqslant \mathsf{T} \tag{91}$$

$$\forall j \in J : \quad x_j : \mathsf{S}'_j \vdash P_j : \mathsf{T}'_j \tag{92}$$

By Lemma C.14.6, there are σ', S such that:

$$\vdash \mathsf{v} : \mathsf{S} \text{ and } \sigma \equiv \mathsf{p}!\ell(\mathsf{S})\cdot\sigma' \text{ and } \vdash h : \sigma' \tag{93}$$

Now, let:

$$\Gamma' = \{\mathsf{p} : (\sigma_\mathsf{p}, \mathop{\&}_{j\in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j), \mathsf{q} : (\mathsf{p}!\ell(\mathsf{S})\cdot\sigma', \mathsf{T}_\mathsf{q})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{94}$$

Then, we have:

$$\Gamma' \leqslant \Gamma \qquad \text{(by (93), (91), (90), (94))} \tag{95}$$

$$\Gamma' \text{ is live} \qquad \text{(by (95) and Lemma C.13).} \tag{96}$$

By (80), (96) and Definition 4.4 we get a contradiction.

[ERR-EVAL]: We have

$$\mathcal{M} = \mathsf{p} \triangleleft \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \mid \mathsf{p} \triangleleft h \mid \mathcal{M}_1 \tag{97}$$

$$\nexists \mathsf{v} : \mathsf{e} \downarrow \mathsf{v} \tag{98}$$

$$\mathcal{M}_1 = \prod_{i \in I} (\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{99}$$

Assume to the contrary that there exists Γ such that:

$$\Gamma \vdash \mathcal{M} \tag{100}$$

$$\Gamma \text{ is live} \tag{101}$$

By Lemma C.14.7,

$$\vdash \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 : \mathsf{T} \tag{102}$$

$$\vdash h : \sigma \tag{103}$$

$$\forall i \in I \quad \vdash P_i : \mathsf{T}_i \tag{104}$$

$$\forall i \in I \quad \vdash h_i : \sigma_i \tag{105}$$

$$\Gamma = \{\mathsf{p} : (\sigma, \mathsf{T})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{106}$$

From (102) and by Lemma C.14.4:

$$\vdash P_1 : \mathsf{T} \tag{107}$$

$$\vdash P_2 : \mathsf{T} \tag{108}$$

$$\vdash \mathsf{e} : \mathtt{bool} \tag{109}$$

By Lemma C.17, there is a value v such that $\mathsf{e} \downarrow \mathsf{v}$, which leads to contradiction with assumption (98).

[ERR-EVAL2]: We have

$$\mathcal{M} = \mathsf{p} \triangleleft \mathsf{q}!\ell\langle \mathsf{e} \rangle.P \mid \mathsf{p} \triangleleft h \mid \mathcal{M}_1 \tag{110}$$

$$\nexists \mathsf{v} : \mathsf{e} \downarrow \mathsf{v} \tag{111}$$

$$\mathcal{M}_1 = \prod_{i \in I} (\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{112}$$

Assume to the contrary that there exists Γ such that:

$$\Gamma \vdash \mathcal{M} \tag{113}$$

$$\Gamma \text{ is live} \tag{114}$$

By Lemma C.14.7,

$$\vdash \mathsf{p} \triangleleft \mathsf{q}!\ell\langle \mathsf{e} \rangle.P : \mathsf{T} \tag{115}$$

$$\vdash h : \sigma \tag{116}$$

$$\forall i \in I \quad \vdash P_i : \mathsf{T}_i \tag{117}$$

$$\forall i \in I \quad \vdash h_i : \sigma_i \tag{118}$$

$$\Gamma = \{\mathsf{p} : (\sigma, \mathsf{T})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{119}$$

From (115) and by Lemma C.14.3:

$$\mathsf{q}!\ell(\mathsf{S}_1).\mathsf{T}_1 \leqslant \mathsf{T} \tag{120}$$

$$\vdash \mathsf{e} : \mathsf{S} \tag{121}$$

$$\mathsf{S}_1 \leq: \mathsf{S} \tag{122}$$

By Lemma C.17, there is a value v such that $\mathsf{e} \downarrow \mathsf{v}$, which leads to contradiction with assumption (111).

[ERR-OPHN]: We have:

$$\mathcal{M} = \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft P_{\mathsf{q}} \mid \mathsf{q} \triangleleft (\mathsf{p}, \ell(\mathsf{v}))\cdot h \mid \mathcal{M}_1 \tag{123}$$
$$q? \notin \mathtt{act}(P) \tag{124}$$
$$\mathcal{M}_1 = \prod_{i\in I}(\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{125}$$

By Lemma C.14.7,

$$\vdash P : \mathsf{T}_{\mathsf{p}} \tag{126}$$
$$\vdash (\mathsf{p}, \ell(\mathsf{v}))\cdot h : \sigma_{\mathsf{q}} \tag{127}$$
$$\vdash P_{\mathsf{q}} : \mathsf{T}_{\mathsf{q}} \tag{128}$$
$$\vdash h_{\mathsf{p}} : \sigma_{\mathsf{p}} \tag{129}$$
$$\forall i \in I : \quad \vdash P_i : \mathsf{T}_i \tag{130}$$
$$\forall i \in I : \quad \vdash h_i : \sigma_i \tag{131}$$
$$\Gamma = \{\mathsf{p} : (\sigma_{\mathsf{p}}, \mathsf{T}_{\mathsf{p}}), \mathsf{q} : (\sigma_{\mathsf{q}}, \mathsf{T}_{\mathsf{q}})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{132}$$

By Lemma C.14.6, there are σ', S' such that:

$$\vdash \mathsf{v} : \mathsf{S}' \quad \text{and} \quad \sigma_{\mathsf{q}} \equiv \mathsf{p}!\ell_k(\mathsf{S}')\cdot\sigma' \quad \text{and} \quad \vdash h : \sigma' \tag{133}$$

Let

$$\Gamma' = \{\mathsf{p} : (\sigma_{\mathsf{p}}, \mathsf{T}_{\mathsf{p}}), \mathsf{q} : (\mathsf{p}!\ell_k(\mathsf{S}')\cdot\sigma', \mathsf{T}_{\mathsf{q}})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{134}$$

Then we have

$$\Gamma \equiv \Gamma' \qquad \text{(by (132) and (137))} \tag{135}$$
$$\Gamma' \text{ is live} \qquad \text{(by Lemma C.1)} \tag{136}$$

By (126) and Lemma C.19 we have that there is T' such that $\mathsf{T}' \leqslant \mathsf{T}_{\mathsf{p}}$ and $\vdash P : \mathsf{T}'$ and $\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(P)$. Now let

$$\Gamma'' = \{\mathsf{p} : (\sigma_{\mathsf{p}}, \mathsf{T}'), \mathsf{q} : (\mathsf{p}!\ell_k(\mathsf{S}')\cdot\sigma', \mathsf{T}_{\mathsf{q}})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{137}$$

Then we have $\Gamma'' \leqslant \Gamma'$, and hence, Γ'' is live (by Lemma C.13). This gives a contradiction with (124), $\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(P)$ and Definition 4.4 (since $\mathsf{q}? \notin \mathtt{act}(\mathsf{T}')$ holds, the message in the queue of q will never be received in any reduction of Γ'').

[ERR-STRV]: We have

$$\mathcal{M} = \mathsf{p} \triangleleft \sum_{j\in J} \mathsf{q}?\ell_j(x_j).P_j \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft P_{\mathsf{q}} \mid \mathsf{q} \triangleleft h_{\mathsf{q}} \mid \mathcal{M}_1 \tag{138}$$
$$\mathsf{p}! \notin \mathtt{act}(P_{\mathsf{q}}) \tag{139}$$
$$h_{\mathsf{q}} \not\equiv (\mathsf{p}, -(-)) \cdot h'_{\mathsf{q}} \tag{140}$$
$$\mathcal{M}_1 = \prod_{i\in I}(\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{141}$$

Assume to the contrary that there exists Γ such that:

$$\Gamma \vdash \mathcal{M} \tag{142}$$
$$\Gamma \text{ is live} \tag{143}$$

By Lemma C.14.7,

$$\vdash \sum_{j \in J} \mathsf{q}?\ell_j(x_j).P_j : \mathsf{T}_\mathsf{p} \tag{144}$$
$$\vdash h_\mathsf{q} : \sigma_\mathsf{q} \tag{145}$$
$$\vdash P_\mathsf{q} : \mathsf{T}_\mathsf{q} \tag{146}$$
$$\vdash h_\mathsf{p} : \sigma_\mathsf{p} \tag{147}$$
$$\forall i \in I : \quad \vdash P_i : \mathsf{T}_i \tag{148}$$
$$\forall i \in I : \quad \vdash h_i : \sigma_i \tag{149}$$
$$\Gamma = \{\mathsf{p} : (\sigma_\mathsf{p}, \mathsf{T}_\mathsf{p}), \mathsf{q} : (\sigma_\mathsf{q}, \mathsf{T}_\mathsf{q})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{150}$$

By Lemma C.14.2, there are $\mathsf{T}'_j, \mathsf{S}'_j$ (for $j \in J$) such that:

$$\&_{j \in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j \leqslant \mathsf{T}_\mathsf{p} \tag{151}$$
$$\forall j \in J : \quad x_j : \mathsf{S}'_j \vdash P_j : \mathsf{T}'_j \tag{152}$$

Now, let:

$$\Gamma' = \{\mathsf{p} : (\sigma_\mathsf{p}, \&_{j \in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j), \mathsf{q} : (\sigma_\mathsf{q}, \mathsf{T}_\mathsf{q})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{153}$$

Then, we have:

$$\Gamma' \leqslant \Gamma \qquad \text{(by (150), (151), (153))} \tag{154}$$
$$\Gamma' \text{ is live} \qquad \text{(by (154) and Lemma C.13)} \tag{155}$$

By (140), (145) and Lemma C.18 we have that $\sigma_\mathsf{q} \not\equiv \mathsf{p}! - (-) \cdot \sigma'$. By (146) and Lemma C.19 there is T' such that $\mathsf{T}' \leqslant \mathsf{T}_\mathsf{q}$ and $\vdash P_\mathsf{q} : \mathsf{T}'$ and $\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(P_\mathsf{q})$. Now let

$$\Gamma'' = \{\mathsf{p} : (\sigma_\mathsf{p}, \&_{j \in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j), \mathsf{q} : (\sigma_\mathsf{q}, \mathsf{T}')\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{156}$$

Then we have $\Gamma'' \leqslant \Gamma'$, and hence, Γ'' is live (by Lemma C.13). This gives a contradiction with (139) and $\mathtt{act}(\mathsf{T}') \subseteq \mathtt{act}(P_\mathsf{q})$ and $\sigma_\mathsf{q} \not\equiv \mathsf{p}! - (-) \cdot \sigma'$ and Definition 4.4 (since $\mathsf{p}! \notin \mathtt{act}(\mathsf{T}')$ and $\sigma_\mathsf{q} \not\equiv \mathsf{p}! - (-) \cdot \sigma'$ hold, none of the active inputs in p will be activated in any reduction of Γ'').

Inductive step:

[R-STRUCT]. Assume that $\mathcal{M} \longrightarrow \mathsf{error}$ is derived from:

$$\mathcal{M} \equiv \mathcal{M}_1 \tag{157}$$
$$\mathcal{M}_1 \longrightarrow \mathsf{error} \tag{158}$$

By the induction hypothesis, there is no live Γ_1 such that $\Gamma_1 \vdash \mathcal{M}_1$. Assume on the contrary that there is live Γ such that $\Gamma \vdash \mathcal{M}$. Then, by Lemma C.15.3, there is Γ_2 such that $\Gamma \equiv \Gamma_2$ and $\Gamma_2 \vdash \mathcal{M}_1$. Since Γ is live, by Lemma C.1 we obtain Γ_2 is live, which is a contradiction with the induction hypothesis. ◀

▶ **Lemma C.21.** *If $\Gamma \equiv \Gamma'$ then $\Gamma \leqslant \Gamma'$.*

Proof. Directly by the definitions of $\Gamma \equiv \Gamma'$ and $\Gamma \leqslant \Gamma'$. ◀

▶ **Theorem 4.5** (Subject Reduction). *Assume $\Gamma \vdash \mathcal{M}$ with Γ live. If $\mathcal{M} \longrightarrow \mathcal{M}'$, then there are live type environments Γ', Γ'' such that $\Gamma'' \leqslant \Gamma$, $\Gamma'' \longrightarrow^* \Gamma'$ and $\Gamma' \vdash \mathcal{M}'$.*

Proof. Assume:

$$\Theta \cdot \Gamma \vdash \mathcal{M} \quad \text{(by hypothesis)} \tag{159}$$
$$\Gamma \text{ is live} \quad \text{(by hypothesis)} \tag{160}$$
$$\mathcal{M} \longrightarrow \mathcal{M}' \quad \text{(by hypothesis)} \tag{161}$$

The proof is by induction on the derivation of $\mathcal{M} \longrightarrow \mathcal{M}'$.
Base cases:
[R-SEND]: We have

$$\mathcal{M} = \mathsf{p} \triangleleft \mathsf{q}!\ell\langle \mathsf{e}\rangle.P \mid \mathsf{p} \triangleleft h \mid \mathcal{M}_1 \tag{162}$$
$$\mathsf{e} \downarrow \mathsf{v} \tag{163}$$
$$\mathcal{M}' = \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h \cdot (\mathsf{q}, \ell(\mathsf{v})) \mid \mathcal{M}_1 \tag{164}$$
$$\mathcal{M}_1 = \prod_{i \in I} (\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{165}$$

By Lemma C.14.7,

$$\vdash \mathsf{q}!\ell\langle \mathsf{e}\rangle.P : \mathsf{T} \tag{166}$$
$$\vdash h : \sigma \tag{167}$$
$$\forall i \in I \quad \vdash P_i : \mathsf{T}_i \tag{168}$$
$$\forall i \in I \quad \vdash h_i : \sigma_i \tag{169}$$
$$\Gamma = \{\mathsf{p} : (\sigma, \mathsf{T})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{170}$$

