Recursive Logical Relations for Intuitionistic Linear Logic Session Types

Stephanie Balzer[1], Farzaneh Derakshan[2], Robert Harper[1], and Yue Yao[1]

[1] Carnegie Mellon University, Pittsburgh PA
[2] Illinois Institute of Technology, Chicago IL

Abstract. Program *equivalence* is the fulcrum for reasoning about and proving properties of programs. To assert noninterference, for example, a program is shown to be equivalent to itself up to the confidentiality level of an observer. A powerful enabler for such proofs are *logical relations*, which, guided by the type structure, prescribe when two programs are indistinguishable. Logical relations enjoy ample exploration in functional languages, including languages with general recursion and a higher-order store—yet logical relations for session types only exist for terminating languages. This paper scales logical relations to *general recursive* session types. It develops a logical relation for *progress-sensitive equivalence* for *intuitionistic linear logic session types*, tackling the challenges non-termination and concurrency pose. In particular, the relation only equates a diverging program with another diverging one and accounts for nondeterminism of scheduling. The logical relation has two distinguishing characteristics: it is (i) indexed with an *intuitionistic linear sequent*, validating *cut reductions* and affording *biorthogonal closure*, and (ii) bound by an *observation index*, stratifying the logical relation in the presence of recursion. Biorthogonal closure validates the logical relation, proving that the induced equivalence is *sound and complete* with regard to closure of weak bisimilarity under parallel composition. Soundness guarantees that the equivalence has enough discriminatory power, completeness ensures that it is maximally permissive. The logical relation is then put to test on the example of noninterference.

Keywords: Logical relations · biorthogonality · step-indexing · intuitionistic linear logic session types · progress-sensitive noninterference.

1 Introduction

Whether two programs are *equivalent* is at the heart of many program verification problems, such as proofs of parametricity, compiler correctness, and noninterference. This paper develops a *logical relation* to reason about program equivalence of *session-typed processes* and proves soundness and completeness of the relation via a *biorthogonality* argument.

Message-passing concurrency. Rooted in process calculi [25, 41, 42], message-passing concurrency enjoys widespread adoption, including languages such as

Erlang, Go, and Rust. A program in this setting amounts to a number of *processes* connected by *channels*, which compute by exchanging *messages* along these channels. Messages can even amount to channels themselves, giving rise to so-called *higher-order* channels, as present in the π-calculus [43,53]. This feature is equally empowering as daunting because it changes the process topology dynamically. Originally untyped, the π-caluclus [43] has gradually been enriched with types [53] to prescribe the kinds of messages that can be exchanged over a channel. More advanced type systems [30, 31, 33, 34] additionally assert correctness properties, such as deadlock freedom and data race freedom.

To prescribe not only the types of exchanged messages but also the *protocol* underlying the exchange, *session types* [26, 27] were introduced. Session types rely on a *linear* treatment of channels to model the state transitions induced by a protocol. This foundation manifests in a correspondence between linear logic and the session-typed π-calculus [10,62], resulting in two families of session type languages: *intuitionistic linear logic session types (ILLST)* [10, 11, 59, 60] and *classical linear logic session types (CLLST)* [35, 37, 38, 62]. Due to their logical foundation well-typed ILLST/CLLST processes not only are protocol-compliant (a.k.a., preservation), but also free of data races and deadlocks (a.k.a., progress).

This paper studies logical relations for ILLST-typed processes. ILLST reject linear negation and distinguish the *provider* from the *client* side of a channel. Run-time configurations of processes thus give raise to a *rooted tree*, with a *child* node as the provider and the *parent* node as the client.

Program equivalence. Today's predominant techniques for reasoning about program equivalence of stateful programs are Kripke logical relations (KLRs) [3,19, 28,45,49,57] and bisimulations [36,52,55,56]. KLRs tend to be used for sequential, ML-like languages, bisimulations for concurrent process calculi. KLRs are phrased by structural induction on types, bisimulations by coinduction. Hur et al. [29] note that KLRs and bisimulations have mutually disjoint strengths and weaknesses. Whereas KLRS naturally support higher-order features, but struggle with recursive types, bisimulations naturally support recursion, but struggle with higher-order features. To support recursive types KLRs employ step-indexing [1,2,4], complicating proofs with step arithmetic [8] and challenging transitivity of the logical relation [29].

This paper scales logical relations to session-typed concurrency in the presence of *general recursive types*, contributing a *recursive session logical relation (RSLR)* for intuitionistic linear logic session types (ILLST). In contrast to KLRs, which index the logical relation with the type of the considered expression, RSLRs index the logical relation with an *intuitionistic linear sequent*, denoting the types of the free channels along which the considered process configuration exchanges messages with the outside world. The usual term interpretation clause

$$(e_1; e_2) \in \mathcal{E}[\![\tau]\!] \text{ iff } \ldots$$

for expressions e_1 and e_2 and a type τ in KLRs thus takes the form

$$(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash A]\!] \text{ iff } \ldots$$

in RSLRs, for configurations $\mathcal{D}_1$ and $\mathcal{D}_1$ with the types of free channels denoted by the sequent $\Delta \Vdash A$. This generalization is necessitated by the underlying computational model (message-passing concurrency) and the goal to accommodate various program verification problems, including noninterference. The use of a sequent as an index makes explicit the *duality* of session types, allowing a type to be associated with two interpretations: as a *provider* of a session of that type and a *client*. For example, as a provider of a session $\&\{\mathit{left}{:}A, \mathit{right}{:}B\}$, two related configurations may *assume* to receive related messages, i.e., either they both receive *left* or *right*. As a client of such a session, conversely, related configurations must *assert* to either both send *left* or *right*. The latter condition, crucial for establishing noninterference, is lost when only indexing the logical relation with one type (i.e., the type of the provider). Duality is also central to enforcing a *resource* semantics for channels. For example, as a provider of a session $A \otimes B$, promising to send a channel of type A, related configurations must give up access to the sent channel, whereas a client of such a session will become its owner. Thanks to the validity of semantic cut (Lem. 2), the usual, single-type-index interpretation falls out for free as a special case.

To accommodate general recursive types, RSLRs use an *observation index* to stratify the logical relation. An observation index m is associated with the free channels in $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash A]\!]^m$ and bounds the number of messages that $\mathcal{D}_1$ and $\mathcal{D}_2$ exchange along those channels. The index is thus associated with an *externally observable* event, symmetrically bounding both configurations. In contrast KLRs employ a step index [2,4,18], which is tied to the internal computation or unfolding steps of one of the two terms, making the relation asymmetric.

We establish several metatheoretic properties of RSLRs. In particular, we show that RSLRs entail an *equivalence relation*, whose proof of transitivity (Lem. 5) benefits from the extensionality of the observation index. Given the possibility of divergence, the equivalence is *progress-sensitive* [23] in that it equates a divergent program only with another diverging one. We also show that RSLRs are *sound and complete* (Thm. 1) with regard to closure of weak *bisimilarity* under parallel composition, using a *biorthogonality* argument à la Pitts [48]. The *duality* of session types establishes a natural connection to biorthogonality. To put the RSLR to test, we employ it to verify *progress-sensitive noninterference*.

Contributions. In summary, this paper makes the following contributions:

- *Recursive session logical relation (RSLR)*, a binary logical relation for *progress-sensitive equivalence* of ILLST processes with general recursion, featuring an *intuitionistic linear sequent* and *observation index*, to capture duality and to support general recursion without compromising extensionality, resp.
- Closure of the RSLR under parallel composition, a.k.a., *semantic cut* (Lem. 2).
- RSLR induces an *equivalence relation*, including *transitivity* (Lem. 5).
- Soundness and completeness of the RSLR with regard to *weak asynchronous bisimilarity*, a.k.a., *adequacy* (Lem. 6).
- Biorthogonal ($\top\top$-) closure of the induced equivalence relation (Thm. 1), guaranteeing that the equivalence has enough discriminatory power (soundness), while being maximally permissive (completeness).

– Small case study applying the RSLR to *progress-sensitive noninterference*.

The appendix includes the complete formalization and all the proofs.

2 Background

This section familiarizes with intuitionistic linear logic session types (ILLST).

ILLST type system. Our development is based on a variant of ILLST languages [10, 11, 59, 60] that supports general recursive types [5, 60]. We refer to this language as SESSION.

SESSION's connectives are drawn from intuitionistic linear logic and obey the following grammar

$$A, B ::= \oplus\{\ell{:}A_\ell\}_{\ell \in L} \mid \&\{\ell{:}A_\ell\}_{\ell \in L} \mid A \otimes B \mid A \multimap B \mid 1 \mid Y,$$

where L ranges over finite, non-empty sets of labels denoted by ℓ and k, the primitive values in SESSION. Type variable Y is a fixed point whose definition $Y = A$ is collected in a global signature Σ. General recursive types are supported through this definition mechanism. They must be *contractive* [21], demanding a message exchange before recurring, and *equi-recursive* [14], avoiding explicit (un)fold messages and relating types up to their unfolding.

Process terms are typed using the *intuitionistic sequent*

$$\Omega \vdash_\Sigma P :: x{:}A$$

to be read as *"process P provides a session of type A along channel variable x, given the typing of sessions offered along channel variables in Ω and given the type and process definitions in Σ"*. Ω is a *linear* context, consisting of assumptions $y_i{:}B_i$, indicating for each channel variable y_i its session type B_i. The signature Σ contains global type and process definitions and facilitates recursive type and process definitions. Bound channel variables are substituted with channels that are created at run-time upon process spawning. When the distinction is clear from the context we refer to either as "channels" for brevity.

ILLST rejects linear negation, allowing the singleton right-hand side of the sequent to be interpreted as the type of the *providing* process, which, conversely, is the *client* of the sessions in Ω. Channels can thus be typed with the type of the providing process, rather than typing the two channel endpoints dually, as done in classical linear logic session types (CLLST) [37, 38, 62]. To express the *duality* in behavior the ILLST type system is given as *sequent calculus*, comprising both a *right* and a *left* rule per connective. Right rules define a communication from the point of view of the provider, left rules from the point of view of the client.

Fig. 1 summarizes the typing rules. Cut reduction in the sequent calculus invites a computational, bottom-up reading of the rules, where the type of the conclusion denotes the protocol state of the provider before the message exchange and the type of the premise the protocol state after the message exchange. The

$$\frac{\Omega \vdash_\Sigma P :: x{:}A_k \qquad k \in L}{\Omega \vdash_\Sigma x.k; P :: x{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}} \oplus R \qquad \frac{\Omega, x{:}A_k \vdash_\Sigma Q_k :: y{:}C \quad \forall k \in L}{\Omega, x{:} \oplus \{\ell : A_\ell\}_{\ell \in L} \vdash_\Sigma \mathbf{case}\, x(\ell \Rightarrow Q_\ell)_{\ell \in L} :: y{:}C} \oplus L$$

$$\frac{\Omega \vdash_\Sigma Q_k :: x{:}A_k \qquad \forall k \in L}{\Omega \vdash_\Sigma \mathbf{case}\, x(\ell \Rightarrow Q_\ell)_{\ell \in I} :: x{:}\&\{\ell : A_\ell\}_{\ell \in L}} \& R \qquad \frac{\Omega, x{:}A_k \vdash_\Sigma P :: y{:}C \qquad k \in L}{\Omega, x{:}\&\{\ell : A_\ell\}_{\ell \in I} \vdash_\Sigma x.k; P :: y{:}C} \& L$$

$$\frac{\Omega \vdash_\Sigma P :: x{:}B}{\Omega, z{:}A \vdash_\Sigma \mathbf{send}\, z\, x; P :: x{:}A \otimes B} \otimes R \qquad \frac{\Omega, z{:}A, x{:}B \vdash_\Sigma P :: y{:}C}{\Omega, x{:}A \otimes B \vdash_\Sigma z \leftarrow \mathbf{recv}\, x; P :: y{:}C} \otimes L$$

$$\frac{\Omega, z{:}A \vdash_\Sigma P :: x{:}B}{\Omega \vdash_\Sigma z \leftarrow \mathbf{recv}\, x; P :: x{:}A \multimap B} \multimap R \qquad \frac{\Omega, x{:}B \vdash_\Sigma P :: y{:}C}{\Omega, z{:}A, x{:}A \multimap B \vdash_\Sigma \mathbf{send}\, z\, x; P :: y{:}C} \multimap L$$

$$\frac{}{\cdot \vdash_\Sigma \mathbf{close}\, x :: x : 1} 1R \qquad \frac{\Omega \vdash_\Sigma Q :: y : C}{\Omega, x : 1 \vdash_\Sigma \mathbf{wait}\, x; Q :: y : C} 1L$$

$$\frac{\Omega_1' \vdash X = P :: x'{:}A \in \Sigma \qquad \Omega_1, x{:}A \Vdash \gamma :: \Omega_1', x'{:}A \qquad \Omega_2, x{:}A \vdash_\Sigma Q :: y{:}C}{\Omega_1, \Omega_2 \vdash_\Sigma (x \leftarrow X[\gamma] \leftarrow \Omega_1); Q :: y{:}C} \textsc{Spawn}$$

$$\frac{\begin{array}{c} z'{:}C \vdash F_Y = \mathtt{Fwd}_{C,y' \leftarrow z'} :: y'{:}C \in \Sigma \\ Y = C \in \Sigma \qquad z{:}C, y{:}C \Vdash \gamma :: z'{:}C, y'{:}C \end{array}}{z{:}Y \vdash_\Sigma F_Y[\gamma] :: y{:}C} \textsc{D-Fwd}$$

$$\frac{Y = A \in \Sigma \qquad \Omega \vdash_\Sigma P :: x{:}A}{\Omega \vdash_\Sigma P :: x{:}Y} \textsc{TVar}_R \qquad \frac{Y = A \in \Sigma \qquad \Omega, x{:}A \vdash_\Sigma P :: z{:}C}{\Omega, x{:}Y \vdash_\Sigma P :: z{:}C} \textsc{TVar}_L$$

$$\frac{}{\Vdash_\Sigma (\cdot)\ \mathbf{sig}} \Sigma_1 \qquad \frac{\Vdash_\Sigma A\ \mathbf{contr} \qquad \Vdash_\Sigma \Sigma'\ \mathbf{sig}}{\Vdash_\Sigma Y = A, \Sigma'\ \mathbf{sig}} \Sigma_2 \qquad \frac{\Omega \vdash_\Sigma P :: x{:}A \qquad \Vdash_\Sigma \Sigma'\ \mathbf{sig}}{\Vdash_\Sigma \Omega \vdash_\Sigma X = P :: x{:}A, \Sigma'\ \mathbf{sig}} \Sigma_3$$

Fig. 1: Process term typing rules and signature checking rules of SESSION.

polarity of type moreover determines the direction of communication: for positive connectives, the provider sends and the client receives, for negative connectives, the provider receives and the client sends. We review each rule in turn next.

The additive connectives $\oplus\{\ell{:}A_\ell\}_{\ell \in L}$ and $\&\{\ell{:}A_\ell\}_{\ell \in L}$ denote *labelled choices* of sessions A_ℓ, where the labels ℓ range over the finite, non-empty set L. The two connectives differ in who sends a label and thus makes a choice. For $\oplus\{\ell{:}A_\ell\}_{\ell \in L}$, the provider chooses, for $\&\{\ell{:}A_\ell\}_{\ell \in L}$, the client chooses, giving the connectives the names *internal choice* and *external choice*, resp. The duality of behavior of a provider and client is conveyed by the process terms of the right and left rules, resp. For example, a provider of an internal choice (rule $\oplus R$) executes the process term $x.k; P$ to send the label k along its providing channel x, after which it will continue with P of type A_k. Conversely, a client of an internal choice (rule $\oplus L$) executes the process term $\mathbf{case}\, x(\ell \Rightarrow Q_\ell)_{\ell \in L}$ to receive any label $k \in L$ along x, after which it will continue with the chosen branch Q_k of type A_k.

The multiplicative connectives $A \otimes B$ (tensor) and $A \multimap B$ (lolli) allow channels themselves to be sent over channels, changing the process topology at run-time. These connectives endow SESSION with *higher-order channels*, making channels first-class values. Again, the two connectives differ in who sends a channel: for $A \otimes B$, the provider sends, for $A \multimap B$, the client sends. Due to linearity, absence of weakening specifically, the sender loses access to the sent channel in its continuation; as can be seen in the premises of rules $\otimes R$ and $\multimap L$. We use $\leftarrow$ to denote variable binding. For example, the process term $z \leftarrow \mathbf{recv}\, x; P$ in the conclusion of rule $\otimes L$ binds the received channel to y, with scope P.

The unit of $\otimes$, 1, allows a process to terminate. Here, the provider sends a close message, $\mathbf{close}\ x$, along its providing channel x (rule $1R$), awaited by the client. Due to absence of weakening, rule $1R$ demands that the typing context Ω be empty, ensuring that no channels become orphans. Consequently, the client continuation Q loses access to x after receipt of the close message (rule $1L$).

Rule Spawn types a process that spawns another process using the process definition X. The invocation $x \leftarrow X[\gamma] \leftarrow \Omega_1$ binds the newly spawned process to x and gives it the argument channels Ω_1. After the invocation, the process continues with executing Q, now with $x{:}A$ in its context. For the invocation to succeed, a corresponding process definition $\Omega_1' \vdash X = P :: x'{:}A$ must exist in the signature Σ. The definition associates a name X with the process term P and indicates the names and types of argument channels Ω_1' and the name and type of the providing channel $x'{:}A$. To account for the difference in variable names, the programmer must provide a corresponding mapping $\Omega_1, x{:}A \Vdash \gamma :: \Omega_1', x'{:}A$ as part of the invocation, assigning to each variable in the definition $\Omega_1', x'{:}A$ a corresponding variable of the same type in the invocation $\Omega_1, x{:}A$.

Rule D-Fwd allows a process to delegate requests to its singleton child z by passing any messages received along y to z and vice versa. The rule uses a forwarder definition $z'{:}C \vdash F_Y = \mathtt{Fwd}_{C,y' \leftarrow z'} :: y'{:}C$, for the type definition $Y = C$, both to be present in the signature Σ. A corresponding forwarder $\mathtt{Fwd}_{C,y' \leftarrow z'}$ of type C is defined by structural induction on the type of every user-provided type definition, amounting to an *identity expansion*, and can be automatically generated. In combination with rule D-Fwd, we get a coinductive definition. (see Def. 10 in the Appendix). The programmer can use the syntactic sugar $y \leftarrow z$ instead, which can be expanded into the corresponding forwarder.

Rules TVar$_R$ and TVar$_L$ allow us to unfold type definitions. The rules insist that a corresponding definition exists in the global signature $Y = A \in \Sigma$.

Rules Σ_1, Σ_2, and Σ_3 type check the signature itself. Rule Σ_2 type checks type definitions, requiring the type be contractive ($\Vdash_\Sigma A\ \mathbf{contr}$) and thus not a type variable itself [21]. Rule Σ_3 type checks the body P of a process definition X, whose existence is assumed when spawning the process in rule Spawn.

We illustrate the typing rules on a simple PIN-based authentication example, shown in Fig. 2. Fig. 2 introduces two recursive type definitions, pin and ver, as well as two process definitions, $\mathsf{Verifier}$ and Pin. A verifier expects to receive a PIN channel from its client, which it validates. If validation is successful, the verifier sends the message *succ*, otherwise the message *fail* to the client. In either case, the verifier also returns the PIN channel. The PIN itself is encoded as an internal choice of labels tok_i, with one being the correct security token.

$$\mathsf{ver} = \mathsf{pin} \multimap \oplus\{\mathit{succ}{:}\mathsf{pin} \otimes \mathsf{ver}, \mathit{fail}{:}\mathsf{pin} \otimes \mathsf{ver}\} \qquad \mathsf{pin} = \oplus\{\mathit{tok_1}{:}\mathsf{pin}, \ldots, \mathit{tok_n}{:}\mathsf{pin}\}$$

$$\begin{array}{l} \cdot \vdash \mathsf{Verifier} :: x{:}\mathsf{ver} = (\; u \leftarrow \mathbf{recv}\, x \quad // \; u{:}\mathsf{pin} \vdash x{:} \oplus \{\mathit{succ} : \mathsf{pin} \otimes \mathsf{ver}, \mathit{fail}{:}\mathsf{pin} \otimes \mathsf{ver}\} \\ \qquad\qquad\qquad\qquad\qquad \mathbf{case}\, u\, (\mathit{tok}_j \Rightarrow x.\mathit{succ}; \mathbf{send}\, u\, x; x \leftarrow \mathsf{Verifier} \leftarrow \cdot \\ \qquad\qquad\qquad\qquad\qquad\qquad \mid \mathit{tok}_{i \neq j} \Rightarrow x.\mathit{fail}; \mathbf{send}\, u\, x; x \leftarrow \mathsf{Verifier} \leftarrow \cdot)) \end{array}$$

$$\cdot \vdash \mathsf{Pin} :: u{:}\mathsf{pin} = (u.\mathit{tok}_j; u \leftarrow \mathsf{Pin} \leftarrow \cdot)$$

Fig. 2: Example: PIN-based authentication.

ILLST configuration typing At runtime, SESSION programs become configurations of processes, forming *rooted trees* and obeying the following grammar:

$$\begin{array}{lll} \mathcal{C}, \mathcal{D}, \mathcal{F}, \mathcal{T} & ::= & \mathbf{proc}(x_\alpha, \hat{\delta}(P))\, \mathcal{C} \;\mid\; \mathbf{msg}(M)\, \mathcal{C} \;\mid\; \cdot \\ M & ::= & x_\alpha.k \;\mid\; \mathbf{send}\, z_\beta\, x_\alpha \;\mid\; \mathbf{close}\, y_\alpha \end{array}$$

Configurations consist of a set of processes of the form $\mathbf{proc}(y_\alpha, \hat{\delta}(P))$ and a set of messages of the form $\mathbf{msg}(M)$. Every spawn results in a process $\mathbf{proc}(y_\alpha, \hat{\delta}(P))$, every send in a message $\mathbf{msg}(M)$, making SESSION's dynamics asynchronous. The metavariable y_α in $\mathbf{proc}(y_\alpha, \hat{\delta}(P))$ denotes the process' providing channel and the metavariable P the process' source code. Run-time channels y_α can be distinguished from channel variables y by their generation subscript α, whose meaning we clarify in § 2. The substitution δ maps variables to channels; $\hat{\delta}(P)$ yields the term with all free channel variables substituted by channels.

The configuration typing rules of SESSION are given in Fig. 3, using the judgment $\Delta_0 \Vdash \mathcal{C} :: \Delta$. We allow configurations to have *free channels*, which amount to Δ_0 and Δ for the given judgment. The channels in Δ comprise the channels for which there exist providing processes in the configuration $\mathcal{C}$, the channels in Δ_0 comprise the channels for which there exist client processes in the configuration $\mathcal{C}$. It may be surprising that we refer to these channels as free, given that they occur in C. Would linearity not imply that they occur "exactly once", and thus must be bound? It is more fruitful to think of channels as shared resources, like memory locations, *shared* among processes. ILLST then ensures that at most two processes, a provider and a client, share a channel at any point in time, where each "owns" one endpoint of (i.e., a reference to) the channel. The careful accounting of "resources" is visible in the configuration typing rules. For example, rule **proc** insists that for all the channel endpoints used by process $\mathbf{proc}(x_\alpha, \hat{\delta}(P))$ there exists either a provider in the sub-configuration $\mathcal{C}$, making the channel bound, or the endpoint is free and occurs in Δ_0'.

ILLST asynchronous dynamics SESSION's asynchronous dynamics is given in Fig. 4. It is phrased as multiset rewriting rules [12]. Multiset rewriting rules express the dynamics as state transitions between configurations $C \mapsto C'$ and are *local* in that they only mention the parts of a configuration they rewrite.

Being *asynchronous*, the dynamics implements sends by spawning off a message $\mathbf{msg}(M)$ that carries the sent message M. In rule $\otimes_{\mathsf{snd}}$, for example, the

$$\frac{}{x_\alpha{:}A \Vdash \cdot :: x_\alpha{:}A}\ \mathbf{emp_1} \qquad \frac{}{\cdot \Vdash \cdot :: \cdot}\ \mathbf{emp_2} \qquad \frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0' \Vdash \mathcal{C}_1 :: x_\alpha{:}A}{\Delta_0, \Delta_0' \Vdash \mathcal{C}\,\mathcal{C}_1 :: \Delta, x_\alpha{:}A}\ \mathbf{comp}$$

$$\frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0', \Delta, x_\alpha{:}A \vdash \delta :: \Omega_0', \Omega, x{:}A \qquad \Omega_0', \Omega \vdash_\Sigma P :: x{:}A}{\Delta_0, \Delta_0' \Vdash \mathcal{C}\ \mathbf{proc}(x_\alpha, \hat{\delta}(P)) :: x_\alpha{:}A}\ \mathbf{proc}$$

$$\frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0', \Delta \vdash M :: x_\alpha{:}A}{\Delta_0, \Delta_0' \Vdash \mathcal{C}\ \mathbf{msg}(M) :: x_\alpha{:}A}\ \mathbf{msg} \qquad \frac{k \in L}{y_{\alpha+1}{:}A_k \vdash y_\alpha.k :: y_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}}\ \mathbf{m}_{\oplus R}$$

$$\frac{k \in L}{x_\alpha{:}\&\{\ell : A_\ell\}_{\ell \in I} \vdash x_\alpha.k :: x_{\alpha+1}{:}A_k}\ \mathbf{m}_{\& L} \qquad \frac{}{z_\beta{:}A, y_{\alpha+1}{:}B \vdash \mathbf{send}\, z_\beta\, y_\alpha :: y_\alpha{:}A \otimes B}\ \mathbf{m}_{\otimes R}$$

$$\frac{}{z_\beta{:}A, x_\alpha{:}A \multimap B \vdash \mathbf{send}\, z_\beta\, x_\alpha :: x_{\alpha+1}{:}B}\ \mathbf{m}_{\multimap L} \qquad \frac{}{\cdot \vdash \mathbf{close}\, y_\alpha :: y_\alpha{:}1}\ \mathbf{m}_{1R}$$

Fig. 3: Configuration typing rules of $\mathsf{SESSION}$.

provider $\mathbf{proc}(y_\alpha, \mathbf{send}\, x_\beta\, y_\alpha; P)$ generates the message $\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)$, indicating that the channel x_β is sent over channel y_α. In rule $\otimes_{\mathsf{rcv}}$, the client then consumes the message when receiving the channel. To preserve the invariant stipulated by typing that at most two processes share a channel at any point in time, a new providing channel must be generated for the continuation of the provider. This new channel must be linked to the old providing channel, otherwise proper sequencing of messages would be violated.

An elegant way to achieve proper sequencing are *channel generations*. Rather than creating an entirely fresh channel name for each send, we keep the name of a channel constant but increment its generation, provided as a subscript to the channel name. For example, the provider $\mathbf{proc}(y_\alpha, \mathbf{send}\, x_\beta\, y_\alpha; P)$ in rule $\otimes_{\mathsf{snd}}$ steps to its continuation $\mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P)$, creating a new generation $\alpha + 1$ of its providing channel y_α. The receiving client process $\mathbf{proc}(u_\eta, w \leftarrow \mathbf{recv}\, y_\alpha; P)$ will then increment the channel generation of the carrier channel y_α upon receipt in its continuation $\mathbf{proc}(u_\eta, [x_\beta/w][y_{\alpha+1}/y_\alpha]P)$ (see rule $\otimes_{\mathsf{rcv}}$). Like bound variables, bound channels are subject to α-variance and capture-avoiding substitution, as usual [53], with the understanding that only the name y of a channel y_α can be renamed, but not its generation.

