Fair Subtyping for Open Session Types

Fair Subtyping for Open Session Types

Luca Padovani

Università di Torino, Dipartimento di Informatica, Italy
luca.padovani@unito.it

Abstract. Standard subtyping for session types may compromise session liveness, that is the ability for some participant of a session to make progress. We define a fair subtyping relation for possibly open session types that preserves session liveness and is a pre-congruence with respect to all the operators of the type language. Even if fair subtyping and the well-known should-testing pre-congruence share the same definition, it turns out that fair subtyping is coarser than should-testing, admits an intuitive characterization as a refinement of standard subtyping, is fully axiomatizable, and can be decided efficiently.

1 Introduction

Session types [6,7] describe the type, order, and direction of messages that can be sent over channels. In essence, session types are simple CCS-like processes using a reduced set of operators [2,1]: termination, external and internal choices respectively guarded by input and output actions, and recursion. For example, the session type $T = \mu X.(!\mathtt{a}.X \oplus !\mathtt{b})$ denotes a channel for sending an arbitrary number of `a` messages followed by a single `b` message. In an almost dual manner, the session type $S = \mu X.(?\mathtt{a}.X + ?\mathtt{b}.!\mathtt{c})$ denotes a channel for receiving an arbitrary number of `a` messages, or a single `b` message after which it is possible to send a `c` message. A term $T \mid S \mid ?\mathtt{c}$ describes a session as the parallel composition of the behavior of its participants. In this case the session has three participants that interact through three session channels of type T, S, and $?\mathtt{c}$. Session type systems check that processes use session channels according to a session type. For instance, the typing derivation below proves that the process $\mathtt{rec}\ X.k!\langle m\rangle.X$ sending the message m on channel k is well typed in the channel environment $k : T$ provided that "m is a message of type `a`" (the exact interpretation of this property is irrelevant):

$$\dfrac{\dfrac{\dfrac{\vdash m : \mathtt{a} \qquad \dfrac{}{\mathscr{X} \mapsto \{k : x\}; k : x \vdash \mathscr{X}}\,[\textsc{Var}]}{\mathscr{X} \mapsto \{k : x\}; k : !\mathtt{a}.x \vdash k!m.\mathscr{X}}\,[\textsc{Output}] \qquad !\mathtt{a}.x \oplus !\mathtt{b} \leqslant !\mathtt{a}.x}{\mathscr{X} \mapsto \{k : x\}; k : !\mathtt{a}.x \oplus !\mathtt{b} \vdash k!m.\mathscr{X}}\,[\textsc{Sub}]}{k : \mu X.(!\mathtt{a}.X \oplus \mathtt{b}) \vdash \mathtt{rec}\ \mathscr{X}.k!m.\mathscr{X}}\,[\textsc{Rec}]$$

Rule [Rec] opens the recursive session type T in correspondence with recursion in the process and augments the process environment with the association $\mathscr{X} \mapsto \{k : x\}$. In this way, an occurrence of the process variable $\mathscr{X}$ in a channel environment where the channel k has type x can be declared well typed. Rule [Sub] applies subsumption on k so that its type $!\mathtt{a}.x \oplus !\mathtt{b}$ matches the actual behavior of the process, which is described

by $!\mathtt{a}.x$ (thinking of channels as objects and of messages as methods, the process is invoking the `a` method on an object that has both `a` and `b` methods available). Rule [OUTPUT] checks that the output performed by the process on channel k is consistent with the type of k. Finally, the continuation of the process after $k!m$ is checked against the residual type of k after $!\mathtt{a}$.

The original subtyping for session types [5] establishes that $\leqslant$ is contravariant for outputs: indeed the derivation above relies on the law $!\mathtt{a}.x \oplus !\mathtt{b} \leqslant !\mathtt{a}.x$, where the larger session type permits fewer outputs. This law is consistent with other subtyping relations for channel types [11,4], but there are contexts in which its unconditioned application may compromise session liveness. For instance, consider the session described earlier as $T \mid S \mid ?\mathtt{c}$ and observe that all non-terminated participants retain the potential to make progress: it is always *possible* for the first participant to send a `b` message and terminate. If that happens, the second participant sends the `c` message to the third participant, at which point all participants terminate. Accepting as correct the typing derivation above means accepting as correct also the session $\mu x.!\mathtt{a}.x \mid S \mid ?\mathtt{c}$, where the first participant sends only `a` messages. In this session, however, the third participant is no longer able to make any progress, so the liveness of the session with respect to the third participant is compromised. This example proves that the original subtyping relation for session types, for which $!\mathtt{a}.x \oplus !\mathtt{b} \leqslant !\mathtt{a}.x$ holds, is not liveness preserving in general: there exist a context $\mathscr{C} = \mu x.[\,]$ and two behaviors $S \mid ?\mathtt{c}$ such that the session described by $\mathscr{C}[!\mathtt{a}.x \oplus !\mathtt{b}] \mid S \mid ?\mathtt{c}$ does have the liveness property while $\mathscr{C}[!\mathtt{a}.x] \mid S \mid ?\mathtt{c}$ does not.

The contribution of this work is the definition and characterization of a new subtyping relation, which we dub *fair subtyping*, as the coarsest liveness-preserving refinement for possibly *open* session types (like $!\mathtt{a}.x \oplus !\mathtt{b}$ and $!\mathtt{a}.x$ above) that is a *pre-congruence* for all the operators of the type language. With this definition in place, we are able to reject a derivation like the one above because it is based on the law $!\mathtt{a}.x \oplus !\mathtt{b} \leqslant !\mathtt{a}.x$ which is *invalid* for fair subtyping. A behavioral refinement called *should-testing* with all the above properties has been extensively studied in [12]. There, should-testing is shown to be the coarsest liveness-preserving pre-congruence of a process algebra considerably richer than session types. Therefore, given the correspondence between session types and processes, we could just take should-testing as the defining notion for fair subtyping, but we find this shortcut unsatisfactory for a number of reasons: first, should-testing implies trace equivalence between related processes. In our context, this would amount to requiring *invariance* of outputs, essentially collapsing subtyping to type equality. Second, no complete axiomatization is known for should-testing and its alternative characterization is based on a complex denotational model. As a consequence, it is difficult to understand the basic laws that underlie should-testing. Third, the decision algorithm for should-testing is linear exponential, that is remarkably more expensive compared to the quadratic algorithm for the original subtyping [5]. Instead, by restricting the language of processes to that of session types, we are able to show that:

- Fair subtyping is coarser than should-testing and *does not* imply trace equivalence.
- Fair subtyping admits a complete axiomatization obtained from that of the original subtyping by plugging in a simple auxiliary relation in just two strategic places.
- Fair subtyping can be decided in $O(n^4)$ time.

In the rest of the paper we formalize session types as an appropriate subset of CCS (Section 2) and define fair subtyping as the relation that preserves session liveness in every context (Definition 2.2). Then, we provide a coinductive characterization of fair subtyping that unveils its properties (Section 3). The pre-congruence property is subtle to characterize because fair subtyping is *context sensitive* (two session types may or may not be related depending on the context in which they occur). For example, we have seen that $!\mathtt{a}.x \oplus !\mathtt{b} \not\leqslant !\mathtt{a}.x$ and yet $!\mathtt{a}.(!\mathtt{a}.x \oplus !\mathtt{b}) \oplus !\mathtt{b} \leqslant !\mathtt{a}.!\mathtt{a}.x \oplus !\mathtt{b}$ despite the unrelated terms $!\mathtt{a}.x \oplus !\mathtt{b}$ and $!\mathtt{a}.x$ occur in corresponding positions in the latter pair of related session types. The coinductive characterization also paves the way to the complete axiomatization of fair subtyping and to its decision algorithm (Section 4). In turn, the axiomatization shows how to incrementally patch the original subtyping for session types to ensure liveness preservation. We conclude with a more detailed comparison with related work (Section 5) and a few remarks on the relevance of this research (Section 6). *Proofs of the results can be found in the appendix, which it is not formally part of the submission.*

Table 1. Syntax of session types and sessions.

$T ::=$	**Session Type**	$M ::=$	**Session**
end	(termination)	T	(endpoint)
$\mid x$	(variable)	$\mid (M \mid M)$	(composition)
$\mid \sum_{i\in I} ?\mathtt{a}_i.T_i$	(input)		
$\mid \bigoplus_{i\in I} !\mathtt{a}_i.T_i$	(output)		
$\mid \mu x.T$	(recursion)		

2 Syntax and Semantics of Session Types

We assume given an infinite set $\mathbf{V}$ of variables x, y, ... and an infinite set of messages $\mathtt{a}$, $\mathtt{b}$, ... We let X, Y, ... range over subsets of $\mathbf{V}$. The syntax of sessions and session types is given by the grammar in Table 1. Sessions M, N, ... are abstracted as parallel compositions of session types T, S, ... which are informally described below. The term end denotes the type of channels on which no further operations are possible. We will often omit trailing occurrences of end. A term $\sum_{i\in I} ?\mathtt{a}_i.T_i$ is the type of a channel for receiving a message in the set $\{\mathtt{a}_i\}_{i\in I}$. According to the received message $\mathtt{a}_i$, the channel must be used according to T_i afterwards. Terms $\bigoplus_{i\in I} !\mathtt{a}_i.T_i$ are analogous, but they denote the type of channels that can be used for sending messages. Note that output session types represent *internal choices* (the process using a channel with output type can choose any message in the set $\{\mathtt{a}_i\}_{i\in I}$) while input session types are *external choices* (the process using a channel with input type must be ready to deal with any message in the set $\{\mathtt{a}_i\}_{i\in I}$). We assume that the set I in input and output session types is always finite and non-empty and that choices are deterministic, in the sense that $\mathtt{a}_i = \mathtt{a}_j$ implies $i = j$ for every $i, j \in I$. We will sometimes use an infix notation for choices writing $?\mathtt{a}_1.T_1 + \cdots + ?\mathtt{a}_n.T_n$ and $!\mathtt{a}_1.T_1 \oplus \cdots \oplus !\mathtt{a}_n.T_n$ instead of $\sum_{1\leq i\leq n} ?\mathtt{a}_i.T_i$

and $\bigoplus_{1\le i\le n} !\mathtt{a}_i.T_i$ respectively. Terms $\mu x.T$ and x are used for building recursive session types, as usual. We assume that session types are contractive, namely that they do not contain any subterm of the form $\mu x_1 \cdots \mu x_n.x_1$. The notions of free and bound variables are standard and so are the definitions of open and closed session types. We take an equirecursive point of view and identify session types modulo renaming of bound variables and folding/unfolding of recursions. That is, $\mu x.T = T\{\mu x.T/x\}$ where $T\{S/x\}$ is the capture-avoiding substitution of every free occurrence of x in T with S.

Table 2. Transition system of sessions.

$$[\text{T-OUTPUT}]\quad !\mathtt{a}.T \xrightarrow{!\mathtt{a}} T$$

$$[\text{T-CHOICE}]\quad \frac{k\in I}{\bigoplus_{i\in I} !\mathtt{a}_i.T_i \xrightarrow{\tau} !\mathtt{a}_k.T_k}$$

$$[\text{T-INPUT}]\quad \frac{k\in I}{\sum_{i\in I} ?\mathtt{a}_i.T_i \xrightarrow{?\mathtt{a}_k} T_k}$$

$$[\text{T-PAR LEFT}]\quad \frac{M \xrightarrow{\ell} M'}{M\,|\,N \xrightarrow{\ell} M'\,|\,N}$$

$$[\text{T-PAR RIGHT}]\quad \frac{N \xrightarrow{\ell} N'}{M\,|\,N \xrightarrow{\ell} M\,|\,N'}$$

$$[\text{T-PAR COMM}]\quad \frac{M \xrightarrow{\overline{\alpha}} M' \qquad N \xrightarrow{\alpha} N'}{M\,|\,N \xrightarrow{\tau} M'\,|\,N'}$$

We define the operational semantics of sessions by means of a labeled transition system mimicking the actions performed by processes that behave according to session types (in fact, we are abstracting processes into types). The transition system makes use of actions α of the form $?\mathtt{a}$ and $!\mathtt{a}$ describing the input/output of $\mathtt{a}$ messages and labels ℓ that are either actions or the invisible move τ. Axioms and rules of the transition system are defined in Table 2 and briefly described hereafter. Rules [T-OUTPUT], [T-CHOICE], and [T-INPUT] deal with prefixed terms. The first and last ones are standard. Rule [T-CHOICE] states that a process behaving according to the type $\bigoplus_{i\in I} !\mathtt{a}_i.T_i$ may internally choose, through an invisible move τ, to send any message from the set $\{\mathtt{a}_i\}_{i\in I}$. Rules [T-PAR LEFT], [T-PAR RIGHT] propagate labels across compositions while [T-PAR COMM] is the synchronization rule between complementary actions resulting into an invisible move (we let $\overline{?\mathtt{a}} = !\mathtt{a}$ and $\overline{!\mathtt{a}} = ?\mathtt{a}$).