By Lemma C.14.3, there are T', S' such that:

$$\mathsf{q}!\ell(\mathsf{S}).\mathsf{T}' \leqslant \mathsf{T} \tag{171}$$
$$\vdash \mathsf{e} : \mathsf{S}' \quad \text{and} \quad \mathsf{S}' \leq: \mathsf{S} \tag{172}$$
$$\vdash P : \mathsf{T}' \tag{173}$$

Now, let:

$$\Gamma'' = \{\mathsf{p} : (\sigma, \mathsf{q}!\ell(\mathsf{S}).\mathsf{T}')\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{174}$$
$$\Gamma' = \{\mathsf{p} : (\sigma \cdot \mathsf{q}!\ell(\mathsf{S}), \mathsf{T}')\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{175}$$

Then, we conclude:

$$\Gamma'' \leqslant \Gamma \quad \text{(by (174), (171), and (170))} \tag{176}$$
$$\Gamma'' \text{ is live} \quad \text{(by (176) and Lemma C.13)} \tag{177}$$
$$\Gamma'' \longrightarrow \Gamma' \quad \text{(by (174), (175), and [E-SEND] of Def. 4.3)} \tag{178}$$
$$\Gamma' \text{ is live} \quad \text{(by (177), (178), and Proposition C.2)}$$
$$\Gamma' \vdash \mathcal{M}' \quad \text{((164), (175), and Def. 4.1)}$$

[R-RCV]: We have

$$\mathcal{M} = \mathsf{p} \triangleleft \sum_{j \in J} \mathsf{q}?\ell_j(x_j).P_j \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft P_{\mathsf{q}} \mid \mathsf{q} \triangleleft (\mathsf{p}, \ell_k(\mathsf{v}))\cdot h \mid \mathcal{M}_1 \quad (\text{for some } k \in J) \tag{179}$$

$$\mathcal{M}' = \mathsf{p} \triangleleft P_k\{\mathsf{v}/x_k\} \mid \mathsf{p} \triangleleft h_{\mathsf{p}} \mid \mathsf{q} \triangleleft P_{\mathsf{q}} \mid \mathsf{q} \triangleleft h \mid \mathcal{M}_1 \tag{180}$$

$$\mathcal{M}_1 = \prod_{i \in I} (\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{181}$$

By Lemma C.14.7,

$$\vdash \sum_{j \in J} \mathsf{q}?\ell_j(x_j).P_j : \mathsf{T} \tag{182}$$

$$\vdash (\mathsf{p}, \ell_k(\mathsf{v}))\cdot h : \sigma \tag{183}$$

$$\vdash P_{\mathsf{q}} : \mathsf{T}_{\mathsf{q}} \tag{184}$$

$$\vdash h_{\mathsf{p}} : \sigma_{\mathsf{p}} \tag{185}$$

$$\forall i \in I : \quad \vdash P_i : \mathsf{T}_i \tag{186}$$

$$\forall i \in I : \quad \vdash h_i : \sigma_i \tag{187}$$

$$\Gamma = \{\mathsf{p} : (\sigma_{\mathsf{p}}, \mathsf{T}), \mathsf{q} : (\sigma, \mathsf{T}_{\mathsf{q}})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{188}$$

By Lemma C.14.2, there are $\mathsf{T}'_j, \mathsf{S}'_j$ (for $j \in J$) such that:

$$\mathop{\&}_{j \in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j \leqslant \mathsf{T} \tag{189}$$

$$\forall j \in J : \quad x_j : \mathsf{S}'_j \vdash P_j : \mathsf{T}'_j \tag{190}$$

By Lemma C.14.6, there are σ', S' such that:

$$\vdash \mathsf{v} : \mathsf{S}' \text{ and } \sigma \equiv \mathsf{p}!\ell_k(\mathsf{S}')\cdot\sigma' \tag{191}$$

Now, let:

$$\Gamma'' = \{\mathsf{p} : (\sigma_{\mathsf{p}}, \mathop{\&}_{j \in J} \mathsf{q}?\ell_j(\mathsf{S}'_j).\mathsf{T}'_j), \mathsf{q} : (\mathsf{p}!\ell_k(\mathsf{S}')\cdot\sigma', \mathsf{T}_{\mathsf{q}})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{192}$$

Then, we have:

$$\Gamma'' \leqslant \Gamma \qquad \text{(by (192), (191), (189), (188))} \tag{193}$$

$$\Gamma'' \text{ is live} \qquad \text{(by (193) and Lemma C.13)} \tag{194}$$

Observe that we also have:

$$\mathsf{S}' \leq: \mathsf{S}'_k \tag{195}$$

To prove (195), assume (by contradiction) that $\mathsf{S}' \not\leq: \mathsf{S}'_k$. Then, the premise of [E-RCV] in Def. 4.3 does not hold, and thus, p's external choice in Γ'' cannot possibly synchronise with q's queue type. But then, by Def. 4.4, Γ'' is *not* live, which gives a contradiction with (194). Therefore, it must be the case that (195) holds.

Now, let:

$$\Gamma' = \{\mathsf{p} : (\sigma_{\mathsf{p}}, \mathsf{T}'_k), \mathsf{q} : (\sigma', \mathsf{T}_{\mathsf{q}})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{196}$$

And we conclude:

$$\Gamma'' \longrightarrow \Gamma' \qquad \text{(by (192), (195), (196), and rule [E-RCV] of Def. 4.3)} \tag{197}$$

$$\Gamma' \text{ is live} \qquad \text{(by (194), (197), and Proposition C.2)}$$

$$\Gamma' \vdash \mathcal{M}' \qquad \text{(by (190), (195), (180), Lemma C.16, (196), and Def. 4.1)}$$

<u>[R-COND-T] ([R-COND-F])</u>: We have:

$$\mathcal{M} = \mathsf{p} \triangleleft \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q \mid \mathsf{p} \triangleleft h \mid \mathcal{M}_1 \tag{198}$$

$$\mathcal{M}' = \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft h \mid \mathcal{M}_1 \quad (\mathcal{M}' = \mathsf{p} \triangleleft Q \mid \mathsf{p} \triangleleft h \mid \mathcal{M}_1) \tag{199}$$

$$\mathcal{M}_1 = \prod_{i \in I} (\mathsf{p}_i \triangleleft P_i \mid \mathsf{p}_i \triangleleft h_i) \tag{200}$$

By Lemma C.14.7,

$$\vdash \mathsf{if}\ \mathsf{e}\ \mathsf{then}\ P\ \mathsf{else}\ Q : \mathsf{T} \tag{201}$$

$$\vdash h : \sigma \tag{202}$$

$$\forall i \in I : \quad \vdash P_i : \mathsf{T}_i \tag{203}$$

$$\forall i \in I : \quad \vdash h_i : \sigma_i \tag{204}$$

$$\Gamma = \{\mathsf{p} : (\sigma, \mathsf{T})\} \cup \{\mathsf{p}_i : (\sigma_i, \mathsf{T}_i) : i \in I\} \tag{205}$$

By Lemma C.14.4:

$$\vdash P : \mathsf{T} \tag{206}$$

$$\vdash Q : \mathsf{T} \tag{207}$$

$$\vdash \mathsf{e} : \mathtt{bool} \tag{208}$$

Then, letting $\Gamma' = \Gamma'' = \Gamma$, we have:

$$\Gamma'' \leqslant \Gamma \qquad \text{(by reflexivity of } \leqslant) \tag{209}$$

$$\Gamma'' \longrightarrow^* \Gamma' \tag{210}$$

$$\Gamma' \vdash \mathcal{M}' \qquad \text{(by (199), (202), (206) or (207), and Def. 4.1)} \tag{211}$$

Inductive step:
<u>[R-STRUCT]</u> Assume that $\mathcal{M} \longrightarrow \mathcal{M}'$ is derived from:

$$\mathcal{M} \equiv \mathcal{M}_1 \tag{212}$$

$$\mathcal{M}_1 \longrightarrow \mathcal{M}_1' \tag{213}$$

$$\mathcal{M}' \equiv \mathcal{M}_1' \tag{214}$$

From (159), (212), by Lemma C.15, there is Γ_1 such that

$$\Gamma_1 \equiv \Gamma \tag{215}$$

$$\Gamma_1 \vdash \mathcal{M}_1. \tag{216}$$

By induction hypothesis, there is are live type environments Γ_1', Γ'' such that:

$$\Gamma'' \leqslant \Gamma_1 \ \text{ and } \ \Gamma'' \longrightarrow^* \Gamma_1' \tag{217}$$

$$\Gamma'_1 \vdash \mathcal{M}'_1 \tag{218}$$

Now, by (214) and Lemma C.15, there is a live environment Γ' such that

$$\Gamma' \equiv \Gamma'_1 \;\text{ and }\; \Gamma' \vdash \mathcal{M}' \tag{219}$$

We conclude:

$$\Gamma'' \leqslant \Gamma \qquad \text{(by (217), (215), Lemma C.21 and transitivity of } \leqslant\text{)} \tag{220}$$

$$\Gamma'' \longrightarrow^* \Gamma' \qquad \text{(by (217), (219) and rule [E-STRUCT] of Def. 4.3)} \tag{221}$$

◀

We may now show Type Safety and Progress results, that are both corollaries of Subject Reduction and Error subject reduction results.

▶ **Corollary C.22** (Type Safety and Progress). *Let* $\Gamma \vdash \mathcal{M}$ *with* Γ *live. Then,* $\mathcal{M} \longrightarrow^* \mathcal{M}'$ *implies* $\mathcal{M}' \neq \mathsf{error}$*; also, either* $\mathcal{M}' \equiv \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$*, or* $\exists \mathcal{M}''$ *such that* $\mathcal{M}' \longrightarrow \mathcal{M}'' \neq \mathsf{error}$*.*

Proof. Assume $\Gamma \vdash \mathcal{M}$ with Γ live, and $\mathcal{M} \longrightarrow^* \mathcal{M}'$. By Theorem 4.5 (subject reduction), there is some live Γ' such that:

$$\Gamma' \vdash \mathcal{M}' \tag{222}$$

This implies that $\mathcal{M}'$ cannot be an (untypable) error — which is the first part of the thesis. For the *"also..."* part of the statement, we have two possibilities:

1. $\mathcal{M}' \equiv \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$. This is the thesis; or,
2. $\mathcal{M}' \not\equiv \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$. We have two sub-cases:
   a. there is $\mathcal{M}''$ such that $\mathcal{M}' \longrightarrow \mathcal{M}'' \neq \mathsf{error}$. This is the thesis; or,
   b. there is no $\mathcal{M}''$ such that $\mathcal{M}' \longrightarrow \mathcal{M}'' \neq \mathsf{error}$. This case is impossible, because it means that either:
      i. $\nexists \mathcal{M}'' : \mathcal{M}' \longrightarrow \mathcal{M}''$. This case is impossible. In fact, it would imply that $\mathcal{M}'$ cannot reduce by rules [R-SEND], [R-RCV], [R-COND-T], or [R-COND-F] (possibly via [R-STRUCT]) in Table 2. Since $\mathcal{M}' \not\equiv \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$, this can only happen because $\mathcal{M}'$ has some process stuck on an external choice without a matching message, or has some expression (in conditionals or outputs) that cannot be evaluated. But then, by the rules in Table 2, we have $\mathcal{M}' \longrightarrow \mathsf{error}$ by at least one of the rules [ERR-MISM], [ERR-DLOCK], [ERR-EVAL] or [ERR-EVAL2]. This leads to case 2(b)ii below;
      ii. $\mathcal{M}' \longrightarrow \mathsf{error}$. This case is impossible. In fact, if we admit this case, by (222) and the contrapositive of Lemma C.20, we have that Γ' is not live, which means (by the contrapositive of Proposition C.2) that Γ is not live — contradiction.

◀

D Appendix of Section 5

Step1: subtyping negation

The rules in Table 4 relate two SISO trees when:

- their sets of actions are disjunctive ([N-OUT], [N-INP], [N-OUT-R], [N-INP-R]);

- their top prefixes are both inputs or outputs, targeting the same participant, and the label of the LHS is not equal to the label of the RHS ([N-INP-ℓ], [N-OUT-ℓ]);
- their top prefixes are both inputs or outputs, targeting the same participant, with matching labels followed by mismatching sorts or mismatching continuations ([N-INP-S], [N-OUT-S], [N-INP-W], [N-OUT-W]);
- they both consider input prefixes, targeting the same participant, where the input on the RHS is preceded by a finite number of inputs from other participant, and the label of the LHS is not equal to the label of the RHS ([N-$\mathcal{A}$-ℓ]);
- they both consider input prefixes, targeting the same participant, where the input on the RHS is preceded by a finite number of inputs from other participants, with matching labels followed by mismatching sorts or mismatching continuations ([N-$\mathcal{A}$-S], [N-$\mathcal{A}$-W]);
- top prefix on the LHS is input and the top sequence of the prefixes on the RHS consists of a finite number of inputs from other participants and/or outputs ([N-I-O-1], [N-I-O-2]);
- they both consider output prefixes, targeting the same participant, where the output on the RHS is preceded by a finite number of outputs to other participants and/or inputs, and the label of the LHS is not equal to the label of the RHS ([N-$\mathcal{B}$-ℓ]);
- they both consider output prefixes, targeting the same participant, where the output on the RHS is preceded by a finite number of outputs to other participants and/or inputs, with matching labels followed by mismatching sorts or mismatching continuations ([N-$\mathcal{B}$-S], [N-$\mathcal{B}$-W]).

We aim to prove that for every pair of SISO trees that are not related by coinductevely defined relation $\lesssim$, we can derive that they are related by inductively defined relation $\not\lesssim$. Let us consider all possible pairs $(\mathsf{W}_1, \mathsf{W}'_1)$ of regular SISO trees. We can undoubtedly divide them in cases with $\mathtt{act}(\mathsf{W}_1) = \mathtt{act}(\mathsf{W}'_1)$ and $\mathtt{act}(\mathsf{W}_1) \neq \mathtt{act}(\mathsf{W}'_1)$. In the former case, we make the classification taking for W_1 one of the three possible forms, and list all possible forms of W'_1 with respect to the position of the first appearance of the top (if any) prefix of W_1. In what follows, we list all the pairs we get with such a reasoning.