3 Recursive session logical relation (RSLR)

This section develops a logical relation for ILLST with general recursive types (§ 3.1) and shows that the logical relation is closed under parallel composition (§ 3.2), amounts to an equivalence relation (§ 3.3), and adequate, i.e., a weak asynchronous bisimilarity (§ 3.4). The logical relation is shown to have enough discriminatory power (soundness), while being maximally permissive (completeness) via a biorthogonality argument (Thm. 1).

$$
\begin{array}{lll}
\textsc{Spawn} & \mathbf{proc}(y_\alpha, (x \leftarrow X[\gamma] \leftarrow \Delta_1); Q) \;\mapsto & (\Omega_1' \vdash X = P :: x' : B \in \Sigma) \\
 & \mathbf{proc}(x_0, \hat{\gamma}([x_0/x]P))\ \mathbf{proc}(y_\alpha, [x_0/x]Q) & (\Omega_1' \Vdash \gamma : \Delta_1,\ x_0\, \mathit{fresh}) \\
\textsc{D-Fwd} & \mathbf{proc}(y_\alpha, F_Y[\gamma]) \;\mapsto & (x' : C \vdash F_Y = \mathtt{Fwd}_{C, y' \leftarrow x'} :: y' : C \in \Sigma) \\
 & \mathbf{proc}(y_\alpha, \mathtt{Fwd}_{C, y_\alpha \leftarrow x_\beta}) & (x', y' \Vdash \gamma : x_\beta, y_\alpha) \\
1_{\mathsf{snd}} & \mathbf{proc}(y_\alpha, \mathbf{close}\, y_\alpha) \;\mapsto\; \mathbf{msg}(\mathbf{close}\, y_\alpha) & \\
1_{\mathsf{rcv}} & \mathbf{msg}(\mathbf{close}\, y_\alpha)\ \mathbf{proc}(x_\beta, \mathbf{wait}\, y_\alpha; Q) \;\mapsto\; \mathbf{proc}(x_\beta, Q) & \\
\oplus_{\mathsf{snd}} & \mathbf{proc}(y_\alpha, y_\alpha.k; P) \;\mapsto\; \mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P)\ \mathbf{msg}(y_\alpha.k) & \\
\oplus_{\mathsf{rcv}} & \mathbf{msg}(y_\alpha.k)\ \mathbf{proc}(u_\eta, \mathbf{case}\, y_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L}) \;\mapsto\; \mathbf{proc}(u_\eta, [y_{\alpha+1}/y_\alpha]P_k) & \\
\&_{\mathsf{snd}} & \mathbf{proc}(y_\alpha, x_\beta.k; P) \;\mapsto\; \mathbf{msg}(x_\beta.k)\ \mathbf{proc}(y_\alpha, [x_{\beta+1}/x_\beta]P) & \\
\&_{\mathsf{rcv}} & \mathbf{proc}(y_\alpha, \mathbf{case}\, y_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L})\ \mathbf{msg}(y_\alpha.k) \;\mapsto\; \mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P_k) & \\
\otimes_{\mathsf{snd}} & \mathbf{proc}(y_\alpha, \mathbf{send}\, x_\beta\, y_\alpha; P) \;\mapsto\; \mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P)\ \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) & \\
\otimes_{\mathsf{rcv}} & \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)\ \mathbf{proc}(u_\eta, w \leftarrow \mathbf{recv}\, y_\alpha; P) \;\mapsto\; \mathbf{proc}(u_\eta, [x_\beta/w][y_{\alpha+1}/y_\alpha]P) & \\
\multimap_{\mathsf{snd}} & \mathbf{proc}(y_\alpha, \mathbf{send}\, x_\beta\, u_\eta; P) \;\mapsto\; \mathbf{msg}(\mathbf{send}\, x_\beta\, u_\eta)\ \mathbf{proc}(y_\alpha, [u_{\gamma+1}/u_\gamma]P) & \\
\multimap_{\mathsf{rcv}} & \mathbf{proc}(y_\alpha, w \leftarrow \mathbf{recv}\, y_\alpha; P)\ \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \;\mapsto\; \mathbf{proc}(y_{\alpha+1}, [x_\beta/w][y_{\alpha+1}/y_\alpha]P) &
\end{array}
$$

Fig. 4: Asynchronous dynamics of $\mathsf{SESSION}$.

3.1 Logical relation

Logical relations have been developed for session types [9, 15–17, 24, 46, 47, 50], both in unary and binary forms, but *without* considering general recursion. We contribute a logical relation for ILLST with support of *general recursive types*. In devising the resulting *recursive session logical relation (RSLR)*, we had to tackle the following challenges:

– *Well-foundedness:* Recursive types mandate use of a measure like step-indexing or later modalities [2,4,18] to keep the logical relation well-founded. Our logical relation makes use of a more *extensional* notion, an *observation index*, which bounds the number of messages exchanged with the outside world along the free channels of a configuration. The observation index facilitates proofs of various results shown in this section.
– *Non-termination:* In the presence of general recursive types, divergence is a possible outcome. Our logical relation is *progress-sensitive* [23] with regard to divergence; it equates a divergent program only with another diverging one.
– *Nondeterminism:* Although ILLST are confluent, processes run *concurrently*. As a result, the logical relation has to account for the relatedness of messages that may not be sent in the same order due to nondeterministic scheduling.

A logical relation for message-passing concurrency. It may be helpful to pause a moment and ask what a logical relation for session-typed processes should amount to. Logical relations for functional languages relate terms by defining their equality at their type when evaluated to values. This can be phrased in terms of two mutually recursive relations, a value interpretation $\mathcal{V}[\![\tau]\!]$ and a term interpretation $\mathcal{E}[\![\tau]\!]$, defined by structural induction on the type τ.

The former defines equality of values by inducting over a type, with values at ground type forming the base cases, and the latter steps the terms until they reach a value, demanding that these must be related by the value interpretation. In imperative languages, logical relations are enriched with Kripke possible worlds to model dynamically evolving shapes of storage [3, 19, 28, 45, 49, 57].

To answer our question, it is helpful to remind ourselves that logical relations are a means to prove observational equivalence [44], which equates two programs, if the same observations can be made about them. In a pure functional setting, the observables are the values to which the terms evaluate. In an imperative setting, the observables can additionally comprise the values of locations in the store. The observables in a message-passing concurrent setting are the messages that a configuration of processes exchanges with the outside world.

A logical relation for session-typed processes thus relates pairs of process configurations $(\mathcal{D}_1; \mathcal{D}_2)$, such that $\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}A$ and $\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}A$, demanding that the same messages can be observed along the free channels Δ and $x_\alpha{:}A$. Fig. 5 shows the resulting logical relation for well-typed process configurations in $\mathsf{SESSION}$, which is indexed with an *intuitionistic linear sequent* $\Delta \Vdash A$, comprising the free channels. We find it convenient to distinguish a value interpretation $\mathcal{V}$ from a term interpretation $\mathcal{E}$. As expected, the term interpretation steps the configurations until they reach a "value". But what does it mean for a configuration to be a value in a message-passing concurrent setting? It means that a configuration is *ready to send* or *receive* along any of the free channels in $\Delta \cup \{x_\alpha{:}A\}$. For any such channel, the term interpretation invokes the value interpretation, which demands that the exchanged messages be related, and then invokes the term interpretation for the configurations resulting after the message exchange. The mutually recursive relations are defined by clauses of the form

$$(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash A]\!] \quad \text{iff} \quad \ldots (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{E}[\![\Delta' \Vdash A']\!].$$

In a terminating setting, the relations are defined *multiset induction over the structure of types of the free channels* [15] such that $\Delta' \Vdash A' < \Delta \Vdash A$. To support general recursive types, we will augment the relations with an observation index, as discussed shortly, yielding an inductive definition.

The sequent $\Delta \Vdash A$ singles out the typing A of the providing free channel of the configurations $\mathcal{D}_1$ and $\mathcal{D}_2$. If the providing free channel is not observable, we write _. We use the metavariable K to stand for either $x_\alpha{:}A$ or simply _. The ability to mark some free channels as unobservable becomes important when we apply the logical relation to noninterference in § 4.

The transition from the value to the term interpretation amounts to an *observation* of a message exchange along a free channel in $\Delta \Vdash K$. The exchange will advance the protocol state of the concerned process, and as a result transition the sequent to $\Delta' \Vdash K'$, describing the post-state of the configurations $(\mathcal{D}_1'; \mathcal{D}_2')$ after the message exchange. The logical relation comprises two value interpretation clauses for each connective, one for the communications occurring on the *right* along K, and one for those occurring on the *left* along Δ.

(1) $(\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{V}\llbracket \cdot \Vdash y_\alpha{:}1 \rrbracket^{m+1}_{\cdot;y_\alpha}$	iff $\mathcal{D}_1 = \mathbf{msg}(\mathbf{close}\, y_\alpha)$ and $\mathcal{D}_2 = \mathbf{msg}(\mathbf{close}\, y_\alpha)$
(2) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta \Vdash y_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in I} \rrbracket^{m+1}_{\cdot;y_\alpha}$	iff $\exists k_1, k_2 \in I. k_1 = k_2$ and $\mathcal{D}_1 = \mathcal{D}_1'\mathbf{msg}(y_\alpha.k_1)$ and $\mathcal{D}_2 = \mathcal{D}_2'\mathbf{msg}(y_\alpha.k_2)$ and $(\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{E}\llbracket \Delta \Vdash y_{\alpha+1}{:}A_{k_1} \rrbracket^m$
(3) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta \Vdash y_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell \in I} \rrbracket^{m+1}_{y_\alpha;\cdot}$	iff $\forall k_1, k_2 \in I.$ if $k_1 = k_2$ then $(\mathcal{D}_1\mathbf{msg}(y_\alpha.k_1);\mathcal{D}_2\mathbf{msg}(y_\alpha.k_2)) \in \mathcal{E}\llbracket \Delta \Vdash y_{\alpha+1}{:}A_{k_1} \rrbracket^m$
(4) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta', \Delta'' \Vdash y_\alpha{:}A \otimes B \rrbracket^{m+1}_{\cdot;y_\alpha}$	iff $\exists x_\beta$ s.t. $\mathcal{D}_1 = \mathcal{D}_1'\mathcal{T}_1\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)$ and $\mathcal{D}_2 = \mathcal{D}_2'\mathcal{T}_2\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)$ and $(\mathcal{T}_1;\mathcal{T}_2) \in \mathcal{E}\llbracket \Delta'' \Vdash x_\beta{:}A \rrbracket^m$ and $(\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{E}\llbracket \Delta' \Vdash y_{\alpha+1}{:}B \rrbracket^m$
(5) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta \Vdash y_\alpha{:}A \multimap B \rrbracket^{m+1}_{y_\alpha;\cdot}$	iff $\forall x_\beta \notin dom(\Delta, y_\alpha{:}A \multimap B)$. $(\mathcal{D}_1\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha);\mathcal{D}_2\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)) \in$ $\mathcal{E}\llbracket \Delta, x_\beta{:}A \Vdash y_{\alpha+1}{:}B \rrbracket^m$
(6) $(\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{V}\llbracket \Delta, y_\alpha{:}1 \Vdash K \rrbracket^{m+1}_{y_\alpha;\cdot}$	iff $(\mathbf{msg}(\mathbf{close}\, y_\alpha)\mathcal{D}_1;\mathbf{msg}(\mathbf{close}\, y_\alpha)\mathcal{D}_2) \in \mathcal{E}\llbracket \Delta \Vdash K \rrbracket^m$
(7) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta, y_\alpha : \oplus\{\ell{:}A_\ell\}_{\ell \in I} \Vdash K \rrbracket^{m+1}_{y_\alpha;\cdot}$	iff $\forall k_1, k_2 \in I.$if $k_1 = k_2$ then $(\mathbf{msg}(y_\alpha.k_1)\mathcal{D}_1;\mathbf{msg}(y_\alpha.k_2)\mathcal{D}_2) \in \mathcal{E}\llbracket \Delta, y_{\alpha+1}{:}A_{k_1} \Vdash K \rrbracket^m$
(8) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta, y_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell \in I} \Vdash K \rrbracket^{m+1}_{\cdot;y_\alpha}$	iff $\exists k_1, k_2 \in I. k_1 = k_2$ and $\mathcal{D}_1 = \mathbf{msg}(y_\alpha.k_1)\mathcal{D}_1'$ and $\mathcal{D}_2 = \mathbf{msg}(y_\alpha.k_2)\mathcal{D}_2'$ and $(\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{E}\llbracket \Delta, y_{\alpha+1}{:}A_{k_1} \Vdash K \rrbracket^m$
(9) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta, y_\alpha{:}A \otimes B \Vdash K \rrbracket^{m+1}_{y_\alpha;\cdot}$	iff $\forall x_\beta \notin dom(\Delta, y_\alpha{:}A \otimes B, K)$. $(\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\mathcal{D}_1;\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\mathcal{D}_2) \in$ $\mathcal{E}\llbracket \Delta, x_\beta{:}A, y_{\alpha+1}{:}B \Vdash K \rrbracket^m$
(10) $(\mathcal{D}_1;\mathcal{D}_2) \in$ $\mathcal{V}\llbracket \Delta', \Delta'', y_\alpha{:}A \multimap B \Vdash K \rrbracket^{m+1}_{\cdot;y_\alpha}$	iff $\exists x_\beta$ s.t. $\mathcal{D}_1 = \mathcal{T}_1\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\, \mathcal{D}_1''$ and $\mathcal{D}_2 = \mathcal{T}_2\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\, \mathcal{D}_2''$ and $(\mathcal{T}_1;\mathcal{T}_2) \in \mathcal{E}\llbracket \Delta' \Vdash x_\beta{:}A \rrbracket^m$ and $(\mathcal{D}_1'';\mathcal{D}_2'') \in \mathcal{E}\llbracket \Delta'', y_{\alpha+1}{:}B \Vdash K \rrbracket^m$
(11) $(\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}\llbracket \Delta \Vdash K \rrbracket^{m+1}$	iff $(\mathcal{D}_1;\mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $\forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.$ if $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$ then $\exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$, and $\Upsilon_1 \subseteq \Upsilon_2$ and $\forall\, y_\alpha \in \mathbf{Out}(\Delta \Vdash K).$ if $y_\alpha \in \Upsilon_1.$ then $(\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}\llbracket \Delta \Vdash K \rrbracket^{m+1}_{\cdot;y_\alpha}$ and $\forall\, y_\alpha \in \mathbf{In}(\Delta \Vdash K).$if $y_\alpha \in \Theta_1.$ then $(\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}\llbracket \Delta \Vdash K \rrbracket^{m+1}_{y_\alpha;\cdot}$
(12) $(\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}\llbracket \Delta \Vdash K \rrbracket^0$	iff $(\mathcal{D}_1;\mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$

Fig. 5: Recursive session logical relation (RSLR) for ILLST. (Clauses (1)-(10) are only defined for well-typed configurations, a condition we elided for concision.)

Assume and assert—a pas de deux. A key characteristic of logical relations is *extensionality*. In a functional context this means that relatedness has to be shown for values of types in positive positions, but values of types in negative positions can be assumed to be related. This idea translates equally to a message-passing concurrent setting, embodied by the logical relation shown in Fig. 5 as:

– *Positive* types: *assert* sending of related messages when communicating on the *right*; *assume* receipt of related messages when communicating on the *left*.
– *Negative* types: *assume* receipt of related messages when communicating on the *right*; *assert* sending of related messages when communicating on the *left*.

At base types (1, $\oplus$, and $\&$), relatedness of messages means that the same messages are being exchanged. For example, clause (2) in Fig. 5 for $\oplus$-right asserts existence of messages $\mathbf{msg}(y_\alpha.k_1)$ and $\mathbf{msg}(y_\alpha.k_2)$ in $\mathcal{D}_1$ and $\mathcal{D}_2$, resp., such that $k_1 = k_2$. Conversely, clause (3) in Fig. 5 for $\&$-right assumes receipt of messages $\mathbf{msg}(y_\alpha.k_1)$ and $\mathbf{msg}(y_\alpha.k_2)$ to be added to $\mathcal{D}_1$ and $\mathcal{D}_2$ in the post-states, resp., such that $k_1 = k_2$, for arbitrary k_1 and k_2.

For higher-order types ($\otimes$ and $\multimap$) relatedness means that future observations to be made along the exchanged channels must be related. For example, clause (10) in Fig. 5 for $\multimap$-left asserts existence of a message $\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)$ and of subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ in $\mathcal{D}_1$ and $\mathcal{D}_2$, resp. The subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ are rooted at the sent channel x_β and will be transferred (and thus lost) together with the sent channel. The clause comprises two invocations of the term relation, $(\mathcal{T}_1;\mathcal{T}_2) \in \mathcal{E}[\![\Delta' \Vdash x_\beta{:}A]\!]$, asserting that future observations to be made along the sent channel x_β are related, and $(\mathcal{D}_1'';\mathcal{D}_2'') \in \mathcal{E}[\![\Delta'', y_{\alpha+1}{:}B \Vdash K]\!]$, asserting that the continuations $\mathcal{D}_1''$ and $\mathcal{D}_2''$ are related. [3] The channel endpoint x_β is existentially quantified and occurs either bound or free in both $\mathcal{D}_1$ and $\mathcal{D}_2$. Conversely, clause (5) in Fig. 5 for $\multimap$-right assumes receipt of a message $\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)$ that carries a universally quantified free channel x_β, different from the free channels available in the pre-state ($\forall x_\beta \notin \mathit{dom}(\Delta, y_\alpha{:}A \multimap B)$). The sequent is enlarged with the received channel x_β for the invocation of the term relation $(\mathcal{D}_1 \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha); \mathcal{D}_2 \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)) \in \mathcal{E}[\![\Delta, x_\beta{:}A \Vdash y_{\alpha+1}{:}B]\!]$.

The *duality* of ILLST, established by "cutting" the left and right rule of a connective, translates equally to our logical relation, allowing us to compose related configurations in parallel, such that the assumptions made by one pair of configurations are discharged by the other. We prove this result in § 3.2 and exploit it connect our development to biorthogonality in § 3.5.

Syntactic well-typedness. The value and term interpretations of our logical relation relate pairs of process configurations only if both are syntactically well-typed. The predicate $(\mathcal{D}_1;\mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ present in the term interpretation (clauses (11)-(12) in Fig. 5) stipulates this requirement and is defined as $\mathcal{D}_1 \in \mathsf{Tree}(\Delta \Vdash K)$ and $\mathcal{D}_2 \in \mathsf{Tree}(\Delta \Vdash K)$, with $\mathcal{D}_i \in \mathsf{Tree}(\Delta \Vdash K)$ defined as $\Delta \Vdash \mathcal{D}_i :: K$ (see Def. 11 in the Appendix). For concision, we elide appeals to the $\mathsf{Tree}()$ predicate in the value interpretation (clauses (1)-(10) in Fig. 5). Syntactic typing ensures that configurations form *rooted trees*, a property our proof of semantic cut (Lem. 2 in § 3.2) relies upon.

Well-foundedness. In the presence of general recursive types, it is no longer sound to define the logical relation by multiset induction over the structure of

[3] We will momentarily explain the superscript and subscripts of these invocations.

types in its sequent $\Delta \Vdash K$. To restore well-foundedness a measure like step-indexing [2, 4] can be employed. Step-indexing is typically tied to the number of type unfolding or computation steps by which one of the two programs is bound. A more natural and symmetric measure for our setting is the number of *observations* that can be made along the free channels $\Delta \Vdash K$ of the logical relation. We thus bound the value and term interpretation of our logical relation by this number and attach the resulting *observation index* as the superscript m to the sequent $\Delta \Vdash K$, yielding clauses of the form

$$(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1} \ \text{ iff } \ \ldots (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{E}[\![\Delta' \Vdash K']\!]^{m}$$

The observation index gets decremented whenever the value interpretation "observes" a message exchange. If $m = 0$, well-typed configurations are trivially related, as expressed by clause (12) in Fig. 5.

Non-termination and nondeterminism. It is now time to take a closer look at the definition of the term interpretation of our logical relation. Its definition is challenged by the possibility of divergence and nondeterminism of scheduling. For the former, a progress-sensitive statement of equivalence demands that the relation relates a divergent program only with another diverging one. For the latter, the term interpretation has to account for the possibility that two configurations may not *simultaneously* be ready to send or receive along a free channel in $\Delta \Vdash K$. To address these challenges, the term interpretation is phrased as follows (we are repeating clause (11) in Fig. 5):

$$\begin{array}{l}
(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1} \ \text{ iff} \\
(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K) \ \text{and} \ \forall\, \Upsilon_1, \Theta_1, \mathcal{D}_1'. \text{if } \mathcal{D}_1 \ \mapsto^{*\Upsilon_1;\Theta_1} \ \mathcal{D}_1' \ \text{then} \\
\exists \Upsilon_2, \mathcal{D}_2' \ \text{such that } \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2' \ \text{and } \Upsilon_1 \subseteq \Upsilon_2 \ \text{and} \\
\forall\, y_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ y_\alpha \in \Upsilon_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot; y_\alpha} \\
\text{and} \ \forall\, y_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\ y_\alpha \in \Theta_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{y_\alpha; \cdot}
\end{array}$$

The term interpretation uses the transition $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, amounting to the iterated application of the rewriting rules defined in Fig. 4. The star expresses that zero to multiple steps can be taken. The superscript Υ_1 denotes the set of channels in $\Delta \Vdash K$ for which there exist messages in $\mathcal{D}_1'$ to be sent along these channels. The superscript Θ_1 denotes set of channels in $\Delta \Vdash K$ for which there exist processes in $\mathcal{D}_1'$ waiting to receive along these channels.

To ensure progress-sensitivity, the term interpretation asserts, that, whenever $\mathcal{D}_1$ can step, so can $\mathcal{D}_2$, and that the messages ready to be sent to the outside world in $\mathcal{D}_2'$, are at least the ones ready to be sent in $\mathcal{D}_1'$, i.e., $\Upsilon_1 \subseteq \Upsilon_2$. Due to the dynamics being asynchronous, a receipt is not an observable event. As such, the set of processes in $\mathcal{D}_1'$ and $\mathcal{D}_2'$ that are waiting to receive along a free channel are not required to correspond, and thus, we do not consider the set Θ_2 for $\mathcal{D}_2'$. The existentially quantified $\mathcal{D}_2'$ is justified by nondeterminism of scheduling, allowing the term interpretation to choose an appropriate way of stepping $\mathcal{D}_2$ to catch up with $\mathcal{D}_1'$.

The set $\mathbf{Out}(\Delta \Vdash K)$ includes all channels whose types in Δ, K indicate a send. This includes channels in Υ_1 for which the messages are ready to be sent. Similarly, the set $\mathbf{In}(\Delta \Vdash K)$ includes all channels whose types in Δ, K indicate a receive, including those in Θ_1 for which the processes are ready to receive.

To resolve the issue of simultaneity, the term interpretation makes use of two *focus* channels ranging over the sets Υ_1 and Θ_1. These are added as a subscript $\_;\_$ to the value interpretation, where $\cdot; y_\alpha$ indicates that $y_\alpha \in \Upsilon_1$ and $y_\alpha; \cdot$ that $y_\alpha \in \Theta_1$. The term interpretation invokes the value interpretation for all the channels in Υ_1 and Θ_1, thus ensuring that any messages ready to be sent in $\mathcal{D}_1'$ and $\mathcal{D}_2'$ and any processes waiting to receive in $\mathcal{D}_1'$ will be "observed" by the value interpretation. Non-productive configurations are trivially accommodated by the term interpretation, by allowing the sets Υ_1 and Θ_1 to be empty.

A reader may wonder why the logical relation seems oblivious of recursive type unfoldings. Since the relation is indexed by the number of external observations, it is invariant in type unfoldings. For example, for the pair of configurations $(\mathcal{D}_1; \mathcal{D}_2)$ it holds true that $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash x_\alpha{:}\mathsf{pin}]\!]^{m+1}$ as well as $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash x_\alpha{:}\oplus\{tok_i{:}\mathsf{pin}\}_{i\leq n}]\!]^{m+1}$, because the value interpretation with focus channel x_α is only triggered when a message along x_α (i.e., token) is ready. Any subsequent invocation of the term interpretation by the value interpretation upon consumption of the message will then happen at index m.

3.2 Semantic cut: closure under parallel composition

The use of an observation index rather than a step index by the RSLR is fundamental in proving various metatheoretic properties in the remainder of this section. It allows us to keep the observation index *constant* in the term interpretation, while stepping the configurations internally, permitting disagreement in the number of internal messages exchanged. As a result, internal steps cannot depend on the observation index, facilitating proof of the following lemma:

Lemma 1 (Moving existential over universal quantifier).
If

$$\begin{array}{l}
\forall m.\, \forall\, \Upsilon_1, \Theta_1, \mathcal{D}_1'.\ \text{if}\ \mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1',\ \text{then} \\
\exists \Upsilon_2, \mathcal{D}_2'\ \text{such that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \text{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \text{and} \\
\ \forall\, x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then} \\
\qquad\qquad (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}\ \text{and} \\
\ \forall\, x_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\ x_\alpha \in \Theta_1.\, \mathbf{then} \\
\qquad\qquad (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot},
\end{array}$$

then

$$\begin{array}{l}
\forall\, \Upsilon_1, \Theta_1, \mathcal{D}_1'.\text{if}\ \mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1',\ \text{then} \\
\exists \Upsilon_2, \mathcal{D}_2'\ \text{such that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \text{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \text{and} \\
\ \forall\, x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then} \\
\qquad\qquad \forall\, m.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}\ \text{and} \\
\ \forall\, x_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\ x_\alpha \in \Theta_1.\, \mathbf{then} \\
\qquad\qquad \forall m.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.
\end{array}$$

Proof. See proof of Lem. 26 in the Appendix. The proof relies on confluence (Lem. 21 in the Appendix) and backwards closure (Lem. 23 in the Appendix).

With this lemma in hand, we can now prove that the logical relation is closed under parallel composition.

Lemma 2 (Semantic cut). *$\forall m. (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta, u_\alpha{:}T \Vdash K]\!]^m$ iff for all $\mathcal{T}_1$ and $\mathcal{T}_2$ if $\forall m. (\mathcal{T}_1; \mathcal{T}_2) \in \mathcal{E}[\![\Delta' \Vdash u_\alpha{:}T]\!]^m$ then $\forall k. (\mathcal{T}_1\mathcal{D}_1; \mathcal{T}_2\mathcal{D}_2) \in \mathcal{E}[\![\Delta', \Delta \Vdash K]\!]^k$.*

Proof. See proof of Corollary 2 in the Appendix. The proof relies on Lem. 1.

The semantic cut lemma is analogous to the syntactic cut-elimination theorem, with T serving as the analog of the cut formula. Consider the left-to-right direction. For internal steps in $\mathcal{T}_1\mathcal{D}_1$ invoked by the term interpretation that do not involve the channel $u_\alpha : T$, we can perform corresponding internal steps in $\mathcal{T}_1$ or $\mathcal{D}_1$. Using the fact that $\mathcal{T}_1$ and $\mathcal{D}_1$ are related to $\mathcal{T}_2$ and $\mathcal{D}_2$, resp., we can construct the next step(s) of $\mathcal{T}_2\mathcal{D}_2$ as required by the term interpretation. For internal steps in $\mathcal{T}_1\mathcal{D}_1$ along the channel $u_\alpha : T$, we use the fact that $\mathcal{T}_1$ is ready to send a message along u_α and $\mathcal{D}_1$ is ready to receive along u_α (or vice versa). Here, we take advantage of the duality in the value interpretation for the provider and client of each connective to show that the assertions made by the sender are sufficient to establish the assumptions of the receiver. With this, we invoke the definition of value interpretations to transition to a new term interpretation involving subformula(s) of type T in the sequent and apply the inductive hypothesis. This step is analogous to internal cut reductions in the cut-elimination proof. For any communication in $\mathcal{T}_1\mathcal{D}_1$ along its external channel, we use the fact that either $\mathcal{T}_1$ or $\mathcal{D}_1$ can perform the same communication. Using their respective value interpretations and the inductive hypothesis, we establish the value interpretation for $\mathcal{T}_1\mathcal{D}_1$ and $\mathcal{T}_2\mathcal{D}_2$. This step corresponds to external cut reductions in the cut-elimination proof.