We use φ, ψ, ... to range over strings of actions, ε to denote the empty string, and $\le$ to denote the usual prefix order between strings. We write $\overset{\tau}{\Longrightarrow}$ for the reflexive, transitive closure of $\xrightarrow{\tau}$ and $\overset{\alpha}{\Longrightarrow}$ for the composition $\overset{\tau}{\Longrightarrow}\xrightarrow{\alpha}\overset{\tau}{\Longrightarrow}$. We extend this notation to strings of actions so that $\overset{\alpha_1\cdots\alpha_n}{\Longrightarrow}$ stands for the composition $\overset{\alpha_1}{\Longrightarrow}\cdots\overset{\alpha_n}{\Longrightarrow}$. We write $T \overset{\alpha}{\Longrightarrow}$ (respectively $T \overset{\varphi}{\Longrightarrow}$) if there exists S such that $T \overset{\alpha}{\Longrightarrow} S$ (respectively $T \overset{\varphi}{\Longrightarrow} S$). We write $T \overset{\varphi}{\not\Longrightarrow}$ if not $T \overset{\varphi}{\Longrightarrow}$. We let $\mathtt{tr}(T)$ denote the set of *traces* of T, namely $\mathtt{tr}(T) \overset{\text{def}}{=} \{\varphi \mid T \overset{\varphi}{\Longrightarrow}\}$.

We say that a session M is successful if every computation starting from M can be extended to a state that emits the action $!\mathtt{OK}$, where $\mathtt{OK}$ is a special message that we assume is not used for synchronization purposes. Formally:

Definition 2.1 (success). *We say that M is* successful *if $M \overset{\tau}{\Longrightarrow} N$ implies $N \overset{!\mathtt{OK}}{\Longrightarrow}$.*

Example 2.1. Consider $T = \mu x.(!\mathtt{a}.x \oplus !\mathtt{b})$ and $S = \mu x.(?\mathtt{a}.x + ?\mathtt{b}.!\mathtt{c})$ from the introduction. Then $T \mid S \mid ?\mathtt{c}.!\mathtt{OK}$ is a successful session because, no matter how many $\mathtt{a}$ messages the first participant sends, it is always possible that a $\mathtt{b}$ message and a $\mathtt{c}$ message are sent. At that point, the third participant emits $!\mathtt{OK}$. The session $!\mathtt{b} \mid S \mid ?\mathtt{c}.!\mathtt{OK}$ is successful as well. By contrast $\mu x.!\mathtt{a}.x \mid S \mid ?\mathtt{c}.!\mathtt{OK}$ is unsuccessful even if that the first two participants keep interacting with each other, because none of them is sending the $\mathtt{c}$ message that the third participant is waiting for. ■

We can now define fair subtyping as the relation that preserves session success. To make sure that fair subtyping is a pre-congruence, we quantify over all possible contexts that apply to the two session types being compared. *Contexts*, ranged over by $\mathscr{C}$, are just session types with a hole $[\,]$ in place of some subterm and are generated by the grammar

$$\mathscr{C} \;::=\; [\,] \;\mid\; ?\mathtt{a}.\mathscr{C} + \textstyle\sum_{i\in I} ?\mathtt{a}_i.T_i \;\mid\; !\mathtt{a}.\mathscr{C} \oplus \bigoplus_{i\in I} !\mathtt{a}_i.T_i \;\mid\; \mu x.\mathscr{C}$$

where the set I indexing choices is finite and possibly empty. We write $\mathscr{C}[T]$ for the session type obtained by filling the hole in $\mathscr{C}$ with T. This operation differs from variable substitution in that $\mathscr{C}$ may capture variables occurring free in T.

Fair subtyping is defined thus:

Definition 2.2 (fair subtyping). *We say that T is a* fair subtype *of S, written $T \leqslant S$, if $M \mid \mathscr{C}[T]$ successful implies $M \mid \mathscr{C}[S]$ successful for every M and $\mathscr{C}$.*

Note the left-to-right substitution of behaviors that characterizes Definition 2.2 as opposed to the right-to-left substitution of values that is normally associated with subtyping relations. In fact, these are just different viewpoints for the same intuition: when we replace a channel of type S with another one of type T within a (well-typed) process, the process keeps behaving according to S even if it is acting on a channel of type T. If the channel of type T was meant to work successfully in a session $M \mid T$ and $T \leqslant S$, then success of the session is not compromised as $M \mid S$ is still successful.

Showing that $T \leqslant S$ is difficult in general, because of the universal quantification over sessions M and contexts $\mathscr{C}$ in Definition 2.2. Until we characterize precisely the properties of $\leqslant$ in Section 3, we can only argue informally as to why a relation holds.

Example 2.2. Consider $T = \mu x.(!\mathtt{a}.x \oplus !\mathtt{b})$ and $S_1 = \mu x.(!\mathtt{a}.!\mathtt{a}.x \oplus !\mathtt{b})$ and take $M = \mu x.(?\mathtt{a}.x + ?\mathtt{b}.!\mathtt{OK})$. Note that M relies on the eventual reception of a $\mathtt{b}$ message to emit $!\mathtt{OK}$ so M is somehow the "most demanding" session such that $M \mid T$ is successful. Now, since $M \mid S_1$ is successful as well, we can argue that $T \leqslant S_1$ does hold. It is easier to find session types *not* related by fair subtyping. For instance, given $S_2 = \mu x.!\mathtt{a}.x$ we have $T \not\leqslant S_2$ because $M \mid T$ is successful but $M \mid S_2$ is not (no $\mathtt{b}$ message is ever sent). Now, for $\mathscr{C} = \mu x.[\,]$ we have $T = \mathscr{C}[!\mathtt{a}.x \oplus \mathtt{b}]$ and $S_2 = \mathscr{C}[!\mathtt{a}.x]$, therefore $!\mathtt{a}.x \oplus \mathtt{b} \not\leqslant !\mathtt{a}.x$. ■

3 Fair Subtyping

In this section we develop an alternative characterization of fair subtyping. To ease the presentation of the development, we split it up in three steps corresponding to three increasingly accurate approximations of fair subtyping. The last approximation will be shown to coincide with fair subtyping.

3.1 Unfair Subtyping

We begin by defining a subtyping relation for session types, which we dub "unfair subtyping", as a first approximation of $\leqslant$.

Definition 3.1 (unfair subtyping). *We say that $\mathscr{U}$ is a* coinductive subtyping *if $T\ \mathscr{U}\ S$ implies either (1) $T = S = x$ or (2) $T = S = \mathsf{end}$ or (3) $T = \sum_{i\in I} ?\mathsf{a}_i.T_i$ and $S = \sum_{i\in I} ?\mathsf{a}_i.S_i$ and $T_i\ \mathscr{U}\ S_i$ for every $i \in I$ or (4) $T = \bigoplus_{i\in I\cup J} !\mathsf{a}_i.T_i$ and $S = \bigoplus_{i\in I} !\mathsf{a}_i.S_i$ and $T_i\ \mathscr{U}\ S_i$ for every $i \in I$.* Unfair subtyping*, denoted by $\leqslant_{\mathsf{U}}$, is the largest coinductive subtyping.*

Clauses **(1–2)** state the reflexivity of $\leqslant_{\mathsf{U}}$ for end and type variables, while clauses **(3–4)** respectively state invariance and contravariance of $\leqslant_{\mathsf{U}}$ with respect to external and internal choices. Unfair subtyping is essentially the standard subtyping relation for session types presented in [5], except that standard subtyping is covariant with respect to external choices. This difference is motivated by the synchronous communication model we have adopted (see Section 5 for a more detailed discussion). The appeal for unfair subtyping comes from its semplicity and intuitive rationale. The key clause **(4)** states that the larger session type allows in general for fewer kinds of messages to be sent: when $T \leqslant_{\mathsf{U}} S$, a process behaving according to S can be safely placed in a context where a process behaving according to T is expected because S expresses a more deterministic behavior compared to T. Reducing non-determinism is generally perceived as harmless, but sometimes it may compromise liveness.

Example 3.1. Consider the session types $T = !\mathsf{a}.x \oplus !\mathsf{b}$ and $S = !\mathsf{a}.x$ and the context $\mathscr{C} = \mu x.[\,]$. Then both $\{(T,S),(x,x)\}$ and $\{(\mathscr{C}[T],\mathscr{C}[S])\}$ are coinductive subtyping relations, from which we deduce $T \leqslant_{\mathsf{U}} S$ and $\mathscr{C}[T] \leqslant_{\mathsf{U}} \mathscr{C}[S]$. Yet Example 2.2 shows that neither $\mathscr{C}[T] \leqslant \mathscr{C}[S]$ nor $T \leqslant S$ do hold. ■

Unfair subtyping is a necessary but not sufficient condition for fair subtyping.

Theorem 3.1. $\leqslant \subsetneq \leqslant_{\mathsf{U}}$.

3.2 Towards the Characterization of Fair Subtyping

We introduce some handy notation for referring to the residual of a session type after some sequence of actions.

Definition 3.2 (continuation). *Let $\alpha \in \mathtt{tr}(T)$. The* continuation *of T after α is the session type S such that $T \stackrel{\varepsilon}{\Longrightarrow} \stackrel{\alpha}{\longrightarrow} S$ (note that S is uniquely determined because branches in session types are guarded by distinct actions). We extend the notion of continuation to sequences of actions so that $T(\varepsilon) = T$ and $T(\alpha\varphi) = T(\alpha)(\varphi)$ when $\alpha\varphi \in \mathtt{tr}(T)$.*

Example 3.1 shows that some branches of internal choices cannot be pruned without compromising session success. Therefore, we must identify additional conditions to restrict clause **(4)** of Definition 3.1 so that $T \leqslant_{\mathsf{U}} S$ implies $T \leqslant S$. The condition

$$\mathtt{tr}(T) \subseteq \mathtt{tr}(S) \tag{1}$$

Table 3. Convergence relation.

$$\frac{\forall \varphi \in \mathtt{tr}(T) \setminus \mathtt{tr}(S) : \exists \psi \leq \varphi, \mathtt{a} : T(\psi!\mathtt{a}) \sqsubseteq S(\psi!\mathtt{a})}{T \sqsubseteq S}$$

is clearly sufficient but also overly restrictive: it imposes invariance of fair subtyping with respect to outputs, collapsing fair subtyping to equality. Condition (1) plays nonetheless a key role: we will show that, when $T \leqslant S$, there must be an "inevitable" pair of corresponding continuations of T and S at some "finite distance" from T and S for which condition (1) holds. To illustrate, consider the session types $T = \mu x.(!\mathtt{a}.(!\mathtt{a}.x \oplus !\mathtt{c}) \oplus !\mathtt{b})$ and $S = \mu x.(!\mathtt{a}.!\mathtt{a}.x \oplus !\mathtt{b})$ depicted as the graphs in Figure 1(a). Comparing T and S we see that they differ because of the missing $!\mathtt{c}$-labeled branch in S. We also observe that the difference occurs along a loop having a $!\mathtt{b}$-labeled exit path shared by both T and S and leading to an end state where condition (1) holds ($\mathtt{tr}(\mathsf{end}) \subseteq \mathtt{tr}(\mathsf{end})$). This state is "inevitable" in the sense that the path leading to it departs from an internal choice. By contrast, consider now $T' = \mu x.(?\mathtt{a}.(!\mathtt{a}.x \oplus !\mathtt{c}) + ?\mathtt{b})$ and $S' = \mu x.(?\mathtt{a}.!\mathtt{a}.x + ?\mathtt{b})$ which are similar to T and S above except that the outermost choice is now an external one. Their graphs (Figure 1(b)) do indeed share a loop with an exit path leading to a region of T' and S' in which the traces of the former are included in those of the latter. However, this path starts from an external choice and therefore the end state is not "inevitable".

We formalize the inevitable reachability of an indistinguishable region of T and S saying that T *converges into* S, where the convergence relation is inductively defined in Table 3. To get acquainted with convergence, keep in mind that it is inductively defined and observe that its base case corresponds exactly to enforcing condition (1):

$$\frac{\mathtt{tr}(T) \subseteq \mathtt{tr}(S)}{T \sqsubseteq S}$$

In the general case, imagine some session M that strives to detect any difference between T and S, in the sense that M succeeds (emitting $!\mathtt{OK}$) as soon as it enters some trace of T that is not present in S. In order to achieve its goal, M will try to drive the interaction along some path $\varphi \in \mathtt{tr}(T) \setminus \mathtt{tr}(S)$. This is where convergence comes into play: the rule in Table 3 states that after following some prefix ψ of φ that is shared by both T and S, M encounters an internal choice of S having a branch (corresponding to

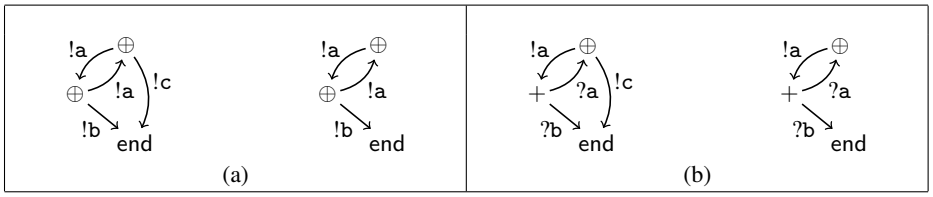


Fig. 1. Graph representation of session types.

some action $!\mathtt{a}$) that may divert the interaction to a new stage where the residual behaviors of T and S (respectively $T(\psi!\mathtt{a})$ and $S(\psi!\mathtt{a})$) have sets of traces that are slightly less different. We say "slightly less different" because $T(\psi!\mathtt{a})$ and $S(\psi!\mathtt{a})$ are one step closer to the top of the derivation of $T \sqsubseteq S$, whose leaves imply condition (1). Since the convergence relation is defined inductively, this means that T and S are a *finite number* of stages away from the point where condition (1) holds. In conclusion, when $T \sqsubseteq S$ holds, it is impossible for M to solely rely on the traces in $\mathtt{tr}(T) \setminus \mathtt{tr}(S)$ in order to succeed; M can always be veered into a stage of the interaction where (some continuations of) T and S are no longer distinguishable as all the traces of T are also traces of S.