1. $\mathtt{act}(\mathsf{W}_1) = \mathtt{act}(\mathsf{W}'_1)$
   a. $\mathsf{W}_1 = \mathtt{end}$ and $\mathsf{W}'_1 = \mathtt{end}$;
   b. $\mathsf{W}_1 = \mathsf{p}?\ell(\mathsf{S}).\mathsf{W}'$ and
      i. $\mathsf{W}'_1 = \mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}'$, or
      ii. $\mathsf{W}'_1 = \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}'$, or
      iii. $\mathsf{W}'_1 = \mathsf{q}!\ell'(\mathsf{S}').\mathsf{W}'$ and $\mathsf{p}? \in \mathtt{act}(\mathsf{W}')$, or
      iv. $\mathsf{W}'_1 = \mathcal{A}^{(\mathsf{p})}.\mathsf{q}!\ell'(\mathsf{S}').\mathsf{W}'$ and $\mathsf{p}? \in \mathtt{act}(\mathsf{W}')$;
   c. $\mathsf{W}_1 = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}'$ and
      i. $\mathsf{W}'_1 = \mathsf{p}!\ell'(\mathsf{S}').\mathsf{W}'$, or
      ii. $\mathsf{W}'_1 = \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell'(\mathsf{S}').\mathsf{W}'$;
2. $\mathtt{act}(\mathsf{W}_1) \neq \mathtt{act}(\mathsf{W}'_1)$
   a. $\mathsf{p}? \in \mathtt{act}(\mathsf{W}_1)$ and $\mathsf{p}? \notin \mathtt{act}(\mathsf{W}'_1)$, for some p, or
   b. $\mathsf{p}! \in \mathtt{act}(\mathsf{W}_1)$ and $\mathsf{p}! \notin \mathtt{act}(\mathsf{W}'_1)$, for some p, or
   c. $\mathsf{p}? \notin \mathtt{act}(\mathsf{W}_1)$ and $\mathsf{p}? \in \mathtt{act}(\mathsf{W}'_1)$, for some p, or
   d. $\mathsf{p}! \notin \mathtt{act}(\mathsf{W}_1)$ and $\mathsf{p}! \in \mathtt{act}(\mathsf{W}'_1)$, for some p.

Every pair, except $(\mathtt{end}, \mathtt{end})$ is in the conclusion of at least one rule in Table 4. If we know that a pair $(\mathsf{W}_1, \mathsf{W}'_1)$ is related by $\not\lesssim$, i.e. it can be derived by the rules in Table 4 applying the rules downwards, then the pair can also be verified if we apply the rules upwards.

In a finite number of steps, staring from $(\mathsf{W}_1, \mathsf{W}'_1)$ and applying the rules upwards, we will end by application of a base rule. Consequently, if the upward verification procedure applied on some pair $(\mathsf{W}_1, \mathsf{W}'_1)$ does not terminate, it applies in each step one of the non base rules: [N-INP-W], [N-OUT-W], [N-$\mathcal{A}$-W], [N-$\mathcal{B}$-W].

▶ **Lemma D.1.** *Let* W *and* W' *be SISO trees. If* $\neg(\mathsf{W} \lesssim \mathsf{W}')$ *then* $\mathsf{W} \not\lesssim \mathsf{W}'$ *is derivable.*

Proof. The proof is by contraposition: we shall prove that $\mathsf{W} \not\lesssim \mathsf{W}'$ is not derivable implies $\mathsf{W} \lesssim \mathsf{W}'$. Let us assume that $\mathsf{W} \not\lesssim \mathsf{W}'$ is not derivable. The case $(\mathsf{W}, \mathsf{W}') = (\mathtt{end}, \mathtt{end})$ follows directly. Take $(\mathsf{W}, \mathsf{W}') \neq (\mathtt{end}, \mathtt{end})$ and notice that it certainly appears in a conclusion of a rule in Table 4. Consider now an algorithmic procedure that applies the rules given in Table 4 upwards, starting from $\mathsf{W} \not\lesssim \mathsf{W}'$. In each step of the procedure, we can apply only one of the four rules: [N-INP-W], [N-OUT-W], [N-$\mathcal{A}$-W], or [N-$\mathcal{B}$-W]. Otherwise, $\mathsf{W} \not\lesssim \mathsf{W}'$ is derivable, since all other rules are base cases. Also, each premise that is reached by the procedure has the form $\mathsf{W}_1 \not\lesssim \mathsf{W}'_1$ with $\mathtt{act}(\mathsf{W}_1) = \mathtt{act}(\mathsf{W}'_1)$. In case $\mathtt{act}(\mathsf{W}_1) \neq \mathtt{act}(\mathsf{W}'_1)$, the procedure reaches one of the forms of the conclusion of [N-OUT], [N-INP], [N-OUT-R] [N-INP-R].The procedure terminates only in case it reaches in a premise $\mathtt{end} \not\lesssim \mathtt{end}$ (which happens in case W and W' are finite and related by $\lesssim$).

Let $\mathcal{R}(\mathsf{W}, \mathsf{W}')$ be the minimal (possibly infinite) set of pairs of trees satisfying the following properties:

(1) $(\mathsf{W}, \mathsf{W}') \in \mathcal{R}(\mathsf{W}, \mathsf{W}')$ and

(2) if $(\mathsf{W}_1, \mathsf{W}'_1) \in \mathcal{R}(\mathsf{W}, \mathsf{W}')$ and $\mathsf{W}_1 \not\lesssim \mathsf{W}'_1$ is the conclusion and $\mathsf{W}_2 \not\lesssim \mathsf{W}'_2$ is in the premise of one of the rules [N-INP-W], [N-OUT-W], [N-$\mathcal{A}$-W],[N-$\mathcal{B}$-W] (with all other assumptions in the premise satisfied as well), then $(\mathsf{W}_2, \mathsf{W}'_2) \in \mathcal{R}(\mathsf{W}, \mathsf{W}')$.

It follows directly that $\mathcal{R}(\mathsf{W}, \mathsf{W}')$ complies with the rules in Definition 3.2. ◀

Regular representatives for subtyping negation

In the sequel, we will always consider only regular representatives of SO and SI trees that appear in the definition of the negation of subtyping. Before we adopt that approach, we will prove that whenever there exist a pair of (possibly irregular) representatives $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathsf{V}' \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ with $\mathsf{U} \not\lesssim \mathsf{V}'$, there is also a pair of regular representatives $\mathcal{T}(\mathbb{U}_1) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{V}'_1) \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ such that $\mathcal{T}(\mathbb{U}_1) \not\lesssim \mathcal{T}(\mathbb{V}'_1)$.

We start by proving that for each irregular tree $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ there is a regular tree $\mathsf{U}_1 \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ such that U and U_1 overlap in at least top n levels, for a given n. For that purpose, we introduce two auxiliary functions, $\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, i, \mathbb{T}, \mathbb{T}')$ and $\mathtt{mu}^-(\mathbb{T})$. The function $\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, i, \mathbb{T}, \mathbb{T}')$, with $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$, follows in the tree of $\mathbb{T}$ the pattern determined by top i levels of U and extracts (step by step) a type $\mathbb{U}$ with the tree $\mathcal{T}(\mathbb{U})$ that follows the same pattern. The type $\mathbb{U}$ might not be unique, but all such types have the same top i levels. Each step of the procedure applies one of the three options that are introduced and clarified along the following lines.

(1) If $\mathbb{T} = \mu \mathbf{t}.\mathbb{T}_1$, then

$$\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, i, \mu\mathbf{t}.\mathbb{T}_1, \mathbb{T}') = \mu\mathbf{t}.\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, i, \mathbb{T}_1, \mu\mathbf{t}.\mathbb{T}_1), \text{ for every } i \geq 0;$$

The function goes behind $\mu\mathbf{t}$ and in the same time the forth parameter keeps the information on the form of the μ type (it might be needed later on for unfolding).

(2) If $\mathbb{T} \neq \mu\mathbf{t}.\mathbb{T}_1$, for any $\mathbf{t}$ and $\mathbb{T}_1$, and $i > 0$, then

$$\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, i, \mathbb{T}, \mathbb{T}') = \begin{cases} \mathtt{end} & , \mathsf{U} = \mathtt{end} \\ \mathsf{p}!\ell_j(\mathsf{S}_j).\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}_j, i-1, \mathbb{T}_j, \mathbb{T}') & , \mathsf{U} = \mathsf{p}!\ell_j(\mathsf{S}_j).\mathsf{U}_j, j \in K, \\ & \mathbb{T} = \bigoplus_{k \in K} \mathsf{p}!\ell_k(\mathsf{S}_k).\mathbb{T}_k, \\ \&_{k \in K} \mathsf{p}?\ell_k(\mathsf{S}_k).\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}_k, i-1, \mathbb{T}_k, \mathbb{T}') & , \mathsf{U} = \&_{k \in K} \mathsf{p}?\ell_k(\mathsf{S}_k).\mathsf{U}_k, \\ & \mathbb{T} = \&_{k \in K} \mathsf{p}?\ell_k(\mathsf{S}_k).\mathbb{T}_k, \\ \mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, i, \mathbb{T}_1, \mathbb{T}') & , \mathbb{T} = \mathbf{t}, \mathbb{T}' = \mu\mathbf{t}.\mathbb{T}_1 \end{cases}$$

The function extracts from $\mathbb{T}$ the prefix for $\mathbb{U}$ that will induce the same level in its tree as U. If $\mathbb{T} = \mathbf{t}$, it first applies unfolding, recovering the form for substitution from the forth parameter.

(3) If $\mathbb{T} \neq \mu\mathbf{t}.\mathbb{T}_1$, for any $\mathbf{t}$ and $\mathbb{T}_1$, and $i = 0$, then $\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, 0, \mathbb{T}, \mathbb{T}') = \mathbb{U}_1$ for some $\mathcal{T}(\mathbb{U}_1) \in [\![\mathcal{T}(\mathbb{T}\{\mathbb{T}'''/\mathbf{t}_1\})]\!]_{\mathsf{SO}}$, where $\mathbb{T}' = \mu\mathbf{t}_1.\mathbb{T}''$ and $\mathbb{T}''' = \mu\mathbf{t}_2.\mathbb{T}''\{\mathbf{t}_2/\mathbf{t}_1\}$ (we choose here a fresh name $\mathbf{t}_2$). The choice of $\mathbb{U}_1$ is not unique and we will always choose one that satisfies $\mathtt{act}(\mathsf{U}) \supseteq \mathtt{act}(\mathcal{T}(\mathbb{U}_1))$.

One can notice that the previous procedure might create some terms of the form $\mu\mathbf{t}.\mathbb{U}'$ with $\mathbf{t} \notin \mathbb{U}'$. These terms are cleaned up by $\mathtt{mu}^-(\mathbb{T})$, that is defined as follows:

$$\mathtt{mu}^-(\mathbb{T}) = \begin{cases} \mathtt{end} & \mathbb{T} = \mathtt{end} \\ \bigoplus_{k \in K} \mathsf{p}!\ell_k(\mathsf{S}_k).\mathtt{mu}^-(\mathbb{T}_k) & \mathbb{T} = \bigoplus_{k \in K} \mathsf{p}!\ell_k(\mathsf{S}_k).\mathbb{T}_k \\ \&_{k \in K} \mathsf{p}?\ell_k(\mathsf{S}_k).\mathtt{mu}^-(\mathbb{T}_k) & \mathbb{T} = \&_{k \in K} \mathsf{p}?\ell_k(\mathsf{S}_k).\mathbb{T}_k \\ \mu\mathbf{t}.\mathbb{T}' & \mathbb{T} = \mu\mathbf{t}.\mathbb{T}', \mathbf{t} \in \mathbb{T}' \\ \mathtt{mu}^-(\mathbb{T}') & \mathbb{T} = \mu\mathbf{t}.\mathbb{T}', \mathbf{t} \notin \mathbb{T}' \end{cases}$$

▶ **Lemma D.2.** *If $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ then there is $\mathbb{U}_1$ such that $\mathcal{T}(\mathbb{U}_1) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{U}_1)$ overlaps with U at top n levels.*

Proof. If U is finite, then we choose $\mathbb{U}_1 = \mathsf{U}$. If U is infinite, then we choose $\mathbb{U}_1 = \mathtt{mu}^-(\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, n, \mathbb{T}, \mathbb{T}))$. ◀

In the following two examples we illustrate the procedure on some interesting cases.