3.3 Logical equivalence

To prove that the RSLR induces an equivalence relation, we introduce the notation $\Delta \Vdash \mathcal{D}_1 :: K \equiv \Delta \Vdash \mathcal{D}_2 :: K$, defined below, where $(\mathcal{C}_1; \mathcal{C}_2) \in \mathsf{Forest}(\Delta' \Vdash \Delta)$ stands for $\Delta' \Vdash \mathcal{C}_1 :: \Delta$ and $\Delta' \Vdash \mathcal{C}_2 :: \Delta$:

Definition 1 (Logical equivalence).
- *We define the relation $\Delta \Vdash \mathcal{D}_1 :: K \equiv \Delta \Vdash \mathcal{D}_2 :: K$ as $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $\forall m. (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m. (\mathcal{D}_2; \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$.*
- *We define the relation $\Delta \dashv \mathcal{C}_1[\,]\mathcal{F}_1 \dashv K \equiv \Delta \dashv \mathcal{C}_2[\,]\mathcal{F}_2 \dashv K$ as (i) $\exists K'$ such that $K \Vdash \mathcal{F}_1 :: K' \equiv K \Vdash \mathcal{F}_2 :: K'$ and (ii) $\exists \Delta'$ such that $(\mathcal{C}_1; \mathcal{C}_2) \in \mathsf{Forest}(\Delta' \Vdash \Delta)$ and $\forall \mathcal{T}_1 \in \mathcal{C}_1$, and $\forall \mathcal{T}_2 \in \mathcal{C}_2$ with $(\mathcal{T}_1; \mathcal{T}_2) \in \mathsf{Tree}(\Delta'_1 \Vdash K'')$, we have $\Delta'_1 \Vdash \mathcal{T}_1 :: K'' \equiv \Delta'_1 \Vdash \mathcal{T}_2 :: K''$.*

Next, we show that the relation $\equiv$ is reflexive, symmetric, and transitive.

Lemma 3 (Reflexivity). *For all configurations $\Delta \Vdash \mathcal{D} :: x_\alpha{:}T$, we have $(\Delta \Vdash \mathcal{D} :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D} :: x_\alpha{:}T)$.*

Proof. See proof of Lem. 28 in the Appendix.

Lemma 4 (Symmetry). *For all configurations $\mathcal{D}_1$ and $\mathcal{D}_2$, we have $(\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T)$, iff $(\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T)$,*

Proof. The proof is straightforward by the definition of logical equivalence (Def. 1).

Lemma 5 (Transitivity). *For all configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, we have*

$$\begin{array}{l}\mathsf{if}\ (\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T),\ \mathsf{and}\\ \quad(\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_3 :: x_\alpha{:}T)\\ \mathsf{then}\ (\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_3 :: x_\alpha{:}T).\end{array}$$

Proof. See proof of Lem. 30 in the Appendix.

3.4 Adequacy

To show that the RSLR is adequate, we prove that configurations related by the logical relation are bisimilar and vice versa. To facilitate this proof, we first give the definition of weak *asynchronous bisimilarity* [53] (Def. 2). The definition relies on a standard labeled transition system displayed in Fig. 6, which rephrases our dynamics in Fig. 4. We define *weak* transitions as

(1) $\Rightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$,

(2) $\xRightarrow{\alpha}$ is $\Rightarrow\xrightarrow{\alpha}$.

Definition 2 (Weak asynchronous bisimilarity). *Asynchronous bisimilarity, written $\mathcal{D}_1 \approx_{\mathsf{a}} \mathcal{D}_2$, is the largest symmetric relation such that whenever $\mathcal{D}_1 \approx_{\mathsf{a}} \mathcal{D}_2$, we have*

–($\tau - \mathtt{step}$) *if* $\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_{\mathsf{a}} \mathcal{D}_2'$,

–(output) *if* $\mathcal{D}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_{\mathsf{a}} \mathcal{D}_2'$.

–(left input) *for all* $q \notin \mathsf{fn}(\mathcal{D}_1)$, *if* $\mathcal{D}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_{\mathsf{a}} \mathbf{msg}(x_\alpha.q)\mathcal{D}_2'$,

–(right input) *for all* $q \notin \mathsf{fn}(\mathcal{D}_1)$, *if* $\mathcal{D}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_{\mathsf{a}} \mathcal{D}_2'\mathbf{msg}(x_\alpha.q)$.

where $\mathbf{msg}(x_\alpha.q)$ *is defined as* $\mathbf{msg}(\mathbf{close}\, x_\alpha)$ *if* $q = \mathbf{close}$, $\mathbf{msg}(x_\alpha.k)$ *if* $q = k$, *and* $\mathbf{msg}(\mathbf{send}\, z_\delta\, x_\alpha)$ *if* $q = z_\delta$. *Here,* $\mathsf{fn}(\mathcal{D}_1)$ *is the set of free channels in the configuration* $\mathcal{D}_1$*.*

Our adequacy theorem (Lem. 6) shows that RSLRs are sound and complete with regard to asynchronous bisimilarity.

Lemma 6 (Adequacy). *For all* $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$, *we have* $\forall m.(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ *and* $\forall m.(\mathcal{D}_2; \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ *iff* $\mathcal{D}_1 \approx_{\mathsf{a}} \mathcal{D}_2$.

Proof. The result is a corollary of Lem. 34 in the Appendix.

Internal transition $\xrightarrow{\tau}$ *defined as:*

$\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ iff $\mathcal{D}_1 \mapsto \mathcal{D}_1'$

Actions $\xrightarrow{\overline{y_\alpha}\, q}$, $\xrightarrow{\mathbf{L}\, y_\alpha\, q}$ *and* $\xrightarrow{\mathbf{R}\, y_\alpha\, q}$ *defined as below when* $y \in \mathbf{fn}(\mathcal{D}_1)$*:*

$$
\begin{array}{lll}
(1)\, \mathcal{D}_1 \mathbf{msg}(\mathbf{close}\, y_\alpha)\, \mathcal{D}_2 & \xrightarrow{\overline{y_\alpha}\, \mathbf{close}} & \mathcal{D}_1 \mathcal{D}_2 \\
(2) \mathcal{D}_1\, \mathbf{msg}(y_\alpha.k) \mathcal{D}_2 & \xrightarrow{\overline{y_\alpha}\, k} & \mathcal{D}_1 \mathcal{D}_2 \\
(3) \mathcal{D}_1\, \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \mathcal{D}_2 & \xrightarrow{\overline{y_\alpha}\, x_\beta} & \mathcal{D}_1 \mathcal{D}_2 \\
(4)\, \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{wait}\, y_\alpha; P) \mathcal{D}_2 & \xrightarrow{\mathbf{L}\, y_\alpha\, \mathbf{close}} & \mathbf{msg}(\mathbf{close}\, y_\alpha)\, \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{wait}\, y_\alpha; P)\, \mathcal{D}_2 \\
(5)\, \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I})\, \mathcal{D}_2 & \xrightarrow{\mathbf{L}\, y_\alpha\, k} & \mathbf{msg}(y_\alpha.k) \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I}) \mathcal{D}_2 \\
(6) \mathcal{D}_1\, \mathbf{proc}(z_\delta, w \leftarrow \mathbf{recv}\, y_\alpha) \mathcal{D}_2 & \xrightarrow{\mathbf{L}\, y_\alpha\, x_\beta} & \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \mathcal{D}_1\, \mathbf{proc}(z_\delta, w \leftarrow \mathbf{recv}\, y_\alpha)\, \mathcal{D}_2 \\
(7) \mathcal{D}_1\, \mathbf{proc}(y_\alpha, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I}) \mathcal{D}_2 & \xrightarrow{\mathbf{R}\, y_\alpha\, k} & \mathcal{D}_1\, \mathbf{proc}(y_\alpha, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I})\, \mathcal{D}_2 \mathbf{msg}(y_\alpha.k) \\
(8) \mathcal{D}_1 \mathbf{proc}(y_\alpha, w \leftarrow \mathbf{recv}\, y_\alpha)\, \mathcal{D}_2 & \xrightarrow{\mathbf{R}\, y_\alpha\, x_\beta} & \mathcal{D}_1 \mathbf{proc}(y_\alpha, w \leftarrow \mathbf{recv}\, y_\alpha) \mathcal{D}_2\, \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)
\end{array}
$$

Fig. 6: Labeled transition system for SESSION.

3.5 Biorthogonality

To define our main result, that the relation $\equiv$ is sound and complete with regard to closure of weak bisimilarity under parallel composition (Thm. 1), we define an *orthogonality* operation $\top$ [7,40,48], leading to Def. 5. We follow the development of Pitts [48].

We first define the program relations $PRel(\Delta \Vdash K)$ and environment relations $ERel(\Delta \Vdash K)$ (Def. 4), reliant upon a session-typed environment (Def. 3):

Definition 3 (Session-typed environment). *A session-typed environment with the sequent* $\Delta \Vdash K$*, is of the form* $\mathcal{C}[\ ]\mathcal{F}$*, such that for some* Δ' *and* K'*, we have* $\Delta' \Vdash \mathcal{C} :: \Delta$ *and* $K \Vdash \mathcal{F} :: K'$*.*

Definition 4 (Program- and environment- relations). *A session program-relation is a binary relation between session-typed programs, i.e., open configurations of the form* $\Delta \Vdash \mathcal{D} :: K$*. Given the sequent* $\Delta \Vdash K$*, we write* $PRel(\Delta \Vdash K)$ *for the set of all program relations that relate programs* $\Delta \Vdash \mathcal{D} :: K$*.*

A session environment-relation is a binary relation between session-typed environments. Given the sequent $\Delta \Vdash K$*, we write* $ERel(\Delta \Vdash K)$ *for the set of all environment relations that relate environments* $\mathcal{C}[\ ]\mathcal{F}$ *with the sequent* $\Delta \Vdash K$*.*

We can now define the orthogonality operation $\top$ [40, 48].

Definition 5 (The $(\_)^\top$ operation). *Given sequent $\Delta \Vdash K$ and $r \in PRel(\Delta \Vdash K)$, we define $r^\top \in ERel(\Delta \Vdash K)$ by*

$$(\mathcal{C}_1[\ ]\mathcal{F}_1, \mathcal{C}_2[\ ]\mathcal{F}_2) \in r^\top \text{ iff}$$
$$\forall(\mathcal{D}_1, \mathcal{D}_2) \in r.(\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_{\mathsf{a}} \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2),$$

and given $s \in ERel(\Delta \Vdash K)$, we define $s^\top \in PRel(\Delta \Vdash K)$ by

$$(\mathcal{D}_1, \mathcal{D}_2) \in s^\top \text{ iff}$$
$$\forall(\mathcal{C}_1[\ ]\mathcal{F}_1, \mathcal{C}_2[\ ]\mathcal{F}_2) \in s.(\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_{\mathsf{a}} \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2).$$

By definition, the $\top$ operation is inflationary and idempotent [48].

Finally, we state our main result that the logical equivalence relation $\equiv$ induced by our RSLR (Def. 1) is sound and complete with regard to closure of weak bisimilarity under parallel composition.

Theorem 1 ($\top\top$-closure). *Consider $(\mathcal{D}_1, \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$, we have*

$$(\Delta \Vdash \mathcal{D}_1 :: K) \equiv (\Delta \Vdash \mathcal{D}_2 :: K) \textbf{ iff}$$
$$\forall \mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2.\ \mathsf{if}\ (\Delta \dashv \mathcal{C}_1[\ ]\mathcal{F}_1 \dashv K) \equiv (\Delta \dashv \mathcal{C}_2[\ ]\mathcal{F}_2 \dashv K)$$
$$\mathsf{then}\ \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_{\mathsf{a}} \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2.$$

Proof. See proof of Thm. 8 in the Appendix.

The main purpose of Thm. 1 is to validate the logical equivalence induced by our logical relation, ensuring that it has enough discriminatory power (soundness) while being maximally permissive (completeness). To do that, Thm. 1 uses biorthogonal closure. Specifically, it instantiates the program-relation r and environment-relation s in Def. 5 with our logical relation, and proves that related programs cannot be discriminated by related environments ($r \subseteq s^\top$) and that programs that cannot be discriminated by related environments are related ($s^\top \subseteq r$). As detailed in [48], showing that $r = s^\top$ is enough to get $r = r^{\top\top}$ by virtue of $\top$ being a Galois connection. Thm. 1 thus establishes that our logical relation is compositional and extensional (behavioral). It shows that our logical relation includes exactly the programs whose compositions with related environments are observationally equivalent. Soundness ensures that related environments are not very strong and do not have too much discriminatory power with respect to the observable behavior – related environments cannot distinguish between the related programs. Completeness guarantees that related environments are not very weak and have enough discriminatory power – if related environments cannot distinguish between two programs, the programs are indeed related. This aligns with the assert-assume pas de deux described in § 3.1, ensuring that the assumptions made by related programs are exactly those asserted by the related environments and the assertions of the programs are exactly those assumed by the environments.

In the next section, we establish noninterference for session-typed processes as one practical application of our logical relation and its soundness and completeness, a.k.a., $\top\top - closure$.

4 Noninterference

This section applies the RSLR to noninterference, phrased as an equivalence up to the secrecy level ξ of an observer, $\equiv_{\xi}^{\Psi_0}$, given a security lattice Ψ_0 (§ 4.2). The section then illustrates the up-to equivalence $\equiv_{\xi}^{\Psi_0}$ on an example (§ 4.3) and concludes with its metatheoretic properties (§ 4.4).

4.1 Attacker model

Our attacker model is parametric in the secrecy level $\xi \in \Psi_0$ of an attacker and a program P, given prior annotation of the free channels of P with secrecy levels $c \in \Psi_0$. It assumes

- that the attacker knows the source code of P;
- that the attacker can only observe the messages sent along the free channels of P with secrecy level $c \sqsubseteq \xi$;
- that an attacker cannot measure the passing of time;
- that an attacker is oblivious of channel names;
- a nondeterministic scheduler.

4.2 Up-to equivalence

To prove noninterference, we introduce an *equivalence relation up to* the secrecy level ξ of an observer, $\equiv_{\xi}^{\Psi_0}$, defined in Def. 8 below. This relation is reminiscent of logical equivalence $\equiv$ (Def. 1), but expects free channels to be annotated with secrecy levels and allows ignoring those channels with secrecy level $\sqsubseteq \xi$.

We first define two auxiliary notions: *(i)* downward projections on context of free channels $\Gamma \Downarrow \xi$ and on the providing free channel $x_\alpha{:}T[c] \Downarrow \xi$ (Def. 6), as well as *(ii)* the predicates **H-Provider**$^{\xi}$ and **H-Client**$^{\xi}$ (Def. 7). Since channels are now annotated with secrecy levels, we use the metavariables Γ and K^s to range over the linear typing context and providing channel, resp.

Definition 6 (Typing context projections). *Downward projection on security linear contexts Γ and providing channels K^s is defined as follows:*

$$\begin{array}{lll}
\Gamma, x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} \Gamma \Downarrow \xi, x_\alpha{:}T[c] & \text{if } c \sqsubseteq \xi \\
\Gamma, x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} \Gamma \Downarrow \xi & \text{if } c \not\sqsubseteq \xi \\
\cdot \Downarrow \xi & \stackrel{\mathsf{def}}{=} \cdot & \\
x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} x_\alpha{:}T[c] & \text{if } c \sqsubseteq \xi \\
x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} _{:}1[\top] & \text{if } c \not\sqsubseteq \xi
\end{array}$$

The projections are used to keep only the free channels that are observable to an attacker. We close the nonobservable channels off by composing the configuration with high-confidentiality clients (**H-Client**$^{\xi}$) and providers (**H-Provider**$^{\xi}$), which may differ for each configuration. These high-confidentiality clients and providers are connected to the configurations through nonobservable channels, making the messages they exchange with the configurations nonobservable to the attacker as well.

Definition 7 (High provider and High client).

$$
\begin{array}{l}
\cdot \;\in \mathbf{H\text{-}Provider}^{\xi}(\cdot) \\[1ex]
\mathcal{B} \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma, x_\alpha{:}A[c]) \qquad \text{iff} \\
\quad \textbf{either}\; c \not\sqsubseteq \xi \text{ and } \mathcal{B} = \mathcal{B}'\mathcal{T} \text{ and } \mathcal{B}' \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma) \text{ and} \\
\qquad\qquad\qquad\qquad \mathcal{T} \in \mathsf{Tree}(\cdot \Vdash x_\alpha{:}A), \\
\quad \textbf{or}\; c \sqsubseteq \xi \text{ and } \mathcal{B} \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma) \\[2ex]
\mathcal{T} \in \mathbf{H\text{-}Client}^{\xi}(x_\alpha{:}A[c]) \qquad \text{iff} \\
\quad \textbf{either}\; c \not\sqsubseteq \xi \text{ and } \mathcal{T} \in \mathsf{Tree}(x_\alpha{:}A \Vdash \_ : 1), \textbf{or} \\
\quad \textbf{or}\; c \sqsubseteq \xi \text{ and } \mathcal{B} = \cdot
\end{array}
$$

Finally, we can state the relation $\equiv_\xi^{\Psi_0}$. Like $\equiv$ (Def. 1), $\equiv_\xi^{\Psi_0}$ is phrased in terms of the term interpretation of the RSLR, but restricts observations to those free channels that are observable. The remaining free channels will be composed in parallel with arbitrary ILLST-typed configurations.

Definition 8 (Logical equivalence up to observer level). *We define the relation* $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv_\xi^{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ *as*

$$
\begin{array}{l}
\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1) \text{ and } \mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2) \text{ and} \\
\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi = \Gamma \text{ and } x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi = K^s \text{ and} \\
\forall\, \mathcal{B}_1 \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma_1).\, \forall\, \mathcal{B}_2 \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma_2). \\
\forall \mathcal{T}_1 \in \mathbf{H\text{-}Client}^{\xi}(x_\alpha{:}A_1[c_1]).\, \forall \mathcal{T}_2 \in \mathbf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]). \\
\forall\, m.\, (\mathcal{B}_1\mathcal{D}_1\mathcal{T}_1, \mathcal{B}_2\mathcal{D}_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m, \text{ and} \\
\forall\, m.\, (\mathcal{B}_2\mathcal{D}_2\mathcal{T}_2, \mathcal{B}_1\mathcal{D}_1\mathcal{T}_1) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m.
\end{array}
$$

Def. 8 makes use of an erasure operation $|\_|$ to drop the secrecy annotations, to match the sequent of the logical relation whose free channels lack secrecy annotations. As a result, $\equiv_\xi^{\Psi_0}$ can be used for configurations that type check using an information flow control (IFC) type system as well as "plain-vanilla" ILLST-typed configurations, facilitating *semantic typing* [13,39,58] for progress-sensitive noninterference (PSNI).

4.3 Case study: progress-sensitive noninterference (PSNI)

To illustrate $\equiv_\xi^{\Psi_0}$, we instantiate it on a simplistic example, in the spirit of an insecure version of the Verifier encountered earlier (Fig. 2), shown below. Assuming that our security lattice (Ψ_0) is **guest** $\sqsubseteq$ **alice**, and that the attacker level is **guest**, the below process leaks to an attacker y whether authentication was successful or not by either sending the label s or f, resp.

$$
\mathsf{pin} = \&\{tok_1{:}1, tok_2{:}1\} \qquad \mathsf{bit} = \&\{zero{:}1, one{:}1\}
$$

$$
\begin{array}{l}
y{:}\mathsf{bit}[\mathbf{guest}] \vdash \mathsf{X} :: x{:}\mathsf{pin}[\mathbf{alice}] = (\mathbf{case}\, x\, (tok_1 \Rightarrow y.zero; \mathbf{wait}\, y;\, \mathbf{close}\, x \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mid tok_2 \Rightarrow y.one; \mathbf{wait}\, y;\, \mathbf{close}\, x))
\end{array}
$$

To prove that this process is secure, we would have to show that for any substitution that instantiates free variables x and y with free channels x_α and y_β

$$\begin{array}{l} y_\beta : \mathsf{bit}[\mathbf{guest}] \Vdash \mathbf{proc}(x_\alpha, \mathsf{X}) :: x_\alpha{:}\mathsf{pin}[\mathbf{alice}] \;\equiv^{\Psi_0}_{\mathbf{guest}} \\ y_\beta : \mathsf{bit}[\mathbf{guest}] \Vdash \mathbf{proc}(x_\alpha, \mathsf{X}) :: x_\alpha{:}\mathsf{pin}[\mathbf{alice}] \end{array}$$

We compose the process with the following high-secrecy clients $\mathcal{T}_1$ and $\mathcal{T}_2$, sending different tokens along the high-secrecy channel $x_\alpha{:}\mathsf{pin}[\mathbf{alice}]$,

$$\begin{array}{l} \mathcal{T}_1 = \mathbf{proc}(z_\eta, x_\alpha.tok_1; \mathbf{wait}\, x_\alpha; \mathbf{close}\, z_\eta) \\ \mathcal{T}_2 = \mathbf{proc}(z_\eta, x_\alpha.tok_2; \mathbf{wait}\, x_\alpha; \mathbf{close}\, z_\eta), \end{array}$$

leaving us left to show that for all m:

$$\begin{array}{l} (\mathbf{proc}(x_\alpha, \mathsf{X})\, \mathbf{proc}(z_\eta, x_\alpha.tok_1; \mathbf{wait}\, x_\alpha; \mathbf{close}\, z_\eta); \\ \;\mathbf{proc}(x_\alpha, \mathsf{X})\, \mathbf{proc}(z_\eta, x_\alpha.tok_2; \mathbf{wait}\, x_\alpha; \mathbf{close}\, z_\eta)) \in \mathcal{E}[\![y_\beta{:}\mathsf{bit} \vdash \_{:}1]\!]^m \end{array}$$

Obviously this does not hold true, confirming that process X is insecure. Consider a variant of X that sends the bit *zero* in either case. This variant would be accepted by our logical relation, as it should because it is secure.

Alternatively, we could employ an *information flow control (IFC)* type system [51, 54, 61] to verify whether process X is secure. Such type systems label of observables (e.g., output, locations, channels) with secrecy levels drawn from a given security lattice Ψ_0 to prevent "flows from high to lo". Next, we sketch such an IFC type system as a refinement of SESSION. The full refinement type system is included in § D. It is based on existing work [15, 16], and thus we only summarize its main characteristics and discuss the typing rules relevant for the example.

To type process terms, we use the judgment

$$\Psi; \Gamma \vdash_\Sigma P@c :: x : A[d].$$

The new elements, compared to SESSION's judgment (see § 2), is the security lattice Ψ and the possible worlds annotations $@c$ and $[d]$ [6], to denote a process' running secrecy and maximal secrecy, resp. Γ is a linear typing context, like Δ, but where types now have maximal secrecy annotations. The idea is that a process' *maximal secrecy* indicates the maximal level of secret information the process may ever obtain, whereas a process' *running secrecy* denotes the highest level of secret information the process has obtained so far.

To constrain the propagation of information, the following invariants are presupposed on the typing judgment $\Psi; \Gamma \vdash_\Sigma P@c :: x : A[d]$

(i) $\forall y{:}B[d'] \in \Gamma. \Psi \Vdash d' \sqsubseteq d$
(ii) $\Psi \Vdash c \sqsubseteq d$

ensuring that the maximal secrecy of a child node is capped by the maximal secrecy of its parent and that the running secrecy of a process is less than or equal to its maximal secrecy, resp. Additionally, the type system ensures that

1. no channel of high secrecy is sent along one with lower secrecy; and

2. a process does not send any message along a low-secrecy channel after receipt of high secrecy information.

The first condition prevents direct flows and is enforced by requiring that the maximal secrecy level of a carrier channel and the channels sent over it match. The second condition is met by ensuring that a process' running secrecy is a sound approximation of the level of secret information a process has obtained so far. To this end, the type system increases the running secrecy of a process upon receipt to *at least* the maximal secrecy of the sending process and, correspondingly, guards sends by making sure that the running secrecy of the sending process is *at most* the maximal secrecy of the receiving process.

This working can be seen in the typing rules for external choice:

$$\frac{\Psi; \Gamma \vdash_\Sigma Q_k@c :: y{:}A_k[c] \qquad \forall k \in L}{\Psi; \Gamma \vdash_\Sigma (\mathbf{case}\, y^c(\ell \Rightarrow Q_\ell)_{\ell \in I})@d_1 :: y{:}\&\{\ell : A_\ell\}_{\ell \in L}[c]}\ \&R$$

$$\frac{\Psi \Vdash d_1 \sqsubseteq c \qquad \Psi; \Gamma, x{:}A_k[c] \vdash_\Sigma P@d_1 :: y{:}C[c'] \qquad k \in L}{\Psi; \Gamma, x{:}\&\{\ell : A_\ell\}_{\ell \in I}[c] \vdash_\Sigma (x^c.k; P)@d_1 :: y{:}C[c']}\ \&L$$

The right rule increases the provider's running secrecy from d_1 to its maximal secrecy c after receipt of the label k. The left rule guards the send by the premise $\Psi \Vdash d_1 \sqsubseteq c$, demanding that the provider's running secrecy d_1 is less than or equal to the maximal secrecy c of the recipient. It is precisely this guard that prevents process X to type check using the refinement IFC type system. This guard also prevents the variant of X, which sends the bit *zero* in either case, to type check, which is accepted by our logical relation. This outcome is expected because type systems have the benefit of automatic verification, foregoing completeness. Our logical relation thus amounts to a *semantic* logical relation [13, 39, 58] for PSNI, as it only requires configurations to be ILLST-typed, but not IFC-typed. The fact that IFC-typed processes are non-interfering is proved as the *fundamental theorem* of the logical relation, Thm. 7 in § G.

4.4 Metatheoretic properties of up-to equivalence

Unsurprisingly, $\equiv_\xi^{\Psi_0}$ has analogous properties to $\equiv$, except for reflexivity, which only holds for IFC-typed configurations, as demonstrated by our X process in § 4.3.

Semantic cut: closure under parallel composition

Lemma 7 (Semantic cut). *If*
1. $(\Gamma_1' \Vdash \mathcal{C}_1 :: \Gamma) \equiv_{\Psi_0}^{\xi} (\Gamma_2' \Vdash \mathcal{C}_2 :: \Gamma)$ with $\Gamma = \Gamma_1 \Downarrow_\xi = \Gamma_2 \Downarrow_\xi$ and
2. $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv_{\Psi_0}^{\xi} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ and
3. $(K^s \Vdash \mathcal{F}_1 :: K_1^s) \equiv_{\Psi_0}^{\xi} (K^s \Vdash \mathcal{F}_2 :: K_2^s)$
with $K^s = x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi$

then $(\Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: K_1^s) \equiv^{\xi}_{\Psi_0} (\Gamma_2', \Gamma_2^h \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: K_2^s)$*, where* Γ_1^h *is the set of all channels* $w_\eta{:}C[d] \in \Gamma_1$ *with* $d \not\sqsubseteq \xi$ *and* Γ_2^h *is the set of all channels* $w_\eta{:}C[d] \in \Gamma_2$ *with* $d \not\sqsubseteq \xi$*.*

Proof. See Lem. 36 in the Appendix.

Partial equivalence relation (PER)

Lemma 8 (Symmetry). *For all security levels* ξ *and configurations* $\mathcal{D}_1$ *and* $\mathcal{D}_2$*, we have* $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv^{\Psi_0}_{\xi} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2])$*, iff* $(\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2]) \equiv^{\Psi_0}_{\xi} (\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1])$*.*

Proof. See proof of Lem. 32 in the Appendix.

Lemma 9 (Transitivity). *For all security levels* ξ*, and configurations* $\mathcal{D}_1$*,* $\mathcal{D}_2$*, and* $\mathcal{D}_3$*, we have*

$$\begin{array}{l}\mathsf{if}\ (\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv^{\Psi_0}_{\xi} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2])\ \mathsf{and}\\ \ (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2]) \equiv^{\Psi_0}_{\xi} (\Gamma_3 \Vdash \mathcal{D}_3 :: z_\eta{:}T_3[c_3])\\ \mathsf{then}\ (\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv^{\Psi_0}_{\xi} (\Gamma_3 \Vdash \mathcal{D}_3 :: z_\eta{:}T_3[c_3]).\end{array}$$

Proof. See proof of Corollary 3 in the Appendix.