We can now restrict $\leqslant_{\mathsf{U}}$ to converging pairs of session types:

Definition 3.3 (fair coinductive subtyping). *A coinductive subtyping $\mathscr{U}$ is* fair *if $T\,\mathscr{U}\,S$ implies $T \sqsubseteq S$. Let $\leqslant_{\mathsf{F}}$ denote the largest fair coinductive subtyping.*

The relation $\leqslant_{\mathsf{F}}$ does indeed preserve success:

Theorem 3.2. *Let $T \leqslant_{\mathsf{F}} S$. Then $M \mid T$ successful implies $M \mid S$ successful for every M.*

Moreover, $\leqslant_{\mathsf{F}}$ is a pre-congruence for internal and external choices and it fully characterizes fair subtyping for closed session types:

Proposition 3.1. *Let T, S be closed. Then $T \leqslant_{\mathsf{F}} S$ if and only if $T \leqslant S$.*

Example 3.2. We are now able to show that the session types $T = \mu x.(!\mathtt{a}.x \oplus !\mathtt{b})$ and $S = \mu x.(!\mathtt{a}.!\mathtt{a}.x \oplus !\mathtt{b})$ considered in Example 2.2 are related by fair subtyping. Observe that $T \leqslant_{\mathsf{U}} S$ and that $\mathtt{tr}(T) \setminus \mathtt{tr}(S)$ is the language of strings generated by the regular expression $!\mathtt{a}(!\mathtt{a}!\mathtt{a})^*!\mathtt{b}$. Given an arbitrary string in $\mathtt{tr}(T) \setminus \mathtt{tr}(S)$ we can take $\psi = \varepsilon$ and we have $T(!\mathtt{b}) = S(!\mathtt{b}) = \mathsf{end}$ where $\mathsf{end} \sqsubseteq \mathsf{end}$, so we can conclude $T \sqsubseteq S$. In general, the reader may verify that $T(\varphi) \sqsubseteq S(\varphi)$ holds for every $\varphi \in \mathtt{tr}(S)$. Therefore we have $T \leqslant_{\mathsf{F}} S$. Since T and S are closed, we conclude $T \leqslant S$ by Proposition 3.1. ■

Example 3.3. Consider again the session types $T = !\mathtt{a}.x \oplus !\mathtt{b}$ and $S = !\mathtt{a}.x$ and the context $\mathscr{C} = \mu x.[\,]$ and recall that in Example 3.1 we showed $T \leqslant_{\mathsf{U}} S$ and $\mathscr{C}[T] \leqslant_{\mathsf{U}} \mathscr{C}[S]$. Let us try to build a derivation for $\mathscr{C}[T] \sqsubseteq \mathscr{C}[S]$. Note that $\mathtt{tr}(\mathscr{C}[T]) \setminus \mathtt{tr}(\mathscr{C}[S])$ is the language of strings generated by the regular expression $(!\mathtt{a})^*!\mathtt{b}$. Taken $\varphi \in \mathtt{tr}(\mathscr{C}[T]) \setminus \mathtt{tr}(\mathscr{C}[S])$ we have that any prefix ψ of φ that is in $\mathtt{tr}(\mathscr{C}[T]) \cap \mathtt{tr}(\mathscr{C}[S])$ has the form $!\mathtt{a} \cdots !\mathtt{a}$ and now $\mathscr{C}[T](\psi!\mathtt{a}) = \mathscr{C}[T]$ and $\mathscr{C}[S](\psi!\mathtt{a}) = \mathscr{C}[S]$. Therefore, in order to prove $\mathscr{C}[T] \sqsubseteq \mathscr{C}[S]$, we need a derivation for $\mathscr{C}[T] \sqsubseteq \mathscr{C}[S]$. Since convergence is an inductive relation, $\mathscr{C}[T] \sqsubseteq \mathscr{C}[S]$ cannot be proved and $\mathscr{C}[T] \not\leqslant_{\mathsf{F}} \mathscr{C}[S]$ which agrees with the fact that these two session types are not related by fair subtyping.

However, it is easy to build a derivation for $T \sqsubseteq S$. This is because $\mathtt{tr}(T) \setminus \mathtt{tr}(S) = \{!\mathtt{b}\}$ and by taking $\psi = \varepsilon$ we have $T(!\mathtt{a}) = S(!\mathtt{a}) = x$ where $x \sqsubseteq x$. Therefore, $T \leqslant_{\mathsf{F}} S$ even though $T \not\leqslant S$. ■

Example 3.3 shows that $\leqslant_{\mathsf{F}}$ is not yet a full pre-congruence, hence we have:

Corollary 3.1. $\leqslant \subsetneq \leqslant_{\mathsf{F}} \subsetneq \leqslant_{\mathsf{U}}$.

3.3 Fair Subtyping for Open Session Types

Corollary 3.1 shows that $\leqslant_{\mathsf{F}}$ is a better over-approximation of $\leqslant$ than $\leqslant_{\mathsf{U}}$, but we have seen that $\leqslant_{\mathsf{F}}$ is not a pre-congruence for recursion. Part of the problem is that convergence (Table 3) does not distinguish the traces leading to end from the traces leading to variables. For example, $\mathtt{tr}(\mathsf{end}) = \mathtt{tr}(x) = \{\varepsilon\}$ but end and x are clearly different. In this subsection we address this last issue and arrive at the precise characterization of $\leqslant$.

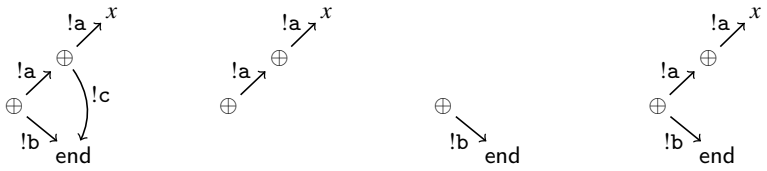
We start by defining the traces of a session type leading to a variable:

Definition 3.4 (X-traces). *The* X-traces *of* T*, denoted by* $X\text{-}\mathtt{tr}(T)$*, are the traces of* T *leading to a variable in* X*. That is,* $X\text{-}\mathtt{tr}(T) = \{\varphi \mid \exists x \in X : T \stackrel{\varphi}{\Longrightarrow} x\}$.

Our goal is to identify the condition that, together with $T \leqslant_{\mathsf{F}} S$, allows us to deduce $\mu x.T \leqslant_{\mathsf{F}} \mu x.S$. To study the problem, consider the session types

$$T = \mathsf{!a}.(\mathsf{!a}.x \oplus \mathsf{!c}) \oplus \mathsf{!b} \qquad S_1 = \mathsf{!a}.\mathsf{!a}.x \qquad S_2 = \mathsf{!b} \qquad S_3 = \mathsf{!a}.\mathsf{!a}.x \oplus \mathsf{!b}$$

depicted as the four graphs below:



From previous discussions we know that $\mu x.T \not\leqslant \mu x.S_1$ while $T \leqslant_{\mathsf{F}} S_1$ because $T \sqsubseteq S_1$. The critical aspect of T and S_1 is that the variable x is *dangerous* in the sense that it lies at the end of a path shared by T and S_1 and that goes through corresponding states of T and S_1 that differ. So, x permits the creation of a loop through which it is possible to detect the differences between the traces of T and those of S_1. On the other hand we have $\mu x.T \leqslant \mu x.S_2$ and therefore $\mu x.T \leqslant_{\mathsf{F}} \mu x.S_2$. In this case no loop can be created because there is no trace in $\{x\}\text{-}\mathtt{tr}(S_2)$ (recall from Definition 3.1 that $X\text{-}\mathtt{tr}(S_2) \subseteq X\text{-}\mathtt{tr}(T)$). This discussion suggests to strengthen condition (1) with

$$\{x\}\text{-}\mathtt{tr}(S) = \emptyset \tag{2}$$

to characterize the pairs of session types T and S that can be safely closed by a context $\mu x.[\,]$ when $T \leqslant_{\mathsf{F}} S$. Just like condition (1), however, condition (2) is excessive. For example, we have seen that $\mu x.T \leqslant \mu x.S_3$ even if $\{x\}\text{-}\mathtt{tr}(S_3) = \{\mathsf{!a}\mathsf{!a}\}$. The crucial difference between S_1 and S_3 is that, while in the former the reachability of the dangerous variable x is *granted* (the only maximal trace of S_1 leads to x), with S_3 the interaction can be diverted into the end state from which x is not reachable any more. The idea is that we should weaken condition (2) by requiring that it must hold "inevitably" at some "finite distance" from S, just like we did when we weakened condition (1) into $\sqsubseteq$. Generalizing this intuition to sets of variables, we arrive at the definition of another convergence relation $T \sqsubseteq_{X;Y} S$ inductively defined in Table 4 and indexed by two sets of variables: X is the set of variables that *may be* dangerous (if they are found to lie

Table 4. Open convergence relation.

$$\frac{\forall \varphi \in (\mathtt{tr}(T) \setminus \mathtt{tr}(S)) \cup Y\text{-}\mathtt{tr}(S) : \exists \psi \le \varphi, \mathtt{a} : T(\psi!\mathtt{a}) \sqsubseteq_{\emptyset;X\cup Y} S(\psi!\mathtt{a})}{T \sqsubseteq_{X;Y} S}$$

on a path that distinguishes T from S); Y is the set of variables that *are* dangerous (because they do lie on a path that distinguishes T from S). We will often write $\sqsubseteq_X$ as an abbreviation for $\sqsubseteq_{X;\emptyset}$.

The rules defining convergence (Table 3) and open convergence (Table 4) are structurally very similar, the only differences being the condition that we want to enforce inevitably (that the traces of T are included in those of S *and* no dangerous variables are reachable) and the use of the indexes X and Y. In particular, the variables in X that may be dangerous (in the conclusion of the rule) become dangerous (in the premise of the rule) when they are discovered to lie on a path that distinguishes T from S (or that leads to the dangerous variables in Y). Open convergence can be explained following the same intuition that we have already used for $\sqsubseteq$. In the case of $T \sqsubseteq_{\{x\}} S$, we should imagine a session M that strives to follow any trace in $\mathtt{tr}(T) \setminus \mathtt{tr}(S)$, concluding that this attempt is not guaranteed to succeed because $M \mid S$ can be diverted into a region of T and S where $\mathtt{tr}(T) \subseteq \mathtt{tr}(S)$ and $\{x\}\text{-}\mathtt{tr}(S) = \emptyset$. Then, when T and S are closed by a context $\mathscr{C} = \mu x.[\,]$, M is not able to drive the interaction along a loop that goes through $\mathscr{C}[T]$ and $\mathscr{C}[S]$ infinitely often so as to detect the differences between the traces of T and those of S, determining $\mathscr{C}[T] \leqslant_{\mathsf{F}} \mathscr{C}[S]$ if $T \leqslant_{\mathsf{F}} S$.