▶ **Example D.3.** Take $\mathbb{T} = \mu\mathbf{t}_1.(\mathsf{p}!\ell_1(\mathsf{S}_1).\mathbf{t}_1 \& \mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2)$ and choose $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ such that $\mathsf{U} = \mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mathsf{p}!\ell_3(\mathsf{S}_3)\ldots$ We show here that the procedure introduced above gives a regular $\mathbb{U}_1$ that overlaps with U (at least) in the top 3 levels and $\mathcal{T}(\mathbb{U}_1) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$.

$$\begin{aligned} \mathbb{U}_1' =& \mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, 3, \mathbb{T}, \mathbb{T}) \\ =& \mu\mathbf{t}_1.\mathtt{reg}_{\mathsf{SO}}(\mathsf{U}, 3, \mathsf{p}!\ell_1(\mathbf{t})_1 \& \mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2, \mathbb{T}) \\ =& \mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathtt{reg}_{\mathsf{SO}}(\mathsf{p}!\ell_2(\mathsf{S}_2).\mathsf{p}!\ell_3(\mathsf{S}_3)\ldots, 2, \mathbf{t}_1, \mathbb{T}) \\ =& \mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathtt{reg}_{\mathsf{SO}}(\mathsf{p}!\ell_2(\mathsf{S}_2).\mathsf{p}!\ell_3(\mathsf{S}_3)\ldots, 2, \mathsf{p}!\ell_1(\mathsf{S}_1).\mathbf{t}_1 \& \mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2, \mathbb{T}) \\ =& \mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mathtt{reg}_{\mathsf{SO}}(\mathsf{p}!\ell_3(\mathsf{S}_3)\ldots, 1, \mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2, \mathbb{T}) \\ =& \mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathtt{reg}_{\mathsf{SO}}(\mathsf{p}!\ell_3(\mathsf{S}_3)\ldots, 1, \mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2, \mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2) \\ =& \mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathtt{reg}_{\mathsf{SO}}(\mathsf{p}!\ell_3(\mathsf{S}_3)\ldots, 0, \mathbf{t}_2, \mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_2) \\ =& \mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mu\mathbf{t}_3.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_3. \end{aligned}$$

After erasure of the meaningless μ terms, we get

$$\mathbb{U}_1 = \mathtt{mu}^-(\mathbb{U}') = \mathtt{mu}^-(\mu\mathbf{t}_1.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mu\mathbf{t}_3.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_3)$$

$$
\begin{aligned}
&=\mathtt{mu}^-(\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mu\mathbf{t}_3.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_3)\\
&=\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}!\ell_2(\mathsf{S}_2).\mathtt{mu}^-(\mu\mathbf{t}_2.\mathsf{p}!\ell_3(\mathsf{S}_3).\mu\mathbf{t}_3.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_3)\\
&=\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_2(\mathsf{S}_2).\mathsf{p}!\ell_3(\mathsf{S}_3).\mathtt{mu}^-(\mu\mathbf{t}_3.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_3)\\
&=\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_2(\mathsf{S}_2).\mathsf{p}!\ell_3(\mathsf{S}_3).\mu\mathbf{t}_3.\mathsf{p}!\ell_3(\mathsf{S}_3).\mathbf{t}_3.
\end{aligned}
$$

▶ **Example D.4.** Take $\mathbb{T} = \mu\mathbf{t}.(\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_4(\mathsf{S}_4).\mathbf{t}\&\mathsf{p}!\ell_2(\mathsf{S}_2).\mathbf{t})$ and choose $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ such that $\mathsf{U} = \mathsf{p}!\ell_1(\mathsf{S}_1)\ldots$ We consutruct here a regular $\mathbb{U}_1$ that overlaps with U at the top level and $\mathcal{T}(\mathsf{U}_1) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$.

$$
\begin{aligned}
\mathbb{U}_1' &=\mathtt{reg}_{\mathsf{SO}}(\mathsf{U},1,\mathbb{T},\mathbb{T})\\
&=\mu\mathbf{t}.\mathtt{reg}_{\mathsf{SO}}(\mathsf{U},1,\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_4(\mathsf{S}_4).\mathbf{t}\&\mathsf{p}!\ell_2(\mathsf{S}_2).\mathbf{t},\mathbb{T})\\
&=\mu\mathbf{t}.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathtt{reg}_{\mathsf{SO}}(\mathsf{U},0,\mathsf{p}?\ell_4(\mathsf{S}_4).\mathbf{t},\mathbb{T})
\end{aligned}
$$

We can now choose any $\mathbb{U}_2$ such that

$$
\mathcal{T}(\mathbb{U}_2) \in [\![\mathcal{T}(\mathsf{p}?\ell_4(\mathsf{S}_4).\mu\mathbf{t}_1.(\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_4(\mathsf{S}_4).\mathbf{t}\&\mathsf{p}!\ell_2(\mathsf{S}_2).\mathbf{t}_1))]\!]_{\mathsf{SO}}.
$$

For example, with $\mathbb{U}_2 = \mathsf{p}?\ell_4(\mathsf{S}_4).\mu\mathbf{t}_2.\mathsf{p}!\ell_2(\mathsf{S}_2).\mathbf{t}_2$ we get

$$
\begin{aligned}
\mathbb{U}_1 &=\mathtt{mu}^-(\mu\mathbf{t}.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathbb{U}_2)\\
&=\mathtt{mu}^-(\mu\mathbf{t}.\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_4(\mathsf{S}_4).\mu\mathbf{t}_2.\mathsf{p}!\ell_2(\mathsf{S}_2).\mathbf{t}_2)\\
&=\mathsf{p}!\ell_1(\mathsf{S}_1).\mathsf{p}?\ell_4(\mathsf{S}_4).\mu\mathbf{t}_2.\mathsf{p}!\ell_2(\mathsf{S}_2).\mathbf{t}_2.
\end{aligned}
$$

▶ **Lemma D.5.** *If $\mathsf{V}' \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SO}}$ then there is $\mathbb{V}_1'$ such that $\mathcal{T}(\mathbb{V}_1') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ and $\mathcal{T}(\mathbb{V}_1')$ overlaps with V' at top n levels.*

Proof. The construction is analogous to the one from the previous lemma. ◀

▶ **Corollary D.6.** *Let $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathsf{V}' \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ be such that $\mathsf{U} \not\lesssim \mathsf{V}'$. Then, there are $\mathbb{U}_1$ and $\mathbb{V}_1'$ such that $\mathcal{T}(\mathbb{U}_1) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{V}_1') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ and $\mathcal{T}(\mathbb{U}_1) \not\lesssim \mathcal{T}(\mathbb{V}_1')$.*

Proof. If $\mathsf{U} \not\lesssim \mathsf{V}'$ was derived in n steps ($n \geq 1$), there is k such that prefixes from top n levels of U that appear in V' are placed in the top k levels of V' (those that are considered for the negation derivation). By Lemma D.2 and Lemma D.5, there are $\mathbb{U}_1$ and $\mathbb{V}_1'$ such that $\mathcal{T}(\mathbb{U}_1) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{V}_1') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ such that $\mathcal{T}(\mathbb{U}_1)$ ovelaps with U in top n levels and $\mathcal{T}(\mathbb{V}_1)$ ovelaps with V' in top k levels. It can be derived in n steps that $\mathcal{T}(\mathbb{U}_1) \not\lesssim \mathcal{T}(\mathbb{V}_1')$. ◀

Step2: characteristic process

The proof that characteristic process of $\mathbb{U}$ is typable by $\mathcal{T}(\mathbb{U})$ is exactly the same as in the case of synchronous multiparty sessions (See [21]). We consider only single-output processes and for such processes there is no difference in typing rules. The whole proof is replicated here, adapted to single-output processes.

▶ **Lemma D.7** (Strengthening). *If $\Theta, X : \mathsf{U}' \vdash P : \mathsf{U}$ and $X \notin \mathsf{fv}\{P\}$ then $\Theta \vdash P : \mathsf{U}$.*

▶ **Lemma D.8** (Weakening). *If $\Theta \vdash P : \mathsf{U}$ and $X \notin dom(\Theta)$ then $\Theta, X : \mathsf{U}' \vdash P : \mathsf{U}$.*

▶ **Lemma D.9.** *If $\Theta, X_t : \mathsf{U}_1 \vdash \mathcal{P}(\mathbb{U}) : \mathcal{T}(\mathbb{U}\sigma)$ where $\mathsf{U}_1 = \mathcal{T}(\mathbb{U}_1)$ (for some $\mathbb{U}_1$), and $\sigma = \{{}^{\Theta(X_{t'})}/t' \mid t' \in \mathsf{fv}(\mathbb{U})\}$, then $\Theta, X_t : \mathsf{U}_2 \vdash \mathcal{P}(\mathbb{U}) : \mathcal{T}(\mathbb{U}\sigma')$, for any $\mathsf{U}_2 = \mathcal{T}(\mathbb{U}_2)$ (for some $\mathbb{U}_2$) and $\sigma' = (\sigma \setminus \{{}^{\mathbb{U}_1}/t\}) \cup \{{}^{\mathbb{U}_2}/t\}$.*

Proof. By induction on the structure of $\mathbb{U}$. ◀

▶ **Lemma D.10.** *For every (possibly open) type* $\mathbb{U}$*, there are* Θ *and* σ *such that* $\mathrm{dom}(\Theta) = \{X_{\boldsymbol{t}} \mid \boldsymbol{t} \in \mathsf{fv}(\mathbb{U})\}$ *and* $\Theta \vdash \mathcal{P}(\mathbb{U}) : \mathcal{T}(\mathbb{U}\sigma)$ *, where* σ *is a substitution such that* $\sigma = \{\mathbb{U}_{\boldsymbol{t}}/\boldsymbol{t} \mid \boldsymbol{t} \in \mathsf{fv}(\mathbb{U})$ *and* $\Theta(X_{\boldsymbol{t}}) = \mathcal{T}(\mathbb{U}_{\boldsymbol{t}})\}$.

Proof. By induction on the structure of $\mathbb{U}$.

- $\mathbb{U} \equiv \mathsf{end} : \mathcal{P}(\mathsf{end}) = \mathbf{0}$ and, by [T-0], $\vdash \mathcal{P}(\mathsf{end}) : \mathsf{end}$.
- $\mathbb{U} \equiv \mathbf{t} : \mathcal{P}(\mathbf{t}) = X_{\mathbf{t}}$
  By [T-VAR], $X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}') \vdash X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}')$ for any $\mathbb{U}'$. For $\sigma = \{\mathbb{U}'/\mathbf{t}\}$, we have

$$X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}') \vdash \mathcal{P}(\mathbf{t}) : \mathcal{T}(\mathbb{U}\sigma)\,.$$

- $\mathbb{U} \equiv \mathop{\&}_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathbb{U}_i : \mathcal{P}(\mathbb{U}) = \sum_{i\in I} \mathsf{p}?\ell_i(x_i).\mathsf{if}\ \underline{\mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)}\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i)$
  By the induction hypothesis, $\Theta_i \vdash \mathcal{P}(\mathbb{U}_i) : \mathcal{T}(\mathbb{U}_i\sigma_i)$, where $\sigma_i = \{\mathbb{U}_{\mathbf{t}}/\mathbf{t} \mid \mathbf{t} \in \mathsf{fv}(\mathbb{U}_i)$ and $\mathcal{T}(\mathbb{U}_{\mathbf{t}}) = \Theta_i(X_{\mathbf{t}})\}$ for some Θ_i, and every $i \in I$. Let us denote by Θ the environment consisting of assignments $X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}_{\mathbf{t}})$ for arbitrarily chosen $\mathbb{U}_{\mathbf{t}}$, where $X_{\mathbf{t}} \in \mathit{dom}(\Theta_i)$ for some $i \in I$. By typing rules, $\Theta, x_i : \mathsf{S}_i \vdash \underline{\mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)} : \mathtt{bool}$, for every $i \in I$. By Lemma D.9, [T-COND] and weakening, for each $i \in I$, we have the judgements:

$$\Theta, x_i : \mathsf{S}_i \vdash \mathsf{if}\ \underline{\mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)}\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i) : \mathcal{T}(\mathbb{U}_i\sigma_i')$$

  where $\sigma_i' = \left\{ \mathbb{U}_{\mathbf{t}}/\mathbf{t} \;\middle|\; \begin{array}{l} \mathbf{t} \in \mathsf{fv}(\mathbb{U}_i) \text{ and} \\ \mathcal{T}(\mathbb{U}_{\mathbf{t}}) = \Theta(X_{\mathbf{t}}) \end{array} \right\}$. Now, by [T-EXT], we have

$$\Theta \vdash \sum_{i\in I} \mathsf{p}?\ell(x_i).\mathsf{if}\ \underline{\mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)}\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i) : \mathop{\&}_{i\in I} \mathsf{p}?\ell(\mathsf{S}_i).\,\mathcal{T}(\mathbb{U}_i\sigma_i')\,.$$

  We conclude this case by remarking that $\mathop{\&}_{i\in I} \mathsf{p}?\ell(\mathsf{S}_i).\,\mathcal{T}(\mathbb{U}_i\sigma_i') = \mathcal{T}(\mathbb{U}\sigma)$ for $\sigma = \cup_{i\in I}\sigma_i' = \{\mathbb{U}_{\mathbf{t}}/\mathbf{t} \mid \mathbf{t} \in \mathsf{fv}(\mathbb{U})$ and $\mathcal{T}(\mathbb{U}_{\mathbf{t}}) = \Theta(X_{\mathbf{t}})\}$.

- $\mathbb{U} \equiv \mathsf{p}!\ell(\mathsf{S}).\mathbb{U}' : \mathcal{P}(\mathbb{U}) = \mathsf{p}!\ell\langle\underline{\mathbf{val}(\!|\mathsf{S}|\!)}\rangle.\mathcal{P}(\mathbb{U}')$
  By the induction hypothesis, $\Theta' \vdash \mathcal{P}(\mathbb{U}') : \mathcal{T}(\mathbb{U}'\sigma')$, where $\sigma' = \{\mathbb{U}_{\mathbf{t}}/\mathbf{t} \mid \mathbf{t} \in \mathsf{fv}(\mathbb{U}')$ and $\mathcal{T}(\mathbb{U}_{\mathbf{t}}) = \Theta'(X_{\mathbf{t}})\}$ for some Θ'. Let us denote by Θ the environment consisting of assignments $X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}_{\mathbf{t}})$ for arbitrarily chosen $\mathbb{U}_{\mathbf{t}}$, where $X_{\mathbf{t}} \in \mathit{dom}(\Theta')$.
  By [T-OUT],

$$\Theta' \vdash \mathsf{p}!\ell(\underline{\mathbf{val}(\!|S|\!)}).\mathcal{P}(\mathbb{U}) : \mathsf{p}!\ell(\mathsf{S}).\,\mathcal{T}(\mathbb{U}'\sigma').$$

- $\mathbb{U} \equiv \mu\mathbf{t}.\mathbb{U}' : \mathcal{P}(\mathbb{U}) = \mu X_{\mathbf{t}}.\mathcal{P}(\mathbb{U}')$
  By induction hypothesis, there is Θ' such that

$$\Theta' \vdash \mathcal{P}(\mathbb{U}') : \mathcal{T}(\mathbb{U}'\sigma') \qquad \text{where } \sigma' = \{\mathbb{U}_{\mathbf{t}'}/\mathbf{t}' \mid \mathbf{t}' \in \mathsf{fv}(\mathbb{U}') \text{ and } \mathcal{T}(\mathbb{U}_{\mathbf{t}'}) = \Theta'(X_{\mathbf{t}'})\}$$

  We have two cases:

  (i) $\mathbf{t} \notin \mathsf{fv}(\mathbb{U}')$. In this case, $X_{\mathbf{t}} \notin \mathsf{fv}(\mathcal{P}(\mathbb{U}'))$ and $\Theta'', X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}'') \vdash \mathcal{P}(\mathbb{U}') : \mathcal{T}(\mathbb{U}'\sigma')$, for some $\mathbb{U}''$ (either $\Theta' = \Theta'', X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}'')$ or it is obtained by weakening of Θ'). By [T-REC], we get $\Theta'' \vdash \mu X_{\mathbf{t}}.\mathcal{P}(\mathbb{U}') : \mathcal{T}(\mathbb{U}'\sigma)$ with $\sigma' = \sigma$.