Adequacy

Definition 9 (Bisimulation with secrecy annotated sequent). *For* $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$*,* $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$ *we define* $\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1] \approx^{\xi}_{\mathsf{a}} \Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2]$ *as*

$$\begin{array}{c}\Gamma_1 \Downarrow \xi = \Gamma_2, \Downarrow \xi\ \ \mathsf{and}\ \ y_\beta{:}A_2[c_2] \Downarrow \xi = x_\alpha{:}A_1[c_1] \Downarrow \xi\ \ \mathsf{and}\\ \forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1).\forall \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2).\\ \forall \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]).\forall \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]).\\ \mathcal{B}_1\mathcal{D}_1\mathcal{T}_1 \approx_{\mathsf{a}} \mathcal{B}_2\mathcal{D}_2\mathcal{T}_2.\end{array}$$

Lemma 10 (Adequacy). *For all* $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$ *and* $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$*, we have* $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\Psi_0}_{\xi} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ *iff* $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \approx^{\xi}_{\mathsf{a}} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$*.*

Proof. See proof of Corollary 4 in the Appendix.

Biorthogonality

Theorem 2 ($\top\top$-closure). *Consider* $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash |x_\alpha{:}A_1[c_1]|)$ *and* $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash |y_\beta{:}A_2[c_2]|)$ *and a given observer level* $\xi \in \Psi_0$*. We have*

$$\begin{array}{l}(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])\ \textbf{iff}\\ \forall \mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2.\\ \mathsf{if}\ (\Gamma_1 \dashv \mathcal{C}_1[\ ]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \dashv \mathcal{C}_2[\ ]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2])\\ \mathsf{then}\ \forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1).\forall \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2).\\ \forall \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]).\forall \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]).\\ \mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1\ \approx_{\mathsf{a}}\ \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2\end{array}$$

Proof. See proof of Thm. 9 in the Appendix.

Completeness ensures that the logical relation relates all secure programs, whereas IFC type systems, by nature, reject infinitely many secure programs. Our logical relation thus becomes generally applicable. For example, we may conceive a more permissive IFC refinement type system that accepts more secure programs. All we would have to do in this case is prove the fundamental theorem for that new refinement type system, guaranteeing that well-typed programs are self-related by our logical relation and thus non-interfering. We conclude this section by noting that all the results presented here are corollaries of the results in Section 3.

5 Related work and discussion

Logical relations for session types. The application of logical relations to session types has focused predominantly on unary logical relations for proving termination [17, 46, 47, 50], except for a binary logical relation for parametricity [9] and noninterference [15, 16, 24]. Our RSLR shares its foundation in linear logic with this line of work, but contributes significantly through its support of general recursive types.

Caires et al. [9] show parametricity for a terminating language and restrict observations to the single offering channel of a configuration, i.e., observations are only made at the root. Thanks to the validity of semantic cut, a single-type-indexed logical relation, such as the authors', falls out as a special case of our RSLR. However, more fine-grained program equivalences, such as noninterference, demand distinguishing the two roles a process configuration may assume. To the best of our understanding, our account of duality ultimately also enabled biorthogonal closure to prove the logical equivalence sound and complete.

Among prior work on logical relations for ILLST, the work by Derakhshan et al. [15] is most closely related to ours. The authors contribute an IFC session type system and develop a logical relation to show noninterference. The refinement type system that we use in our case study (§ 4.3) is based on that IFC system, extended with secrecy-polymorphic processes [16]. Besides the authors' confinement to a terminating language, sidestepping the intricacies of non-termination and nondeterminism, the authors' metatheoretic results are significantly more limited than ours. In particular, the logical relation is dependent on IFC typing, and thus only amounts to a *syntactic* logical relation for noninterference. Moreover, their development lacks metatheoretic results about the entailed program equivalence, i.e., that it is sound with respect to bisimilarity. There are several nuances in the design of our logical relation that enable us to show such results in a more general setting, which also allows for non-termination: Unlike [15], which only considers closed configurations, i.e., it cannot relate two programs without their clients and providers, our logical relation is defined over open configurations and thus allows for connection to biorthogonality. Furthermore, our term interpretation carefully uses the universal and existential quantifiers and thus enables the proof of transitivity for the logical relation, which is not proven in [15]. Our logical relation also handles non-termination via focus channels and the observation index. These features enable defining the logical relation for equivalence in a general context, proving that it is both sound and complete with respect to asynchronous bisimilarity by establishing a connection to biorthogonality, and then instantiating it in various settings, such as noninterference.

Relationship to logical relations for stateful languages. Logical relations have been scaled to accommodate state using Kripke logical relations (KLRs) [49]. KLRs are indexed by a *possible world* W, providing a semantic model of the heap. Invariants can then be imposed that must be preserved by any future worlds W'. KRLs can be combined with step indexing [2, 4] to address circularity arising from higher-order stores [3, 19, 20] and to express state transition systems. For example, Gregersen et al. [22] use the KLR supported in Iris [32] to prove noninterference for a functional language with a higher-order store, recursive types, and impredicative polymorphism. The authors also adopt semantic typing [13, 39, 58], admitting syntactically ill-typed programs, if shown to inhabit the logical relation. Besides the difference in language, the authors consider termination-*insensitive* noninterference, whereas we consider progress-*sensitive* noninterference. Like KLRs, our RSLR is situated in a stateful setting because channels, like locations, are subject to concurrent mutation. However, our RSLR is rooted in ILLST, which guarantees race freedom, stratifies the store (the configuration of processes), and prescribes state transitions. When squinting one's eyes, the sequent $\Delta \Vdash K$ of the logical relation seems reminiscent of a possible world, since it provides the semantic typing of the free channels. However, linearity would not comport well with the usual monotonicity requirement of logical relations. We would like to investigate this connection as future work.

6 Concluding remarks

We have contributed a recursive session logical relation (RSLR) for progress-sensitive equivalence of programs. The RSLR is rooted in intuitionistic linear logic session types, inheriting from it a strong foundation in linear logic, ensuring that channel endpoints are treated as resources. A novel aspect of the RSLR is its use of an *observation index* to keep the logical relation well-defined in the presence of general recursive types. This shift, from a step index / unfolding index associated with *internal* computation steps to an index only associated with *external* observations, facilitates statement and proof of metatheoretic properties of the logical relation: closure under parallel composition, soundness and completeness with regard to weak asynchronous bisimilarity, and biorthogonal closure of the logical equivalence relation induced by the RSLR. A biorthogonality argument arises naturally from the duality inherent to session types. In future work, we would like to explore these connections more deeply, possibly connecting to Kripke logical relations and game semantics.

References

1. Ahmed, A.: Semantics of Types for Mutable State. Ph.D. thesis, Princeton University (2004)
2. Ahmed, A.: Step-indexed syntactic logical relations for recursive and quantified types. In: 15th European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 3924, pp. 69–83. Springer (2006). https://doi.org/10.1007/11693024_6
3. Ahmed, A., Dreyer, D., Rossberg, A.: State-dependent representation independence. In: 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 340–353. ACM (2009). https://doi.org/10.1145/1480881.1480925
4. Appel, A.W., McAllester, D.A.: An indexed model of recursive types for foundational proof-carrying code. ACM Transactions on Programming Languages and Systems (TOPLAS) **23**(5), 657–683 (2001). https://doi.org/10.1145/504709.504712, https://doi.org/10.1145/504709.504712
5. Balzer, S., Pfenning, F.: Manifest sharing with session types. Proceedings of the ACM on Programming Languages **1**(ICFP), 37:1–37:29 (2017). https://doi.org/10.1145/3110281, https://doi.org/10.1145/3110281
6. Balzer, S., Toninho, B., Pfenning, F.: Manifest deadlock-freedom for shared session types. In: 28th European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 11423, pp. 611–639. Springer (2019). https://doi.org/10.1007/978-3-030-17184-1_22, https://doi.org/10.1007/978-3-030-17184-1_22
7. Benton, N., Hur, C.: Biorthogonality, step-indexing and compiler correctness. In: 14th ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 97–108. ACM (2009). https://doi.org/10.1145/1596550.1596567, https://doi.org/10.1145/1596550.1596567
8. Benton, N., Hur, C.: Step-indexing: The good, the bad and the ugly. In: Modelling, Controlling and Reasoning About State. Dagstuhl Seminar Proceedings, vol. 10351, pp. 1–9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany (2010), http://drops.dagstuhl.de/opus/volltexte/2010/2808/
9. Caires, L., Pérez, J.A., Pfenning, F., Toninho, B.: Behavioral polymorphism and parametricity in session-based communication. In: 22nd European Symposium on Programming (ESOP). pp. 330–349 (2013). https://doi.org/10.1007/978-3-642-37036-6_19, https://doi.org/10.1007/978-3-642-37036-6_19
10. Caires, L., Pfenning, F.: Session types as intuitionistic linear propositions. In: 21th International Conference onf Concurrency Theory (CONCUR). Lecture Notes in Computer Science, vol. 6269, pp. 222–236. Springer (2010). https://doi.org/10.1007/978-3-642-15375-4_16, https://doi.org/10.1007/978-3-642-15375-4_16
11. Caires, L., Pfenning, F., Toninho, B.: Linear logic propositions as session types. Mathematical Structures in Computer Science **26**(3), 367–423 (2016). https://doi.org/10.1017/S0960129514000218, https://doi.org/10.1017/S0960129514000218
12. Cervesato, I., Scedrov, A.: Relating state-based and process-based concurrency through linear logic. Information and Computation **207**(10), 1044–1077 (2009). https://doi.org/10.1016/j.ic.2008.11.006, https://doi.org/10.1016/j.ic.2008.11.006

13. Constable, R.L., Allen, S.F., Bromley, M., Cleaveland, R., Cremer, J.F., Harper, R., Howe, D.J., Knoblock, T.B., Mendler, N.P., Panangaden, P., Sasaki, J.T., Smith, S.F.: Implementing Mathematics with the Nuprl Proof Development System. Prentice Hall (1986), http://dl.acm.org/citation.cfm?id=10510
14. Crary, K., Harper, R., Puri, S.: What is a recursive module? In: ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). pp. 50–63. ACM (1999). https://doi.org/10.1145/301618.301641, https://doi.org/10.1145/301618.301641
15. Derakhshan, F., Balzer, S., Jia, L.: Session logical relations for noninterference. In: 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–14. IEEE Computer Society (2021). https://doi.org/10.1109/LICS52264.2021.9470654, https://doi.org/10.1109/LICS52264.2021.9470654
16. Derakhshan, F., Balzer, S., Yao, Y.: Regrading policies for flexible information flow control in session-typed concurrency. In: 38th European Conference on Object-Oriented Programming (ECOOP). LIPIcs, vol. 313, pp. 11:1–11:29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPICS.ECOOP.2024.11, https://doi.org/10.4230/LIPIcs.ECOOP.2024.11
17. DeYoung, H., Pfenning, F., Pruiksma, K.: Semi-axiomatic sequent calculus. In: 5th International Conference on Formal Structures for Computation and Deduction (FSCD). LIPIcs, vol. 167, pp. 29:1–29:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.FSCD.2020.29, https://doi.org/10.4230/LIPIcs.FSCD.2020.29
18. Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. In: 24th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 71–80. IEEE Computer Society (2009). https://doi.org/10.1109/LICS.2009.34
19. Dreyer, D., Neis, G., Birkedal, L.: The impact of higher-order state and control effects on local relational reasoning. In: 15th ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 143–156. ACM (2010). https://doi.org/10.1145/1863543.1863566, https://doi.org/10.1145/1863543.1863566
20. Dreyer, D., Neis, G., Birkedal, L.: The impact of higher-order state and control effects on local relational reasoning. Journal of Functional Programming **22**(4-5), 477–528 (2012). https://doi.org/10.1017/S095679681200024X, https://doi.org/10.1017/S095679681200024X
21. Gay, S.J., Hole, M.: Subtyping for session types in the pi calculus. Acta Informatica **42**(2-3), 191–225 (2005). https://doi.org/10.1007/s00236-005-0177-z, https://doi.org/10.1007/s00236-005-0177-z
22. Gregersen, S.O., Bay, J., Timany, A., Birkedal, L.: Mechanized logical relations for termination-insensitive noninterference. Proceedings of the ACM on Programming Languages **5**(POPL), 1–29 (2021). https://doi.org/10.1145/3434291, https://doi.org/10.1145/3434291
23. Heidin, D., Sabelfeld, A.: A perspective on information-flow control. Tech. rep., Marktoberdorf (2011)
24. van den Heuvel, B., Derakhshan, F., Balzer, S.: Information flow control in cyclic process networks. In: 38th European Conference on Object-Oriented Programming (ECOOP). LIPIcs, vol. 313, pp. 40:1–40:30. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024). https://doi.org/10.4230/LIPICS.ECOOP.2024.40, https://doi.org/10.4230/LIPIcs.ECOOP.2024.40
25. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall (1985)
26. Honda, K.: Types for dyadic interaction. In: 4th International Conference on Concurrency Theory (CONCUR). Lecture Notes in Computer Science, vol. 715, pp.

509–523. Springer (1993). https://doi.org/10.1007/3-540-57208-2_35, https://doi.org/10.1007/3-540-57208-2_35
27. Honda, K., Vasconcelos, V.T., Kubo, M.: Language primitives and type discipline for structured communication-based programming. In: 7th European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 1381, pp. 122–138. Springer (1998). https://doi.org/10.1007/BFb0053567, https://doi.org/10.1007/BFb0053567
28. Hur, C., Dreyer, D.: A kripke logical relation between ML and assembly. In: 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 133–146. ACM (2011). https://doi.org/10.1145/1926385.1926402, https://doi.org/10.1145/1926385.1926402
29. Hur, C., Dreyer, D., Neis, G., Vafeiadis, V.: The marriage of bisimulations and kripke logical relations. In: 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 59–72. ACM (2012). https://doi.org/10.1145/2103656.2103666, https://doi.org/10.1145/2103656.2103666
30. Igarashi, A., Kobayashi, N.: A generic type system for the pi-calculus. In: 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 128–141 (2001). https://doi.org/10.1145/360204.360215, https://doi.org/10.1145/360204.360215
31. Igarashi, A., Kobayashi, N.: A generic type system for the pi-calculus. Theoretical Computer Science **311**(1-3), 121–163 (2004). https://doi.org/10.1016/S0304-3975(03)00325-6, https://doi.org/10.1016/S0304-3975(03)00325-6
32. Jung, R., Krebbers, R., Jourdan, J., Bizjak, A., Birkedal, L., Dreyer, D.: Iris from the ground up: A modular foundation for higher-order concurrent separation logic. Journal of Functional Programming **28**, e20 (2018). https://doi.org/10.1017/S0956796818000151, https://doi.org/10.1017/S0956796818000151
33. Kobayashi, N.: A partially deadlock-free typed process calculus. In: 12th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 128–139. IEEE Computer Society (1997). https://doi.org/10.1109/LICS.1997.614941, https://doi.org/10.1109/LICS.1997.614941
34. Kobayashi, N.: A new type system for deadlock-free processes. In: 17th International Conference on Concurrency Theory (CONCUR). LNCS, vol. 4137, pp. 233–247 (2006). https://doi.org/10.1007/11817949_16, https://doi.org/10.1007/11817949_16
35. Kokke, W., Montesi, F., Peressotti, M.: Better late than never: A fully-abstract semantics for classical processes. Proceedings of the ACM on Programming Languages **3**(POPL), 24:1–24:29 (2019). https://doi.org/10.1145/3290337, https://doi.org/10.1145/3290337
36. Koutavas, V., Wand, M.: Small bisimulations for reasoning about higher-order imperative programs. In: 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 141–152. ACM (2006). https://doi.org/10.1145/1111037.1111050, https://doi.org/10.1145/1111037.1111050
37. Lindley, S., Morris, J.G.: A semantics for propositions as sessions. In: 24th European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 9032, pp. 560–584. Springer (2015). https://doi.org/10.1007/978-3-662-46669-8_23, https://doi.org/10.1007/978-3-662-46669-8_23
38. Lindley, S., Morris, J.G.: Talking bananas: Structural recursion for session types. In: 21st ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 434–447. ACM (2016). https://doi.org/10.1145/2951913.2951921, https://doi.org/10.1145/2951913.2951921

39. Martin-Löf, P.: Constructive mathematics and computer programming. In: Logic, Methodology and Philosophy of Science VI, Studies in Logic and the Foundations of Mathematics, vol. 104, pp. 153–175. Elsevier (1982). https://doi.org/https://doi.org/10.1016/S0049-237X(09)70189-2, https://www.sciencedirect.com/science/article/pii/S0049237X09701892
40. Melliès, P., Vouillon, J.: Recursive polymorphic types and parametricity in an operational framework. In: 20th Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 82–91. IEEE Computer Society (2005). https://doi.org/10.1109/LICS.2005.42, https://doi.org/10.1109/LICS.2005.42
41. Milner, R.: A Calculus of Communicating Systems, Lecture Notes in Computer Science, vol. 92. Springer (1980). https://doi.org/10.1007/3-540-10235-3, https://doi.org/10.1007/3-540-10235-3
42. Milner, R.: Communication and Concurrency. PHI Series in Computer Science, Prentice Hall (1989)
43. Milner, R.: Communicating and Mobile Systems: the π-calculus. Cambridge University Press (1999)
44. Morris, J.H.: Lambda-Calculus Models of Programming Languages. Ph.D. thesis, Massachusetts Institute of Technology (1968)
45. Neis, G., Dreyer, D., Rossberg, A.: Non-parametric parametricity. Journal of Functional Programming **21**(4-5), 497–562 (2011). https://doi.org/10.1017/S0956796811000165, https://doi.org/10.1017/S0956796811000165
46. Pérez, J.A., Caires, L., Pfenning, F., Toninho, B.: Linear logical relations for session-based concurrency. In: 21st European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 7211, pp. 539–558. Springer (2012). https://doi.org/10.1007/978-3-642-28869-2_27, https://doi.org/10.1007/978-3-642-28869-2_27
47. Pérez, J.A., Caires, L., Pfenning, F., Toninho, B.: Linear logical relations and observational equivalences for session-based concurrency. Information and Computation **239**, 254–302 (2014). https://doi.org/10.1016/j.ic.2014.08.001, https://doi.org/10.1016/j.ic.2014.08.001
48. Pitts, A.M.: Parametric polymorphism and operational equivalence. Mathematical Structures in Computer Science **10**(3), 321–359 (2000), http://journals.cambridge.org/action/displayAbstract?aid=44651
49. Pitts, A.M., Stark, I.: Operational reasoning for functions with local state. Higher Order Operational Techniques in Semantics (HOOTS) pp. 227–273 (1998)
50. Rocha, P., Caires, L.: Safe session-based concurrency with shared linear state. In: 32nd European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 13990, pp. 421–450. Springer (2023). https://doi.org/10.1007/978-3-031-30044-8_16
51. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE Journal of Selected Areas in Communications **21**(1), 5–19 (2003). https://doi.org/10.1109/JSAC.2002.806121
52. Sangiorgi, D., Kobayashi, N., Sumii, E.: Environmental bisimulations for higher-order languages. In: 22nd Annual IEEE Symposium on Logic in Computer Science (LICS). pp. 293–302. IEEE Computer Society (2007). https://doi.org/10.1109/LICS.2007.17, https://doi.org/10.1109/LICS.2007.17
53. Sangiorgi, D., Walker, D.: The π-calculus: a Theory of Mobile Processes. Cambridge University Press (2001)
54. Smith, G., Volpano, D.M.: Secure information flow in a multi-threaded imperative language. In: 25th ACM SIGPLAN-SIGACT Symposium on Principles of Pro-

gramming Languages (POPL). pp. 355–364. ACM (1998). https://doi.org/10.1145/268946.268975, https://doi.org/10.1145/268946.268975

55. Støvring, K., Lassen, S.B.: A complete, co-inductive syntactic theory of sequential control and state. In: 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL). pp. 161–172. ACM (2007). https://doi.org/10.1145/1190216.1190244, https://doi.org/10.1145/1190216.1190244
56. Sumii, E., Pierce, B.C.: A bisimulation for type abstraction and recursion. Journal of the ACM **54**(5), 26 (2007). https://doi.org/10.1145/1284320.1284325, https://doi.org/10.1145/1284320.1284325
57. Thamsborg, J., Birkedal, L.: A kripke logical relation for effect-based program transformations. In: 16th ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 445–456. ACM (2011). https://doi.org/10.1145/2034773.2034831, https://doi.org/10.1145/2034773.2034831
58. Timany, A., Krebbers, R., Dreyer, D., Birkedal, L.: A logical approach to type soundness. Journal of the ACM (JACM) **71**(6), 40:1–40:75 (2024). https://doi.org/10.1145/3676954, https://doi.org/10.1145/3676954
59. Toninho, B.: A Logical Foundation for Session-Based Concurrent Computation. Ph.D. thesis, Carnegie Mellon University and New University of Lisbon (2015)
60. Toninho, B., Caires, L., Pfenning, F.: Higher-order processes, functions, and sessions: A monadic integration. In: 22nd European Symposium on Programming (ESOP). Lecture Notes in Computer Science, vol. 7792, pp. 350–369. Springer (2013). https://doi.org/10.1007/978-3-642-37036-6_20, https://doi.org/10.1007/978-3-642-37036-6_20
61. Volpano, D.M., Irvine, C.E., Smith, G.: A sound type system for secure flow analysis. Journal of Computer Security **4**(2/3), 167–188 (1996). https://doi.org/10.3233/JCS-1996-42-304, https://doi.org/10.3233/JCS-1996-42-304
62. Wadler, P.: Propositions as sessions. In: ACM SIGPLAN International Conference on Functional Programming (ICFP). pp. 273–286. ACM (2012). https://doi.org/10.1145/2364527.2364568, https://doi.org/10.1145/2364527.2364568

Appendices

In the appendix, we also present an IFC refinement type system, along with proofs of progress, preservation, and noninterference. The results related to the IFC type system can be found in Appendix C, Appendix D, Appendix E.B, Appendix G.B, and Appendix G.C.

A Abstract syntax

The abstract syntax of SESSION is given below. Lines without a left-hand side are separated by | from their preceding line.

Sort		**Abstract Form**	**Remarks**
Metavariable	$\triangleq$	Ψ_0	concrete security lattice $\langle \mathcal{L}, \mathcal{E}_0, \sqcup, \sqcap \rangle$
		$\iota, \eta \in \mathcal{L}$	set of concrete security levels
		$\xi \in \mathcal{L}$	concrete security level of observer
		$E_0 \in \mathcal{E}_0$	set of relations E_0 of form $\iota \sqsubseteq \iota'$
		Ψ	security theory $\langle \mathcal{V}, \mathcal{E}, \sqcup, \sqcap \rangle$
		$\psi, \omega \in \mathcal{V}$	set of security variables
		$c, d, e, f \in \mathcal{S}$	security terms of Ψ
		$p \in \mathcal{S} \times \mathcal{S}$	pair of security terms of Ψ
		$E \in \mathcal{E}$	set of relations E of form $c \sqsubseteq d$
		$x_\alpha, x_\beta, x_\gamma, x_\delta$	channel
		x, y, z, u, v, w	channel variable
		$j, k, \ell \in I, L$	set of labels
		Δ, Λ	linear typing contexts(channels)
		Ω	linear typing contexts (variables)
		Γ	linear security typing contexts (channels)
		Ξ	linear security typing contexts(variables)
		K	linear typing context singleton(channel)
		K^s	linear security typing context singleton(channel)
		$\gamma_{\mathsf{sec}}, \hat{\gamma_{\mathsf{sec}}}, \delta_{\mathsf{sec}}, \hat{\delta_{\mathsf{sec}}}$	order-preserving substitution of security elements
		$\gamma, \hat{\gamma}, \delta, \hat{\delta}$	channel variable substitution
		$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{T}$	process configuration in SESSION
		$\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}, \mathbb{T}$	process configuration in CONSESSION
Type A, B, C, T	$\triangleq$	$\oplus\{\ell{:}A_\ell\}_{\ell \in L}$	internal choice, at least one label
		$\&\{\ell{:}A_\ell\}_{\ell \in L}$	external choice, at least one label
		$A \otimes B$	channel output
		$A \multimap B$	channel input
		$\mathbf{1}$	termination
		Y	type variable
Definition X, Y	$\triangleq$	$\Psi; \Delta \vdash_\Sigma X = P@\psi_0 :: x{:}A[\psi]$	process definition
		$Y = A$	type definition
Process P, Q	$\triangleq$	$x.k; P$	label output
		$\mathbf{case}\, x(\ell \Rightarrow P_\ell)_{\ell \in L}$	label input
		$\mathbf{send}\, y\, x; P$	channel output

		$y \leftarrow \mathbf{recv}\, x; P_y$	channel input
		$\mathbf{close}\, x$	terminate process
		$\mathbf{wait}\, x; Q$	wait for process to terminate
		$x \leftarrow X[\gamma] \leftarrow \Delta; Q_x$	spawn
		$x \leftarrow y$	forward x to y
		F_Y	forwarder process variable for Y
Messages M, N	$\triangleq$	$x.k$	label output
		$\mathbf{send}\, y\, x$	channel output
		$\mathbf{close}\, x$	terminate process

B ILLST type system and asynchronous dynamics

This section defines the rules to type SESSION programs, both for source code as well as for run-time configurations.

B.1 Process term typing

Fig. 7 summarizes the typing rules for process terms in SESSION.

Rule D-FWD in Fig. 7 relies on a forwarder for each user-defined type definition, amounting to an identity expansion automatically generated as defined in Def. 10.