We summarize here a few properties of open convergence:

Proposition 3.2. *The following properties hold:*

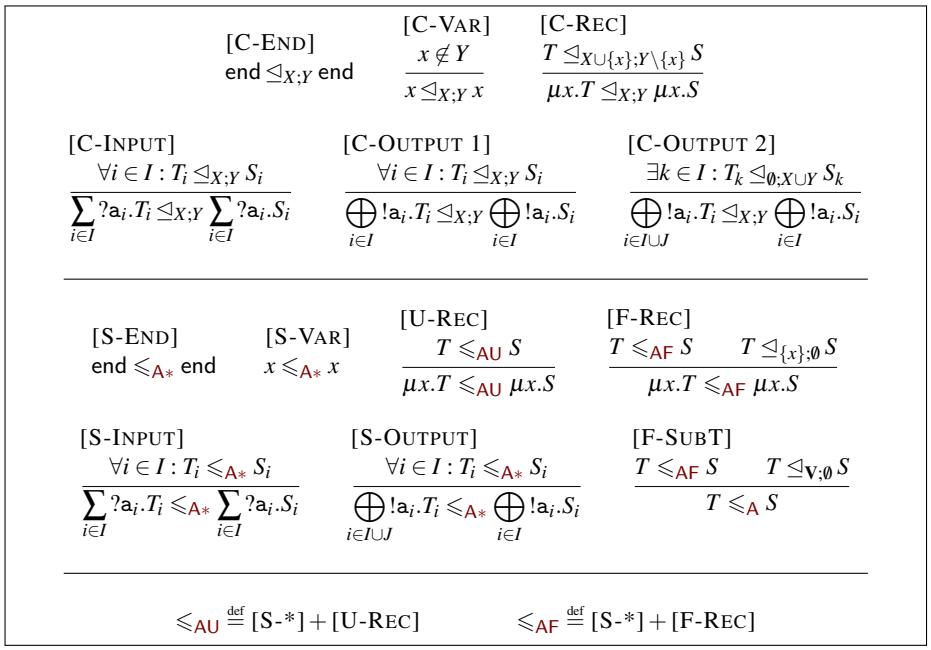
1. $\sqsubseteq_\emptyset = \sqsubseteq$;
2. $Y \subseteq X$ *implies* $\sqsubseteq_X \subseteq \sqsubseteq_Y$;
3. $T \sqsubseteq_{X\cup\{x\}} S$ *implies* $\mu x.T \sqsubseteq_X \mu x.S$.

Property (1) shows that $\sqsubseteq_X$ is a generalization of $\sqsubseteq$. Property (2) shows that $\sqsubseteq_X$ is contravariant in X, which agrees with the intuition that the larger the set of possibly dangerous variables, the more restrictive $\sqsubseteq_X$ is. Finally, property (3) formalizes the idea that $\sqsubseteq_X$ gives the missing condition that makes $\leqslant_{\mathsf{F}}$ a pre-congruence: the X annotation indicates the set of free variables of T and S that can be safely closed by a context $\mathscr{C}$ applied to T and S without compromising convergence. In general, by checking whether $T \sqsubseteq_{\mathbf{V}} S$ holds, we can be sure that $\mathscr{C}[T] \sqsubseteq \mathscr{C}[S]$ holds for all contexts $\mathscr{C}$. In other words, we can characterize $\leqslant$ as the intersection of $\leqslant_{\mathsf{F}}$ and $\sqsubseteq_{\mathbf{V}}$:

Theorem 3.3. $\leqslant \;=\; \leqslant_{\mathsf{F}} \cap \sqsubseteq_{\mathbf{V}}$.

Example 3.4. Take $T = !\mathtt{a}.x \oplus !\mathtt{b}$ and $S = !\mathtt{a}.x$ which were shown to be related by $\leqslant_{\mathsf{F}}$ in Example 3.3. We have $\mathtt{tr}(T) \setminus \mathtt{tr}(S) = \{!\mathtt{b}\}$ hence the only way to have them related by $\sqsubseteq_{\mathbf{V}}$ is to prove $T(!\mathtt{a}) \sqsubseteq_{\emptyset;\mathbf{V}} S(!\mathtt{a})$, but $T(!\mathtt{a}) = S(!\mathtt{a}) = x$ and $x \not\sqsubseteq_{\emptyset;\mathbf{V}} x$. We conclude $T \not\sqsubseteq_{\mathbf{V}} S$, as expected from $T \not\leqslant S$. ■

Table 5. Axiomatization of open convergence, unfair and fair subtyping.



Note the fundamental difference between $\sqsubseteq$ and $\sqsubseteq_{\mathbf{V}}$: the former one is an invariant that must hold for all pairs of corresponding continuations of two session types, while the latter one applies *una tantum* at the top level of two related session types. This distinction is motivated by the observation that applying a context $\mu x.[\,]$ around T creates a loop that necessarily goes through the initial state of T while no context applied to T can create loops strictly inside T. This property contributes to making $\leqslant$ context sensitive, in the sense that the relation between two types may depend on the context in which the types occur. For example, consider the session types $T = !\mathsf{a}.x \oplus !\mathsf{b}$ and $S = !\mathsf{a}.x$ and the context $\mathscr{C} = !\mathsf{a}.[\,] \oplus !\mathsf{b}$ and note that $\mathscr{C}$ does not bind the variable x. Now we have $\mathscr{C}[T] \sqsubseteq_{\{x\}} \mathscr{C}[S]$ while $T \not\sqsubseteq_{\{x\}} S$. Therefore $\mathscr{C}[T] \leqslant \mathscr{C}[S]$ even if $T \not\leqslant S$.

4 Axioms and Algorithms

The characterization developed in Section 3 allows us to produce an axiomatization for fair subtyping. Besides being the first complete axiomatization of a liveness-preserving refinement pre-congruence, the axiomatization shows how to guarantee liveness preservation by patching unfair subtyping or, more generally, standard subtyping for session types [5] with appropriate conditions in just two strategic places.

Table 5 defines four inductive relations $\trianglelefteq_{X;Y}$ (rules [C-*]), $\leqslant_{\mathsf{AU}}$ (rules [S-*] and [U-REC]), $\leqslant_{\mathsf{AF}}$ (rules [S-*] and [F-REC]), and $\leqslant_{\mathsf{A}}$ (rule [F-SUBT]) which will be shown to coincide with $\sqsubseteq_{X;Y}$, $\leqslant_{\mathsf{U}}$, $\leqslant_{\mathsf{F}}$, and $\leqslant$. The axiomatization of $\leqslant_{\mathsf{U}}$ follows directly from clauses **(1–4)** of Definition 3.1 with the addition of the pre-congruence rule [U-REC]

for recursions. The axiomatization of $\leqslant_{\mathsf{F}}$ is defined using the same core rules as unfair subtyping, but the pre-congruence rule [F-REC] for recursion requires the condition $T \trianglelefteq_{\{x\};\emptyset} S$ that verifies whether it is safe to close T and S with the context $\mu x.[\,]$ (see Proposition 3.2(3)). Fair subtyping is defined by [F-SUBT], which is just Theorem 3.3 in the form of inference rule. The axiomatization of open convergence includes a core set of rules where [C-END], [C-INPUT], and [C-OUTPUT 1] enforce trace inclusion (condition $\mathtt{tr}(T) \subseteq \mathtt{tr}(S)$ in Table 4) and rule [C-VAR] checks that x is not a dangerous variable (condition $Y\text{-}\mathtt{tr}(S) = \emptyset$ in Table 4). Rule [C-OUTPUT 2] deals with the case in which the larger session type provides strictly fewer choices with respect to the smaller one and corresponds to the "existential part" of the rule in Table 4. In this case, there must be a common branch ($k \in I$) such that the corresponding continuations are in the open convergence relation where all the possibly dangerous variables have become dangerous ones ($T_k \sqsubseteq_{\emptyset;X\cup Y} S_k$). Finally, the rule [C-REC] deals with recursive contexts $\mu x.[\,]$ by recording x as a possibly dangerous variable.

Theorem 4.1 (correctness). *We have: (1)* $\trianglelefteq_{X;Y} \subseteq \sqsubseteq_{X;Y}$*; (2)* $\leqslant_{\mathsf{AF}} \subseteq \leqslant_{\mathsf{F}}$*; (3)* $\leqslant_{\mathsf{A}} \subseteq \leqslant$.

The axiomatization is complete when session types have recursive terms binding the same variable in corresponding positions. This is not a limitation:

Proposition 4.1. *Let* $T \leqslant_{\mathsf{U}} S$*. Then* $T = T' \leqslant_{\mathsf{AU}} S' = S$ *for some* T' *and* S'.

Theorem 4.2 (completeness). *Let* $T \leqslant_{\mathsf{AU}} S$*. Then: (1)* $T \sqsubseteq_{X;Y} S$ *implies* $T \trianglelefteq_{X;Y} S$*; (2)* $T \leqslant_{\mathsf{F}} S$ *implies* $T \leqslant_{\mathsf{AF}} S$*; (3)* $T \leqslant S$ *implies* $T \leqslant_{\mathsf{A}} S$.

We briefly discuss an algorithm for deciding fair subtyping based on its axiomatization. The only two rules in Table 5 that are not syntax directed are [C-OUTPUT 1] and [C-OUTPUT 2] when $J \setminus I = \emptyset$ because the sets of (possibly) dangerous variables may or may not change when going from the conclusion to the premise of these rules. A naive algorithm would have to backtrack in case the wrong choice is made, leading to exponential complexity. Table 6 presents an alternative set of syntax-directed rules for open convergence. Space constraints prevent us from describing them in detail, but the the guiding principle of these rules, which are directly derived from the axiomatization, is simple: a judgment $T \#_X S \blacktriangleright Y$ *synthesizes*, whenever possible, the *smallest* subset Y of X such that $T \trianglelefteq_{Y;X\setminus Y} S$ holds, where X is the overall set of (possibly) dangerous variables. This way, in [AC-OUTPUT 1] and [AC-OUTPUT 2] the index set X does not change from the conclusion to the premises, so the algorithm can just recur and then verify whether $T_k \#_X S_k \blacktriangleright \emptyset$ for some branch $k \in I$: if this is the case, then [AC-OUTPUT 2] applies; if not and $J \setminus I = \emptyset$, then [AC-OUTPUT 1] applies; otherwise, the algorithm fails. This new set of rules is sound and complete:

Theorem 4.3. *The following properties hold:*

1. $T \#_Y S \blacktriangleright X$ *implies* $T \trianglelefteq_{X;Y\setminus X} S$*;*
2. $T \trianglelefteq_{X;Y} S$ *implies* $T \#_{X\cup Y} S \blacktriangleright Z$ *and* $Z \subseteq X$.

Regarding the complexity of the proposed algorithm, observe that open convergence can be decided in linear time using the rules in Table 6 and that, in Table 5, only [F-REC] and [F-SUBTT] duplicate work. Moreover, rule [F-SUBT] is needed only once for

Table 6. Algorithmic rules for open convergence.

$$\text{[AC-End]}\quad \mathsf{end} \#_X \mathsf{end} \blacktriangleright \emptyset$$

$$\text{[AC-Var 1]}\quad \frac{x \notin X}{x \#_X x \blacktriangleright \emptyset}$$

$$\text{[AC-Var 2]}\quad \frac{x \in X}{x \#_X x \blacktriangleright \{x\}}$$

$$\text{[AC-Rec]}\quad \frac{T \#_{X\cup\{x\}} S \blacktriangleright Y}{\mu x.T \#_X \mu x.S \blacktriangleright Y \setminus \{x\}}$$

$$\text{[AC-Input]}\quad \frac{\forall i \in I : T_i \#_X S_i \blacktriangleright X_i}{\sum_{i\in I} ?\mathtt{a}_i.T_i \#_X \sum_{i\in I} ?\mathtt{a}_i.S_i \blacktriangleright \bigcup_{i\in I} X_i}$$

$$\text{[AC-Output 1]}\quad \frac{\forall i \in I : (T_i \#_X S_i \blacktriangleright X_i \wedge X_i \neq \emptyset)}{\bigoplus_{i\in I} !\mathtt{a}_i.T_i \#_X \bigoplus_{i\in I} !\mathtt{a}_i.S_i \blacktriangleright \bigcup_{i\in I} X_i}$$

$$\text{[AC-Output 2]}\quad \frac{\exists k \in I : T_k \#_X S_k \blacktriangleright \emptyset}{\bigoplus_{i\in I\cup J} !\mathtt{a}_i.T_i \#_X \bigoplus_{i\in I} !\mathtt{a}_i.S_i \blacktriangleright \emptyset}$$

each derivation. Therefore, the algorithm for fair subtyping is quadratic in the size of the proof tree for $T' \leqslant_{\mathsf{A}} S'$, which is the same as $\|S'\|$ (the number of distinct subtrees in S'). Since $\|S'\|$ in the (constructive) proof of Proposition 4.1 is bound by $\|T\| \cdot \|S\|$, the overall complexity for deciding $T \leqslant S$ is $O(n^4)$ where $n = \max\{\|T\|, \|S\|\}$.