(ii) $\mathbf{t} \in \mathsf{fv}(\mathbb{U}')$. In this case, $X_{\mathbf{t}} \in \mathsf{fv}(\mathcal{P}(\mathbb{U}'))$ and $\Theta' = \Theta'', X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}'')$ for some $\mathbb{U}''$. By Lemma D.9,

$$\Theta'', X_{\mathbf{t}} : \mathcal{T}(\mathbb{U}'\sigma') \vdash \mathcal{P}(\mathbb{U}') : \mathcal{T}(\mathbb{U}'\sigma'')$$
$$\text{where} \quad \sigma'' = \{{}^{\mathbb{U}_{\mathbf{t}'}}/_{\mathbf{t}'} \mid \mathbf{t}' \in \mathsf{fv}(\mathbb{U}') \setminus \{\mathbf{t}\} \text{ and } \mathcal{T}(\mathbb{U}_{\mathbf{t}'}) = \Theta''(X_{\mathbf{t}'})\} \cup \{{}^{\mathbb{U}'\sigma'}/_{\mathbf{t}}\}$$

i.e., the difference between σ' and σ'' is that σ'' contains ${}^{\mathbb{U}'\sigma'}/_{\mathbf{t}}$ instead of ${}^{\mathbb{U}''}/_{\mathbf{t}}$. Then, by [T-REC], we conclude $\Theta'' \vdash \mu X_{\mathbf{t}}.\mathcal{P}(\mathbb{U}') : \mathcal{T}(\mathbb{U}'\sigma)$ where $\sigma = \{{}^{\mathbb{U}_{\mathbf{t}}}/_{\mathbf{t}} \mid \mathbf{t} \in \mathsf{fv}(\mathbb{U})$ and $\mathcal{T}(\mathbb{U}_{\mathbf{t}}) = \Gamma''(X_{\mathbf{t}})\} = \sigma'' \setminus \{{}^{\mathbb{U}'\sigma'}/_{\mathbf{t}}\}$.

◀

▶ **Lemma D.11.** *Let* T *be a session type tree. Then*

(i) $\forall \mathsf{U} \in [\![\mathsf{T}]\!]_{\mathsf{SO}}\ \mathsf{U} \leqslant \mathsf{T}$
(ii) $\forall \mathsf{V}' \in [\![\mathsf{T}']\!]_{\mathsf{SI}}\ \mathsf{T}' \leqslant \mathsf{V}'$

Proof. Follows from the definition of decompositions. ◀

▶ **Proposition D.12.** *For all closed types* $\mathbb{T}$ *and* $\mathbb{U}$*, if* $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ *then* $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{T}$.

Proof. As a direct consequence of Lemma D.10 we get $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{U}$. Since by Lemma D.11 for every $\mathbb{U}$ with $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$, $\mathcal{T}(\mathbb{U}) \leqslant \mathcal{T}(\mathbb{T})$, by [T-SUB], $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{T}$. ◀

Step3: characteristic session

▶ **Proposition D.13.** *Let* $\mathbb{V}'$ *be a SI type and* $\mathsf{r} \notin \mathtt{pt}(\mathbb{V}')$*. Let* Q *be a process such that* $\vdash Q : \mathbb{V}'$*. Then, there is a live typing environment* Γ *(see (4)) such that* $\Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$.

Proof. Follows directly from the construction of the characteristic session types and the definition of the live typing environments. ◀

▶ **Proposition D.14.** *Take any* $\mathbb{T}'$*,* $\mathsf{r} \notin \mathtt{pt}(\mathbb{T}')$*, SI type* $\mathbb{V}'$ *such that* $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$*, and* Q *such that* $\vdash Q : \mathbb{T}'$*. Then, there is a live* Γ *(see (4)) such that* $\Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$.

Proof. By Proposition 5.10, $\vdash Q_1 : \mathbb{V}'$ implies there is a live typing environment Γ'' such that

$$\Gamma'' \vdash \mathsf{r} \triangleleft Q_1 \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$$

where $\Gamma'' = \Gamma', \mathsf{r} : (\varnothing, \mathbb{V}')$. By Lemma D.11 we have $\mathbb{T}' \leqslant \mathbb{V}'$. Since for $\Gamma = \Gamma', \mathsf{r} : (\varnothing, \mathbb{T}')$ we have $\Gamma \leqslant \Gamma''$, by Lemma C.13 we obtain that Γ is also live. Hence, if $\vdash Q : \mathbb{T}'$, then we may show

$$\Gamma \vdash \mathsf{r} \triangleleft Q \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$$

and Γ is live. ◀

Step4: completeness

In order to describe the shape of V' type when $\mathsf{U} \not\leqslant \mathsf{V}'$ is derived using cases that involve context $\mathcal{B}^{(\mathsf{p})}$ (for the corresponding projections that satisfy $\mathsf{W} \not\lesssim \mathsf{W}'$) we define context $\mathcal{C}^{(\mathsf{p})}$ with holes, that (as V) have only single inputs, and, in which there are no outputs on p.

$$\text{[N-UV-OUT-ACT]}\ \frac{\mathsf{p}! \notin \mathsf{act}(\mathsf{V}')}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{U} \nleqslant \mathsf{V}'} \qquad \text{[N-UV-INP-ACT]}\ \frac{\mathsf{p}? \notin \mathsf{act}(\mathsf{V}')}{\&_{i\in I}\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i \nleqslant \mathsf{V}'} \qquad \text{[N-UV-OUT-ACT-R]}\ \frac{\mathsf{p}! \notin \mathsf{act}(\mathsf{U})}{\mathsf{U} \nleqslant \bigoplus_{i\in I}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}'_i} \qquad \text{[N-UV-INP-ACT-R]}\ \frac{\mathsf{p}? \notin \mathsf{act}(\mathsf{U})}{\mathsf{U} \nleqslant \mathsf{p}?\ell(\mathsf{S}).\mathsf{V}'}$$

$$\text{[N-UV-INP]}\ \frac{\forall i \in I : \ell_i \neq \ell \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_i \nleqslant \mathsf{V}'}{\&_{i\in I}\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i \nleqslant \mathsf{p}?\ell(\mathsf{S}).\mathsf{V}'} \qquad \text{[N-UV-}\mathcal{A}\text{]}\ \frac{\forall i \in I : \ell_i \neq \ell \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_i \nleqslant \mathcal{A}^{(\mathsf{p})}.\mathsf{V}'}{\&_{i\in I}\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i \nleqslant \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathsf{V}'}$$

$$\text{[N-UV-IN-OUT-1]}\ \frac{}{\&_{i\in I}\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i \nleqslant \bigoplus_{j\in J}\mathsf{q}!\ell_j(\mathsf{S}_j).\mathsf{V}'_j} \qquad \text{[N-UV-IN-OUT-2]}\ \frac{}{\&_{i\in I}\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i \nleqslant \mathcal{A}^{(\mathsf{p})}.\bigoplus_{j\in J}\mathsf{q}!\ell_j(\mathsf{S}_j).\mathsf{V}'_j}$$

$$\text{[N-UV-OUT]}\ \frac{\forall i \in I : \ell \neq \ell_i \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U} \nleqslant \mathsf{V}'_i}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{U} \nleqslant \bigoplus_{i\in I}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}'_i} \qquad \text{[N-UV-}\mathcal{C}\text{]}\ \frac{\forall n \in N\, \forall i \in I_n : \ell \neq \ell_i \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U} \nleqslant (\mathcal{C}^{(\mathsf{p})} \restriction n)[\mathsf{V}'_i]}{\mathsf{p}!\ell(\mathsf{S}).\mathsf{U} \nleqslant \mathcal{C}^{(\mathsf{p})}[\bigoplus_{i\in I_n}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}'_i]^{n\in N}}$$

Table 8 Shapes of unrelated U and V′ type trees

$$\mathcal{C}^{(\mathsf{p})} ::= [\,]^n \ \mid\ \mathsf{q}?\ell(\mathsf{S}).\mathcal{C}^{(\mathsf{p})} \ \mid\ \bigoplus_{i\in I}\mathsf{r}!\ell_i(\mathsf{S}_i).\mathcal{C}^{(\mathsf{p})} \oplus \bigoplus_{i\in I'}\mathsf{r}!\ell_i(\mathsf{S}_i).\mathsf{V}'_i \qquad \mathsf{r} \neq \mathsf{p} \text{ and } \mathsf{p}! \notin \mathsf{act}(\mathsf{V}'_i)$$

Note that, in case of selection, context $\mathcal{C}^{(\mathsf{p})}$ may have holes only in some branches while the rest of the branches contain no outputs on p. This allows us to determine all the "first" outputs appearing in some V' type tree. We write $\mathcal{C}^{(\mathsf{p})}[\,]^{n\in N}$ to denote a context with holes indexed by elements of N and $\mathcal{C}^{(\mathsf{p})}[\mathsf{V}'_n]^{n\in N}$ to denote the same context when the hole $[\,]^n$ has been filled with V'_n. We index the holes in contexts in order to distinguish them. For the rest of the paper we assume $\mathcal{C}^{(\mathsf{p})}$ are nonempty, i.e., it always holds that $\mathcal{C}^{(\mathsf{p})} \neq [\,]$.

Furthermore, for a context $\mathcal{C}^{(\mathsf{p})}[\,]^{n\in N}$ we may apply a mapping $\mathcal{C}^{(\mathsf{p})} \restriction n$ that produces the projection of the context into the path that leads to the hole indexed with n, i.e., it produces the corresponding $\mathcal{B}^{(\mathsf{p})}.[\,]^n$.

▶ **Lemma D.15.** *If* $\neg(\mathsf{U} \leqslant \mathsf{V}')$ *then* $\mathsf{U} \nleqslant \mathsf{V}'$ *can be derived by the inductive rules given in Table 8.*

Proof. If $\neg(\mathsf{U} \leqslant \mathsf{V}')$ then $\forall \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $\forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ holds $\mathsf{W} \not\lesssim \mathsf{W}'$. Now the proof continues by case analysis of the last applied rules for $\mathsf{W} \not\lesssim \mathsf{W}'$.

- Case [N-OUT]: If $\forall \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $\forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ $\mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-OUT], then from $\mathsf{W} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}_1$ we conclude $\mathsf{U} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{U}_1$, by the definition of $[\![\ ]\!]_{\mathsf{SI}}$. Since $\forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ holds $\mathsf{p}! \notin \mathsf{act}(\mathsf{W}')$, we may conclude $\mathsf{p}! \notin \mathsf{act}(\mathsf{V}')$ (by definition of $[\![\ ]\!]_{\mathsf{SO}}$). Hence, we get that U and V' satisfy the clauses of [N-UV-OUT-ACT].
- Cases [N-INP], [N-OUT-R] and [N-INP-R]: By a similar reasoning as in the previous case we may show that U and V' satisfy the clauses of [N-UV-INP-ACT], [U-UV-OUT-ACT-R] and [N-UV-INP-ACT-R], respectively.
- Cases [N-INP-ℓ], [N-INP-S] and [N-INP-W]: Assume $\exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $\exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ such that $\mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-INP-ℓ], [N-INP-S] or [N-INP-W]. Then, from $\mathsf{W} = \mathsf{p}?\ell(\mathsf{S}).\mathsf{W}_1$ and definition of $[\![\ ]\!]_{\mathsf{SI}}$ we may conclude $\mathsf{U} = \&_{i\in I}\mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i$. Also, from $\mathsf{W}' = \mathsf{p}?\ell'(\mathsf{S}').\mathsf{W}'_1$ and definition of

$[\![\ ]\!]_{\mathsf{SO}}$, we conclude $\mathsf{V}' = \mathsf{p}?\ell'(\mathsf{S}').\mathsf{V}_1'$. Since $\forall \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $\forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ holds $\mathsf{W} \not\lesssim \mathsf{W}'$ we distinguish three subcases:

- $\forall \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}\ \forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}\ \mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-INP-ℓ];
- $\exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}\ \exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}\ \mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-INP-S], but could not be derived by [N-INP-ℓ];
- $\exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}\ \exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}\ \mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-INP-W], but could not be derived by [N-INP-ℓ] and [N-INP-W].

In the first subcase we conclude that $\forall i \in I : \ell_i \neq \ell'$; in the second $\exists i \in I : \ell_i = \ell'$ but $\mathsf{S}' \not\leq: \mathsf{S}_i$; while in the third case $\exists i \in I : \ell_i = \ell'$ and $\mathsf{S}' \leq: \mathsf{S}_i$ but $\forall \mathsf{W}_1 \in [\![\mathsf{U}_i]\!]_{\mathsf{SI}} \forall \mathsf{W}_1' \in [\![\mathsf{V}_1']\!]_{\mathsf{SO}}$ holds $\mathsf{W}_1 \not\lesssim \mathsf{W}_1'$. Thus, we derived that $\forall i \in I : \ell_i \neq \ell' \ \vee\ \mathsf{S}' \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_i \not\lesssim \mathsf{V}'$, which are the clauses of [N-UV-INP].