Definition 10. *For all $Y = A \in \Sigma$, we extend Σ by adding the following definition to the signature Σ:*

$$x' : A \vdash F_Y = \mathtt{Fwd}_{A, y' \leftarrow x'} :: y' : A \qquad Y = A \in \Sigma$$

Here F_Y is a specific process variable assigned to the forwarder process for type variable Y, and $\mathtt{Fwd}_{A, y \leftarrow x}$ is defined by induction on the structure of A as a function from type A to process terms as follows:

$$\begin{aligned}
\mathtt{Fwd}_{\oplus\{\ell:A_\ell\}_{\ell \in L}, y \leftarrow x} &:= \mathbf{case}\, x(\ell \Rightarrow y.\ell; \mathtt{Fwd}_{A_\ell, y \leftarrow x})_{\ell \in L} \\
\mathtt{Fwd}_{\&\{\ell:A_\ell\}_{\ell \in L}, y \leftarrow x} &:= \mathbf{case}\, y(\ell \Rightarrow x.\ell; \mathtt{Fwd}_{A_\ell, y \leftarrow x})_{\ell \in L} \\
\mathtt{Fwd}_{A \otimes B, y \leftarrow x} &:= w \leftarrow \mathbf{recv}\, x; \mathbf{send}\, w\, y; \mathtt{Fwd}_{B, y \leftarrow x} \\
\mathtt{Fwd}_{A \multimap B, y \leftarrow x} &:= w \leftarrow \mathbf{recv}\, y; \mathbf{send}\, w\, x; \mathtt{Fwd}_{B, y \leftarrow x} \\
\mathtt{Fwd}_{1, y \leftarrow x} &:= \mathbf{wait}\, x; \mathbf{close}\, y \\
\mathtt{Fwd}_{Y; y \leftarrow x} &:= F_Y[x \mapsto x', y \mapsto y'] \qquad Y = A \in \Sigma
\end{aligned}$$

Forwarders are well-typed by construction, as shown next:

Lemma 11. *Given the extended signature Σ, for all type A, there is a derivation for $x : A \vdash_\Sigma \mathtt{Fwd}_{A, y \leftarrow x} :: y : A$.*

Proof. The proof is by induction on the structure of type A. In a base case, where A is a type variable Y, i.e., $A = Y$ for $Y = C \in \Sigma$, the proof is straightforward by the D-FWD rule since $x : C \vdash F_Y := \mathtt{Fwd}_{C, y \leftarrow x} :: y : A \in \Sigma$. The proof of other cases is straightforward.

$$\frac{\Omega \vdash_\Sigma P :: x{:}A_k \qquad k \in L}{\Omega \vdash_\Sigma x.k; P :: x{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}} \oplus R \qquad \frac{\Omega, x{:}A_k \vdash_\Sigma Q_k :: y{:}C \quad \forall k \in L}{\Omega, x{:} \oplus \{\ell : A_\ell\}_{\ell \in L} \vdash_\Sigma \mathbf{case}\, x(\ell \Rightarrow Q_\ell)_{\ell \in L} :: y{:}C} \oplus L$$

$$\frac{\Omega \vdash_\Sigma Q_k :: x{:}A_k \qquad \forall k \in L}{\Omega \vdash_\Sigma \mathbf{case}\, x(\ell \Rightarrow Q_\ell)_{\ell \in I} :: x{:}\&\{\ell : A_\ell\}_{\ell \in L}} \&R \qquad \frac{\Omega, x{:}A_k \vdash_\Sigma P :: y{:}C \qquad k \in L}{\Omega, x{:}\&\{\ell : A_\ell\}_{\ell \in I} \vdash_\Sigma x.k; P :: y{:}C} \&L$$

$$\frac{\Omega \vdash_\Sigma P :: x{:}B}{\Omega, z{:}A \vdash_\Sigma \mathbf{send}\, z\, x; P :: x{:}A \otimes B} \otimes R \qquad \frac{\Omega, z{:}A, x{:}B \vdash_\Sigma P :: y{:}C}{\Omega, x{:}A \otimes B \vdash_\Sigma z \leftarrow \mathbf{recv}\, x; P :: y{:}C} \otimes L$$

$$\frac{\Omega, z{:}A \vdash_\Sigma P :: x{:}B}{\Omega \vdash_\Sigma z \leftarrow \mathbf{recv}\, x; P :: x{:}A \multimap B} \multimap R \qquad \frac{\Omega, x{:}B \vdash_\Sigma P :: y{:}C}{\Omega, z{:}A, x{:}A \multimap B \vdash_\Sigma \mathbf{send}\, z\, x; P :: y{:}C} \multimap L$$

$$\frac{}{\cdot \vdash_\Sigma \mathbf{close}\; x :: x : 1} 1R \qquad \frac{\Omega \vdash_\Sigma Q :: y : C}{\Omega, x : 1 \vdash_\Sigma \mathbf{wait}\, x; Q :: y : C} 1L$$

$$\frac{\Omega_1' \vdash X = P :: x'{:}A \in \Sigma \qquad \Omega_1, x{:}A \Vdash \gamma :: \Omega_1', x'{:}A \qquad \Omega_2, x{:}A \vdash_\Sigma Q :: y{:}C}{\Omega_1, \Omega_2 \vdash_\Sigma (x \leftarrow X[\gamma] \leftarrow \Omega_1); Q :: y{:}C} \textsc{Spawn}$$

$$\frac{\begin{array}{c} z'{:}C \vdash F_Y = \mathtt{Fwd}_{C, y' \leftarrow z'} :: y'{:}C \in \Sigma \\ Y = C \in \Sigma \qquad z{:}C, y{:}C \Vdash \gamma :: z'{:}C, y'{:}C \end{array}}{z{:}Y \vdash_\Sigma F_Y[\gamma] :: y{:}C} \text{D-}\textsc{Fwd}$$

$$\frac{Y = A \in \Sigma \qquad \Omega \vdash_\Sigma P :: x{:}A}{\Omega \vdash_\Sigma P :: x{:}Y} \textsc{TVar}_R \qquad \frac{Y = A \in \Sigma \qquad \Omega, x{:}A \vdash_\Sigma P :: z{:}C}{\Omega, x{:}Y \vdash_\Sigma P :: z{:}C} \textsc{TVar}_L$$

$$\frac{}{\Vdash_\Sigma (\cdot)\ \mathbf{sig}} \Sigma_1 \qquad \frac{\Vdash_\Sigma A\ \mathbf{contr} \qquad \Vdash_\Sigma \Sigma'\ \mathbf{sig}}{\Vdash_\Sigma Y = A, \Sigma'\ \mathbf{sig}} \Sigma_2 \qquad \frac{\Omega \vdash_\Sigma P :: x{:}A \qquad \Vdash_\Sigma \Sigma'\ \mathbf{sig}}{\Vdash_\Sigma \Omega \vdash_\Sigma X = P :: x{:}A, \Sigma'\ \mathbf{sig}} \Sigma_3$$

Fig. 7: Process term typing rules and signature checking rules of SESSION.

B.2 Configuration typing

Fig. 8 summarizes the typing rules for process configurations and messages in SESSION. Since Σ is fixed, we may drop it in a configuration typing judgment and a process typing judgment, respectively, for brevity.

We introduce the judgments $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$, $\mathcal{D} \in \mathsf{Tree}(\Delta \Vdash K)$, and $\mathcal{B} \in \mathsf{Forest}(\Delta \Vdash \Delta')$ to denote well-typed configurations, defined as follows:

Definition 11 (Well-typed configuration). *Well-typed configuration(s) are defined in terms of the judgments $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $\mathcal{D} \in \mathsf{Tree}(\Delta \Vdash K)$, where K is either of the form $x{:}A$ or $\_{:}1$ and $\_$ stands for an arbitrary channel name along which no observations are made.*

– *We define $\mathcal{D} \in \mathsf{Tree}(\Delta \Vdash K)$ as*

$$\Delta \Vdash \mathcal{D} :: K$$

– *We define $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ as $\mathcal{D}_1 \in \mathsf{Tree}(\Delta \Vdash K)$ and $\mathcal{D}_2 \in \mathsf{Tree}(\Delta \Vdash K)$*

$$\frac{}{x_\alpha{:}A \Vdash \cdot :: x_\alpha{:}A}\ \mathbf{emp_1} \qquad \frac{}{\cdot \Vdash \cdot :: \cdot}\ \mathbf{emp_2} \qquad \frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0' \Vdash \mathcal{C}_1 :: x_\alpha{:}A}{\Delta_0, \Delta_0' \Vdash \mathcal{C}\, \mathcal{C}_1 :: \Delta, x_\alpha{:}A}\ \mathbf{comp}$$

$$\frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0', \Delta, x_\alpha{:}A \vdash \delta :: \Omega_0', \Omega, x{:}A \qquad \Omega_0', \Omega \vdash_\Sigma P :: x{:}A}{\Delta_0, \Delta_0' \Vdash \mathcal{C}\ \mathbf{proc}(x_\alpha, \hat{\delta}(P)) :: x_\alpha{:}A}\ \mathbf{proc}$$

$$\frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0', \Delta \vdash M :: x_\alpha{:}A}{\Delta_0, \Delta_0' \Vdash \mathcal{C}\ \mathbf{msg}(M) :: x_\alpha{:}A}\ \mathbf{msg} \qquad \frac{k \in L}{y_{\alpha+1}{:}A_k \vdash y_\alpha.k :: y_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}}\ \mathbf{m}_{\oplus R}$$

$$\frac{k \in L}{x_\alpha{:}\&\{\ell : A_\ell\}_{\ell \in I} \vdash x_\alpha.k :: x_{\alpha+1}{:}A_k}\ \mathbf{m}_{\& L} \qquad \frac{}{z_\beta{:}A, y_{\alpha+1}{:}B \vdash \mathbf{send}\, z_\beta\, y_\alpha :: y_\alpha{:}A \otimes B}\ \mathbf{m}_{\otimes R}$$

$$\frac{}{z_\beta{:}A, x_\alpha{:}A \multimap B \vdash \mathbf{send}\, z_\beta\, x_\alpha :: x_{\alpha+1}{:}B}\ \mathbf{m}_{\multimap L} \qquad \frac{}{\cdot \vdash \mathbf{close}\, y_\alpha :: y_\alpha{:}1}\ \mathbf{m}_{1R}$$

Fig. 8: Configuration typing rules of SESSION.

– *We define* $\mathcal{B} \in \mathsf{Forest}(\Delta \Vdash \Delta')$ *as*

$$\Delta \Vdash \mathcal{B} :: \Delta'$$

– *We define the notation* $\mathcal{T} \in \mathcal{B}$ *meaning* $\mathcal{T}$ *is a particular tree in a forest of trees* $\mathcal{B}$*: For* $\mathcal{B} \in \mathsf{Forest}(\Delta \Vdash \Delta')$*, we write* $\mathcal{T} \in \mathcal{B}$ *iff* $\mathcal{B} = \mathcal{B}'\mathcal{T}$*, and* $\mathcal{B}' \in \mathsf{Forest}(\Delta_1 \Vdash \Delta_1')$ *and* $\mathcal{T}' \in \mathsf{Tree}(\Delta_2 \Vdash K)$ *with* $\Delta = \Delta_1, \Delta_2$ *and* $\Delta' = \Delta_1', K$*.* ◇

B.3 Asynchronous dynamics

Fig. 9 gives the asynchronous dynamics of SESSION as multiset rewriting rules [12].

C Concrete security lattice and security theories

Process configurations and terms are typed relative to a concrete security lattice and a security theory. A *concrete* lattice Ψ_0 is defined globally for an application and consists of concrete security levels ι. Our running example lattice

$$\mathbf{guest} \sqsubseteq \mathbf{alice} \sqsubseteq \mathbf{bank} \qquad \mathbf{guest} \sqsubseteq \mathbf{bob} \sqsubseteq \mathbf{bank}$$

is an example of a concrete security lattice. Polymorphic process definitions make use of a *security theory* Ψ, ranging over security variables ψ and concrete security levels ι from the given concrete security lattice Ψ_0. At run-time, all security variables occurring in polymorphic processes will be replaced with concrete security levels.

Spawn	$\mathbf{proc}(y_\alpha, (x \leftarrow X[\gamma] \leftarrow \Delta_1); Q) \mapsto$	$(\Omega_1' \vdash X = P :: x' : B \in \Sigma)$
	$\mathbf{proc}(x_0, \hat{\gamma}([x_0/x]P))\ \mathbf{proc}(y_\alpha, [x_0/x]Q)$	$(\Omega_1' \Vdash \gamma : \Delta_1,\ x_0\, \mathit{fresh})$
D-Fwd	$\mathbf{proc}(y_\alpha, F_Y[\gamma]) \mapsto$	$(x' : C \vdash F_Y = \mathtt{Fwd}_{C,y' \leftarrow x'} :: y' : C \in \Sigma)$
	$\mathbf{proc}(y_\alpha, \mathtt{Fwd}_{C,y_\alpha \leftarrow x_\beta})$	$(x', y' \Vdash \gamma : x_\beta, y_\alpha)$
1_{snd}	$\mathbf{proc}(y_\alpha, \mathbf{close}\, y_\alpha) \mapsto \mathbf{msg}(\mathbf{close}\, y_\alpha)$	
1_{rcv}	$\mathbf{msg}(\mathbf{close}\, y_\alpha)\ \mathbf{proc}(x_\beta, \mathbf{wait}\, y_\alpha; Q) \mapsto \mathbf{proc}(x_\beta, Q)$	
$\oplus_{\mathsf{snd}}$	$\mathbf{proc}(y_\alpha, y_\alpha.k; P) \mapsto \mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P)\ \mathbf{msg}(y_\alpha.k)$	
$\oplus_{\mathsf{rcv}}$	$\mathbf{msg}(y_\alpha.k)\ \mathbf{proc}(u_\eta, \mathbf{case}\, y_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L}) \mapsto \mathbf{proc}(u_\eta, [y_{\alpha+1}/y_\alpha]P_k)$	
$\&_{\mathsf{snd}}$	$\mathbf{proc}(y_\alpha, x_\beta.k; P) \mapsto \mathbf{msg}(x_\beta.k)\ \mathbf{proc}(y_\alpha, [x_{\beta+1}/x_\beta]P)$	
$\&_{\mathsf{rcv}}$	$\mathbf{proc}(y_\alpha, \mathbf{case}\, y_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L})\ \mathbf{msg}(y_\alpha.k) \mapsto \mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P_k)$	
$\otimes_{\mathsf{snd}}$	$\mathbf{proc}(y_\alpha, \mathbf{send}\, x_\beta\, y_\alpha; P) \mapsto \mathbf{proc}(y_{\alpha+1}, [y_{\alpha+1}/y_\alpha]P)\ \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)$	
$\otimes_{\mathsf{rcv}}$	$\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)\ \mathbf{proc}(u_\eta, w \leftarrow \mathbf{recv}\, y_\alpha; P) \mapsto \mathbf{proc}(u_\eta, [x_\beta/w][y_{\alpha+1}/y_\alpha]P)$	
$\multimap_{\mathsf{snd}}$	$\mathbf{proc}(y_\alpha, \mathbf{send}\, x_\beta\, u_\eta; P) \mapsto \mathbf{msg}(\mathbf{send}\, x_\beta\, u_\eta)\ \mathbf{proc}(y_\alpha, [u_{\gamma+1}/u_\gamma]P)$	
$\multimap_{\mathsf{rcv}}$	$\mathbf{proc}(y_\alpha, w \leftarrow \mathbf{recv}\, y_\alpha; P)\ \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \mapsto \mathbf{proc}(y_{\alpha+1}, [x_\beta/w][y_{\alpha+1}/y_\alpha]P)$	

Fig. 9: Asynchronous dynamics of SESSION.

Definition 12 (Concrete security lattice and security theory).
Let $\Psi_0 \triangleq \langle \mathcal{L}, \mathcal{E}_0, \sqcup \rangle$ be a concrete join semi-lattice with a partial order $\mathcal{E}_0$ over concrete security levels $\iota, \eta \in \mathcal{L}$ such that $E_0 \in \mathcal{E}_0$ is of the form $\iota \sqsubseteq \iota'$. We define a security theory $\Psi \triangleq \langle \mathcal{V}, \mathcal{E} \rangle$ that augments Ψ_0 with security variables ψ and relations $\mathcal{E}$ over them such that
- *$\psi \in \mathcal{V}$ is a set of security variables*
- *$c, d \in \mathcal{S}$ is a set of security terms of the security theory Ψ, defined by the grammar*
$$c, d := c \sqcup d \mid \iota \mid \psi$$
- *$E \in \mathcal{E}$ is a set of relations over the elements c, d of the security theory where $E = c \sqsubseteq d$.*

For convenience, we define the projection $\mathcal{E}(\Psi)$ to extract the relations $\mathcal{E}$ of Ψ. We write
$$\Psi \Vdash E$$
if E is consequence of the lattice theory based on the relations. The judgment is defined in Fig. 10.

Process definitions and process spawning rely on an order-preserving substitution, defined as follows:

Definition 13 (Order-preserving substitution).
Let $\gamma \in \mathcal{V} \to \mathcal{S}$ be a total function that maps security variables to security terms. Its lifting $\hat{\gamma}$ to other syntactic objects, such as security terms c, process terms P, typing contexts Δ, and security lattices Ψ, is defined by structural induction over the syntactic object, replacing simultaneously all variable occurrences ψ_i in the object with $\gamma(\psi_i)$. The function γ and its lifting $\hat{\gamma}$ must be

$$\frac{c \sqsubseteq d \in \mathcal{E}_0}{\Psi \Vdash c \sqsubseteq d} \qquad \frac{c \sqsubseteq d \in \mathcal{E}(\Psi)}{\Psi \Vdash c \sqsubseteq d} \qquad \frac{}{\Psi \Vdash c \sqsubseteq c} \qquad \frac{\Psi \Vdash c_1 \sqsubseteq c_2 \qquad \Psi \Vdash c_2 \sqsubseteq c_3}{\Psi \Vdash c_1 \sqsubseteq c_3}$$

$$\frac{\Psi \Vdash c \sqsubseteq d' \qquad \Psi \Vdash d \sqsubseteq d'}{\Psi \Vdash c \sqcup d \sqsubseteq d'} \qquad \frac{}{\Psi \Vdash c \sqsubseteq c \sqcup d} \qquad \frac{}{\Psi \Vdash d \sqsubseteq c \sqcup d}$$

Fig. 10: Inductive definition of $\Psi \Vdash E$.

order-preserving, ensuring that if $\Psi' \Vdash E$, then $\hat{\gamma}(\Psi') \Vdash \hat{\gamma}(E)$. To link a spawner and a spawnee, we define an order-preserving substitution $\Psi \Vdash \gamma : \Psi'$

$$\Psi \Vdash \gamma : \Psi' \triangleq \mathit{if}\, \Psi' \Vdash E, \mathit{then}\, \Psi \Vdash \hat{\gamma}(E)$$

Composition $\gamma \circ \gamma'$ of two substitutions γ and γ' is defined as usual. We observe that if both γ and γ' are order-preserving so is their composition.

The type system requires every spawner to provide an order-preserving substitution $\Psi \Vdash \gamma : \Psi'$ for the security variables of the spawnee (rule Spawn). As a result, the security theory Ψ' of the spawnee must be satisfied by the security theory Ψ of the spawner, i.e., if $\Psi' \Vdash E$, then $\Psi \Vdash \hat{\gamma}(E)$. Moreover, if the spawner provides a substitution δ for Ψ_0, i.e., $\Psi_0 \Vdash \delta : \Psi$, so does the spawnee, i.e., $\Psi_0 \Vdash \gamma \circ \delta : \Psi'$.

D IFC refinement type system

This section develops an Information Flow Control (IFC) refinement type system for ILLST, yielding the language CONSESSION. Deviating from the main text, we use here the metavariable Ξ instead of Γ to denote the security typing context.

D.1 Process term typing

Fig. 11 summarizes the typing rules for process terms in CONSESSION.

Rule Spawn relies on two substitutions, γ_{var} that provides a substitution for channel variables to match them up with the definitions in the signature, and γ_{sec} which is an order-preserving substitution $\Psi \Vdash \gamma_{\mathsf{sec}} : \Psi'$, guaranteeing that the security terms provided by the spawner comply with the order expected among those terms by the spawnee. Rule Spawn moreover establishes the invariants necessary to prevent information leakage for the newly spawned process, by the premise $\Psi \Vdash \hat{\gamma}(\psi) \sqsubseteq d$, and allows the newly spawned process to start at least at the spawner's running secrecy d_0, by the premise $d_0 \sqsubseteq \Psi \Vdash \hat{\gamma}(\psi)$.

Rule D-Fwd in Fig. 11 relies on a forwarder for each user-defined type definition, amounting to an identity expansion automatically generated as defined in Def. 14.

$$\oplus R\ \frac{\Psi; \Xi \vdash_\Sigma P@d_1 :: y{:}A_k[c] \qquad k \in L}{\Psi; \Xi \vdash_\Sigma (y^c.k; P)@d_1 :: y{:}\oplus\{\ell{:}A_\ell\}_{\ell \in L}[c]} \qquad \oplus L\ \frac{\Psi \Vdash d_2 = c \sqcup d_1 \qquad \Psi; \Xi, x{:}A_k[c] \vdash_\Sigma Q_k@d_2 :: y{:}C[c'] \quad \forall k \in L}{\Psi; \Xi, x{:}\oplus\{\ell : A_\ell\}_{\ell \in L}[c] \vdash_\Sigma (\mathbf{case}\, x^c(\ell \Rightarrow Q_\ell)_{\ell \in L})@d_1 :: y{:}C[c']}$$

$$\&R\ \frac{\Psi; \Xi \vdash_\Sigma Q_k@c :: y{:}A_k[c] \qquad \forall k \in L}{\Psi; \Xi \vdash_\Sigma (\mathbf{case}\, y^c(\ell \Rightarrow Q_\ell)_{\ell \in I})@d_1 :: y{:}\&\{\ell : A_\ell\}_{\ell \in L}[c]} \qquad \&L\ \frac{\Psi \Vdash d_1 \sqsubseteq c \qquad \Psi; \Xi, x{:}A_k[c] \vdash_\Sigma P@d_1 :: y{:}C[c'] \qquad k \in L}{\Psi; \Xi, x{:}\&\{\ell : A_\ell\}_{\ell \in I}[c] \vdash_\Sigma (x^c.k; P)@d_1 :: y{:}C[c']}$$

$$\otimes R\ \frac{\Psi; \Xi \vdash_\Sigma P@d_1 :: y{:}B[c]}{\Psi; \Xi, z{:}A[c] \vdash_\Sigma (\mathbf{send}\, z\, y^c; P)@d_1 :: y{:}A \otimes B[c]} \qquad \otimes L\ \frac{\Psi \Vdash d_2 = c \sqcup d_1 \qquad \Psi; \Xi, z{:}A[c], x{:}B[c] \vdash_\Sigma P@d_2 :: y{:}C[c']}{\Psi; \Xi, x{:}A \otimes B[c] \vdash_\Sigma (z \leftarrow \mathbf{recv}\, x^c; P)@d_1 :: y{:}C[c']}$$

$$\multimap R\ \frac{\Psi; \Xi, z{:}A[c] \vdash_\Sigma P@c :: y{:}B[c]}{\Psi; \Xi \vdash_\Sigma (z \leftarrow \mathbf{recv}\, y^c; P)@d_1 :: y{:}A \multimap B[c]} \qquad \multimap L\ \frac{\Psi \Vdash d_1 \sqsubseteq c \qquad \Psi; \Xi, x{:}B[c] \vdash_\Sigma P@d_1 :: y{:}C[c']}{\Psi; \Xi, z{:}A[c], x{:}A \multimap B[c] \vdash_\Sigma (\mathbf{send}\, z\, x^c; P)@d_1 :: y{:}C[c']}$$

$$\frac{}{\Psi; \cdot \vdash_\Sigma (\mathbf{close}\, y^c)@d_1 :: y : 1[c]}\ 1R \qquad \frac{\Psi; \Xi \vdash_\Sigma Q@d_1 :: y : C[d]}{\Psi; \Xi, x : 1[c] \vdash_\Sigma (\mathbf{wait}\, x^c; Q)@d_1 :: y : C[d]}\ 1L$$

$$\frac{Y = A \in \Sigma \qquad \Psi; \Xi \vdash_\Sigma P :: x{:}A[c]}{\Psi; \Xi \vdash_\Sigma P :: x{:}Y[c]}\ \textsc{TVar}_R \qquad \frac{Y = A \in \Sigma \qquad \Psi; \Xi, x{:}A[c] \vdash_\Sigma P :: z{:}C[c']}{\Psi; \Xi, x{:}Y[c] \vdash_\Sigma P :: z{:}C[c']}\ \textsc{TVar}_L$$

$$\frac{\begin{array}{c}\Psi'; \Xi_1' \vdash_\Sigma X = P@\psi_0 :: x'{:}A[\psi] \in \Sigma \qquad \Psi \Vdash \gamma_{\mathsf{sec}} : \Psi' \qquad \Psi \Vdash \hat{\gamma_{\mathsf{sec}}}(\psi) \sqsubseteq d \\ \Psi \Vdash d_0 \sqsubseteq \hat{\gamma_{\mathsf{sec}}}(\psi_0) \qquad \Xi_1, x{:}A[\hat{\gamma_{\mathsf{sec}}}(\psi_0)] \Vdash (\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}}) :: \Xi_1', x'{:}A[\psi] \\ \Psi; \Xi_2, x{:}A[\hat{\gamma_{\mathsf{sec}}}(\psi)] \vdash_\Sigma Q@d_0 :: z{:}C[d]\end{array}}{\Psi; \Xi_1, \Xi_2 \vdash_\Sigma ((x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[(\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})] \leftarrow \Xi_1)@\hat{\gamma_{\mathsf{sec}}}(\psi_0); Q)@d_0 :: z{:}C[d]}\ \textsc{Spawn}$$

$$\frac{\begin{array}{c}\psi = \psi; z_1 : C[\psi] \vdash_\Sigma F_Y = \mathtt{Fwd}_{C, x_1^\psi \leftarrow z_1^\psi}@\psi :: x_1{:}C[\psi] \in \Sigma \qquad Y = A \in \Sigma \\ \Psi \Vdash \gamma_{\mathsf{sec}} : \psi = \psi \qquad z{:}C[c], y{:}C[c] \Vdash (\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}}) :: z_1{:}C[\psi], x_1{:}C[\psi]\end{array}}{\Psi; z{:}C[c] \vdash_\Sigma F_Y[(\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})]@c :: y{:}C[c]}\ \textsc{D-Fwd}$$

$$\frac{}{\Vdash_{\Sigma;\Psi_0} (\cdot)\ \mathbf{sig}}\ \Sigma_1 \qquad \frac{\Vdash_{\Sigma;\Psi_0} A\ \mathbf{contr} \qquad \Vdash_{\Sigma;\Psi_0} \Sigma'\ \mathbf{sig}}{\Vdash_{\Sigma;\Psi_0} Y = A, \Sigma'\ \mathbf{sig}}\ \Sigma_2$$

$$\frac{\begin{array}{c}\forall i \in \{1 \ldots n\}. \Psi \Vdash \psi_i \sqsubseteq \psi \\ \Psi \Vdash \psi_0 \sqsubseteq \psi \qquad \Psi; y_1{:}B_1[\psi_1], \ldots, y_n{:}B_n[\psi_n] \vdash_\Sigma P@\psi_0 :: x{:}A[\psi] \qquad \Vdash_{\Sigma;\Psi_0} \Sigma'\ \mathbf{sig}\end{array}}{\Vdash_{\Sigma;\Psi_0} \Psi; y_1{:}B_1[\psi_1], \ldots, y_n{:}B_n[\psi_n] \vdash_\Sigma X = P@\psi_0 :: x{:}A[\psi], \Sigma'\ \mathbf{sig}}\ \Sigma_3$$

Fig. 11: Process term typing rules and signature checking rules of CONSESSION.

Definition 14. *For all $Y = A \in \Sigma$, we extend Σ by adding the following IFC-typed forwarder definition to the signature Σ:*

$$\psi = \psi; x : A[\psi] \vdash F_A = \mathtt{Fwd}_{A, y^\psi \leftarrow x^\psi}@\psi :: y : A[\psi] \qquad Y = A \in \Sigma$$

Here F_Y is a specific process variable assigned to the forwarder process for type variable Y, and $\texttt{Fwd}_{A,y\leftarrow x}$ is defined similar to Def. 10 as

$$
\begin{array}{lll}
\texttt{Fwd}_{\oplus\{\ell:A_\ell\}_{\ell\in L},y\leftarrow x} & := \textbf{case}\, x(\ell \Rightarrow y.\ell; \texttt{Fwd}_{A_\ell,y^c\leftarrow x^c})_{\ell\in L} & \\
\texttt{Fwd}_{\&\{\ell:A_\ell\}_{\ell\in L},y^c\leftarrow x^c} & := \textbf{case}\, y(\ell \Rightarrow x.\ell; \texttt{Fwd}_{A_\ell,y^c\leftarrow x^c})_{\ell\in L} & \\
\texttt{Fwd}_{A\otimes B,y^c\leftarrow x^c} & := w \leftarrow \textbf{recv}\, x; \textbf{send}\, w\, y; \texttt{Fwd}_{B,y^c\leftarrow x^c} & \\
\texttt{Fwd}_{A\multimap B,y^c\leftarrow x^c} & := w \leftarrow \textbf{recv}\, y; \textbf{send}\, w\, x; \texttt{Fwd}_{B,y^c\leftarrow x^c} & \\
\texttt{Fwd}_{1,y^c\leftarrow x^c} & := \textbf{wait}\, x; \textbf{close}\, y & \\
\texttt{Fwd}_{Y;y^c\leftarrow x^c} & := F_Y[(x' \mapsto x, y' \mapsto y), (\psi \mapsto c)] & Y = A \in \Sigma
\end{array}
$$

Forwarders are well-typed by construction, as shown next:

Lemma 12. *Given the extended signature Σ, for all type A, there are derivations for*

(i) $\Psi; x : A[\psi] \vdash_\Sigma \texttt{Fwd}_{A,y^\psi\leftarrow x^\psi}@\psi :: y : A[\psi]$*, and*

(ii) $\Psi; x : A[\psi] \vdash_\Sigma \texttt{Fwd}_{A,y^\psi\leftarrow x^\psi}@\psi' :: y : A[\psi]$*, when* $\Psi \Vdash \psi' \sqsubseteq \psi$ *and* $A \neq Y$ *for a type variable Y.*

Proof. (i) The proof is by induction on the structure of type A. In a base case, where A is a type variable Y, i.e., $A = Y$ for $Y = C \in \Sigma$, the proof is straightforward by the D-Fwd rule since $\psi = \psi; x : C[\psi] \vdash F_Y := \texttt{Fwd}_{C,y^\psi\leftarrow x^\psi}@\psi :: y : A[\psi] \in \Sigma$, and the substitutions γ_{sec} and γ_{var} enforce the premises of the rule. The proof of other cases is straightforward.

(ii) The proof is by case analysis on the structure of type A. In all cases, by the way we defined the process terms, after the very first application of a rule (the first communication which is always a receive), the running secrecy increases to ψ, and we can apply the previous item. Note that by the tree invariant every judgment in our typing derivation satisfies the condition $\Psi \Vdash \psi' \sqsubseteq \psi$.