5 Related Work

It has already been observed [2] that the standard subtyping relation for session types is related to the *must-testing* pre-order, and in Section 1 we have anticipated the close correspondence between fair subtyping and the should-testing pre-congruence [12] which, in turn, is also related to fair testing [9]. Fair subtyping shares with the should-testing pre-congruence its semantic definition (Definition 2.2), but differs in that the authors of [12] work with a richer language of processes. Regarding the differences between should-testing and fair subtyping, though, the single feature that has the biggest impact is the stratification of the language: our type language disallows parallel compositions inside T terms, while in [12] parallel composition can occur anywhere. This feature increases the discriminating power of tests in [12] and ultimately implies that $\mathtt{tr}(T) = \mathtt{tr}(S)$ when $T \leqslant S$. To see why, consider $T = !\mathtt{a} \oplus !\mathtt{b}$ and $S = !\mathtt{a}$ and the context $\mathscr{C} = \mu x.((?\mathtt{a}.x + ?\mathtt{b}.!\mathtt{done}) \mid [\,])$ and notice that parallel composition occurs underneath a recursion. The intuition is that the term $\mathscr{C}[T]$ "restarts" T if T chooses to emit $\mathtt{a}$ and terminates if T chooses to emit $\mathtt{b}$. So, while it is always possible for T to emit $\mathtt{b}$, the term $\mathscr{C}[S]$ is doomed to diverge hopelessly. Indeed $?\mathtt{done}.!\mathtt{OK} \mid \mathscr{C}[T]$ is successful while $?\mathtt{done}.!\mathtt{OK} \mid \mathscr{C}[S]$ is not, determining $T \not\leqslant S$. In our context, the ability to restart a process an arbitrary number of times is too powerful: session types are associated with *linear channels* which have exactly one owner at any given point in time. Also, the type of the channel reduces as the channel is used and the type system forbids "jumps" to previous stages of the protocol described by the session type, unless the session type itself allows to do so by means of an explicit recursion. If the condition $\mathtt{tr}(T) \subseteq \mathtt{tr}(S)$ were to hold also in our context, fair subtyping would collapse to equality (modulo folding/unfolding of recursions) and would therefore lack any interest whatsoever. We conjecture that the restriction of should-testing to finite-state processes (where parallel composition is forbidden underneath recursions) shares the same trace-related properties (and possibly the behavioral characterization) of fair subtyping.

This work generalizes and simplifies our previous work [10], where fair subtyping was introduced for the first time. It is a generalization because it extends fair subtyping to open session types, whereas [10] only considered closed ones. Also, in the present work we purposefully adopt a notion of "session correctness" (Definition 2.1) that is weaker (i.e. more general) than the notions of session correctness usually adopted in conventional session type theories. For instance, dyadic sessions are based on complementary behaviors [5] and multi-party session type theories either on fair termination (e.g., [10,3]) or progress (e.g., [8]) of all participants. These stronger notions of session correctness can be encoded in terms of success. Since fair subtyping is defined as the relation that preserves success (Definition 2.2), the net effect is that the results presented here apply to all session type theories based on stronger notions of session correctness. With respect to [10], this work simplifies the characterization of fair subtyping. In particular, the convergence relation $T \sqsubseteq S$ is defined in [10] as a test for type emptiness for a derived term $T - S$ representing the behavioral difference of T and S. Here we give a direct definition of $\sqsubseteq$ with no need for auxiliary operators or notions of type emptiness.

Session type theories usually assume an asynchronous communication model. Besides being practically more relevant than the synchronous model we adopt here, asynchrony makes it possible to relax subtyping so that it is covariant with respect to external choices. Conversely, the synchronous communication model we have chosen imposes invariance (see clause **(3)** of Definition 3.1) because input actions are observable. This mismatch is irrelevant as far as the characterization and the essential properties of fair subtyping are concerned, because the additional branches of external choices resulting from covariance play no role in the characterization of fair subtyping, which is instead entirely focused on determining which branches of internal choices can be safely pruned without compromising success. In fact, the additional discriminating power given by synchronous communication makes our fair subtyping relation *finer* than necessary regarding external choices. Therefore, the results presented in this work are sound in contexts adopting weaker communication models and can be extended in a straightforward way to the standard subtyping relation of [5]. Finally, synchronous communication allows us to keep the formal model manageable and easier to compare with related ones [9,12] that are all based on synchronous communication.

Another simplification we have made is that messages are abstracted as atomic names. Actions of conventional session types are usually annotated with more precise type information like $!\mathtt{a}\langle int\rangle$, which specifies the type of arguments carried by messages. Higher-order session types are also considered. These aspects turn out to have no impact on the properties of fair subtyping, and have been neglected in the present paper for the sake of simplicity. The reader interested in the combined treatment of fairness and higher-order session types may refer to the extended version of [10].

6 Concluding Remarks

Our long-term goal is the definition of type systems for reasoning on liveness properties of programs structured around sessions. Using types for this purpose poses lots of unexpected challenges. For instance, we realized that the widely adopted typing rules

for recursive processes that take advantage of equirecursive types are unsound when the subtyping relation is liveness preserving. To illustrate the issue, consider the derivation

$$\dfrac{\dfrac{\vdash m : \mathtt{a} \qquad \dfrac{}{\mathscr{X} \mapsto \{k : T\}; k : T \vdash \mathscr{X}}\,[\textsc{Var}]}{\mathscr{X} \mapsto \{k : T\}; k : \mathtt{!a}.T \vdash k!m.\mathscr{X}}\,[\textsc{Output}] \qquad \mathtt{!a}.T \oplus \mathtt{!b} \leqslant \mathtt{!a}.T}{\dfrac{\mathscr{X} \mapsto \{k : T\}; k : \mathtt{!a}.T \oplus \mathtt{!b} \vdash k!m.\mathscr{X}}{k : T \vdash \mathtt{rec}\ x.k!m.\mathscr{X}}\,[\textsc{Rec}]}\,[\textsc{Sub}]$$

which is essentially the same derivation of Section 1 except that, instead of opening the session type $T = \mu x.(\mathtt{!a}.x \oplus \mathtt{!b})$ and associating the process variable $\mathscr{X}$ with the type variable x in the process environment, we identify T with its own unfolding $\mathtt{!a}.T \oplus \mathtt{!b}$. Now, we would like to reject this derivation for the same reasons we wanted to reject the one in Section 1. There, we argued that the derivation had to be rejected because it relied on the invalid law $\mathtt{!a}.x \oplus \mathtt{!b} \leqslant \mathtt{!a}.x$. But in the derivation above, the law $\mathtt{!a}.T \oplus \mathtt{!b} \leqslant \mathtt{!a}.T$ holds even for fair subtyping! The problem originates from using subsumption within a recursion: even if the (valid) relation $\mathtt{!a}.T \oplus \mathtt{!b} \leqslant \mathtt{!a}.T$ is applied *once* in the derivation, in practice it is used *infinitely many times* for enforcing $T \leqslant \mu x.\mathtt{!a}.x$, which is invalid. We are not aware of alternative ways of tackling this issue other than forbidding subsumption within recursions, or using a theory of open session types like the one developed in the present paper.

References

1. Franco Barbanera and Ugo de'Liguoro. Two notions of sub-behaviour for session-based client/server systems. In *Proceedings of PPDP'10*, pages 155–164. ACM, 2010.
2. Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, Elena Giachino, and Luca Padovani. Foundations of session types. In *Proceedings of PPDP'09*, pages 219–230. ACM, 2009.
3. Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, and Luca Padovani. On Global Types and Multi-Party Sessions. *Logical Methods in Computer Science*, 8:1–45, 2012.
4. Giuseppe Castagna, Rocco De Nicola, and Daniele Varacca. Semantic subtyping for the pi-calculus. *Theoretical Computer Science*, 398(1-3):217–242, 2008.
5. Simon Gay and Malcolm Hole. Subtyping for session types in the π-calculus. *Acta Informatica*, 42(2-3):191–225, 2005.
6. Kohei Honda. Types for dyadic interaction. In *Proceedings of CONCUR'93*, LNCS 715, pages 509–523. Springer, 1993.
7. Kohei Honda, Vasco T. Vasconcelos, and Makoto Kubo. Language primitives and type disciplines for structured communication-based programming. In *Proceedings of ESOP'98*, LNCS 1381, pages 122–138. Springer, 1998.
8. Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In *Proceedings of POPL'08*, pages 273–284. ACM, 2008.
9. V. Natarajan and Rance Cleaveland. Divergence and fair testing. In *Proceedings of ICALP'95*, LNCS 944, pages 648–659. Springer, 1995.
10. Luca Padovani. Fair Subtyping for Multi-Party Session Types. In *Proceedings of COORDINATION'11*, volume LNCS 6721, pages 127–141. Springer, 2011. Extended version available at `http://www.di.unito.it/~padovani/Papers/FairSubtypingLong.pdf`.
11. Benjamin Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–453, 1996.
12. Arend Rensink and Walter Vogler. Fair testing. *Information and Computation*, 205(2):125–198, 2007.

A Supplement to Section 3

Proof of Theorem 3.1

Theorem A.1 (Theorem 3.1). $\leqslant \subseteq \leqslant_{\mathsf{U}}$.

Proof. We must prove that $\leqslant$ is a coinductive subtyping. Assume $T \leqslant S$. We reason by cases on the structure of T.

- ($T = \mathsf{end}$) We reason by contradiction to show that $S = \mathsf{end}$:
  - Suppose $S \xrightarrow{?\mathtt{a}}$ and let $M = !\mathtt{a}.!\mathtt{c} \,|\, !\mathtt{b} \,|\, ?\mathtt{b}.!\mathtt{OK} + ?\mathtt{c}$ where $\mathtt{b}$ and $\mathtt{c}$ occur nowhere else. Then $T \,|\, M$ is successful and $S \,|\, M$ is not, which is absurd.
  - Suppose $S \xRightarrow{!\mathtt{a}}$ and let $M = ?\mathtt{a} + ?\mathtt{b}.!\mathtt{OK} \,|\, !\mathtt{b}$ where $\mathtt{b}$ occurs nowhere else. Then $T \,|\, M$ is successful and $S \,|\, M$ is not, which is absurd.
  - Suppose $S = x$ for some x. Let $\mathscr{C} = \mu x.?\mathtt{a}.[\,]$ and $M = !\mathtt{a}.!\mathtt{a} \,|\, ?\mathtt{a}.!\mathtt{OK}$. We have $\mathscr{C}[T] \,|\, M$ successful and $\mathscr{C}[S] \,|\, M$ unsuccessful, which is absurd.
- ($T = x$) We reason by contradiction to show that $S = x$:
  - Suppose $S = \mathsf{end}$ or $S = y \neq x$. Let $\mathscr{C} = \mu x.?\mathtt{a}.[\,]$ and $M = !\mathtt{a}.!\mathtt{a}.!\mathtt{OK}$. Then $\mathscr{C}[T] \,|\, M$ is successful and $\mathscr{C}[S] \,|\, M$ is not, which is absurd.
  - If $S \xrightarrow{?\mathtt{a}}$ or $S \xRightarrow{!\mathtt{a}}$ we can reason as for the case $T = \mathsf{end}$.
- ($T = \sum_{i\in I} ?\mathtt{a}_i.T_i$) Let $M = !\mathtt{a}_k.!\mathtt{b}_1.\cdots!\mathtt{b}_n \,|\, ?\mathtt{b}_1.R_1 \,|\, \cdots \,|\, ?\mathtt{b}_n.R_n$ where $k \in I$ and $T_k \,|\, R_1 \,|\, \cdots \,|\, R_n$ successful and $\mathtt{b}_1, \ldots, \mathtt{b}_n$ occur nowhere else. Then $T \,|\, M$ is successful. We deduce $S = \sum_{i\in J} ?\mathtt{a}_i.S_i$ with $I \subseteq J$ because k is an arbitrary element of I. Also, $T_i \leqslant S_i$ for every $i \in I$ because $R_1 \,|\, \cdots \,|\, R_n$ is arbitrary.
- ($T = \bigoplus_{i\in I} !\mathtt{a}_i.T_i$) Let $M = \sum_{i\in I} ?\mathtt{a}_i.!\mathtt{b}_{i1}.\cdots!\mathtt{b}_{in} \,|\, \sum_{i\in I} ?\mathtt{b}_{i1}.R_{i1} \,|\, \cdots \,|\, \sum_{i\in I} ?\mathtt{b}_{in}.R_{in}$ where $T_i \,|\, R_{i1} \,|\, \cdots \,|\, R_{in}$ is successful and the $\mathtt{b}_{i1}, \ldots, \mathtt{b}_{in}$ occur nowhere else for every $i \in I$ (we can assume that the number n of components required to build a successful session for T_i is the same for every i by adding suitable end session types). Then $T \,|\, M$ is successful. We deduce $S = \bigoplus_{i\in J} !\mathtt{a}_i.S_i$ with $J \subseteq I$. Also, $T_i \leqslant S_i$ for every $i \in J$ because $R_{i1} \,|\, \cdots \,|\, R_{in}$ is an arbitrary successful test for T_i. □

Proof of Theorem 3.2

Proposition A.1. *Let $T \leqslant_{\mathsf{F}} S$ and $M \,|\, T$ be successful and $M \,|\, S \xRightarrow{\tau} N \,|\, S'$. Then there exists $T' \leqslant_{\mathsf{F}} S'$ such that $M \,|\, T \xRightarrow{\tau} N \,|\, T'$.*

Definition A.1 (coinductive divergence). *We say that $\mathscr{D}$ is a* coinductive divergence *if $T\ \mathscr{D}\ S$ implies that there exists $\varphi \in \mathtt{tr}(T) \setminus \mathtt{tr}(S)$ such that for every $\psi \leq \varphi$ and $\mathtt{a}$ with $\psi!\mathtt{a} \in \mathtt{tr}(S)$ we have $T(\psi!\mathtt{a})\ \mathscr{D}\ S(\psi!\mathtt{a})$.*

Proposition A.2. *$\not\sqsubseteq$ is the largest coinductive divergence.*

Proof. Follows from the definition of $\sqsubseteq$. □

Proposition A.3. *Let $T \leqslant_{\mathsf{U}} S$ and $T \not\sqsubseteq S$. Then either*

1. $T = \sum_{i\in I} ?\mathtt{a}_i.T_i$ *and* $S = \sum_{i\in I} ?\mathtt{a}_i.S_i$ *and* $T_i \not\sqsubseteq S_i$ *for some* $i \in I$, *or*
2. $T = \bigoplus_{i\in I\cup J} !\mathtt{a}_i.T_i$ *and* $S = \bigoplus_{i\in I} !\mathtt{a}_i.S_i$ *and* $T_i \not\sqsubseteq S_i$ *for every* $i \in I$.