- Cases [N-$\mathcal{A}$-ℓ], [N-$\mathcal{A}$-S] and [N-$\mathcal{A}$-W]: Follow by a similar reasoning as in the previous item, only deriving that U and V' satisfy rule [N-UV-$\mathcal{A}$].
- Cases [N-I-O-1] and [N-I-O-2]: Assume $\exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $\exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ such that $\mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-I-O-1] or [N-I-O-2]. Using the definition of $[\![\ ]\!]_{\mathsf{SO}}$ and $[\![\ ]\!]_{\mathsf{SI}}$ we conclude $\mathsf{U} = \&_{i\in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i$ and, $\mathsf{V}' = \bigoplus_{j\in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathsf{V}_j'$ or $\mathsf{V}' = \mathcal{A}^{(\mathsf{p})}.\bigoplus_{j\in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathsf{V}_j'$, i.e., U and V' satisfy [N-UV-IN-OUT-1] or [N-UV-IN-OUT-2], respectively.
- Cases [N-OUT-ℓ], [N-OUT-S] and [N-OUT-W]: Follow by a similar reasoning as in the item with [N-INP-ℓ], [N-INP-S] and [N-INP-W], only deriving that U and V' satisfy rule [N-UV-OUT].
- Cases [N-$\mathcal{B}$-ℓ], [N-$\mathcal{B}$-S] and [N-$\mathcal{B}$-W]: Assume $\exists \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ and $\exists \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ such that $\mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-$\mathcal{B}$-ℓ], [N-$\mathcal{B}$-S] or [N-$\mathcal{B}$-W]. Then, $\mathsf{W} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}_1$ and $\mathsf{W}' = \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell'(\mathsf{S}').\mathsf{W}_1'$. By definition of $[\![\ ]\!]_{\mathsf{SI}}$ we directly obtain $\mathsf{U} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{U}_1$. By definition of $[\![\ ]\!]_{\mathsf{SO}}$ we obtain $\mathsf{p}! \in \mathsf{act}(\mathsf{V}')$ and $\mathsf{V}' \neq \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_i'$, i.e., $\forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ either $\mathsf{W}' = \mathcal{B}^{(\mathsf{p})}.\mathsf{p}!\ell''(\mathsf{S}'').\mathsf{W}_1''$ or $\mathsf{p}! \notin \mathsf{act}(\mathsf{W}')$. Thus, we may conclude $\mathsf{V}' = \mathcal{C}^{(\mathsf{p})}[\bigoplus_{i\in I_n} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_i']^{n\in N}$. Furthermore, since $\forall \mathsf{W} \in [\![\mathsf{U}]\!]_{\mathsf{SI}}$ (that have form $\mathsf{W} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}_1$) and $\forall \mathsf{W}' \in [\![\mathsf{V}']\!]_{\mathsf{SO}}$ (that have form $\mathsf{W}' = (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}_i']$ or $\mathsf{p}! \notin \mathsf{act}(\mathsf{W}')$) hold $\mathsf{W} \not\lesssim \mathsf{W}'$, we may distinguish three cases:

  - $\forall \mathsf{W} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}_1\ \forall \mathsf{W}' = (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}_i']\ \mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-$\mathcal{B}$-ℓ];
  - $\exists \mathsf{W} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}_1\ \exists \mathsf{W}' = (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}_i']\ \mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-$\mathcal{B}$-S], but could not be derived by [N-$\mathcal{B}$-ℓ];
  - $\exists \mathsf{W} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{W}_1\ \exists \mathsf{W}' = (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}_i']\ \mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-$\mathcal{B}$-W], but could not be derived by [N-$\mathcal{B}$-ℓ] or [N-$\mathcal{B}$-S].

  In the first case we get that $\forall n \in N \forall i \in I_n\ \ell \neq \ell_i$; in the second that $\exists n \in N \exists i \in I_n$ such that $\ell = \ell_i$, and $\mathsf{S} \not\leq: \mathsf{S}_i$; and in the third that $\exists n \in N \exists i \in I_n$ such that $\ell = \ell_i$ and $\mathsf{S} \leq: \mathsf{S}_i$, but $\mathsf{W} \not\lesssim (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{W}_i']$. Notice that in the third case $\forall \mathsf{W} \in [\![\mathsf{p}!\ell(\mathsf{S}).\mathsf{U}_1]\!]_{\mathsf{SI}}$ and $\forall \mathsf{W}' \in [\![(\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{p}!\ell(\mathsf{S}_i).\mathsf{V}_i']]\!]_{\mathsf{SO}}$, where $\mathsf{S} \leq: \mathsf{S}_i$, we have $\mathsf{W} \not\lesssim \mathsf{W}'$ is derived by [N-$\mathcal{B}$-W]. Then, we may conclude that $\forall \mathsf{W}_1 \in [\![\mathsf{U}_1]\!]_{\mathsf{SI}}$ and $\forall \mathsf{W}_1' \in [\![(\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{V}_i']]\!]_{\mathsf{SO}}$, we have $\mathsf{W}_1 \not\lesssim \mathsf{W}_1'$, i.e., we obtain $\mathsf{U}_1 \not\lesssim (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{V}_i']$. Therefore, we concluded that $\mathsf{U} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{U}_1$, and $\mathsf{V}' = \mathcal{C}^{(\mathsf{p})}[\bigoplus_{i\in I_n} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_i']^{n\in N}$, and that $\forall n \in N \forall i \in I_n : \ell \neq \ell_i \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_1 \not\lesssim (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{V}_i']$, that are the clauses of [N-UV-$\mathcal{C}$].

◀

▶ **Lemma D.16.** *If* $\mathsf{S} \not\leq: \mathsf{S}'$ *there is no* v *such that* $\mathbf{expr}(\!|\mathbf{val}(\!|S|\!), S'|\!) \downarrow \mathsf{v}$.

Proof. By case analysis, we consider expression $\mathbf{expr}(\!|\mathbf{val}(\!|S|\!), S'|\!)$:

- $\mathtt{int} \not\leq: \mathtt{nat} : \mathbf{expr}(\!|-1, \mathtt{nat}|\!) = (\mathsf{succ}(-1) > 0)$;
- $\mathtt{bool} \not\leq: \mathtt{nat} : \mathbf{expr}(\!|\mathsf{true}, \mathtt{nat}|\!) = (\mathsf{succ}(\mathsf{true}) > 0)$;

- $\mathtt{nat} \not\leq: \mathtt{bool}$: $\mathbf{expr}(\!|1, \mathtt{bool}|\!) = \neg 1$;
- $\mathtt{int} \not\leq: \mathtt{bool}$: $\mathbf{expr}(\!|-1, \mathtt{bool}|\!) = \neg(-1)$;
- $\mathtt{bool} \not\leq: \mathtt{int}$: $\mathbf{expr}(\!|\mathsf{true}, \mathtt{int}|\!) = (\mathsf{inv}(\mathsf{true}) > 0)$.

In each case, the expression is undefined since the following expressions are undefined: $\mathsf{succ}(-1), \mathsf{succ}(\mathsf{true}), \neg 1, \neg(-1), \mathsf{inv}(\mathsf{true})$. ◀

▶ **Lemma D.17.** *If* $\mathsf{S} \leq: \mathsf{S}'$ *then* $\mathbf{expr}(\!|\mathbf{val}(\!|S|\!), S'|\!) \downarrow \mathsf{true}$ *or* $\mathbf{expr}(\!|\mathbf{val}(\!|S|\!), S'|\!) \downarrow \mathsf{false}$.

Proof. By case analysis:

- $\mathtt{nat} \leq: \mathtt{nat}$: $\mathbf{expr}(\!|1, \mathtt{nat}|\!) = (\mathsf{succ}(1) > 0) \downarrow \mathsf{true}$;
- $\mathtt{int} \leq: \mathtt{int}$: $\mathbf{expr}(\!|-1, \mathtt{int}|\!) = (\mathsf{inv}(-1) > 0) \downarrow \mathsf{true}$;
- $\mathtt{nat} \leq: \mathtt{int}$: $\mathbf{expr}(\!|1, \mathtt{int}|\!) = (\mathsf{inv}(1) > 0) \downarrow \mathsf{false}$.

◀

▶ **Proposition D.18.** *Let* $\mathbb{U}$ *and* $\mathbb{V}'$ *be session types such that* $\mathcal{T}(\mathbb{U}) \not\leqslant \mathcal{T}(\mathbb{V}')$ *and* $\mathtt{pt}(\mathbb{V}') \subseteq \{\mathsf{p}_k : 1 \leq k \leq m\}$ *and* $\mathbb{U}_{\mathsf{p}_k} = \mathtt{cyclic}(\mathbb{V}', \mathsf{p}_k, \mathsf{r})$. *If*

$$\mathsf{M} \equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \prod_{1 \leq k \leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}_k}) \mid \mathsf{p}_k \triangleleft \varnothing)$$

$$\mathsf{M}' \equiv \mathsf{r} \triangleleft P \mid \mathsf{r} \triangleleft h_{\mathsf{p}} \mid \prod_{1 \leq k \leq m} (\mathsf{p}_k \triangleleft P_k \mid \mathsf{p}_k \triangleleft h_k),$$

where $\mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \longrightarrow^* \mathsf{r} \triangleleft P \mid \mathsf{r} \triangleleft h_{\mathsf{p}}$ *and*

$$\prod_{1 \leq k \leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}_k}) \mid \mathsf{p}_k \triangleleft \varnothing) \longrightarrow^* \prod_{1 \leq k \leq m} (\mathsf{p}_k \triangleleft P_k \mid \mathsf{p}_k \triangleleft h_k)$$

then, $\mathsf{M} \longrightarrow^* \mathsf{M}'$ *and* $\mathsf{M}' \longrightarrow^* \mathsf{error}$.

Proof. The proof is by induction on the derivation of $\mathcal{T}(\mathbb{U}) \not\leqslant \mathcal{T}(\mathbb{V}')$. We extensively use notation $\mathsf{U} = \mathcal{T}(\mathbb{U})$ and $\mathsf{V}' = \mathcal{T}(\mathbb{V}')$. The cases for the last rule applied are derived from Table 8.

We first consider the cases with $\mathsf{act}(\mathsf{U}) \neq \mathsf{act}(\mathsf{V}')$.

$\boxed{\mathsf{act}(\mathsf{U}) \neq \mathsf{act}(\mathsf{V}')}$

[N-UV-OUT-ACT] : $\mathsf{U} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{U}_1$ and $\mathsf{p}! \notin \mathsf{act}(\mathsf{V}')$.
In this case, if $\mathsf{p} \in \mathtt{pt}(\mathbb{V}')$ by definition of characteristic session type and characteristic process, $\mathsf{r}? \notin \mathcal{P}(\mathbb{U}_{\mathsf{p}})$. Thus,

$$\begin{aligned}
\mathsf{M} &\equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1' \\
&\equiv \mathsf{r} \triangleleft \mathcal{P}(\mathsf{p}!\ell(\mathsf{S}).\mathbb{U}_1) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1' \\
&\longrightarrow \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_1) \mid \mathsf{r} \triangleleft (\mathsf{p}, \ell(\mathbf{val}(\!|\mathsf{S}|\!))) \mid \mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1' \longrightarrow \mathsf{error} \qquad (\text{by [ERR-OPHN]})
\end{aligned}$$

If $\mathsf{p} \notin \mathtt{pt}(\mathbb{V}')$ we use $\mathsf{M} \equiv \mathsf{M} \mid \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$ and derive the analogous proof as above.
[N-UV-INP-ACT] : $\mathsf{U} = \&_{i \in I}\, \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i$ and $\mathsf{p}? \notin \mathsf{act}(\mathsf{V}')$.
In this case, if $\mathsf{p} \in \mathtt{pt}(\mathbb{V}')$ by definition of characteristic session type and characteristic process, $\mathsf{r}! \notin \mathcal{P}(\mathbb{U}_{\mathsf{p}})$. Thus,

$$\mathsf{M} \equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1'$$

$$\equiv \mathsf{r} \triangleleft \mathcal{P}\left(\mathop{\&}_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathbb{U}_i\right) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

$$\equiv \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathbb{U}_{\mathsf{p}}) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1 \longrightarrow \mathsf{error} \qquad \text{(by [ERR-STRV])}$$

If $\mathsf{p} \notin \mathtt{pt}(\mathbb{V}')$ we use $\mathsf{M} \equiv \mathsf{M} \mid \mathsf{p} \triangleleft \mathbf{0} \mid \mathsf{p} \triangleleft \varnothing$ and derive the analogous proof as above.
[N-UV-OUT-ACT-R] : $\mathsf{p}! \notin \mathsf{act}(\mathsf{U})$ and $\mathsf{V}' = \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}'_i$.
In this case, by definition of characteristic process, $\mathsf{p}! \notin \mathcal{P}(\mathbb{U})$.

$$\mathsf{M} \equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}'_i, \mathsf{p}, \mathsf{r})\right) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

$$\equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1 \longrightarrow \mathsf{error} \qquad \text{(by [ERR-STRV])}$$

[N-UV-INP-ACT-R] : $\mathsf{p}? \notin \mathsf{act}(\mathsf{U})$ and $\mathsf{V}' = \mathsf{p}?\ell(\mathsf{S}).\mathsf{V}'_1$.
In this case, by definition of characteristic process, $\mathsf{p}? \notin \mathcal{P}(\mathbb{U})$.

$$\mathsf{M} \equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}'_1, \mathsf{p}, \mathsf{r})) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

$$\equiv \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathsf{r}!\ell\langle \mathbf{val}(\!|\mathsf{S}|\!)\rangle.P \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

$$\longrightarrow \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft P \mid \mathsf{p} \triangleleft (\mathsf{r}, \ell(\mathbf{val}(\!|\mathsf{S}|\!))) \mid \mathsf{M}'_1 \longrightarrow \mathsf{error} \qquad \text{(by [ERR-OPHN])}$$

In the following cases, U type tree is rooted with an external choice.