D.2 Configuration typing

Fig. 12 summarizes the typing rules for process configurations and messages in CONSESSION. Since Ψ_0 and Σ are fixed, we may drop Ψ_0 and Σ in a configuration typing judgment and a process typing judgment, respectively, for brevity.

D.3 Security erasure

We define erasure of typing contexts, signatures, and configurations to state noninterference as a logical equivalence between SESSION configurations. To this end, Thm. 3 proves that well-typed CONSESSION configurations are well-typed SESSION configurations.

Definition 15. *Define security-erasure $|\Sigma|$ of the signature Σ as*

$$
\begin{array}{ll}
|\cdot| & = \cdot \\
|Y = A, \Sigma'| & = Y = A, |\Sigma'| \\
|\Psi; \Xi \vdash X = P@\psi_0 :: x{:}A[\psi],\ \Sigma'| & = |\Xi| \vdash X = P :: x{:}A\,, |\Sigma'|
\end{array}
$$

$$\frac{}{\Psi_0; x{:}A[d] \Vdash \cdot :: (x{:}A[d])}\ \mathbf{emp_1} \qquad \frac{}{\Psi_0; \cdot \Vdash \cdot :: (\cdot)}\ \mathbf{emp_2} \qquad \frac{\Psi_0; \Gamma_0 \Vdash \mathbb{C} :: \Gamma \qquad \Psi_0; \Gamma_0' \Vdash \mathcal{C}_1 :: x{:}A[d]}{\Psi_0; \Gamma_0, \Gamma_0' \Vdash \mathbb{C}, \mathcal{C}_1 :: \Gamma, x{:}A[d]}\ \mathbf{comp}$$

$$\frac{\begin{array}{c}\Psi_0 \Vdash d_1 \sqsubseteq d \qquad \forall y{:}B[d'] \in \Gamma_0', \Gamma\ (\Psi_0 \Vdash d' \sqsubseteq d)\\ \Psi_0; \Gamma_0 \Vdash \mathbb{C} :: \Gamma \\ \Gamma_0', \Gamma, x_\alpha{:}A[d] \vdash \delta :: \Xi_0', \Xi, x{:}A[d] \qquad \Psi_0; \Xi_0', \Xi \vdash_\Sigma P@d_1 :: (x{:}A[d])\end{array}}{\Psi_0; \Gamma_0, \Gamma_0' \Vdash \mathbb{C}, \mathbf{proc}(x[d], P@d_1) :: (x{:}A[d])}\ \mathbf{proc}$$

$$\frac{\forall y{:}B[d'] \in \Gamma_0', \Gamma\ (\Psi_0 \Vdash d' \sqsubseteq d) \qquad \Psi_0; \Gamma_0 \Vdash \mathbb{C} :: \Gamma \qquad \Psi_0; \Gamma_0', \Gamma \vdash M :: (x{:}A[d])}{\Psi_0; \Gamma_0, \Gamma_0' \Vdash \mathbb{C}, \mathbf{msg}(M) :: (x{:}A[d])}\ \mathbf{msg}$$

$$\frac{k \in L}{\Psi_0; y_{\alpha+1}{:}A_k[c] \vdash y_\alpha.k :: y_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}[c]}\ \mathbf{m}_{\oplus R} \qquad \frac{k \in L}{\Psi_0; x_\alpha{:}\&\{\ell : A_\ell\}_{\ell \in I}[c] \vdash (x_\alpha.k; P) :: x_{\alpha+1}{:}A_k[c]}\ \mathbf{m}_{\& L}$$

$$\frac{}{\Psi_0; z_\beta{:}A[c], y_{\alpha+1}{:}B[c] \vdash \mathbf{send}\, z_\beta\, y_\alpha :: y_\alpha{:}A \otimes B[c]}\ \mathbf{m}_{\otimes R}$$

$$\frac{}{\Psi_0; z_\beta{:}A[c], x_\alpha{:}A \multimap B[c] \vdash \mathbf{send}\, z_\beta\, x_\alpha :: x_{\alpha+1}{:}B[c]}\ \mathbf{m}_{\multimap L} \qquad \frac{}{\Psi_0; \cdot \vdash \mathbf{close}\, y_\alpha :: y_\alpha{:}1[c]}\ \mathbf{m}_{1R}$$

Fig. 12: Configuration typing rules of CONSESSION.

Definition 16. *Define security-erasures $|\Xi|$, $|x{:}A[c]|$, $|\Gamma|$, and $|K^s|$ of the security linear variable context Ξ, security channel variable, security linear channel context Γ, and security channel singleton K^s, respectively, as*

$$\begin{array}{ll}
|\Xi, x{:}A[c]| & \stackrel{\text{def}}{=} |\Xi|, x{:}A\\
|x{:}A[c]| & \stackrel{\text{def}}{=} x{:}A\\
|\Gamma, x_\alpha{:}A[c]| & \stackrel{\text{def}}{=} |\Gamma|, x_\alpha{:}A\\
|\cdot| & \stackrel{\text{def}}{=} \cdot\\
|x_\alpha{:}A[c]| & \stackrel{\text{def}}{=} x_\alpha{:}A\\
|_{:}1[\top]| & \stackrel{\text{def}}{=} _{:}1
\end{array}$$

Definition 17. *Define a security-erasure $|\mathbb{C}|$ of the configuration $\mathbb{C}$ as*

$$\begin{array}{ll}
|\cdot| & = \cdot\\
|\mathbb{C}, \mathbf{proc}(x[d], P@d_1)| & = |\mathbb{C}|, \mathbf{proc}(x, P)\\
|\mathbb{C}, \mathbf{msg}(M)| & = |\mathbb{C}|, \mathbf{msg}(M)
\end{array}$$

Theorem 3. *Every IFC well-typed configuration $\Psi_0; \Gamma \Vdash \mathbb{C} :: K^s$ is session-typed $|\mathbb{C}| \in \mathsf{Tree}(|\Gamma| \Vdash |K^s|)$.*

Proof. By induction on the derivation of $\Psi_0; \Gamma \Vdash \mathbb{C} :: K^s$.

E Progress and preservation

This section proves type safety, first for SESSION and then for CONSESSION.

E.1 Session-typed processes

This section proves type safety of session-typed processes in SESSION. We first start with defining the notion of a poised configuration and proofs of necessary lemmas.

Poised configuration and configuration permutation Progress relies on the notion of a *poised* configuration. A configuration is poised if it is empty or cannot take any internal steps but wants to engage in a message exchange along any of its free channels.

Definition 18 (Poised Configuration). *A configuration $\Delta_1, \Delta_2 \Vdash \mathcal{C}_1, \mathcal{C}_2 :: \Lambda, w{:}A'$ is poised iff either $\mathcal{C}_1, \mathcal{C}_2$ is empty or $\Delta_1 \Vdash \mathcal{C}_1 :: \Lambda$ is poised and $\Delta_2 \Vdash \mathcal{C}_2 :: w{:}A'$ is poised. The configuration $\Delta_2 \Vdash \mathcal{C}_2 :: w{:}A'$ is poised iff it cannot take any steps and at least one of the following conditions hold:*

1. *$\mathcal{C}_2$ is an empty configuration.*
2. *$\mathcal{C}_2 = \mathcal{C}_2' \, \mathbf{msg}(M) \, \mathcal{C}_2''$ such that $\mathbf{msg}(M)$ is a negative message along $y_\alpha \in \Delta_2$, i.e. $y_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell \in L} \Vdash \mathbf{msg}(M) :: y_{\alpha+1}{:}A_k$ or $y_\alpha{:}A \multimap B, z_\beta{:}A \Vdash \mathbf{msg}(M) :: y_{\alpha+1}{:}B$, and both sub-configurations $\mathcal{C}_2'$ and $\mathcal{C}_2''$ are poised.*
3. *$\mathcal{C}_2 = \mathcal{C}_2' \, \mathbf{proc}(x, P) \, \mathcal{C}_2''$ such that $\mathbf{proc}(x, P)$ attempts to receive along a channel $y_\alpha {\in} \Delta_2$, and both sub-configurations $\mathcal{C}_2'$ and $\mathcal{C}_2''$ are poised.*
4. *$\mathcal{C}_2 = \mathcal{C}_2' \, \mathbf{msg}(P)$ such that $\mathbf{msg}(M)$ is a positive message sent along $w_\beta{:}A'$, i.e. $w_{\beta+1}{:}A_k \Vdash \mathbf{msg}(M) :: w_\beta{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}$ or $w_{\beta+1}{:}B, z_\gamma{:}A \Vdash \mathbf{msg}(M) :: w_\beta{:}A \otimes B$, or $\cdot \Vdash \mathbf{msg}(M) :: w{:}1$, and sub-configuration $\mathcal{C}_2'$ is poised.*
5. *$\mathcal{C}_2 = \mathcal{C}_2' \, \mathbf{proc}(w, P)$ such that $\mathbf{proc}(w, P)$ attempts to receive along $w{:}A'$, and sub-configuration $\mathcal{C}_2'$ is poised.*

◇

The dynamics (see Fig. 9) is expressed in terms of multiset rewriting rules, which update a configuration locally, without regard for the remaining configuration. As a result, the updated configuration may not necessarily be well-typed, according to the rules in the configuration typing. For example, stepping the configuration

$$\mathcal{C}_1 \mathcal{T} \ \mathbf{proc}(y_\alpha, (\mathbf{send}\, x_\beta \, y_\alpha; P)) \ \mathcal{C}_2 \ \mapsto$$

using rule $\otimes_{\mathsf{snd}}$ (see Fig. 9), yields the configuration

$$\mathcal{C}_1 \mathcal{T} \ \mathbf{proc}(y_{\alpha+1}, ([y_{\alpha+1}/y_\alpha]P)) \ \mathbf{msg}(\mathbf{send}\, x_\beta \, y_\alpha) \ \mathcal{C}_2$$

For well-typedness, the subtree $\mathcal{T}$ rooted at the message $\mathbf{msg}(\mathbf{send}\, x_\beta \, y_\alpha)$ would have to be moved left to the message. Our proofs account for this possibility and only require that the dynamics yield a valid permutation of a well-typed configuration, assuming that the pre-state is a valid permutation of well-typed configuration as well. We define the notion of a valid permutation next. A valid permutation may rearrange the order of processes and messages in a configuration as long as parent-child relationships are preserved.

Definition 19 (Valid configuration permutation).

For a well-typed configuration $\Psi_0; \Delta \Vdash \mathcal{C} :: \Delta'$, a valid permutation $\mathcal{P}(\mathcal{C})$ can be derived by simultaneously changing the position of a process or message in $\mathcal{C}$, yielding the permutation $\mathcal{C}'$, i.e., $\mathcal{P}(\mathcal{C}) = \mathcal{C}'$, as long as the following conditions are met:

1. *For a process $\mathbf{proc}(z_\alpha, P)$ in $\mathcal{C}'$ such that $\mathcal{C}' = \mathcal{C}_1 \mathbf{proc}(z_\alpha, P)\mathcal{C}_2$ and for all $y_\beta \notin \mathsf{dom}(\Delta)$ that P is using, there must exist either a process $\mathbf{proc}(y_\beta, \_)$ or a message $\mathbf{msg}(\_\,\langle y_\beta\rangle)$ in $\mathcal{C}_1$.*
2. *For a positive message $\mathbf{msg}(M\langle z_\alpha\rangle)$ in $\mathcal{C}'$ such that $\mathcal{C}' = \mathcal{C}_1 \mathbf{msg}(M\langle z_\alpha\rangle)\mathcal{C}_2$, if $z_\alpha \notin \mathsf{dom}(\Delta)$, there must exist either a process $\mathbf{proc}(z_\alpha, \_)$ or a message $\mathbf{msg}(\_\,\langle z_\alpha\rangle)$ in $\mathcal{C}_1$. Moreover, for all $y \notin \mathsf{dom}(\Delta)$ that P is using, there must exist either a process $\mathbf{proc}(y, \_)$ or a message $\mathbf{msg}(y.\_)$ in $\mathcal{C}_1$.*
3. *For a negative message $\mathbf{msg}(P\langle v_\beta\rangle)$ in $\mathcal{C}'$ such that $\mathcal{C}' = \mathcal{C}_1 \mathbf{msg}(P\langle v_\beta\rangle)\mathcal{C}_2$, if $v_\beta \notin \mathsf{dom}(\Delta)$, there must exist either a process $\mathbf{proc}(v_\beta, \_)$ or a message $\mathbf{msg}(\_\,\langle w\rangle)$ in $\mathcal{C}_1$. Moreover, for all $y_\alpha \notin \mathsf{dom}(\Delta)$ that P is using, there must exist either a process $\mathbf{proc}(y_\alpha, \_)$ or a message $\mathbf{msg}(y_\alpha.\_)$ in $\mathcal{C}_1$.*

◇

Lemmas

Lemma 13 (Term variable substitution). *The following substitutions are type-preserving and thus admissible:*

1. *If $\Delta \vdash_\Sigma P :: x : A$ then, for any fresh $y : A$, we have $\Delta \vdash_\Sigma [y/x]P :: y : A$.*
2. *If $\Delta, y : B \vdash_\Sigma P :: x : A$ then, for any fresh $z : B$, we have $\Delta, z : B \vdash_\Sigma [z/y]P :: x : A$.*

Proof. The proof is by induction on the process term typing rules.

The next lemma allows us to break up an open forest into two sub-forests, as illustrated in Fig. 13.

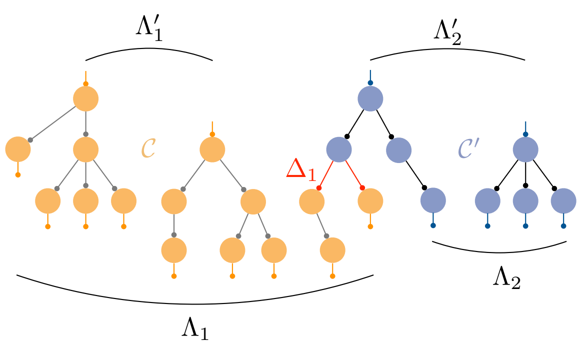


Fig. 13: Schematic illustration of Lem. 14, allowing us to break up an open forest into two open sub-forests.

Lemma 14 (Making two forests out of one). *If $\Delta \Vdash \mathcal{C}\,\mathcal{C}' :: \Delta'$, then for some Δ_1 we have $\Lambda_1 \Vdash \mathcal{C} :: \Lambda_1', \Delta_1$ and $\Lambda_2, \Delta_1 \Vdash \mathcal{C}' :: \Lambda_2'$, where $\Delta = \Lambda_1, \Lambda_2$ and $\Delta' = \Lambda_1', \Lambda_2'$.*

Proof. The proof is by a straightforward induction on the configuration typing rules.

Progress

Theorem 4 (Progress).
For any configuration $\mathcal{C}$, if $\mathcal{C}$ is a valid permutation of a configuration $\mathcal{C}''$ such that $\Delta \Vdash \mathcal{C}'' :: \Delta'$, then either $\mathcal{C} \mapsto \mathcal{C}'$ or $\mathcal{C}$ is poised.

Proof. The proof is standard and can be found in the literature of intuitionistic linear session types.

Preservation

Theorem 5 (Preservation). *For any configuration $\mathcal{C}$ that is a valid permutation of a configuration $\mathcal{C}''$ such that $\Delta \Vdash \mathcal{C}'' :: \Delta'$, if $\mathcal{C} \mapsto \mathcal{C}'$, then $\mathcal{C}'$ is a valid permutation of a configuration $\mathcal{C}'''$ such that $\Delta \Vdash \mathcal{C}''' :: \Delta'$.*

Proof. The proof is standard and can be found in the literature for intuitionistic linear session types.

E.2 IFC-typed processes

This section proves type safety of CONSESSION.

Lemma 15 (Term variable substitution). *The following substitutions are type-preserving and thus admissible:*
1. *If $\Psi; \Xi \vdash_\Sigma P@c :: x : A[c']$ with the tree invariant satisfied, then, for any fresh $y : A$, we have $\Psi; \Xi \vdash_\Sigma [y/x]P@c :: y : A[c']$ with the tree invariant still satisfied.*
2. *If $\Psi; \Xi, y : B[c''] \vdash_\Sigma P@c :: x : A[c']$ with the tree invariant satisfied, then, for any fresh $z : B$, we have $\Psi; \Xi, z : B[c''] \vdash_\Sigma [z/y]P@c :: x : A[c']$ with the tree invariant still satisfied.*

Lemma 16 (Making two forests out of one). *If $\Psi_0; \Gamma \Vdash \mathbb{C}\,\mathbb{C}' :: \Gamma'$, then for some Γ_1 we have $\Psi_0; \Gamma_1'' \Vdash \mathbb{C} :: \Gamma_1''', \Gamma_1$ and $\Psi_0; \Gamma_2'', \Gamma_1 \Vdash \mathbb{C}' :: \Gamma_2'''$, where $\Gamma = \Gamma_1'', \Gamma_2''$ and $\Gamma' = \Gamma_1''', \Gamma_2'''$.*

Proof. The proof is a straightforward induction on the configuration typing rules.

Lemma 17 (Security variable substitution). *If $\Psi; \Xi \vdash_\Sigma P@c :: x : B[d]$ with the tree invariant satisfied, then for every substitution $\Psi' \Vdash \gamma : \Psi$, we have*

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_\Sigma \hat{\gamma}(P)@\hat{\gamma}(c) :: x{:}B[\hat{\gamma}(d)],$$

with the tree invariant satisfied as well.

Proof. The proof is by induction on process term typing derivations $\Psi; \Xi \vdash_\Sigma P@c :: x : B[d]$. We consider cases for the last step in the derivation.
Case 1.

$$\frac{\Psi; \Xi \vdash_\Sigma Q_k@c :: y : A_k[c] \quad \forall k \in L}{\Psi; \Xi \vdash_\Sigma (\mathbf{case}\, y^c(\ell \Rightarrow Q_\ell)_{\ell \in L})@d_1 :: y : \&\{\ell : A_\ell\}_{\ell \in L}[c]} \ \&R$$

By the induction hypothesis, for all $k \in L$, we have

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_\Sigma \hat{\gamma}(Q_k)@\hat{\gamma}(c) :: y{:}A_k[\hat{\gamma}(c)],$$

which preserves the invariant.
By the $\&R$ rule, we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_\Sigma \mathbf{case}\, y^{\hat{\gamma}(c)}(\ell \Rightarrow \hat{\gamma}(Q_\ell))_{\ell \in L})@\hat{\gamma}(d_1) :: y : \&\{\ell : A_\ell\}_{\ell \in L}[\hat{\gamma}(c)].$$

By the first part of the tree invariant for the original sequent, we get $\Psi \Vdash d_1 \sqsubseteq c$, and thus $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(d_1) \sqsubseteq \hat{\gamma}(c)$. The second part of the tree invariant is guaranteed by the induction hypothesis on the premise.
By a simple rewrite according to how the lifting of γ is defined we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_\Sigma \hat{\gamma}(\mathbf{case}\, y^c(\ell \Rightarrow Q_\ell)_{\ell \in L})@\hat{\gamma}(d_1) :: y{:}\&\{\ell : A_\ell\}_{\ell \in L}[\hat{\gamma}(c)].$$

Case 2.

$$\frac{\begin{array}{c}\Psi \Vdash d_1 \sqsubseteq c \\ \Psi; \Xi, x : A_k[c] \vdash_\Sigma P@d_1 :: y : T[d] \quad k \in L\end{array}}{\Psi; \Xi, x : \&\{\ell : A_\ell\}_{\ell \in L}[c] \vdash_\Sigma (x^c.k; P)@d_1 :: y : T[d]} \ \&L$$

By the induction hypothesis on the first premise and definition of the lifting of γ we have

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), x{:}A_k[\hat{\gamma}(c)] \vdash_\Sigma \hat{\gamma}(P)@\hat{\gamma}(d_1) :: y{:}T[\hat{\gamma}(d)],$$

which preserves the tree invariant. By applying substitution on the second premise ($\Psi \Vdash d_1 \sqsubseteq c$) we get $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(d_1) \sqsubseteq \hat{\gamma}(c)$.
By the $\&L$ rule, we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), x{:}\&\{A_\ell\}_{\ell \in L}[\hat{\gamma}(c)] \vdash_\Sigma x^{\hat{\gamma}(c)}.k; \hat{\gamma}(P)@\hat{\gamma}(d_1) :: y : T[\hat{\gamma}(d)],$$

which satisfies the tree invariant since the premise satisfies it. Again by a simple rewrite according to how the lifting of γ is defined we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi, x{:}\&\{A_\ell\}_{\ell \in L}[c]) \vdash_\Sigma \hat{\gamma}(x^c.k; P)@\hat{\gamma}(d_1) :: y : T[\hat{\gamma}(d)].$$

Case 3.

$$\frac{\Psi; \Xi \vdash_\Sigma P@c :: y : B[d]}{\Psi; \Xi, z : A[d] \vdash_\Sigma (\mathbf{send}\, z^d\, y; P)@c :: y : (A[d] \otimes B)\,[d]} \ \otimes R$$

By the induction hypothesis on the premise we have

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_\Sigma \hat{\gamma}(P)@\hat{\gamma}(c) :: y{:}B[\hat{\gamma}(d)],$$

which preserves the tree invariant and thus $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(d) \sqsubseteq \hat{\gamma}(d)$.
By the $\otimes R$ rule, we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), z{:}A[\hat{\gamma}(d)] \vdash_\Sigma \mathbf{send}\, z^{\hat{\gamma}(d)}\, y; \hat{\gamma}(P)@\hat{\gamma}(c) :: y : (A \otimes B)[\hat{\gamma}(d)],$$

the resulting sequent satisfies the tree invariant. Again by a simple rewrite according to how the lifting of γ is defined we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi, z{:}A[d]) \vdash_\Sigma \hat{\gamma}(\mathbf{send}\, z^d\, y; P)@\hat{\gamma}(c) :: y : (A \otimes B)[\hat{\gamma}(d)].$$

Case 4.

$$\frac{\begin{array}{c}\Psi \Vdash c' = c \sqcup d \\ \Psi; \Xi, z : A[d], x : B[d] \vdash_\Sigma P@c' :: y : T[d_1]\end{array}}{\Psi; \Xi, x : (A \otimes B)\,[d] \vdash_\Sigma (z^d \leftarrow \mathbf{recv}\, x; P)@c :: y : T[d_1]}\ \otimes L$$

By the induction hypothesis on the second premise and definition of the lifting of γ we have

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), z{:}A[\hat{\gamma}(d)], x{:}B[\hat{\gamma}(d)] \vdash_\Sigma \hat{\gamma}(P)@\hat{\gamma}(c') :: y{:}T[\hat{\gamma}(d_1)],$$

which preserves the tree invariant. Moreover, by applying the substitution on the first premise, we get $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(c') = \hat{\gamma}(c) \sqcup \hat{\gamma}(d)$.
By the $\otimes L$ rule, we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), x{:}(A \otimes B)[\hat{\gamma}(d)] \vdash_\Sigma z^{\hat{\gamma}(d)} \leftarrow \mathbf{recv}\, x; \hat{\gamma}(P)@\hat{\gamma}(c) :: y{:}T[\hat{\gamma}(d_1)],$$

It is straightforward to observe that the resulting sequent satisfies the tree invariant and can be rewritten as before according to the definition of the lifting of γ.

Case 5. Here is the interesting case!

$$\frac{\begin{array}{c}\Psi'; \Xi_1' \vdash_\Sigma X = P@\psi_0 :: x'{:}A[\psi] \in \Sigma \\ \Psi \Vdash \delta_{\mathsf{sec}} : \Psi' \qquad \Psi \Vdash \hat{\delta_{\mathsf{sec}}}(\psi) \sqsubseteq d \\ \Psi \Vdash d_0 \sqsubseteq \hat{\delta_{\mathsf{sec}}}(\psi_0) \qquad \Xi_1, x{:}A[\hat{\delta_{\mathsf{sec}}}(\psi_0)] \Vdash (\delta_{\mathsf{sec}}, \delta_{\mathsf{var}}) :: \Xi_1', x'{:}A[\psi] \\ \Psi; \Xi_2, x{:}A[\hat{\delta_{\mathsf{sec}}}(\psi)] \vdash_\Sigma Q@d_0 :: z{:}C[d]\end{array}}{\Psi; \Xi_1, \Xi_2 \vdash_\Sigma ((x^{[\hat{\delta_{\mathsf{sec}}}(\psi)]} \leftarrow X[(\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})] \leftarrow \Xi_1)@\hat{\delta_{\mathsf{sec}}}(\psi_0); Q)@d_0 :: z{:}C[d]}\ \textsc{Spawn}$$

By the induction hypothesis we have

$$\star\, \hat{\gamma}(\Psi); \hat{\gamma}(\Xi_2), x{:}A[\hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi))] \vdash_\Sigma \hat{\gamma}(Q)@\hat{\gamma}(d_0) :: y{:}\hat{\gamma}(T)[\hat{\gamma}(d)],$$

which preserves the tree invariants. By applying γ on the 3rd and 4th premises, we get

$$\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi)) \sqsubseteq \hat{\gamma}(d),\ \hat{\gamma}(d_0) \sqsubseteq \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi_0)).$$

By applying the substitution γ on the 2nd premise, we get $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}) :: \Psi'$.
And from the 5th premise, we get

$$\hat{\gamma}(Xi_1), x{:}A[\hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi_0))] \Vdash (\hat{\gamma}(\delta_{\mathsf{sec}}), \delta_{\mathsf{var}}) :: \Xi_1', x'{:}A[\psi]$$

By the Spawn rule for the substitution $\gamma' = (\delta_{\mathsf{var}}, \gamma \circ \delta_{\mathsf{sec}})$, we get

$$\hat{\gamma}(\Psi); \hat{\gamma}(\Xi_1), \hat{\gamma}(\Xi_2) \vdash_\Sigma ((x^{[\hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi))]} \leftarrow X[\gamma'] \leftarrow \hat{\gamma}(\Xi_1)@\hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi_0))); \hat{\gamma}(Q))@\hat{\gamma}(d_0) :: y : \hat{\gamma}(T)[\hat{\gamma}(d)].$$

To show that this sequent satisfies the tree invariant, we use (a) the fact that the sequent $\star$ satisfies the tree invariant and (b) the condition on process definitions in the signature. The condition we imposed on process definitions ensures that $\Psi' \Vdash \psi_0 \sqsubseteq \psi$ and $\forall y{:}A[\psi_i] \in \Xi_1'.\Psi' \Vdash \psi_i \sqsubseteq \psi$.
By $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}) :: \Psi'$, we can rewrite the above judgment as

$$\forall y{:}A[\psi_i] \in \Xi_1'.\ \hat{\gamma}(\Psi) \Vdash \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi_i)) \sqsubseteq \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi)).$$

By the tree invariant of $\star$ (i.e., $\hat{\gamma}(\Psi) \Vdash \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi)) \sqsubseteq \hat{\gamma}(d)$),

$$\forall y{:}A[\psi_i] \in \Xi_1'.\ \hat{\gamma}(\Psi) \Vdash \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi_i)) \sqsubseteq \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi)) \sqsubseteq \hat{\gamma}(d).$$

By $(\delta_{\mathsf{var}}, \hat{\gamma}(\hat{\delta_{\mathsf{sec}}})(\Xi_1'')) = \hat{\gamma}(\Xi_1)$,

$$\forall y{:}\hat{\gamma}(A)[\hat{\gamma}(\psi_i)] \in \hat{\gamma}(\Xi_1).\ \hat{\gamma}(\Psi) \Vdash \hat{\gamma}((\psi_i)) \sqsubseteq \hat{\gamma}(\hat{\delta_{\mathsf{sec}}}(\psi)) \sqsubseteq \hat{\gamma}(d).$$

Case 6. The case for D-Fwd is similar to the previous case.

Progress Progress for IFC-typed processes in $\mathsf{CONSESSION}$ follows from the progress of $\mathsf{SESSION}$, as IFC typing can be viewed as a refinement typing using the same dynamics as the session typed processes.