Proof. Follows from the definition of $\not\sqsubseteq$. □

Theorem A.2 (Theorem 3.2). *Let* $T \leqslant_{\mathsf{F}} S$. *Then* $M \mid T$ *successful implies* $M \mid S$ *successful for every* M.

Proof. Let $M \mid T$ be successful and consider a derivation $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S'$. We have to prove $N \mid S' \stackrel{!\mathtt{OK}}{\Longrightarrow}$. From Proposition A.1 we deduce that there exists $T' \leqslant_{\mathsf{F}} S'$ such that $M \mid T \stackrel{\tau}{\Longrightarrow} N \mid T'$. Then $T' \sqsubseteq S'$. We do an induction on $T' \sqsubseteq S'$ to show $N \mid S' \stackrel{!\mathtt{OK}}{\Longrightarrow}$. From the hypothesis $N \mid T'$ successful we deduce $N \stackrel{\overline{\varphi}!\mathtt{OK}}{\Longrightarrow}$ and $T' \stackrel{\varphi}{\Longrightarrow}$ for some φ. Without loss of generality, we can assume that φ is a string of minimal length leading N to success, that is $N \stackrel{\overline{\psi}!\mathtt{OK}}{\not\Longrightarrow}$ for every $\psi < \varphi$.

In the base case we have $\mathtt{tr}(T') \subseteq \mathtt{tr}(S')$, therefore we conclude $S' \stackrel{\varphi}{\Longrightarrow}$.

In the inductive case, suppose $\varphi \in \mathtt{tr}(T') \setminus \mathtt{tr}(S')$ for otherwise there is nothing left to prove. By definition of $\sqsubseteq$ we deduce that there exist $\psi \leq \varphi$ and $\mathtt{a}$ such that $T'(\psi!\mathtt{a}) \sqsubseteq S'(\psi!\mathtt{a})$. From the hypothesis $N \mid T'$ successful and the fact that φ is a string of minimal length leading N to success we deduce $N \stackrel{\overline{\psi}?\mathtt{a}}{\Longrightarrow} N'$ where $N' \mid T'(\psi!\mathtt{a})$ is successful. We conclude by induction hypothesis. □

Proof of Proposition 3.1

Lemma A.1. *Let* $T \leqslant_{\mathsf{U}} S$ *and* $T \not\sqsubseteq S$. *Then* $T \not\leqslant S$.

Proof. We show how to build an $\mathscr{M}(T,S)$ such that $\mathscr{M}(T,S) \mid T$ is successful and $\mathscr{M}(T,S) \mid S$ is not under the hypothesis $T \leqslant_{\mathsf{U}} S$ and $T \not\sqsubseteq S$. Let $\mathscr{M}(T,S)$ be the session type coinductively defined by the equations:

$$\mathscr{M}(T,S) \stackrel{\text{def}}{=} \begin{cases} \bigoplus\limits_{i\in I, T_i \not\sqsubseteq S_i} !\mathtt{a}_i.\mathscr{M}(T_i,S_i) & \text{if } T = \sum\limits_{i\in I} ?\mathtt{a}_i.T_i \text{ and } S = \sum\limits_{i\in I} ?\mathtt{a}_i.S_i \\ \sum\limits_{i\in I} ?\mathtt{a}_i.\mathscr{M}(T_i,S_i) + \sum\limits_{i\in J\setminus I} ?\mathtt{a}_i.!\mathtt{OK} & \text{if } T = \bigoplus\limits_{i\in I\cup J} !\mathtt{a}_i.T_i \text{ and } S = \bigoplus\limits_{i\in I} !\mathtt{a}_i.S_i \end{cases}$$

First of all we show that $\mathscr{M}(T,S)$ is well defined, which amounts to showing that all of its choices have at least one branch. This is a straightforward consequence of Proposition A.3.

Now consider a derivation of $\mathscr{M}(T,S) \mid T \stackrel{\tau}{\Longrightarrow} R \mid T'$ where $\mathscr{M}(T,S) \stackrel{\overline{\varphi}}{\Longrightarrow} R \stackrel{!\mathtt{OK}}{\not\Longrightarrow}$ and $T \stackrel{\varphi}{\Longrightarrow} T'$. Without loss of generality we may assume $R = \mathscr{M}(T,S)(\overline{\varphi})$ and $T' = T(\varphi)$. If this is not the case, by the very definition of $\mathscr{M}(T,S)$ the computation can always be extended by one more synchronization so that this assumption holds. From the hypothesis $R \stackrel{!\mathtt{OK}}{\not\Longrightarrow}$ we deduce that $\varphi \in \mathtt{tr}(S)$. Moreover, by definition of $\mathscr{M}(T,S)$ we know $T(\varphi) \not\sqsubseteq S(\varphi)$. We deduce that there exists $\varphi' \in \mathtt{tr}(T(\varphi)) \setminus \mathtt{tr}(S(\varphi))$ such that $R \stackrel{\overline{\varphi'}!\mathtt{OK}}{\Longrightarrow}$. We conclude that $\mathscr{M}(T,S) \mid T$ is successful.

To see that $\mathscr{M}(T,S) \mid S$ is unsuccessful it suffices to observe that $\mathscr{M}(T,S) \stackrel{\overline{\varphi}!\mathtt{OK}}{\Longrightarrow}$ implies $S \stackrel{\varphi}{\not\Rightarrow}$ by construction of $\mathscr{M}(T,S)$. □

Proposition A.4 (Proposition 3.1). *Let T,S be closed. Then $T \leqslant_{\mathsf{F}} S$ if and only if $T \leqslant S$.*

Proof. ($\Rightarrow$) It is enough to show that $T \sqsubseteq S$ implies $\mathscr{C}[T] \sqsubseteq \mathscr{C}[S]$ for every $\mathscr{C}$ when T and S are closed. This follows from an easy induction on $\mathscr{C}$.

($\Leftarrow$) We prove that $T \not\leqslant_{\mathsf{F}} S$ implies $T \not\leqslant S$. If $T \not\leqslant_{\mathsf{U}} S$, then we conclude $T \not\leqslant S$ from Theorem 3.1. If $T \leqslant_{\mathsf{U}} S$ and $T \not\sqsubseteq S$, then we conclude $T \not\leqslant S$ from Lemma A.1. □

Proof of Proposition 3.2

Lemma A.2. *$T \sqsubseteq_{\emptyset;X\cup\{x\}} S$ implies $T\{T_0/x\} \sqsubseteq_{\emptyset;X} S\{S_0/x\}$.*

Proof. By induction on the derivation of $T \sqsubseteq_{\emptyset;X\cup\{x\}} S$. In the base case we have $\mathtt{tr}(T) \subseteq \mathtt{tr}(S)$ and $(X\cup\{x\})\text{-}\mathtt{tr}(S) = \emptyset$, therefore $T\{T_0/x\} = T$ and $S\{S_0/x\} = S$ and we conclude immediately. In the inductive case, consider $\varphi \in (\mathtt{tr}(T\{T_0/x\}) \setminus \mathtt{tr}(S\{S_0/x\})) \cup X\text{-}\mathtt{tr}(S\{S_0/x\})$. Then we deduce either $\varphi \in \mathtt{tr}(T) \setminus \mathtt{tr}(S)$ or $\varphi \in X\text{-}\mathtt{tr}(S)$ or $\varphi = \varphi_1\varphi_2$ where $\varphi_1 \in \{x\}\text{-}\mathtt{tr}(S)$ and $\varphi_2 \in (\mathtt{tr}(T_0) \setminus \mathtt{tr}(S_0)) \cup X\text{-}\mathtt{tr}(S_0)$. In any of these cases, from the hypothesis $T \sqsubseteq_{\emptyset;X\cup\{x\}} S$ we deduce that there exist $\psi \le \varphi$ and $!\mathtt{a}$ such that $T(\psi!\mathtt{a}) \sqsubseteq_{\emptyset;X\cup\{x\}} S(\psi!\mathtt{a})$. By induction hypothesis we deduce $T(\psi!\mathtt{a})\{T_0/x\} \sqsubseteq_{\emptyset;X} S(\psi!\mathtt{a})\{S_0/x\}$. We conclude because φ is arbitrary. □

Lemma A.3. *$T \sqsubseteq_{X\cup\{x\};Y} S$ implies $\mu x.T \sqsubseteq_{X;Y} \mu x.S$.*

Proof. Let $T_0 = \mu x.T$ and $S_0 = \mu x.S$. Let $\varphi \in \mathtt{tr}(T_0) \setminus \mathtt{tr}(S_0) \cup Y\text{-}\mathtt{tr}(S_0)$. Then $\varphi = \varphi_1 \cdots \varphi_n$ where $\{\varphi_1,\ldots,\varphi_{n-1}\} \subseteq \{x\}\text{-}\mathtt{tr}(S)$ and $\varphi_n \in \mathtt{tr}(T) \setminus \mathtt{tr}(S) \cup Y\text{-}\mathtt{tr}(S)$. From the hypothesis $T \sqsubseteq_{X\cup\{x\};Y} S$ we deduce that there exist $\psi' \le \varphi_n$ and $\mathtt{a}$ such that $T(\psi'!\mathtt{a}) \sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S(\psi'!\mathtt{a})$. From Lemma A.2 we deduce $T\{T_0/x\}(\psi'!\mathtt{a}) \sqsubseteq_{\emptyset;X\cup Y} S\{S_0/x\}(\psi'!\mathtt{a})$. Take $\psi = \varphi_1 \cdots \varphi_{n-1}\psi'$ and observe that $\psi \le \varphi$ and $T_0(\psi!\mathtt{a}) = T\{T_0/x\}(\psi'!\mathtt{a})$ and $S_0(\psi!\mathtt{a}) = S\{S_0/x\}(\psi'!\mathtt{a})$. Since this arguments holds for an arbitrary φ, we can conclude $T_0 \sqsubseteq_{X;Y} S_0$. □

Lemma A.4 (Proposition 3.2). *The following properties hold:*

1. *$\sqsubseteq_\emptyset = \sqsubseteq$;*
2. *$Y \subseteq X$ implies $\sqsubseteq_X \subseteq \sqsubseteq_Y$;*
3. *$T \sqsubseteq_{X\cup\{x\}} S$ implies $\mu x.T \sqsubseteq_X \mu x.S$.*

Proof. Items (1) and (2) follow directly from the definition of $\sqsubseteq_{X;Y}$, while item (3) is Lemma A.3. □

Proof of Theorem 3.3

Definition A.2 (open coinductive divergence). *We say that $\mathscr{R}$ is an* open coinductive divergence *if $(X,Y,T,S) \in \mathscr{R}$ implies that there exists $\varphi \in (\mathtt{tr}(T) \setminus \mathtt{tr}(S)) \cup Y\text{-}\mathtt{tr}(S)$ such that for every $\psi \leq \varphi$ and $\mathtt{a}$ with $\psi!\mathtt{a} \in \mathtt{tr}(S)$ we have $(\emptyset, X \cup Y, T(\psi!\mathtt{a}), S(\psi!\mathtt{a})) \in \mathscr{R}$.*

Proposition A.5. *The following properties hold:*

1. *$\{(X,Y,T,S) \mid T \not\sqsubseteq_{X;Y} S\}$ is an open coinductive divergence;*
2. *if $\mathscr{R}$ is an open coinductive divergence and $(X,Y,T,S) \in \mathscr{R}$, then $T \not\sqsubseteq_{X;Y} S$.*

Lemma A.5. *$T \not\sqsubseteq_{X\cup\{x\};Y} S$ implies $\mu x.T \not\sqsubseteq_{X;Y} \mu x.S$.*

Proof. Let $T_0 = \mu x.T$ and $S_0 = \mu x.S$ and let

$$\begin{aligned}\mathscr{R} \stackrel{\text{def}}{=} \quad & \{(X,Y,T'\{T_0/x\},S'\{S_0/x\}) \mid T' \not\sqsubseteq_{X\cup\{x\};Y} S'\} \\ \cup\ & \{(\emptyset,X\cup Y,T'\{T_0/x\},S'\{S_0/x\}) \mid T' \not\sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S'\}\end{aligned}$$

We proceed showing that $\mathscr{R}$ is an open coinductive divergence. Then the proof is concluded, because $(X,Y,T_0,S_0) \in \mathscr{R}$ by definition of $\mathscr{R}$. Consider $(X',Y',T'',S'') \in \mathscr{R}$. From the definition of $\mathscr{R}$ it must be $T'' = T'\{T_0/x\}$ and $S'' = S'\{S_0/x\}$ for some T' and S' such that either $T' \not\sqsubseteq_{X\cup\{x\};Y} S'$ or $T' \not\sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S'$. We distinguish the two possibilities:

- $(T' \not\sqsubseteq_{X\cup\{x\};Y} S')$ From Proposition A.5 we deduce that there exists $\varphi \in \mathtt{tr}(T') \setminus \mathtt{tr}(S') \cup Y\text{-}\mathtt{tr}(S')$ such that for every $\psi \leq \varphi$ and $\mathtt{a}$ with $\psi!\mathtt{a} \in \mathtt{tr}(S')$ we have $T'(\psi!\mathtt{a}) \not\sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S'(\psi!\mathtt{a})$. Then $\varphi \in \mathtt{tr}(T'') \setminus \mathtt{tr}(S'') \cup Y\text{-}\mathtt{tr}(S'')$ and $\{\psi!\mathtt{a} \mid \psi \leq \varphi \wedge \psi!\mathtt{a} \in \mathtt{tr}(S') = \{\psi!\mathtt{a} \mid \psi \leq \varphi \wedge \psi!\mathtt{a} \in \mathtt{tr}(S'')\}$. We conclude that for every $\psi \leq \varphi$ and $\mathtt{a}$ with $\psi!\mathtt{a} \in \mathtt{tr}(S'')$ we have $(\emptyset, X \cup Y, T''(\psi!\mathtt{a}), S''(\psi!\mathtt{a})) \in \mathscr{R}$ by construction of $\mathscr{R}$, as required by Definition A.2.
- $(T' \not\sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S')$ From Proposition A.5 we deduce that there exists $\varphi_1 \in \mathtt{tr}(T') \setminus \mathtt{tr}(S') \cup (X \cup Y \cup \{x\})\text{-}\mathtt{tr}(S')$ such that for every $\psi \leq \varphi_1$ and $\mathtt{a}$ with $\psi!\mathtt{a} \in \mathtt{tr}(S')$ we have $T'(\psi!\mathtt{a}) \not\sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S'(\psi!\mathtt{a})$. We distinguish two subcases:
  - $(\varphi_1 \in \mathtt{tr}(T') \setminus \mathtt{tr}(S') \cup (X \cup Y)\text{-}\mathtt{tr}(S'))$ We conclude by taking $\varphi = \varphi_1$.
  - $(\varphi_1 \in \{x\}\text{-}\mathtt{tr}(S'))$ From the hypothesis $T \not\sqsubseteq_{X\cup\{x\};Y} S$ we deduce that there exists $\varphi_2 \in \mathtt{tr}(T) \setminus \mathtt{tr}(S) \cup Y\text{-}\mathtt{tr}(S)$ such that for every $\psi \leq \varphi_2$ and $\mathtt{a}$ with $\psi!\mathtt{a} \in \mathtt{tr}(S)$ we have $T(\psi!\mathtt{a}) \not\sqsubseteq_{\emptyset;X\cup Y\cup\{x\}} S(\psi!\mathtt{a})$. We conclude by taking $\varphi = \varphi_1\varphi_2$ and observing that $T'(\varphi_1) = S'(\varphi_1) = x$. □

Lemma A.6. *Let (1) $T \sqsubseteq S$ and (2) $T_0 \sqsubseteq S_0$. Then $T\{T_0/x\} \sqsubseteq S\{S_0/x\}$.*

Proof. An induction on **(1)** using **(2)** in the base case. □

Lemma A.7. *Let (1) $T \leqslant_{\mathsf{F}} S$ and (2) $T \sqsubseteq_{X\cup\{x\}} S$. Then $\mu x.T \leqslant_{\mathsf{F}} \mu x.S$ and $\mu x.T \sqsubseteq_X \mu x.S$.*

Proof. Let $T_0 = \mu x.T$ and $S_0 = \mu x.S$. From **(1)** and Theorem 3.1 we deduce $T \leqslant_{\mathsf{U}} S$. Since $\leqslant_{\mathsf{U}}$ is a full pre-congruence we have $T_0 \leqslant_{\mathsf{U}} S_0$. From **(2)** and Lemma A.3(3) we deduce $T_0 \sqsubseteq_X S_0$. All that remains to show is that $T_0(\varphi) \sqsubseteq S_0(\varphi)$ for every $\varphi \in \mathtt{tr}(S_0)$. Given any such trace φ, it must be the case that $\varphi = \varphi_1 \cdots \varphi_n$ where $\{\varphi_1, \ldots, \varphi_{n-1}\} \subseteq \{x\}\text{-}\mathtt{tr}(S)$ and $\varphi_n \in \mathtt{tr}(S)$. Then $T_0(\varphi) = T(\varphi_n)\{T_0/x\}$ and $S_0(\varphi) = S(\varphi_n)\{S_0/x\}$. From **(1)** we deduce $T(\varphi_n) \sqsubseteq S(\varphi_n)$. We conclude by Lemma A.6. □

Lemma A.8. *Let* ***(1)*** $T \leqslant_{\mathsf{F}} S$ *and* ***(2)*** $T \sqsubseteq_{\mathbf{V}} S$. *Then* $\mathscr{C}[T] \leqslant_{\mathsf{F}} \mathscr{C}[S]$ *and* $\mathscr{C}[T] \sqsubseteq_{\mathbf{V}} \mathscr{C}[S]$ *for every* $\mathscr{C}$.

Proof. By induction on $\mathscr{C}$ and by cases on its structure. The only interesting case is when $\mathscr{C} = \mu x.[\,]$ which is solved by Lemma A.7. □

Theorem A.3 (Theorem 3.3). $\leqslant = \leqslant_{\mathsf{F}} \cap \sqsubseteq_{\mathbf{V}}$.

Proof. ($\subseteq$) From Corollary 3.1 we know $\leqslant \subseteq \leqslant_{\mathsf{F}}$. Suppose, by contradiction, that there exist T and S such that $T \leqslant S$ and $T \not\sqsubseteq_{\mathbf{V}} S$. Then $T \not\sqsubseteq_X S$ where X is the set of free variables occurring in S. By Lemma A.5 we deduce that there exists $\mathscr{C}$ such that $\mathscr{C}[T] \not\sqsubseteq \mathscr{C}[S]$. By Lemma A.1 we deduce that that there exists M such that $M \,|\, \mathscr{C}[T]$ is successful and $M \,|\, \mathscr{C}[S]$ is not. This contradicts the hypothesis $T \leqslant S$.

($\supseteq$) Let M and $\mathscr{C}$ be such that $M \,|\, \mathscr{C}[T]$ is successful. By Lemma A.8 we have $\mathscr{C}[T] \leqslant_{\mathsf{F}} \mathscr{C}[S]$. We conclude that $M \,|\, \mathscr{C}[S]$ is successful by Theorem 3.2. □

B Supplement to Section 4

Proof of Theorem 4.1

Lemma B.1. $\trianglelefteq_{X;Y} \subseteq \sqsubseteq_{X;Y}$.

Proof. We proceed by induction on the derivation of **(1)** $T \trianglelefteq_{X;Y} S$ and by cases on the last rule applied.

- [C-END] and [C-VAR] are trivial because $\mathtt{tr}(T) \subseteq \mathtt{tr}(S)$ and $Y\text{-}\mathtt{tr}(S) = \emptyset$.
- [C-REC] Then $T \equiv \mu x.T'$ and $S \equiv \mu x.S'$ and $T' \trianglelefteq_{X\cup\{x\};Y} S'$. By induction hypothesis we deduce $T' \sqsubseteq_{X\cup\{x\};Y} S'$. We conclude by Lemma A.3.
- [C-INPUT] Then $T \equiv \sum_{i\in I} ?\mathsf{a}_i.T_i$ and $S \equiv \sum_{i\in I} ?\mathsf{a}_i.S_i$ and $T_i \trianglelefteq_{X;Y} S_i$ for every $i \in I$. By induction hypothesis we have $T_i \sqsubseteq_{X;Y} S_i$ for every $i \in I$. Let $\varphi \in \mathtt{tr}(T) \setminus \mathtt{tr}(S) \cup Y\text{-}\mathtt{tr}(S)$. Then $\varphi = ?\mathsf{a}_k\varphi'$ for some $k \in I$ and $\varphi' \in \mathtt{tr}(T_k) \setminus \mathtt{tr}(S_k) \cup Y\text{-}\mathtt{tr}(S_k)$. We deduce that there exist $\psi' \leq \varphi'$ and a such that $T_k(\psi'!\mathsf{a}) \sqsubseteq_{\emptyset;X\cup Y} S_k(\psi'!\mathsf{a})$. We conclude by taking $\psi = ?\mathsf{a}_k\psi'$.
- [C-OUTPUT 1] Analogous to the previous case.
- [C-OUTPUT 2] Then $T \equiv \bigoplus_{i\in I\cup J} !\mathsf{a}_i.T_i$ and $S \equiv \bigoplus_{i\in I} !\mathsf{a}_i.S_i$ and $J \setminus I \neq \emptyset$ and $T_k \trianglelefteq_{\emptyset;X\cup Y} S_k$ for some $k \in I$. By induction hypothesis we have $T_k \sqsubseteq_{\emptyset;X\cup Y} S_k$. Let $\varphi \in \mathtt{tr}(T) \setminus \mathtt{tr}(S) \cup Y\text{-}\mathtt{tr}(S)$. We conclude by taking $\psi = \varepsilon$ and $\mathsf{a} = \mathsf{a}_k$. □

Theorem B.1 (Theorem 4.1). *We have:* ***(1)*** $\trianglelefteq_{X;Y} \subseteq \sqsubseteq_{X;Y}$; ***(2)*** $\leqslant_{\mathsf{AF}} \subseteq \leqslant_{\mathsf{F}}$; ***(3)*** $\leqslant_{\mathsf{A}} \subseteq \leqslant$.

Proof. Item **(1)** follows from Lemma B.1. The proof of item **(2)** is a straightforward induction on the derivation of $T \leqslant_{\mathsf{AF}} S$, using the fact that $\leqslant$ is a pre-congruence for choices and Lemma A.7 for the case [A-REC]. Item **(3)** follows from Theorem 3.3. □

Proof of Theorem 4.2

Lemma B.2 (Proposition 4.1). *Let* $T_0 \leqslant_{\mathsf{U}} S_0$. *Then* $T_0 = T' \leqslant_{\mathsf{AU}} S' = S_0$ *for some* T' *and* S'.

Proof. We begin by defining a number of auxiliary objects:

- Let
$$\Delta_0 \stackrel{\text{def}}{=} \{(T_1,S_1),\ldots,(T_n,S_n)\} \stackrel{\text{def}}{=} \{(T_0(\varphi),S_0(\varphi)) \mid \varphi \in \mathtt{tr}(S_0)\}$$
be the set of pairs of corresponding continuations of T_0 and S_0. This set is finite because T_0 and S_0 are regular tree. Moreover, $n \leq \|T_0\| \cdot \|S_0\|$.
- Let $x_1,\ldots,x_n$ be n fresh variables.
- Let $\sigma_1 = \{T_i/x_i\}_{i=1..n}$ and $\sigma_2 = \{S_i/x_i\}_{i=1..n}$ be substitutions.
- Let $\Gamma = \{(T_i,S_i) \mapsto x_i\}_{i=1..n}$ be the map associating each pair of corresponding continuations of T_0 and S_0 to the i-th variable x_i. Note that $\mathtt{dom}(\Gamma) = \Delta_0$.
- Let Δ range over subsets of Δ_0.