$\boxed{\mathsf{U} = \mathop{\&}_{i\in I} \mathsf{p}?\ell_i(\mathsf{S}_i).\mathsf{U}_i}$

In these cases, we have

$$\mathcal{P}(\mathbb{U}) \equiv \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i, \text{ where}$$

$$P_i \equiv \mathsf{if}\ \mathbf{expr}(\!|x_i, \mathsf{S}_i|\!)\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i)$$

According to Table 8, we distinguish four cases (not already considered), depending on the form of V'.
[N-UV-INP]: $\mathsf{V}' = \mathsf{p}?\ell(\mathsf{S}).\mathsf{V}'_1$ and $\forall i \in I : \ell_i \neq \ell \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_i \not\lesssim \mathsf{V}'_1$.
Now we have

$$\mathsf{M} \equiv \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}'_1, \mathsf{p}, \mathsf{r})) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

$$= \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathsf{r}!\ell\langle \mathbf{val}(\!|\mathsf{S}|\!)\rangle.P' \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

$$\longrightarrow \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft P' \mid \mathsf{p} \triangleleft (\mathsf{r}, \ell(\mathbf{val}(\!|\mathsf{S}|\!))) \mid \mathsf{M}'_1$$

$$=: \mathsf{M}_1$$

We now distinguish three cases.

- $\forall i \in I : \ell_i \neq \ell$: Session M_1 reduces to error by [ERR-MISM].
- $\exists i \in I : \ell_i = \ell \ \wedge\ \mathsf{S} \not\leq: \mathsf{S}_i$:

$$\mathsf{M}_1 \longrightarrow \mathsf{r} \triangleleft \mathsf{if}\ \mathbf{expr}(\!|\mathbf{val}(\!|\mathsf{S}|\!), \mathsf{S}_i|\!)\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft P' \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}'_1$$

  By Lemma D.16 and [ERR-EVAL], the session reduces to error.

- $\exists i \in I : \ell_i = \ell \ \wedge\ \mathsf{S} \leq: \mathsf{S}_i \ \wedge\ \mathsf{U}_i \not\leqslant \mathsf{V}_1'$:

$$\mathsf{M}_1 \longrightarrow \mathsf{r} \triangleleft \mathsf{if}\ \underline{\mathbf{expr}(\!|\underline{\mathbf{val}(\!|\mathsf{S}|\!)}, \mathsf{S}_i|\!)}\ \mathsf{then}\ \mathcal{P}(\mathbb{U}_i)\ \ \mathsf{else}\ \mathcal{P}(\mathbb{U}_i)\ \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft P' \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1'$$

By Lemma D.17, we further derive

$$\mathsf{M}_1 \longrightarrow^* \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_i)\ \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft P' \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1'$$

where (assuming $\mathsf{p} = \mathsf{p}_1 = \mathsf{p}_{m+1}$)

$$P' \equiv \mathsf{p}_2!\ell(\mathsf{true}).\mathsf{p}_m?\ell(x).$$
$$\mathsf{if}\ \underline{\mathbf{expr}(\!|\underline{\mathbf{val}(\!|x|\!)}, \mathtt{bool}|\!)}\ \mathsf{then}\ \mathtt{cyclic}(\mathbb{V}_1', \mathsf{p}_1, \mathsf{r})\ \mathsf{else}\ \mathtt{cyclic}(\mathbb{V}_1', \mathsf{p}_1, \mathsf{r})$$

$$\mathsf{M}_1' \equiv \prod_{2\leq k\leq m} (\mathsf{p}_k \triangleleft \mathsf{p}_{k-1}?\ell(x).\mathsf{if}\ \underline{\mathbf{expr}(\!|\underline{\mathbf{val}(\!|x|\!)}, \mathtt{bool}|\!)}\ \mathsf{then}\ Q_k\ \mathsf{else}\ Q_k \mid \mathsf{p}_k \triangleleft \varnothing)$$

where $Q_k = \mathsf{p}_{k+1}!\ell(\mathsf{true}).\mathcal{P}(\mathtt{cyclic}(\mathbb{V}_1', \mathsf{p}_k, \mathsf{r}))$. Hence, we have

$$\mathsf{M}_1 \longrightarrow^* \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_i)\ \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathtt{cyclic}(\mathbb{V}_1', \mathsf{p}, \mathsf{r}) \mid \mathsf{p} \triangleleft \varnothing \mid$$
$$\prod_{2\leq k\leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathbb{V}_1', \mathsf{p}_k, \mathsf{r}))\ \mid \mathsf{p}_k \triangleleft \varnothing)$$

Since $\mathsf{U}_i \not\leqslant \mathsf{V}_1'$, by induction hypothesis the session reduces to error.

[N-UV-$\mathcal{A}$]: $\mathsf{V}' = \mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathsf{V}_1'$ and $\forall i \in I : \ell_i \neq \ell \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_i \not\leqslant \mathcal{A}^{(\mathsf{p})}.\mathsf{V}_1'$.
Assuming $\mathsf{p}_1 = \mathsf{p}_{m+1} = \mathsf{p}$, we have

$$\mathsf{M} \equiv \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}\Big(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}_1', \mathsf{p}, \mathsf{r})\Big)\ \mid \mathsf{p} \triangleleft \varnothing \mid$$
$$\prod_{2\leq k\leq m} (\mathsf{p}_k \triangleleft \mathcal{P}\Big(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{p})}.\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}_1', \mathsf{p}_k, \mathsf{r})\Big)\ \mid \mathsf{p}_k \triangleleft \varnothing)$$

By induction on context $\mathcal{A}^{(\mathsf{p})}$ we may show that

$$\mathsf{M} \longrightarrow^* \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}_1', \mathsf{p}, \mathsf{r}))\ \mid \mathsf{p} \triangleleft \varnothing \mid$$
$$\prod_{2\leq k\leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}_1', \mathsf{p}_k, \mathsf{r}))\ \mid \mathsf{p}_k \triangleleft h_k)$$

where for all $2 \leq k \leq m$: $h_k = (\mathsf{r}, \ell_1(\underline{\mathbf{val}(\!|\mathsf{S}_1|\!)})) \cdot \ldots \cdot (\mathsf{r}, \ell_n(\underline{\mathbf{val}(\!|\mathsf{S}_n|\!)}))$ when

$$\mathcal{A}^{(\mathsf{p})} = \mathcal{A}_1^{(\mathsf{p}_k)}.\mathsf{p}_k?\ell_1(S_1).\mathcal{A}_2^{(\mathsf{p}_k)}.\ldots.\mathcal{A}_n^{(\mathsf{p}_k)}.\mathsf{p}_k?\ell_n(S_n).\mathcal{A}_{n+1}^{(\mathsf{p}_k)}$$

where instead of $\mathcal{A}_i^{(\mathsf{p}_k)}$ contexts there could also be empty contexts, and if $\mathsf{p}_k? \notin \mathtt{act}(\mathcal{A}^{(\mathsf{p})})$ then $h_k = \varnothing$. Using the last observation, we have

$$\mathsf{M} \longrightarrow^* \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathsf{r}!\ell\langle\underline{\mathbf{val}(\!|\mathsf{S}|\!)}\rangle.P' \mid \mathsf{p} \triangleleft \varnothing \mid$$
$$\prod_{2\leq k\leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}_1', \mathsf{p}_k, \mathsf{r}))\ \mid \mathsf{p}_k \triangleleft h_k)$$
$$\longrightarrow \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft P' \mid \mathsf{p} \triangleleft (\mathsf{r}, \ell(\underline{\mathbf{val}(\!|\mathsf{S}|\!)})) \mid$$
$$\prod_{2\leq k\leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathsf{p}?\ell(\mathsf{S}).\mathbb{V}_1', \mathsf{p}_k, \mathsf{r}))\ \mid \mathsf{p}_k \triangleleft h_k)$$

We now distinguish three cases

- $\forall i \in I : \ell_i \neq \ell$;
- $\exists i \in I : \ell_i = \ell \ \wedge\ \mathsf{S} \not\leq: \mathsf{S}_i$;
- $\exists i \in I : \ell_i = \ell \ \wedge\ \mathsf{S} \leq: \mathsf{S}_i \ \wedge\ \mathsf{U}_i \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{V}'_1$.

In the first two cases M reduces to error by the same arguments that are presented for the case of [N-UV-INP]. For the third case, again using the same arguments as in the case of [N-UV-INP], we have that

$$\begin{aligned}\mathsf{M} \longrightarrow^* & \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_i) \ \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathtt{cyclic}(\mathbb{V}'_1, \mathsf{p}, \mathsf{r}) \mid \mathsf{p} \triangleleft \varnothing \mid \\ & \prod_{2 \leq k \leq m} (\mathsf{p}_k \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathbb{V}'_1, \mathsf{p}_k, \mathsf{r})) \ \mid \mathsf{p}_k \triangleleft h_k) \\ & =: \mathsf{M}'\end{aligned}$$

Since, $\mathsf{U}_i \not\lesssim \mathcal{A}^{(\mathsf{p})}.\mathsf{V}'_1$ and

$$\begin{aligned}\mathsf{M}'' = & \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_i) \ \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathtt{cyclic}(\mathcal{A}^{(\mathsf{p})}.\mathbb{V}'_1, \mathsf{p}, \mathsf{r}) \mid \mathsf{p} \triangleleft \varnothing \mid \\ & \prod_{2 \leq k \leq m} (\mathsf{p}_k \triangleleft \mathcal{P}\Big(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{p})}.\mathbb{V}'_1, \mathsf{p}_k, \mathsf{r})\Big) \ \mid \mathsf{p}_k \triangleleft \varnothing) \\ \longrightarrow^* & \mathsf{M}'\end{aligned}$$

which can also be shown in the same way as for M (by induction on $\mathcal{A}^{(\mathsf{p})}$), by induction hypothesis we get $\mathsf{M}'' \longrightarrow^* \mathsf{M}' \longrightarrow^* \mathsf{error}$, and hence, $\mathsf{M} \longrightarrow^* \mathsf{M}' \longrightarrow^* \mathsf{error}$.

[N-UV-IN-OUT-1]: $\mathsf{V}' = \bigoplus_{j \in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathsf{V}'_j$.
Assuming $\mathsf{p}_1 = \mathsf{p}_{m+1} = \mathsf{q}$ we have

$$\begin{aligned}\mathsf{M} \equiv & \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \ \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{q} \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathbb{V}', \mathsf{q}, \mathsf{r})) \ \mid \mathsf{q} \triangleleft \varnothing \mid \mathsf{M}'_1 \\ = & \mathsf{r} \triangleleft \sum_{i \in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{q} \triangleleft \sum_{j \in J} \mathsf{r}?\ell(x_j).P_j^{\mathsf{q}} \mid \mathsf{q} \triangleleft \varnothing \mid \mathsf{M}'_1\end{aligned}$$

where

$$\mathsf{M}'_1 \equiv \prod_{2 \leq k \leq m} (\mathsf{p}_k \triangleleft \sum_{j \in J} \mathsf{p}_{k-1}?\ell_j(x_j).P_j^{\mathsf{p}_k} \ \mid \mathsf{p}_k \triangleleft \varnothing)$$

The session reduces to error by [ERR-DLOCK].

[N-UV-IN-OUT-2]: $\mathsf{V}' = \mathcal{A}^{(\mathsf{p})}.\bigoplus_{j \in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathsf{V}'_j$.
Let us first assume $\mathsf{q} \neq \mathsf{p}$. Denoting $\mathsf{p}_1 = \mathsf{p}_{m+1} = \mathsf{q}$ and $\mathsf{p}_2 = \mathsf{p}$ we have

$$\begin{aligned}\mathsf{M} \equiv & \mathsf{r} \triangleleft \sum_{i \in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \\ & \prod_{1 \leq k \leq m} (\mathsf{p}_k \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{p})}.\bigoplus_{j \in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathbb{V}'_j, \mathsf{p}_k, \mathsf{r})\right) \ \mid \mathsf{p}_k \triangleleft \varnothing)\end{aligned}$$

Using a similar reasoning as in the case of [N-UV-$\mathcal{A}$] (by induction on $\mathcal{A}^{(\mathsf{p})}$) we obtain

$$\begin{aligned}\mathsf{M} \longrightarrow^* & \mathsf{r} \triangleleft \sum_{i \in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \\ & \prod_{1 \leq k \leq m} (\mathsf{p}_k \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\bigoplus_{j \in J} \mathsf{q}!\ell_j(\mathsf{S}_j).\mathbb{V}'_j, \mathsf{p}_k, \mathsf{r})\right) \ \mid \mathsf{p}_k \triangleleft h_k)\end{aligned}$$

$$=: \mathsf{M}'$$

where for all $1 \le k \le m$: $h_k = (\mathsf{r}, \ell_1(\mathbf{val}(\!|\mathsf{S}_1|\!))) \cdot \ldots \cdot (\mathsf{r}, \ell_n(\mathbf{val}(\!|\mathsf{S}_n|\!)))$ when

$$\mathcal{A}^{(\mathsf{p})} = \mathcal{A}_1^{(\mathsf{p}_k)}.\mathsf{p}_k?\ell_1(S_1).\mathcal{A}_2^{(\mathsf{p}_k)}.\ldots.\mathcal{A}_n^{(\mathsf{p}_k)}.\mathsf{p}_k?\ell_n(S_n).\mathcal{A}_{n+1}^{(\mathsf{p}_k)}$$

where instead of $\mathcal{A}_i^{(\mathsf{p}_k)}$ contexts there could also be empty contexts, and if $\mathsf{p}_k? \notin \mathtt{act}(\mathcal{A}^{(\mathsf{p})})$ then $h_k = \varnothing$. Since $\mathsf{p}? \notin \mathtt{act}(\mathcal{A}^{(\mathsf{p})})$, for the last derived session we have

$$\begin{aligned}\mathsf{M}' =& \mathsf{r} \triangleleft \sum_{i\in I} \mathsf{p}?\ell_i(x_i).P_i \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{q} \triangleleft \sum_{j\in J} \mathsf{r}?\ell(x_j).P_j^{\mathsf{q}} \mid \mathsf{q} \triangleleft h_1 \mid \\ & \mathsf{p} \triangleleft \sum_{j\in J} \mathsf{q}?\ell_j(x_j).P_j^{\mathsf{p}_k} \mid \mathsf{p} \triangleleft \varnothing \mid \prod_{3\le k\le m} (\mathsf{p}_k \triangleleft \sum_{j\in J} \mathsf{p}_{k-1}?\ell_j(x_j).P_j^{\mathsf{p}_k} \mid \mathsf{p}_k \triangleleft h_k)\end{aligned}$$

the session reduces to error by [ERR-DLOCK]. In case $\mathsf{q} = \mathsf{p}$, the proof follows similar lines.