Preservation

Theorem 6 (Preservation). *For any configuration $\mathbb{C}$ that is a valid permutation of a configuration $\mathbb{C}''$ such that $\Psi_0; \Gamma \Vdash \mathbb{C}'' :: \Gamma'$, if $|\mathbb{C}| \mapsto \mathcal{C}'$, then there exists a security annotated program $\mathbb{C}'$ such that $|\mathbb{C}'| = \mathcal{C}'$ and is a valid permutation of a configuration $\mathbb{C}'''$ such that $\Psi_0; \Gamma \Vdash \mathbb{C}''' :: \Gamma'$. Moreover, the stepping preserves the tree invariant.*

Proof. The proof is by considering different cases of $|\mathbb{C}| \mapsto \mathcal{C}'$. For each step we provide a security annotated configuration $\mathbb{C}'$ and then by inversion on the typing derivations show that it is IFC-typed. For the purpose of presentation, we put security annotations of the post-steps for all possible steps in Fig. 14. We provide the detailed proof by inversion for a couple of cases. The rest of the cases are similar.

Case 1. $(\otimes)$

$$
\begin{array}{lll}
\textsc{Fwd} & \mathbf{proc}(y_\alpha[c], (y_\alpha \leftarrow x_\beta)@d_1) \ \mapsto \ [x_\beta/y_\alpha] & (y_\alpha \notin \Delta') \\
\textsc{Spawn} & \mathbf{proc}(y_\alpha[c], (x^d \leftarrow X[\gamma] \leftarrow \Gamma_1)@d_2; Q@d_1) \ \mapsto & (\Psi'; \Gamma_1' \vdash X = P@\psi' :: x' : B[\psi] \in \Sigma) \\
& \mathbf{proc}(x_0[d], \hat{\gamma}(P))\ \mathbf{proc}(y_\alpha[c], [x_0/x]Q@d_1) & (\Gamma_1', x'[\psi], \psi' \Vdash \gamma : \Gamma_1, x_0[d], d_2\ x_0\, \mathit{fresh}) \\
\textsc{D-Fwd} & \mathbf{proc}(y_\alpha[c], F_Y[\gamma]) \ \mapsto & (\psi = \psi; x'[\psi] : C \vdash F_y = \mathtt{Fwd}_{C, y'\psi \leftarrow x'\psi}@\psi :: y' : C[\psi] \in \Sigma) \\
& \mathbf{proc}(y_\alpha[c], \mathtt{Fwd}_{C, y^c_\alpha \leftarrow x^c_\beta}@c) & (x'[\psi], y'[\psi] \Vdash \gamma : x_\beta[c], y_\alpha[c]) \\
1_{\mathsf{snd}} & \mathbf{proc}(y_\alpha[c], (\mathbf{close}\, y_\alpha)@d_1) \ \mapsto \ \mathbf{msg}(\mathbf{close}\, y_\alpha) & \\
1_{\mathsf{rcv}} & \mathbf{msg}(\mathbf{close}\, y_\alpha)\ \mathbf{proc}(x_\beta[c'], (\mathbf{wait}\, y_\alpha; Q)@d_1) \ \mapsto \ \mathbf{proc}(x_\beta[c'], Q@(d_1 \sqcup c)) & \\
\oplus_{\mathsf{snd}} & \mathbf{proc}(y_\alpha[c], y_\alpha.k; P@d_1) \ \mapsto \ \mathbf{proc}(y_{\alpha+1}[c], ([y_{\alpha+1}/y_\alpha]P)@d_1)\ \mathbf{msg}(y_\alpha.k) & \\
\oplus_{\mathsf{rcv}} & \mathbf{msg}(y_\alpha[c].k)\ \mathbf{proc}(u_\gamma[c'], \mathbf{case}\, y_\alpha((\ell \Rightarrow P_\ell)_{\ell \in L})@d_1) \ \mapsto \ \mathbf{proc}(u_\gamma[c'], ([y_{\alpha+1}/y_\alpha]P_k)@(d_1 \sqcup c)) & \\
\&_{\mathsf{snd}} & \mathbf{proc}(y_\alpha[c], (x_\beta.k; P)@d_1) \ \mapsto \ \mathbf{msg}(x_\beta.k)\ \mathbf{proc}(y_\alpha[c], ([x_{\beta+1}/x_\beta]P)@d_1) & \\
\&_{\mathsf{rcv}} & \mathbf{proc}(y_\alpha[c], (\mathbf{case}\, y_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L})@d_1)\ \mathbf{msg}(y_\alpha.k) \ \mapsto \ \mathbf{proc}(v_\delta[c], ([y_{\alpha+1}/y_\alpha]P_k)@c) & \\
\otimes_{\mathsf{snd}} & \mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x_\beta\, y_\alpha; P)@d_1) \ \mapsto \ \mathbf{proc}(y_{\alpha+1}[c], ([y_{\alpha+1}/y_\alpha]P)@d_1)\ \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) & \\
\otimes_{\mathsf{rcv}} & \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)\ \mathbf{proc}(u_\gamma[c'], (w \leftarrow \mathbf{recv}\, y_\alpha; P)@d_1) \ \mapsto \ \mathbf{proc}(u_\gamma[c'], ([x_\beta/w][y_{\alpha+1}/y_\alpha]P)@(d_1 \sqcup c)) & \\
\multimap_{\mathsf{snd}} & \mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x_\beta\, u_\gamma; P)@d_1) \ \mapsto \ \mathbf{msg}(\mathbf{send}\, x_\beta\, u_\gamma)\ \mathbf{proc}(y_\alpha[c], ([u_{\gamma+1}/u_\gamma]P)@d_1) & \\
\multimap_{\mathsf{rcv}} & \mathbf{proc}(y_\alpha[c], (w \leftarrow \mathbf{recv}\, y_\alpha; P)@d_1)\ \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \ \mapsto \ \mathbf{proc}(v_\delta[c], ([x_\beta/w][y_{\alpha+1}/y_\alpha]P)@c) &
\end{array}
$$

Fig. 14: Annotated asynchronous dynamics–proof of preservation.

Subcase 1. (send)

$$\mathbb{C}_1\mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x^c_\beta\, y_\alpha; P)@d_1)\mathbb{C}_2 \ \mapsto \ \mathbb{C}_1\mathbf{proc}(y_{\alpha+1}[c], ([y_{\alpha+1}/y_\alpha]P)@d_1)\mathbf{msg}(\mathbf{send}\, x^c_\beta\, y_\alpha)\mathbb{C}_2$$

By assumption of the theorem: $\Psi_0; \Gamma \Vdash \mathbb{C}_1\mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x^c_\beta\, y_\alpha; P)@d_1)\mathbb{C}_2 :: \Gamma'$.
By Lem. 16: $\Psi_0; \Gamma^1_1 \Vdash \mathbb{C}_1 :: \Gamma_1, \Gamma_2, {\Gamma^1}'_1$ and $\Psi_0; \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c] \Vdash \mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x^c_\beta\, y_\alpha; P)@d_1) :: y_\alpha{:}(A \otimes B)[c]$ and $\Psi_0; \Gamma^1_3, \Gamma_2 \Vdash \mathbb{C}_2 :: \Gamma^{1'}_2$.
Where $\Gamma = \Gamma^1_1, \Gamma^1_2, \Gamma^1_3$ and $\Gamma' = \Gamma^{1'}_1, \Gamma^{1'}_2$. If $x_\beta{:}A[c] \in \Gamma$ we have $\Gamma'_1 = \Gamma_1$ and $\Gamma^{1''}_2, x_\beta{:}A[c] = \Gamma^1_2$, and otherwise $\Gamma'_1, x_\beta{:}A[c] = \Gamma_1$ and $\Gamma^{1''}_2 = \Gamma^1_2$.
By inversion on proc rule $\Psi_0; \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c] \vdash (\mathbf{send}\, x^c_\beta\, y^c_\alpha; P)@d_1 :: y_\alpha{:}(A \otimes B)[c]$. Moreover, $(\star)$ $\forall u_\gamma{:}T[d_2] \in \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c].\, \Psi_0 \Vdash d_2 \sqsubseteq c$.
By inversion on $\otimes R$ rule $\Psi_0; \Gamma^{1''}_2, \Gamma'_1 \vdash P@d_1 :: y_\alpha{:}B[c]$.
By substitution of $y_{\alpha+1}$ for y_α:

$$\Psi_0; \Gamma^{1''}_2, \Gamma'_1 \vdash [y^{\alpha+1}/y^\alpha]P@d_1 :: y_{\alpha+1}{:}B[c].$$

By proc **rule and** $(\star)$**:** $\dagger$ $\Psi_0; \Gamma^{1''}_2, \Gamma'_1 \Vdash \mathbf{proc}(y_{\alpha+1}[c], [y^{\alpha+1}/y^\alpha]P@d_1) :: y_{\alpha+1}{:}B[c]$.
By $\otimes R$ and fwd **rule:** $\Psi_0; x_\beta{:}A[c], y_{\alpha+1}{:}B[c] \vdash (\mathbf{send}\, x^c_\beta\, y^c_\alpha; y^c_\alpha \leftarrow y^c_{\alpha+1})@c :: y_\alpha{:}(A \otimes B)[c]$.
By msg**, $\star$, and $\dagger$:**

$$\Psi_0; \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c] \vdash \mathbf{proc}(y_{\alpha+1}[c], [y^{\alpha+1}/y^\alpha]P@d_1)\mathbf{msg}(\mathbf{send}\, x^c_\beta\, y^c_\alpha; y^c_\alpha \leftarrow y^c_{\alpha+1}) :: y_\alpha{:}(A \otimes B)[c].$$

By configuration typing rules: $\Psi_0; \Gamma \Vdash \mathcal{C}_1\mathbf{proc}(y_{\alpha+1}[c], [y^{\alpha+1}/y^{\alpha}]P@d_1)\mathbf{msg}(\mathbf{send}\, x^c_\beta\, y^c_\alpha; y^c_\alpha \leftarrow y^c_{\alpha+1})\mathcal{C}_2 :: \Gamma'$

Remark: To keep the proofs concise we may use Ψ instead of the term secrecy lattice Ψ_0, whenever we are clearly working with configurations defined in the run-time. Moreover, from now on, we write $\Psi_0; \Delta \Vdash \mathcal{C} :: \Delta'$ also when $\mathcal{C}$ is a valid permutation of a configuration $\mathcal{C}'$ such that $\Psi_0; \Delta \Vdash \mathcal{C}' :: \Delta'$.

F Session logical relation

This section introduces the session logical relation and supporting definitions.

F.1 Open configuration transition

Definition 20. *The configuration $\Delta \Vdash \mathcal{C} :: \Delta'$ is ready to send along $\Lambda \subseteq \Delta, \Delta'$ iff for all $y_\alpha{:}C \in \Lambda$, there is a message $\mathbf{msg}(M)$ in $\mathcal{C}$ sending along y_α, i.e. M is of the form $y_\alpha.k$, $\mathbf{send}\, x_\beta\, y_\alpha$, or $\mathbf{close}\, y_\alpha$.*
Similarly, the configuration $\Delta \Vdash \mathcal{C} :: \Delta'$ is ready to receive along $\Lambda \subseteq \Delta, \Delta'$ iff for all $y_\alpha{:}C \in \Lambda$, there is a process $\mathbf{proc}(z_\delta, P)$ in $\mathcal{C}$ waiting to receive along y_α, i.e. P is of the form $\mathbf{case}\, y_\alpha(\ell \Rightarrow Q_\ell)_{\ell \in I}$, $w \leftarrow \mathbf{recv} y_\alpha; Q$, or $\mathbf{wait}\, y_\alpha; Q$.

Definition 21. *The set $\mathbf{Out}(\Delta \Vdash K)$, is defined as all channels with the sending semantics in Δ, K, i.e., $y_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ iff for some positive type T, we have $y_\alpha{:}T \in \Delta$ or for some negative type T, we have $y_\alpha{:}T \in K$.*
Similarly, the set $\mathbf{In}(\Delta \Vdash K)$, is defined as all channels with the receiving semantics in Δ, K, i.e., $y_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ iff for some negative type T, we have $y_\alpha{:}T \in \Delta$ or for some positive type T, we have $y_\alpha{:}T \in K$.

Definition 22. *$dom(\Delta)$ is a set defined inductively as:*

$$\begin{aligned} dom(\Delta, y_\alpha : A) &= dom(\Delta) \cup \{y\} \\ dom(\cdot) &= \emptyset \end{aligned}$$

Definition 23 (Open configuration transitions). *We provide some notations used in the logical relation and proofs.*

- *The notation $\mapsto^*$ refers to taking none or many steps with $\mapsto$. The notation $\mapsto^j$ refers to taking j steps with $\mapsto$.*
- *We write $\mathcal{D} \mapsto^{*\Upsilon} \mathcal{D}'$ stating that $\mathcal{D} \mapsto^* \mathcal{D}'$ and $\mathcal{D}'$ is ready to send along Υ.*
- *We write $\mathcal{D} \mapsto^{*\Upsilon;\Theta} \mathcal{D}'$ stating that $\mathcal{D} \mapsto^* \mathcal{D}'$ and $\mathcal{D}'$ is ready to send along Υ and ready to receive along Θ.*

F.2 Session Logical Relation

Fig. 15 defines the logical relation for intuitionistic linear logic session types with general recursive types.

$$
\begin{array}{ll}
(1)\ (\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{V}[\![\cdot \Vdash y_\alpha{:}1]\!]^{m+1}_{\cdot;y_\alpha} & \text{iff } \mathcal{D}_1 = \mathbf{msg}(\mathbf{close}\, y_\alpha) \text{ and } \mathcal{D}_2 = \mathbf{msg}(\mathbf{close}\, y_\alpha) \\[4pt]
(2)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \exists k_1, k_2 \in I. k_1 = k_2 \text{ and} \\
\quad \mathcal{V}[\![\Delta \Vdash y_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell\in I}]\!]^{m+1}_{\cdot;y_\alpha} & \quad \mathcal{D}_1 = \mathcal{D}_1'\mathbf{msg}(y_\alpha.k_1) \text{ and } \mathcal{D}_2 = \mathcal{D}_2'\mathbf{msg}(y_\alpha.k_2) \text{ and} \\
 & \quad (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{E}[\![\Delta \Vdash y_{\alpha+1}{:}A_{k_1}]\!]^m \\[4pt]
(3)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \forall k_1, k_2 \in I. \text{ if } k_1 = k_2 \text{ then} \\
\quad \mathcal{V}[\![\Delta \Vdash y_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell\in I}]\!]^{m+1}_{y_\alpha;\cdot} & \quad (\mathcal{D}_1\mathbf{msg}(y_\alpha.k_1);\mathcal{D}_2\mathbf{msg}(y_\alpha.k_2)) \in \mathcal{E}[\![\Delta \Vdash y_{\alpha+1}{:}A_{k_1}]\!]^m \\[4pt]
(4)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \exists x_\beta \text{ s.t.} \\
\quad \mathcal{V}[\![\Delta', \Delta'' \Vdash y_\alpha{:}A \otimes B]\!]^{m+1}_{\cdot;y_\alpha} & \quad \mathcal{D}_1 = \mathcal{D}_1'\mathcal{T}_1\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \text{ and} \\
 & \quad \mathcal{D}_2 = \mathcal{D}_2'\mathcal{T}_2\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha) \text{ and} \\
 & \quad (\mathcal{T}_1;\mathcal{T}_2) \in \mathcal{E}[\![\Delta'' \Vdash x_\beta{:}A]\!]^m \text{ and} \\
 & \quad (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{E}[\![\Delta' \Vdash y_{\alpha+1}{:}B]\!]^m \\[4pt]
(5)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \forall x_\beta \notin dom(\Delta, y_\alpha{:}A \multimap B). \\
\quad \mathcal{V}[\![\Delta \Vdash y_\alpha{:}A \multimap B]\!]^{m+1}_{y_\alpha;\cdot} & \quad (\mathcal{D}_1\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha);\mathcal{D}_2\mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)) \in \\
 & \quad\ \mathcal{E}[\![\Delta, x_\beta{:}A \Vdash y_{\alpha+1}{:}B]\!]^m \\[4pt]
(6)\ (\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{V}[\![\Delta, y_\alpha{:}1 \Vdash K]\!]^{m+1}_{y_\alpha;\cdot} & \text{iff } (\mathbf{msg}(\mathbf{close}\, y_\alpha)\mathcal{D}_1;\mathbf{msg}(\mathbf{close}\, y_\alpha)\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m \\[4pt]
(7)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \forall k_1, k_2 \in I. \text{if } k_1 = k_2 \text{ then} \\
\quad \mathcal{V}[\![\Delta, y_\alpha : \oplus\{\ell{:}A_\ell\}_{\ell\in I} \Vdash K]\!]^{m+1}_{y_\alpha;\cdot} & \quad (\mathbf{msg}(y_\alpha.k_1)\mathcal{D}_1;\mathbf{msg}(y_\alpha.k_2)\mathcal{D}_2) \in \mathcal{E}[\![\Delta, y_{\alpha+1}{:}A_{k_1} \Vdash K]\!]^m \\[4pt]
(8)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \exists k_1, k_2 \in I. k_1 = k_2 \text{ and} \\
\quad \mathcal{V}[\![\Delta, y_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell\in I} \Vdash K]\!]^{m+1}_{\cdot;y_\alpha} & \quad \mathcal{D}_1 = \mathbf{msg}(y_\alpha.k_1)\mathcal{D}_1' \text{ and } \mathcal{D}_2 = \mathbf{msg}(y_\alpha.k_2)\mathcal{D}_2' \text{ and} \\
 & \quad (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{E}[\![\Delta, y_{\alpha+1}{:}A_{k_1} \Vdash K]\!]^m \\[4pt]
(9)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \forall x_\beta \notin dom(\Delta, y_\alpha{:}A \otimes B, K). \\
\quad \mathcal{V}[\![\Delta, y_\alpha{:}A \otimes B \Vdash K]\!]^{m+1}_{y_\alpha;\cdot} & \quad (\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\mathcal{D}_1;\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\mathcal{D}_2) \in \\
 & \quad\ \mathcal{E}[\![\Delta, x_\beta{:}A, y_{\alpha+1}{:}B \Vdash K]\!]^m \\[4pt]
(10)\ (\mathcal{D}_1;\mathcal{D}_2) \in & \text{iff } \exists x_\beta \text{ s.t.} \\
\quad \mathcal{V}[\![\Delta', \Delta'', y_\alpha{:}A \multimap B \Vdash K]\!]^{m+1}_{\cdot;y_\alpha} & \quad \mathcal{D}_1 = \mathcal{T}_1\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\,\mathcal{D}_1'' \text{ and} \\
 & \quad \mathcal{D}_2 = \mathcal{T}_2\mathbf{msg}(\mathbf{send} x_\beta\, y_\alpha)\,\mathcal{D}_2'' \text{ and} \\
 & \quad (\mathcal{T}_1;\mathcal{T}_2) \in \mathcal{E}[\![\Delta' \Vdash x_\beta{:}A]\!]^m \text{ and} \\
 & \quad (\mathcal{D}_1'';\mathcal{D}_2'') \in \mathcal{E}[\![\Delta'', y_{\alpha+1}{:}B \Vdash K]\!]^m \\[4pt]
(11)\ (\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1} & \text{iff } (\mathcal{D}_1;\mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K) \text{ and } \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'. \text{ if } \mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1' \\
 & \quad \text{then } \exists \Upsilon_2, \mathcal{D}_2' \text{ such that } \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2', \text{ and } \Upsilon_1 \subseteq \Upsilon_2 \text{ and} \\
 & \quad \forall\, y_\alpha \in \mathbf{Out}(\Delta \Vdash K). \text{ if } y_\alpha \in \Upsilon_1. \text{ then } (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;y_\alpha} \text{ and} \\
 & \quad \forall\, y_\alpha \in \mathbf{In}(\Delta \Vdash K). \text{if } y_\alpha \in \Theta_1. \text{ then } (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{y_\alpha;\cdot} \\[4pt]
(12)\ (\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{0} & \text{iff } (\mathcal{D}_1;\mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)
\end{array}
$$

Fig. 15: Session logical relation

G Noninterference

This section introduces the fundamental theorem for progress-sensitive noninterference for CONSESSION. We first start with defining supporting notions and proofs of necessary lemmas.

G.1 Up-to equivalence, projections, and splitting up closed configuration

The fundamental theorem Thm. 7 is stated in terms of the logical equivalence $(\Delta_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv_\xi^{\Psi_0} (\Delta_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$, expressing that two open configurations $\mathcal{D}_1$ and $\mathcal{D}_2$ are being related by the term interpretation when plugged into arbitrary closing contexts and observed for any number of message exchanges m. Next we define this equivalence as well as typing context projections, on which the former relies.

Definition 24 (Typing context projections). *Downward projection on security linear contexts and K^s is defined as follows:*

$$\begin{array}{lll}
\Gamma, x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} \Gamma \Downarrow \xi, x_\alpha{:}T[c] & \text{if } c \sqsubseteq \xi \\
\Gamma, x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} \Gamma \Downarrow \xi & \text{if } c \not\sqsubseteq \xi \\
\cdot \Downarrow \xi & \stackrel{\mathsf{def}}{=} \cdot & \\
x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} x_\alpha{:}T[c] & \text{if } c \sqsubseteq \xi \\
x_\alpha{:}T[c] \Downarrow \xi & \stackrel{\mathsf{def}}{=} _{:}1[\top] & \text{if } c \not\sqsubseteq \xi
\end{array}$$

Definition 25 (High provider and High client).

$\cdot \in \textbf{H-Provider}^\xi(\cdot)$

$$\begin{array}{lll}
\mathcal{B} \in \textbf{H-Provider}^\xi(\Gamma, x_\alpha{:}A[c]) & \text{iff} & c \not\sqsubseteq \xi \text{ and } \mathcal{B} = \mathcal{B}'\mathcal{T} \text{ and } \mathcal{B}' \in \textbf{H-Provider}^\xi(\Gamma) \text{ and} \\
& & \mathcal{T} \in \mathsf{Tree}(\cdot \Vdash x_\alpha{:}A), \textbf{or} \\
& & c \sqsubseteq \xi \text{ and } \mathcal{B} \in \textbf{H-Provider}^\xi(\Gamma)
\end{array}$$

$$\begin{array}{lll}
\mathcal{T} \in \textbf{H-Client}^\xi(x_\alpha{:}A[c]) & \text{iff} & c \not\sqsubseteq \xi \text{ and } \mathcal{T} \in \mathsf{Tree}(x_\alpha{:}A \Vdash _ : 1), \textbf{or} \\
& & c \sqsubseteq \xi \text{ and } \mathcal{B} = \cdot
\end{array}$$

Definition 26 (Equivalence of trees by the logical relation upto the observer level). *We define the relation*

$$(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv_\xi^{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2]) \text{ as}$$

$\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$ and $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$ and
$\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi = \Gamma$ and $x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi = K^s$ and
$\forall \mathcal{B}_1 \in \textbf{H-Provider}^\xi(\Gamma_1).\, \forall \mathcal{B}_2 \in \textbf{H-Provider}^\xi(\Gamma_2).\, \forall \mathcal{T}_1 \in \textbf{H-Client}^\xi(x_\alpha{:}A_1[c_1]).\, \forall \mathcal{T}_2 \in \textbf{H-Client}^\xi(y_\beta{:}A_2[c_2]).$

$$\begin{array}{l}
\forall m.\, (\mathcal{B}_1\mathcal{D}_1\mathcal{T}_1, \mathcal{B}_2\mathcal{D}_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m, \text{ and} \\
\forall m.\, (\mathcal{B}_2\mathcal{D}_2\mathcal{T}_2, \mathcal{B}_1\mathcal{D}_1\mathcal{T}_1) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m.
\end{array}$$

G.2 Quasi-running secrecy and relevant nodes

The proof of the fundamental theorem Thm. 7 relies on the notion of a relevant node, , maintaining the invariant that the relevant nodes of the two program runs execute the same code. The latter is guaranteed by Lem. 18. Next we define the notion of a relevant node, expressed locally for an asynchronous semantics (). The notion of a relevant node relies on the notion of quasi-running secrecy, also defined below.

Definition 27 (Quasi-running secrecy). *In an open configuration $\Psi_0; \Gamma \Vdash \mathbb{C} :: \Gamma'$, the quasi-running secrecy of a message or process is determined by its running secrecy, its process term, and the running secrecy of its parent.*

- *If the node is a process with a process term other than* **recv** *or* **case***, then its quasi-running secrecy is equal to its running secrecy.*
- *If the process term is of the form* $\mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L}@d_1$ *or* $x \leftarrow \mathbf{recv}\, y^c_\alpha; P_x@d_1$*, then its quasi-running secrecy is* $d_1 \sqcup c$*.*
- *If the node is a message of a negative type along channel* y^c_α*, its quasi-running secrecy is* c*.*
- *If the node is a message of a positive type along channel* y^c_α *and it has a parent with quasi-running secrecy* d_1*, its quasi-running secrecy is* $d_1 \sqcup c$*, otherwise its quasi-running secrecy is* c*.*

The quasi-running secrecy can be determined by traversing the tree top to bottom.

Definition 28 (Relevant node). *Consider a configuration $\Gamma \Vdash \mathbb{D} :: K^s$ and observer level ξ. A channel is relevant in $\mathbb{D}$ if (1) it has a maximal secrecy level less than or equal to ξ, and (2) it is either an observable channel or it shares a process or message with quasi-running secrecy less than or equal to ξ with a relevant channel. (A channel shares a process with another channel if they are siblings or one is the parent of another.) A process or message is relevant if (1) it has quasi-running secrecy less than or equal to ξ, and (2) it has at least one relevant channel. We denote the relevant processes and messages (i.e., nodes) in $\mathbb{D}$ by $\mathbb{D}\Downarrow\xi$. We write $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$ if the relevant nodes in $\mathbb{D}_1$ are identical to those in $\mathbb{D}_2$ up to renaming of channels with higher or incomparable secrecy than the observer.*

We can build the set of all relevant nodes in a configuration by traversing the tree bottom-up as explained in . Def. 28 provides us with a local guide to identify whether a process is relevant or not. We note that if K^s is observable, then by the tree invariant, every channel in $\mathbb{D}$ is relevant.

Now we can state and prove the invariant that the relevant nodes of the two program runs execute the same code. The proof of the fundamental theorem Thm. 7 crucially relies on this lemma.

Lemma 18 (Keeping relevant nodes in sync). *Consider $\Psi_0; \Gamma \Vdash \mathbb{D}_i :: K^s$ for $i \in \{1, 2\}$ with the relevant nodes in $\mathbb{D}_1$ and $\mathbb{D}_2$ are identical, i.e., $\mathbb{D}_1\Downarrow\xi = \mathbb{D}_2\Downarrow\xi$, with $\Gamma \Downarrow \xi = \Gamma$ and $K^s \Downarrow \xi = K^s$. If $|\mathbb{D}_1| \mapsto |\mathbb{D}'_1|$, then there exists a $\mathbb{D}'_2$ such that $|\mathbb{D}_2| \mapsto^{0,1} |\mathbb{D}'_2|$, i.e., $|\mathbb{D}_2|$ steps to $|\mathbb{D}'_2|$ in zero or one step, and $\mathbb{D}_1\Downarrow\xi = \mathbb{D}_2\Downarrow\xi$.*

Proof. The proof is by cases on the possible steps of $|\mathbb{D}_1| \mapsto |\mathbb{D}^1_1|$. In each case we prove that either the step does not change relevancy of any process in $\mathbb{D}_1$ or we can step $|\mathbb{D}_2|$ such that the same change of relevancy occurs in $\mathbb{D}_2$ too. Note that in all cases, we get $(\mathbb{D}^1_1; \mathbb{D}^1_2) \in \mathsf{Tree}(|\Gamma| \Vdash |K^s|)$ by the preservation of session-typed processes.
In the following cases, we annotate the post-step of the configuration to reflect the annotations required by the preservation of IFC-typed configurations.