We define a function $\mathtt{refold}(\Delta,T,S)$ that takes a context $\Delta\subseteq\Delta_0$ and two session types T and S such that $(T,S)\in\Delta_0$ and produces two new session types T' and S' such that $T=T'\leqslant_{\mathsf{AU}}S'=T'$:

$$\mathtt{refold}(\Delta,T,S)=\begin{cases}(x_k,x_k) & \text{if } (T,S)\in\Delta \text{ and } \Gamma(T,S)=x_k\\ (x,x) & \text{if } T=S=x\\ (\mathsf{end},\mathsf{end}) & \text{if } T=S=\mathsf{end}\\ (\mu x_k.\textstyle\sum_{i\in I}?\mathtt{a}_i.T_i',\mu x_k.\sum_{i\in I}?\mathtt{a}_i.S_i') & \text{if } T=\sum_{i\in I}?\mathtt{a}_i.T_i \text{ and } S=\sum_{i\in I}?\mathtt{a}_i.S_i\\ & \text{and } \mathtt{refold}(\Delta\cup\{(T,S)\},T_i,S_i)=(T_i',S_i')\\ & \text{for every } i\in I \text{ and } \Gamma(T,S)=x_k\\ (\mu x_k.\bigoplus_{i\in I\cup J}!\mathtt{a}_i.T_i',\mu x_k.\bigoplus_{i\in I}!\mathtt{a}_i.S_i') & \text{if } T=\bigoplus_{i\in I\cup J}!\mathtt{a}_i.T_i \text{ and } S=\bigoplus_{i\in I}!\mathtt{a}_i.S_i\\ & \text{and } \mathtt{refold}(\Delta\cup\{(T,S)\},T_i,S_i)=(T_i',S_i')\\ & \text{for every } i\in I \text{ and } T_i'=T_i \text{ for every } i\in J\setminus I\\ & \text{and } \Gamma(T,S)=x_k\end{cases}$$

By definition of $\mathtt{refold}$, we have that $\mathtt{refold}(\Delta,T,S)=(T',S')$ implies $T=T'\sigma_1$ and $S=S'\sigma_2$. All that remains to prove is $T'\leqslant_{\mathsf{AU}}S'$. We show that $\mathtt{refold}(\Delta,T,S)=(T',S')$ implies $T'\leqslant_{\mathsf{AU}}S'$ by induction on $\mathtt{dom}(\Gamma)\setminus\Delta$ and by cases on the definition of $\mathtt{refold}$:

- $((T,S)\in\mathtt{dom}(\Delta))$ Then $T'=S'=\Gamma(T,S)$ and we conclude with an application of rule [AU-VAR].
- $(T=S=x)$ We conclude with an application of rule [AU-VAR].
- $(T=S=\mathsf{end})$ We conclude with an application of rule [AU-END].
- $(T=\sum_{i\in I}?\mathtt{a}_i.T_i$ and $S=\sum_{i\in I}?\mathtt{a}_i.S_i)$ By induction hypothesis we deduce $T_i'\leqslant_{\mathsf{AU}}S_i'$ where $\mathtt{refold}(\Delta\cup\{(T,S)\},T_i,S_i)=(T_i',S_i')$ for every $i\in I$. Then $T'\equiv\mu x_k.\sum_{i\in I}?\mathtt{a}_i.T_i'$ and $S'\equiv\mu x_k.\sum_{i\in I}?\mathtt{a}_i.S_i'$. We conclude $T'\leqslant_{\mathsf{AU}}S'$ with an application of rule [AU-INPUT] followed by an application of rule [AU-REC].
- $(T=\bigoplus_{i\in I\cup J}!\mathtt{a}_i.T_i$ and $S=\bigoplus_{i\in I}!\mathtt{a}_i.S_i)$ Analogous to the previous case. □

Lemma B.3. *Let $T\sqsubseteq_{X;Y}S$. The following properties hold:*

1. *if $T\equiv\sum_{i\in I}?\mathtt{a}_i.T_i$ and $S\equiv\sum_{i\in I}?\mathtt{a}_i.S_i$, then $T_i\sqsubseteq_{X;Y}S_i$ for every $i\in I$;*
2. *if $T\equiv\bigoplus_{i\in I\cup J}!\mathtt{a}_i.T_i$ and $S\equiv\bigoplus_{i\in I}!\mathtt{a}_i.S_i$, then either $T_k\sqsubseteq_{\emptyset;X\cup Y}S_k$ for some $k\in I$ or $J\setminus I=\emptyset$ and $T_i\sqsubseteq_{X;Y}S_i$ for every $i\in I$;*
3. *if $T\equiv\mu x.T'$ and $S\equiv\mu x.S'$, then $T'\sqsubseteq_{X\cup\{x\};Y}S'$.*

Proof. We only prove item (2). Item (1) is a simpler variant of item (2) and item (3) is an immediate consequence of Lemma A.5. If $T_k\sqsubseteq_{\emptyset;X\cup Y}S_k$ for some $k\in I$, then there is nothing left to prove, so assume **(*)** $T_i\not\sqsubseteq_{\emptyset;X\cup Y}S_i$ for every $i\in I$. Let $i\in I$ and $\varphi\in(\mathtt{tr}(T_i)\setminus\mathtt{tr}(S_i))\cup Y\text{-}\mathtt{tr}(S_i)$. Then $!\mathtt{a}_i\varphi\in(\mathtt{tr}(T)\setminus\mathtt{tr}(S))\cup Y\text{-}\mathtt{tr}(S)$. From the hypothesis we deduce that there exist $\psi'\leq!\mathtt{a}_i\varphi$ and $\mathtt{a}$ such that $T(\psi'!\mathtt{a})\sqsubseteq_{\emptyset;X\cup Y}S(\psi'!\mathtt{a})$. It cannot be $\psi'=\varepsilon$, for otherwise we would have $\mathtt{a}=\mathtt{a}_k$ for some $k\in I$ and $T_k\sqsubseteq_{\emptyset;X\cup Y}S_k$, which contradicts **(*)**. Then $\psi'=!\mathtt{a}_i\psi$ for some $\psi\leq\varphi$. We conclude by observing that $T(\psi'!\mathtt{a})=T_i(\psi!\mathtt{a})$ and $S(\psi'!\mathtt{a})=S_i(\psi!\mathtt{a})$. □

Lemma B.4. *Let $T \leqslant_{\mathsf{AU}} S$ and $T \sqsubseteq_{X;Y} S$. Then $T \trianglelefteq_{X;Y} S$.*

Proof. An easy induction on the structure of T and S, using Lemma B.3. □

Lemma B.5. *$\mu x.T \leqslant \mu x.S$ implies $T \leqslant S$.*

Lemma B.6. *Let (1) $T \leqslant S$ and (2) $T \leqslant_{\mathsf{AU}} S$. Then $T \leqslant_{\mathsf{AF}} S$.*

Proof. By induction on the structure of T and S.

- ($T \equiv S \equiv \mathsf{end}$) We conclude by [U-End].
- ($T \equiv S \equiv x$ for some x). We conclude by [U-Var].
- ($T \equiv \sum_{i\in I} ?\mathsf{a}_i.T_i$ and $S \equiv \sum_{i\in I} ?\mathsf{a}_i.S_i$) Then $T_i \leqslant S_i$ and $T_i \leqslant_{\mathsf{U}} S_i$ for every $i \in I$. By induction hypothesis we deduce $T_i \leqslant_{\mathsf{A}} S_i$ for every $i \in I$. We conclude by [U-Input].
- ($T \equiv \bigoplus_{i\in I\cup J} !\mathsf{a}_i.T_i$ and $S \equiv \bigoplus_{i\in I} !\mathsf{a}_i.S_i$) Analogous to the previous case.
- ($T \equiv \mu x.T'$ and $S \equiv \mu x.S'$) Then $T' \leqslant_{\mathsf{U}} S'$. From (1) and Lemma B.5 we deduce $T' \leqslant S'$. By induction hypothesis we derive $T' \leqslant_{\mathsf{AF}} S'$. From (1) and Definition 2.2 we deduce $T \sqsubseteq_{\emptyset;\emptyset} S$. From Lemma B.3(3) we obtain $T' \sqsubseteq_{\{x\};\emptyset} S'$. From Lemma B.4 we obtain $T' \trianglelefteq_{\{x\};\emptyset} S'$. We conclude by [F-Rec]. □

Theorem B.2 (Theorem 4.2). *Let $T \leqslant_{\mathsf{AU}} S$. Then: (1) $T \sqsubseteq_{X;Y} S$ implies $T \trianglelefteq_{X;Y} S$; (2) $T \leqslant_{\mathsf{F}} S$ implies $T \leqslant_{\mathsf{AF}} S$; (3) $T \leqslant S$ implies $T \leqslant_{\mathsf{A}} S$.*

Proof. (1) follows from Lemma B.4; (2) follows from (1) and Lemma B.6; (3) follows from (2) and Theorem 3.3. □

Proof of Theorem 4.3

Proposition B.1. *$T \trianglelefteq_{X;Y\cup Z} S$ implies $T \trianglelefteq_{X\cup Y;Z} S$.*

Proof. Trivial induction on the derivation of $T \trianglelefteq_{X;Y\cup Z} S$. □

Theorem B.3 (Theorem 4.3). *The following properties hold:*

1. *$T \#_Y S \blacktriangleright X$ implies $T \trianglelefteq_{X;Y\setminus X} S$;*
2. *$T \trianglelefteq_{X;Y} S$ implies $T \#_{X\cup Y} S \blacktriangleright X'$ and $X' \subseteq X$.*

Proof. Regarding item (1), we proceed by induction on the derivation of $T \#_Y S \blacktriangleright X$ and by cases on the last rule applied.

- [AC-End] We conclude with an application of rule [C-End].
- [AC-Var 1] Then $T \equiv S \equiv x$ and $X = \emptyset$ and $x \notin Y$. We conclude with an application of [C-Var].
- [AC-Var 2] Then $T \equiv S \equiv x$ and $X = \{x\}$ and $x \in Y$. We conclude with an application of [C-Var].
- [AC-Rec] Then $T \equiv \mu x.T'$ and $S \equiv \mu x.S'$ and $x \notin Z$ and $T' \#_{Y\cup\{x\}} S' \blacktriangleright X'$ and $X = X' \setminus \{x\}$. By induction hypothesis we derive $T' \trianglelefteq_{X';(Y\cup\{x\})\setminus X'} S'$. From Proposition B.1 we derive $T' \trianglelefteq_{X'\cup\{x\};Y\setminus(X'\cup\{x\})} S'$. We conclude with an application of [C-Rec].

- [AC-INPUT] Then $T \equiv \sum_{i\in I} ?\mathsf{a}_i.T_i$ and $S \equiv \sum_{i\in I} ?\mathsf{a}_i.S_i$ and $T_i \mathbin{\#_Y} S_i \blacktriangleright X_i$ for every $i \in I$ and $X = \bigcup_{i\in I} X_i$. By induction hypothesis we derive $T_i \preceq_{X_i;Y\setminus X_i} S_i$ for every $i \in I$. From Proposition B.1 we derive $T_i \preceq_{X;Y\setminus X} S_i$ for every $i \in I$. We conclude with an application of [C-INPUT].
- [AC-OUTPUT 1] Analogous to the previous case.
- [AC-OUTPUT 2] Then $T \equiv \bigoplus_{i\in I\cup J} !\mathsf{a}_i.T_i$ and $S \equiv \bigoplus_{i\in I} !\mathsf{a}_i.S_i$ and $T_k \mathbin{\#_Y} S_k \blacktriangleright \emptyset$ for some $k \in I$ and $X = \emptyset$. By induction hypothesis we derive $T_k \preceq_{\emptyset;Y} S_k$. We conclude with an application of [C-OUTPUT 2].

Regarding item (2), we proceed by induction on the derivation of $T \preceq_{X;Y} S$ and by cases on the last rule applied:

- [C-END] Then $T \equiv S \equiv \mathsf{end}$. We conclude by taking $X' = \emptyset$ with an application of [AC-END].
- [C-VAR] Then $T \equiv S \equiv x$ and $x \notin Y$. If $x \notin X$, then we conclude by taking $X' = \emptyset$ with an application of [AC-VAR 1]. If $x \in X$, then we conclude by taking $X' = \{x\}$ with an application of [AC-VAR 2].
- [C-REC] Then $T \equiv \mu x.T'$ and $S \equiv \mu x.S'$ and $T \preceq_{X\cup\{x\};Y} S$. By induction hypothesis we derive $T' \mathbin{\#_{X\cup Y\cup\{x\}}} S' \blacktriangleright X''$ with $X'' \subseteq X \cup \{x\}$. We conclude by taking $X' = X'' \setminus \{x\}$ with an application of [AC-REC].
- [C-INPUT] Then $T \equiv \sum_{i\in I} ?\mathsf{a}_i.T_i$ and $S \equiv \sum_{i\in I} ?\mathsf{a}_i.S_i$ and $T_i \preceq_{X;Y} S_i$ for every $i \in I$. By induction hypothesis we derive $T_i \mathbin{\#_{X\cup Y}} S_i \blacktriangleright X_i$ and $X_i \subseteq X$ for every $i \in I$. We conclude by taking $X' = \bigcup_{i\in I} X_i$ with an application of [AC-INPUT].
- [C-OUTPUT 1] Then $T \equiv \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ and $S \equiv \bigoplus_{i\in I} !\mathsf{a}_i.S_i$ and $T_i \preceq_{X;Y} S_i$ for every $i \in I$. By induction hypothesis we derive $T_i \mathbin{\#_{X\cup Y}} S_i \blacktriangleright X_i$ and $X_i \subseteq X$ for every $i \in I$. We distinguish two subcases.
  - If $X_k = \emptyset$ for some $k \in I$, then we conclude by taking $X' = \emptyset$ with an application of [AC-OUTPUT 2].
  - If $X_i \neq \emptyset$ for every $i \in I$, then we conclude by taking $X' = \bigcup_{i\in I} X_i$ with an application of [AC-OUTPUT 1].
- [C-OUTPUT 2] Then $T \equiv \bigoplus_{i\in I} !\mathsf{a}_i.T_i$ and $S \equiv \bigoplus_{i\in I} !\mathsf{a}_i.S_i$ and $T_k \preceq_{\emptyset;X\cup Y} S_k$ for some $k \in I$. By induction hypothesis we derive $T_k \mathbin{\#_{X\cup Y}} S_k \blacktriangleright \emptyset$. We conclude with an application of [AC-OUTPUT 2]. □