In the following cases, U type tree is rooted with an internal choice.

$\boxed{\mathsf{U} = \mathsf{p}!\ell(\mathsf{S}).\mathsf{U}_1}$

In these case, we have $\mathcal{P}(\mathbb{U}) \equiv \mathsf{p}!\ell\langle\mathbf{val}(\!|S|\!)\rangle.\mathcal{P}(\mathbb{U}_1)$.
Depending on the form of V', we distinguish two more cases (according to rules in Table 8).

[N-UV-OUT]: $\mathsf{V}' = \bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_i'$ and $\forall i \in I : \ell \neq \ell_i \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_1 \not\lesssim \mathsf{V}_i'$.
Now we have

$$\begin{aligned}\mathsf{M} \equiv& \mathsf{r} \triangleleft \mathsf{p}!\ell\langle\mathbf{val}(\!|S|\!)\rangle.\mathcal{P}(\mathbb{U}_1) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\bigoplus_{i\in I} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}_i', \mathsf{p}, \mathsf{r})\right) \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1' \\ =& \mathsf{r} \triangleleft \mathsf{p}!\ell\langle\mathbf{val}(\!|S|\!)\rangle.\mathcal{P}(\mathbb{U}_1) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \sum_{i\in I} \mathsf{r}?\ell_i(x_i).P_i \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1' \\ \longrightarrow& \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_1) \mid \mathsf{r} \triangleleft (\mathsf{p}, \ell(\mathbf{val}(\!|\mathsf{S}|\!))) \mid \mathsf{p} \triangleleft \sum_{i\in I} \mathsf{r}?\ell_i(x_i).P_i \mid \mathsf{p} \triangleleft \varnothing \mid \mathsf{M}_1'\end{aligned}$$

Now the proof proceeds following the same lines as in the case of [N-UV-INP].

[N-UV-$\mathcal{C}$]: $\mathsf{V}' = \mathcal{C}^{(\mathsf{p})}[\bigoplus_{i\in I_n} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathsf{V}_i']^{n\in N}$ and $\forall n \in N\, \forall i \in I_n : \ell \neq \ell_i \ \vee\ \mathsf{S} \not\leq: \mathsf{S}_i \ \vee\ \mathsf{U}_1 \not\lesssim (\mathcal{C}^{(\mathsf{p})} \upharpoonright n)[\mathsf{V}_i']$.
Let us denote $\mathsf{p}_1 = \mathsf{p}_{m+1} = \mathsf{p}$. We have

$$\begin{aligned}\mathsf{M} \equiv& \mathsf{r} \triangleleft \mathsf{p}!\ell\langle\mathbf{val}(\!|S|\!)\rangle.\mathcal{P}(\mathbb{U}_1) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{C}^{(\mathsf{p})}[\bigoplus_{i\in I_n} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}_i']^{n\in N}, \mathsf{p}, \mathsf{r})\right) \mid \mathsf{p} \triangleleft \varnothing \\ & \mid \prod_{2\le k\le m} (\mathsf{p}_k \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{C}^{(\mathsf{p})}[\bigoplus_{i\in I_n} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}_i']^{n\in N}, \mathsf{p}_k, \mathsf{r})\right) \mid \mathsf{p}_k \triangleleft \varnothing)\end{aligned}$$

By induction on $\mathcal{C}^{(\mathsf{p})}[\ ]^{n\in N}$ we may show that either $\mathsf{M} \longrightarrow^* \mathsf{error}$ or there is $n \in N$ such that

$$\mathsf{M} \longrightarrow^* \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}_1') \mid \mathsf{r} \triangleleft (\mathsf{p}, \ell(\mathbf{val}(\!|\mathsf{S}|\!))) \cdot h_r \mid \mathsf{p} \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\bigoplus_{i\in I_n} \mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}_i', \mathsf{p}, \mathsf{r})\right) \mid \mathsf{p} \triangleleft h_{\mathsf{p}}$$

$$\mid \prod_{2\leq k\leq m}(\mathsf{p}_k \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\bigoplus_{i\in I_n}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}'_i, \mathsf{p}_k, \mathsf{r})\right) \mid \mathsf{p}_k \triangleleft h_k)$$

$$=: \mathsf{M}_1$$

where there exist output-only context $\mathcal{B}^{(\mathsf{r})}$ and $\mathcal{A}^{(\mathsf{r})}$ context (where instead of r we could use any other fresh name) such that

$$\mathsf{M}_2 = \mathsf{r} \triangleleft \mathcal{P}\left(\mathsf{p}!\ell(\mathsf{S}).\mathcal{B}^{(\mathsf{r})}.\mathbb{U}'_1\right) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{r})}.\bigoplus_{i\in I_n}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}'_i, \mathsf{p}, \mathsf{r})\right) \mid \mathsf{p} \triangleleft \varnothing$$

$$\mid \prod_{2\leq k\leq m}(\mathsf{p}_k \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{r})}.\bigoplus_{i\in I_n}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}'_i, \mathsf{p}_k, \mathsf{r})\right) \mid \mathsf{p}_k \triangleleft \varnothing)$$

$$\longrightarrow^* \mathsf{M}_1$$

and where $\mathcal{B}^{(\mathsf{r})}.\mathsf{U}'_1 \nleqslant \mathcal{A}^{(\mathsf{r})}.\mathsf{V}'_i$ can be derived from $\mathsf{U}_1 \nleqslant (\mathcal{C}^{(\mathsf{p})} \restriction n)[\mathsf{V}'_i]$ by applying the rules given in Table 8 from conclusion to premises.

Note that the above $\mathcal{A}^{(\mathsf{r})}$ is actually derived from $(\mathcal{C}^{(\mathsf{p})} \restriction n)$, by taking out all the outputs (that are transformed into the inputs in the characteristic session), since they have found the appropriate outputs in U_1, and also some inputs (that are transformed into outputs in the characteristic session), that have found the appropriate inputs in U_1: these pairs of actions enabled session M to reduce to M_1. Along these lines $\mathcal{B}^{(\mathsf{r})}.\mathsf{U}'_1$ is derived from U_1.

The above also implies that by the assumption $\mathcal{B}^{(\mathsf{r})}.\mathsf{U}'_1 \leqslant \mathcal{A}^{(\mathsf{r})}.\mathsf{V}'_i$ we could derive $\mathsf{U}_1 \leqslant (\mathcal{C}^{(\mathsf{p})} \restriction n)[\mathsf{V}'_i]$.

Since

$$\mathcal{P}\left(\mathtt{cyclic}(\bigoplus_{i\in I_n}\mathsf{p}!\ell_i(\mathsf{S}_i).\mathbb{V}'_i, \mathsf{p}, \mathsf{r})\right) = \sum_{i\in I}\mathsf{r}?\ell_i(x_i).P_i$$

we distinguish three cases

- $\forall i \in I : \ell \neq \ell_i$;
- $\exists i \in I : \ell = \ell_i \ \wedge\ \mathsf{S} \not\leq: \mathsf{S}_i$;
- $\exists i \in I : \ell = \ell_i \ \wedge\ \mathsf{S} \leq: \mathsf{S}_i \ \wedge\ \mathsf{U}_1 \nleqslant (\mathcal{C}^{(\mathsf{p})} \restriction n)[\mathsf{V}'_i]$.

In the first two cases M_1 reduces to error using similar arguments as for [N-UV-INP]. For the third case we have that

$$\mathsf{M}_1 \longrightarrow^* \mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}'_1) \mid \mathsf{r} \triangleleft h_r \mid \mathsf{p} \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathbb{V}'_i, \mathsf{p}, \mathsf{r})) \mid \mathsf{p} \triangleleft h_{\mathsf{p}}$$

$$\mid \prod_{2\leq k\leq m}(\mathsf{p}_k \triangleleft \mathcal{P}(\mathtt{cyclic}(\mathbb{V}'_i, \mathsf{p}_k, \mathsf{r})) \mid \mathsf{p}_k \triangleleft h_k)$$

$$=: \mathsf{M}'$$

Similarly as in the case of [N-UV-$\mathcal{A}$], we have

$$\mathsf{M}'' = \mathsf{r} \triangleleft \mathcal{P}\left(\mathcal{B}^{(\mathsf{r})}.\mathbb{U}'_1\right) \mid \mathsf{r} \triangleleft \varnothing \mid \mathsf{p} \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{r})}.\mathbb{V}'_i, \mathsf{p}, \mathsf{r})\right) \mid \mathsf{p} \triangleleft \varnothing$$

$$\mid \prod_{2\leq k\leq m}(\mathsf{p}_k \triangleleft \mathcal{P}\left(\mathtt{cyclic}(\mathcal{A}^{(\mathsf{r})}.\mathbb{V}'_i, \mathsf{p}_k, \mathsf{r})\right) \mid \mathsf{p}_k \triangleleft \varnothing)$$

$$\longrightarrow^* \mathsf{M}'$$

Since $\mathcal{B}^{(\mathsf{r})}.\mathsf{U}'_1 \nleqslant \mathcal{A}^{(\mathsf{r})}.\mathsf{V}'_i$ can be derived from $\mathsf{U}_1 \nleqslant (\mathcal{C}^{(\mathsf{p})} \restriction n)[\mathsf{V}'_i]$ by applying the rules given in Table 8 from conclusion to premises, we may apply induction hypothesis and obtain $\mathsf{M}'' \longrightarrow^* \mathsf{M}' \longrightarrow^* \mathsf{error}$. Hence, $\mathsf{M} \longrightarrow^* \mathsf{M}' \longrightarrow^* \mathsf{error}$. ◀

▶ **Proposition D.19.** *Let* $\mathbb{T}$ *and* $\mathbb{T}'$ *be session types such that* $\mathbb{T} \not\leqslant \mathbb{T}'$*. Then, there are* $\mathbb{U}$ *and* $\mathbb{V}'$ *with* $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ *and* $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ *and* $\mathbb{U} \not\leqslant \mathbb{V}'$ *such that:*

$$\mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \ \mid \mathsf{r} \triangleleft \varnothing \mid \prod_{1 \leq k \leq m} \mathsf{p}_k \triangleleft (\mathcal{P}(\mathbb{U}_{\mathsf{p}_k}) \ \mid \mathsf{p}_k \triangleleft \varnothing) \longrightarrow^* \mathsf{error},$$

where $\mathtt{pt}(\mathbb{V}') \subseteq \{\mathsf{p}_k : 1 \leq k \leq m\}$ *and* $\mathbb{U}_{\mathsf{p}_k} = \mathtt{cyclic}(\mathbb{V}', \mathsf{p}_k, \mathsf{r})$.

Proof. If $\mathcal{T}(\mathbb{T}) \not\leqslant \mathcal{T}(\mathbb{T}')$, there are U and V such that $\mathsf{U} \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathsf{V}' \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ and $\mathsf{U} \not\leqslant \mathsf{V}'$. By Corollary D.6, there are $\mathbb{U}$ and $\mathbb{V}'$ such that $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SI}}$ and $\mathcal{T}(\mathbb{U}) \not\leqslant \mathcal{T}(\mathbb{V}')$. Now the proof follows by Proposition D.18. ◀

▶ **Theorem 5.13.** *The asynchronous multiparty session subtyping* $\leqslant$ *is complete.*

Proof. Let $\mathbb{T}$ and $\mathbb{T}'$ be such that $\mathcal{T}(\mathbb{T}) \not\leqslant \mathcal{T}(\mathbb{T}')$. Then, by Proposition D.19 there are $\mathbb{U}$ and $\mathbb{V}'$ with $\mathcal{T}(\mathbb{U}) \in [\![\mathcal{T}(\mathbb{T})]\!]_{\mathsf{SI}}$ and $\mathcal{T}(\mathbb{V}') \in [\![\mathcal{T}(\mathbb{T}')]\!]_{\mathsf{SO}}$ and $\mathcal{T}(\mathbb{U}) \not\leqslant \mathcal{T}(\mathbb{V}')$, such that,

$$\mathsf{r} \triangleleft \mathcal{P}(\mathbb{U}) \ \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'} \longrightarrow^* \mathsf{error}, \tag{223}$$

where

$$\mathcal{M}_{\mathsf{r},\mathbb{V}'} = \prod_{1 \leq k \leq m} \mathsf{p}_k \triangleleft (\mathcal{P}(\mathbb{U}_{\mathsf{p}_k}) \ \mid \mathsf{p}_k \triangleleft \varnothing)$$

and $\mathtt{pt}(\mathbb{V}') \subseteq \{\mathsf{p}_k : 1 \leq k \leq m\}$ and $\mathbb{U}_{\mathsf{p}_k} = \mathtt{cyclic}(\mathbb{V}', \mathsf{p}_k, \mathsf{r})$.

By Proposition 5.11, $\vdash Q_1 : \mathbb{T}'$ implies there is a live typing environment Γ such that

$$\Gamma \vdash \mathsf{r} \triangleleft Q_1 \mid \mathsf{r} \triangleleft \varnothing \mid \mathcal{M}_{\mathsf{r},\mathbb{V}'}$$

Since by Proposition 5.6 $\vdash \mathcal{P}(\mathbb{U}) : \mathbb{T}$, we conclude the proof by (223). ◀