Case 1. $\mathbb{D}_1 = \mathbb{D}_1'\mathbf{proc}(y_\alpha[c], y^c_\alpha.k; P@d_1)\mathbb{D}_1''$ **and**

$|\mathbb{D}_1'\mathbf{proc}(y_\alpha[c], y^c_\alpha.k; P@d_1)\mathbb{D}_1''| \mapsto |\mathbb{D}_1'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_1''|$

where $\mathbb{D}^1_1 = \mathbb{D}_1'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_1''$. We consider subcases based on relevancy of process offering along $y_\alpha[c]$:

Subcase 1. $\mathbf{proc}(y_\alpha[c], y^c_\alpha.k; P@d_1)$ is not relevant. By inversion on the typing rules $d_1 \sqsubseteq c$. By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels connected to P including its offering channel y^c_α are relevant. In both cases neither $\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)$, nor $\mathbf{msg}(y^c_\alpha.k)$ are relevant in the post step. Note that from $d_1 \sqsubseteq c$ and $d_1 \not\sqsubseteq \xi$, we get $c \not\sqsubseteq \xi$. Channel y^c_α is not relevant in the pre-step, and both y^c_α and $y^c_{\alpha+1}$ are not relevant in pre-step and post-step configurations. Every not relevant resource of $\mathbf{proc}(y^c_\alpha, y^c_\alpha.k; P@d_1)$ will remain irrelevant in the post-step too.

In this subcase, it is enough to show

$$\mathbb{D}_1'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1'\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi.$$

To prove this we need two observations:

- Neither $\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)$ nor $\mathbf{msg}(y^c_\alpha.k)$ are relevant and they will be dismissed by the projection. (As explained above.)
- Replacing $\mathbf{proc}(y_\alpha[c], y^c_\alpha.k; P@d_1)$ with these two nodes, does not affect relevancy of the rest of processes in $\mathbb{D}_1'\mathbb{D}_1''$. Relevancy of processes in $\mathbb{D}_1''$ remains intact since y^c_α and $y^c_{\alpha+1}$ are irrelevant.
  The relevancy of processes in $\mathbb{D}_1'$ remains intact too as we replace their irrelevant root with another irrelevant process. However, we need to be careful about the changes in the quasi-running secrecy of a process and their effect on its (grand)children. The quasi-running secrecy of the process offering along $y^c_{\alpha+1}$ may be higher or incomparable to d_1 based on the code of P (if it starts with a **recv** or **case**). This is of significance only if $d_1 \sqsubseteq \xi$, and in the pre-step the process has a relevant channel $x : \_[d]$ as its resource where $d \sqsubseteq \xi$. But by the assumption of the subcase, either $d_1 \not\sqsubseteq \xi$ or $x : \_[d]$ is not a relevant channel in the pre-step.

This completes the proof.

Subcase 2. $\mathbf{proc}(y_\alpha[c], y^c_\alpha.k; P@d_1)$ is relevant. By assumption ($\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$) we have :

$$\mathbb{D}_2 = \mathbb{D}_2'\mathbf{proc}(y_\alpha[c], y^c_\alpha.k; P@d_1)\mathbb{D}_2''.$$

and

$$\mathbb{D}_2 \mapsto \mathbb{D}_2'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_2''$$

and $\mathbb{D}^1_2 = \mathbb{D}_2'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k;)\mathbb{D}_2''$.

We need to show:

$$\mathbb{D}_1'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_2'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_2''\Downarrow\xi.$$

- If $c \not\sqsubseteq \xi$, then $\mathbf{msg}(y^c_\alpha.k)$ is not relevant in both runs, and will be dismissed by the projections. Moreover, in this case y^c_α is not relevant in the pre-step and post-step configurations. Thus the relevancy of processes in $\mathbb{D}_1''$ and $\mathbb{D}_2''$ will remain intact.

- If $c \sqsubseteq \xi$, then y_α^c is relevant in the pre-step in both runs. We need to consider the possibility of change in quasi-running secrecy in the post-step. The quasi-running secrecy of the processes offering along $y_{\alpha+1}^c$ may increase based on their code (if the code of P starts with a **recv** or **case**). But in this case, it cannot become irrelevant since by the tree invariant it is always bounded by the max secrecy of the offering channel, $c \sqsubseteq \xi$. Thus, in the post-step of both runs, $y_{\alpha+1}^c$ is relevant. Relevancy of message $\mathbf{msg}(y_\alpha^c.k)$ in the post-steps of the first and second run is determined by the quasi-running secrecies (d and d') of their parents (X and X') in $\mathbb{D}_1''$ and $\mathbb{D}_2''$. If $d \sqsubseteq \xi$, then the parent (X) is relevant in the first run and by assumption is identical to a relevant X' in the second run. Thus messages $\mathbf{msg}(y_\alpha^c.k)$ are relevant in both runs and y_α^c is relevant in the post-step too. The same holds when $d' \sqsubseteq \xi$. Otherwise, in both runs the quasi-running secrecy of the parent is higher than or incomparable to the observer (the parents are both irrelevant). Thus messages $\mathbf{msg}(y_\alpha^c.k)$ are not relevant in the post step of both runs, and will be dismissed by the projections. The channel y_α^c will be irrelevant in the post-step of both runs too. However, this does not affect the processes in $\mathbb{D}_1''$ and $\mathbb{D}_2''$ as the parents of messages (X and X') are already irrelevant in the pre-step.

Finally, we need to show that projections of $\mathbb{D}_1'$ and $\mathbb{D}_2'$ are equal in the post-step too.

Consider the resources of the process that offeres along $y_\alpha[c]$. By our writing convention, they are all in $\mathbb{D}_1'$ and $\mathbb{D}_2'$: (i) Those resources offered by $\mathbb{D}_1'$ and $\mathbb{D}_2'$ with max secrecies higher than or incomparable to the observer ξ in the pre-step will remain higher than or incomparable to the observer and thus irrelevant in the post-step too. (ii) Those resources with max secrecies lower than the observer ξ in the pre-step, i.e., a sub-tree $\mathbb{T}_i$ in $\mathbb{D}_i'$ offering along a channel $w[c']$ where $c' \sqsubseteq \xi$, are relevant in the pre-step and will remain relevant in the post-step (again, in this case, by the tree invariant, the quasi-running secrecy of the processes offering along $y_{\alpha+1}^c$ cannot increase in the post-step).

Case 2. $\mathbb{D}_1 = \mathbb{D}_1'\mathbf{proc}(x_\beta[d], y_\alpha^c.k; P@d_1)\mathbb{D}_1''$ **and**

$$\mathbb{D}_1'\mathbf{proc}(x_\beta[d], y_\alpha^c.k; P@d_1)\mathbb{D}_1'' \mapsto \mathbb{D}_1'\mathbf{msg}(y_\alpha^c.k)\mathbf{proc}(x_\beta[d], [y_{\alpha+1}^c/y_\alpha^c]P@d_1)\mathbb{D}_1''$$

We consider subcases based on relevancy of the process offering along x_d^β:

Subcase 1. $\mathbf{proc}(x_\beta[d], y_\alpha^c.k; P@d_1)$ **is irrelevant.** By inversion on the typing rules, $d_1 \sqsubseteq c \sqsubseteq d$. By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels connected to P including y_α^c and x_β^d are relevant. In both cases, neither $\mathbf{msg}(y_\alpha^c.k)$ nor $\mathbf{proc}(x_\beta[d], [y_{\alpha+1}^c/y_\alpha^c]P@d_1)$ are relevant. Channel x_β^d is irrelevant in the pre-step and post-step configurations.

Channel y_α^c is irrelevant in the pre-step, and both y_α^c and $y_{\alpha+1}^c$ are irrelevant in pre-step and post-step configurations. Every other irrelevant resource of $\mathbf{proc}(x_\beta[d], y_\alpha^c.k; P@d_1)$ will remain irrelevant in the post-step too.

In this subcase, our goal is to show

$$\mathbb{D}_1'\mathbf{msg}(y_\alpha^c.k)\mathbf{proc}(x_\beta[d], [y_{\alpha+1}^c/y_\alpha^c]P@d_1)\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1'\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$$

With a same argument as in **Case 1. Subcase 1.**, we can prove that the relevancy of processes in $\mathbb{D}_1'$ and $\mathbb{D}_1''$ remain intact.

Subcase 2. $\mathbf{proc}(x_\beta[d], y^c_\alpha.k; P@d_1)$ is relevant. By assumption that $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}_2' \mathbf{proc}(x_\beta[d], y^c_\alpha.k; P@d_1)\mathbb{D}_2'',$$

and we have

$$\mathbb{D}_2 \mapsto \mathbb{D}_2' \mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}_2''$$

It remains to show

$$\mathbb{D}_1' \mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}_1'' \Downarrow \xi =_\xi \mathbb{D}_2' \mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}_2'' \Downarrow \xi.$$

- If $d \sqsubseteq \xi$, then x_β is relevant in the pre-steps of both runs and remains relevant in the post-steps. Even if the quasi-running secrecy increases based on the code of P, by the tree invaiant it will be less than or equal to $d \sqsubseteq \xi$, and thus remains observable. As a result, the relevancy of processes in $\mathbb{D}_i''$ remain intact. Moreover, every resource of the processes in $\mathbb{D}_i'$ is relevant in the pre-steps and post-steps of both runs.
- If $d \not\sqsubseteq \xi$, then x_β is irrelevant in the pre-step and remains irrelevant in the post-steps of both runs, and thus the relevancy of processes in $\mathcal{D}_i''$ remain intact. It remains to show that the projections of $\mathcal{D}_1'$ and $\mathcal{D}_2'$ in post-steps are still equal.
  We first condider the trees offered along y^c_α in both runs. The quasi running secrecy of the negative message $\mathbf{msg}(y^c_\alpha.k)$ is c in the post-steps.
  - If $c \sqsubseteq \xi$ then the same message exists in both runs, and the tree offered along y^c_α is relevant in the pre-steps and post-steps.
  - If $c \not\sqsubseteq \xi$ then the message is irrelevant in both runs. y^c_α is irrelevant in the pre-steps of both runs and remain irrelevant in the post-steps too.

  Finally, we need to show that projections of the rest of the trees in $\mathbb{D}_1'$ and $\mathbb{D}_2'$ are equal in the post-step too. Consider the resources of the process that offeres along $x_\beta[d]$. By our writing convention, they are all in $\mathbb{D}_1'$ and $\mathbb{D}_2'$. (i) Those resources offered by $\mathbb{D}_1'$ and $\mathbb{D}_2'$ with max secrecies higher than or incomparable to the observer ξ in the pre-step will remain higher than or incomparable to the observer and thus irrelevant in the post-step too. (ii) Those resources with max secrecies lower than the observer ξ in the pre-step, i.e., a sub-tree $\mathbb{T}_i$ in $\mathbb{D}_i'$ offering along a channel $w[c']$ where $c' \sqsubseteq \xi$, are relevant in the pre-step and thus $\mathbb{T}_1 = \mathbb{T}_2$. The quasi-running secrecy of the process offering along $x_\beta[d]$ after stepping is the same in both runs. If it is still observable, the sub-tree $\mathbb{T}_1 = \mathbb{T}_2$ remains relevant in the post-step of both runs. Otherwise, if the quasi-running secrecies become nonobservable, $\mathbb{T}_1 = \mathbb{T}_2$ may become irrelevant. However, since $\mathbb{T}_1 = \mathbb{T}_2$ and by the tree invariant it only consists of low-secrecy channels, $\mathbb{T}_1$ becomes irrelevant in the post-step of the first run iff $\mathbb{T}_2$ becomes irrelevant in the post-step of the second run. Which completes the proof of this case.

Case 3. $\mathbb{D}_1 = \mathbb{D}_1'\mathbb{T}_1\mathbf{proc}(y_\alpha[c], \mathbf{send} x^d_\beta\, y^c_\alpha @ d_1)\mathbb{D}_1''$ **and**

$|\mathbb{D}_1'\mathbb{T}_1\mathbf{proc}(y_\alpha[c], \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1)\mathbb{D}_1''| \mapsto |\mathbb{D}_1'\mathbb{T}_1\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P @ d_1)\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)\mathbb{D}_1''|$

such that $\Gamma = \Gamma'\Gamma_t$ and $\Psi_0; \Gamma' \Vdash \mathbb{D}_1' :: \Gamma''$ and $\Psi_0; \Gamma_t \Vdash \mathbb{T}_1 :: (x_\beta{:}A[c])$ and $\Gamma'', x_\beta{:}A[c] \Vdash \mathbf{proc}(y_\alpha[c], \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1) :: (y_\alpha{:}A \otimes B[c])$. (In the case that $\mathbb{T}_1$ is empty, we have $\Gamma_t = x_\beta{:}A[c]$.)

We consider subcases based on relevancy of process offering along y^c_α:

Subcase 1. $\mathbf{proc}(y^\alpha_c, \mathbf{send} x^d_\beta\, y^c_\alpha; P @ d_1)$ **is not relevant.**

By inversion on the typing rules $d_1 \sqsubseteq c$. By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels connected to P including y^c_α and x^c_β are relevant. In both cases, neither $\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P @ d_1)$ nor $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ are relevant. Channel y^c_α is irrelevant in the pre-step, and both y^c_α and $y^c_{\alpha+1}$ are irrelevant in pre-step and post-step configurations. In this subcase, our goal is to show

$$\mathbb{D}_1'\mathbb{T}_1\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P @ d_1)\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1'\mathbb{T}_1\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi.$$

We first prove that the relevancy status of $\mathbb{T}_1$ remain intact. Note that all channels in $\mathbb{T}_1$, except x^c_β have the same connections in the pre-step and post-step. So it is enough to consider the changes made to the tree rooted at x^c_β. We show that the relevancy of x^c_β remains intact after the step:

- If $c \not\sqsubseteq \xi$, then x^c_β is irrelevant in the pre-step, and remains irrelevant in the post-step. Moreover, the message is irrelevant in the post-step since its quasi-running secrecy of is higher than or incomparable to the observer.
- If $c \sqsubseteq \xi$, then by $d_1 \sqsubseteq c$, we have $d_1 \sqsubseteq \xi$, and thus both x^β_c and y^c_α must be irrelevant in the pre-step. In the post-step, the parent of the message has to be irrelevant, otherwise y^c_α would be relevant in the pre-step. Since the parent of the message is irrelevant, we know that x^c_β remains irrelevant in the post-step. As a result, the message with three irrelebant channels connected to it is irrelevant in the post-step.

In both cases, the relevancy status of $\mathbb{T}_1$ remains intact: the tree $\mathbb{T}_1$ rooted at x^c_β offers to an irrelevant node before and after the step. With a same argument as in **Case 1. Subcase 1.** and the one given for $\mathbb{T}_1$, we can prove that the relevancy status of processes in $\mathbb{D}_1'$ and $\mathbb{D}_1''$ remain intact.

Subcase 2. $\mathbf{proc}(y^c_\alpha, \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1)$ **is relevant.** By assumption that $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}_2'\mathbb{T}_2''\mathbf{proc}(y_\alpha[c], \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1)\mathbb{D}_2''$$

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}_2'\mathbb{T}_2''\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P @ d_1)\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)\mathbb{D}_2''|$$

If $c \not\sqsubseteq \xi$, then $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ is not relevant in both runs, and will be dismissed by the projections. Moreover, y^c_α is not relevant in the pre-step and post-step configurations. Thus the relevancy of processes in $\mathbb{D}_1''$ and $\mathbb{D}_2''$ will

remain intact. Moreover, in this case x^c_β is irrelevant in both pre-steps and post-steps. Which means that relevancy status of $\mathbb{T}''_1$ and $\mathbb{T}''_2$ remains intact.
If $c \sqsubseteq \xi$, then y^c_α is relevant in the pre-step in both runs. We also know that x_β is relevant in the pre-step and $\mathbb{T}''_1 = \mathbb{T}''_2$ are relevant in the pre-step. In the post-step, $y^c_{\alpha+1}$ is relevant in both runs. Relevancy of messages $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ in the post-steps are determined by the quasi-running secrecy (d' and d'') of their parents (X and X') in $\mathbb{D}''_1$ and $\mathbb{D}''_2$. If $d' \sqsubseteq \xi$, then the parent (X) is relevant in the first run and by assumption is equal to a relevant X' in the second run. Thus messages $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ are relevant in both runs, y^c_α are relevant in the post-step, and trees $\mathbb{T}''_1 = \mathbb{T}''_2$ and their offering channels x^c_β are relevant in the post-steps too. The same holds when $d' \sqsubseteq \xi$.
Otherwise, in both runs the quasi-running secrecy of the parent is higher than or incomparable to the observer level (the parents are both irrelevant). Thus messages $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ are not relevant in the post step of both runs, and will be dismissed by the projections. The channels y^c_α will be irrelevant in the post-step too. However, this does not affect the processes in $\mathbb{D}''_1$ and $\mathbb{D}''_2$ as the parents of messages (X and X') are already irrelevant in the pre-step. The channels x^c_β both become irrelevant in the post-steps. However, we still have $\mathbb{T}''_1 \Downarrow \xi = \mathbb{T}''_2 \Downarrow \xi$ as they have the same type and their parents have the same quasi-running secrecy.
We show that projections of $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are equal in the post-step too. The resources with secrecy level higher than or incomparable to the observer level offered along $\mathbb{D}'_1$ and $\mathbb{D}'_2$ in the pre-step will remain higher than or incomparable to and thus irrelevant in the post-step too. For a relevant resources (w) offered along $\mathbb{D}_1$ and $\mathbb{D}_2$, we need to consider the change in quasi-running secrecy as in **Case 1. Subcase 2.** Moreover, we need to consider the scenario that a relevant resource ($w^{c'}$) in the pre-step loses its relevancy in the post-step because the channel offered along x^c_β is transferred to the message. This case only happens if $c', c \sqsubseteq \xi$ and thus the trees $\mathbb{T}_1$ and $\mathbb{T}_2$ offered along $w^{c'}$ is present in both runs and $\mathbb{T}_1 = \mathbb{T}_2$. We know that $w^{c'}$ is irrelevant in the post-step of both runs, and the quasi-running secrecy of the processes using the resource $w^{c'}$ in both runs are the same.
The relevant and irrelevant processes in $\mathbb{D}''_i$ remain intact.

Case 4. $\mathbb{D}_1 = \mathbb{D}'_1\mathbb{T}''_1\mathbf{proc}(w_\eta[c'], \mathbf{send} x^c_\beta\, y^c_\alpha @ d_1)\mathbb{D}''_1$ **and**

$$|\mathbb{D}'_1\mathbb{T}''_1\mathbf{proc}(w_\eta[c'], \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1)\mathbb{D}''_1| \mapsto |\mathbb{D}'_1\mathbb{T}''_1\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)\mathbf{proc}(w_\eta[c'], [y^c_{\alpha+1}/y^c_\alpha]P @ d_1)\mathbb{D}''_1|$$

such that $\Gamma = \Gamma_1\Gamma_2$ and $\Gamma_1 \Vdash \mathbb{D}'_1 :: \Gamma', y_\alpha{:}(A \multimap B)[c]$ and $\Gamma_2 \Vdash \mathbb{T}''_1 :: (x_\beta{:}A[c])$ and

$$\Gamma', y_\alpha{:}(A \multimap B)[c], x_\beta{:}A[c] \Vdash \mathbf{proc}(w_\eta[c'], \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1) :: (w_\eta{:}C[c']).$$

In the case where $\mathbb{T}''_1$ is empty we have $\Gamma_2 = x_\beta{:}A[c]$. We proceed by considering subcases based on relevancy of the process offering along $w_\eta[c']$:
Subcase 1. $\mathbf{proc}(w_\eta[c'], \mathbf{send} x^c_\beta\, y^c_\alpha; P @ d_1)$ **is not relevant.** By inversion on the typing rules $d_1 \sqsubseteq c \sqsubseteq c'$. By definition either $d_1 \not\sqsubseteq \xi$ or none of the chan-

nels connected to P including y^c_α, and x^c_β are relevant. In both cases, neither $\mathbf{msg}(\mathbf{send}x^c_\beta\, y^c_\alpha)$ nor $\mathbf{proc}(w_\eta[c'], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)$ are relevant. Channel $w^{c'}_\eta$ is irrelevant in the pre-step and post-step configurations.

Channel y^c_α is irrelevant in the pre-step, and both y^c_α and $y^c_{\alpha+1}$ are irrelevant in pre-step and post-step configurations. Every other irrelevant resource of the process in the pre-step will remain irrelevant in the post-step too. See **Case 3. Subcase 1.** for the discussion on the relevancy of $\mathbb{T}''_1$.

$$\mathbb{D}'_1\mathbb{T}''_1\mathbf{msg}(\mathbf{send}x^c_\beta\, y^c_\alpha)\mathbf{proc}(w_\eta[c'], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}''_1\Downarrow\xi =_\xi \mathbb{D}'_1\mathbb{D}''_1\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$$

Subcase 2. $\mathbf{proc}(w^{c'}_\eta, \mathbf{send}x^c_\beta\, y^c_\alpha; P@d_1)$ is relevant. By assumption that $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}'_2\mathbb{T}''_2\mathbf{proc}(w_\eta[c'], \mathbf{send}x^c_\beta\, y^c_\alpha; P@d_1)\mathbb{D}''_2$$

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}'_2\mathbb{T}''_2\mathbf{msg}(\mathbf{send}x^c_\beta\, y^c_\alpha)\mathbf{proc}(w_\eta[c'], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}''_2|$$

With the same argument as in **Case 2. Subcase 2.** we can show that relevancy of $\mathbb{D}''_i$ remains intact.

For $\mathbb{T}''_i$, we argue that if $c \sqsubset \xi$, then $\mathbb{T}''_1 = \mathbb{T}''_2$ is relevant in the pre-step and remains relevant in the post-step too. If $c \not\sqsubseteq \xi$, then the relevancy of $\mathbb{T}''_i$ remain intact from pre-step to post-step. See **Case 3. Subcase 2.** for a more detailed discussion on transferring a tree via message.

The discussion on relevancy of $\mathbb{D}'_i$ is similar to the previous cases.

Case 5. $\mathbb{D}_1 = \mathbb{D}'_1\mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)\mathbb{D}''_1$ and

$$|\mathbb{D}'_1\mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)\mathbb{D}''_1| \mapsto |\mathbb{D}'_1\mathbf{proc}(w_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P_k@(d_1\sqcup c))\mathbb{D}''_1|$$

We consider sub-cases based on relevancy of process offering along w^d_β. Observe that y^c_α is relevant if an only if v^c is relevant, since they share a message of secrecy c.

Subcase 1. $\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)$ is not relevant. By definition either $d_1 \sqcup c \not\sqsubseteq \xi$ or none of the channels connected to P including its offering channel y^c_α are relevant. In both cases the messages $\mathbf{msg}(y^c_\alpha.k)$ and the continuation process $\mathbf{proc}(w_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P_k@(d_1 \sqcup c))$ are not relevant either. It is straightforward to see that

$$\mathbb{D}'_1\mathbf{proc}(w_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P_k@(d_1 \sqcup c))\mathbb{D}''_1\Downarrow\xi =_\xi \mathbb{D}'_1\mathbb{D}''_1\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi.$$

Subcase 2. $\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)$ is relevant. By definition of relevancy, we get that $c \sqcup d_1 \sqsubseteq \xi$ and thus y^c_α is relevant. This means that $\mathbf{msg}(y^c_\alpha.k)$ is relevant too. By assumption that $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}'_2\mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)\mathbb{D}''_2,$$

such that P_ℓ is equal to P_ℓ modulo renaming of some channels with secrecy level higher than or incomparable to the observer.

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}_2'\mathbf{proc}(x_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P_k@(d_1 \sqcup c))\mathbb{D}_2''|$$

This completes the proof of the subcase as we know that the relevancy of channels in $\mathbb{D}_i'$ and $\mathbb{D}_i''$ remain intact.

Case 6. $\mathbb{D}_1 = \mathbb{D}_1'\mathbf{proc}(y_\alpha[c], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_1''$ **and**

$$|\mathbb{D}_1'\mathbf{proc}(y_\alpha[c], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_1''| \mapsto |\mathbb{D}_1'\mathbf{proc}(x_1[c], [y^c_{\alpha+1}/y^c_\alpha]P_k@c)\mathbb{D}_1''|$$

We consider sub-cases based on relevancy of process offering along y^c_α. Observe that y^c_α is relevant if an only if x^c_1 is relevant, since they share a message of secrecy c.

Subcase 1. $\mathbf{proc}(y_\alpha[c], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)$ **is not relevant.** By definition either $d_1 \sqcup c = c \not\sqsubseteq \xi$ or none of the channels connected to P including its offering channel y^c_α are relevant. In both cases, means that $y^c_{\alpha+1}$ is not relevant and $\mathbf{msg}(y^c_\alpha.k)$ is not relevant either. Moreover the continuation process $\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P_k@c)$ won't be relevant. And

$$\mathbb{D}_1'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P_k@c)\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1'\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$$

Subcase 2. $\mathbf{proc}(y_\alpha[c], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)$ **is relevant.** By definition of relevancy, we get that $c \sqcup d_1 = c \sqsubseteq \xi$ and thus y^c_α is relevant. This means that $\mathbf{msg}(y^c_\alpha.k)$ is relevant too. By assumption that $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}_2'\mathbf{proc}(y_\alpha[c], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell\in L}@d_1)\mathbf{msg}(y^c_\alpha.k)\mathbb{D}_2'',$$

such that P_ℓ is equal to P_ℓ modulo renaming of some channels with secrecy level higher than or incomparable to the observer.

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}_2'\mathbf{proc}(y_{\alpha+1}[c], [y^c_{\alpha+1}/y^c_\alpha]P_k@c)\mathbb{D}_2''|$$

This completes the proof of the subcase as we know that the relevancy of channels in $\mathbb{D}_i'$ and $\mathbb{D}_i''$ remain intact.

Case 7. $\mathbb{D}_1 = \mathbb{D}_1'\mathbf{msg}(\mathbf{send} x^c_\eta y^c_\alpha)\mathbf{proc}(v_\eta[c'], w^c \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)\mathbb{D}_1''$ **and**

$$\mathbb{D}_1'\mathbf{msg}(\mathbf{send} x^c_\eta\, y^c_\alpha)\mathbf{proc}(v_\eta[c'], w^c \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)\mathbb{D}_1'' \mapsto \mathbb{D}_1'\mathbf{proc}(v_\eta[c'], [x_\eta/w][y^c_{\alpha+1}/y^c_\alpha]P@c\sqcup d_1)\mathbb{D}_1''$$

We consider sub-cases based on relevancy of process offering along $v^{c'}_\eta$.

Subcase 1. $\mathbf{proc}(v_\eta[c'], w^c \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)$ **is not relevant.** By definition either $d_1 \sqcup c \not\sqsubseteq \xi$ or none of the channels connected to P including y^c_α are relevant.

In both cases by the definition of quasi-running secrecy we know that neither $\mathbf{msg}(\mathbf{send} x^c_\eta y^c_\alpha)$ nor the continuation process $\mathbf{proc}(v_\eta[c'], [x_\eta/w][y^c_{\alpha+a}/y^c_\alpha]P@c\sqcup d_1)$ are relevant. It is then straightforward to see that

$$\mathbb{D}_1'\mathbf{proc}(v_\eta[c'], [x_\eta/w][y^c_{\alpha+1}/y^c_\alpha]P@c \sqcup d_1)\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1'\mathbb{D}_1''\Downarrow\xi =_\xi \mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi.$$

Subcase 2. $\mathbf{proc}(v_\eta[c'], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)$ **is relevant.** By definition of relevancy, we get that $c \sqcup d_1 \sqsubseteq \xi$. This implies that y^c_α is relevant in the pre-step. From relevancy of y^c_α and the quasi-running secrecy lower than or equal to the observer of the positive message $\mathbf{msg}(\mathbf{send} x^c_\eta\, y^c_\alpha;)$ we get that the message is relevant too. By assumption:

$$\mathbb{D}_2 = \mathbb{D}'_2 \mathbf{msg}(\mathbf{send} x^c_\eta\, y^c_\alpha)\mathbf{proc}(u_\gamma[c'], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)\mathbb{D}''_2.$$

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}'_2 \mathbf{proc}(u_\gamma[c'], [x_\eta/w][y^c_{\alpha+1}/y^c_\alpha]P@d_1 \sqcup c)\mathbb{D}''_2|$$

We need to consider that the quasi-running secrecy of the process may increases in the post step based on the code of P. The argument for this case is similar to the previous cases of the proof. See **Case 1.Subcase 2.**. One interesting situation is when the relevancy of chain of positive and relevant messages in the pre-step of $\mathbb{D}'_i$ changes in the post-step. By relevancy in the pre-step we know that these chains exist in both runs, so the same chain of messages will become irrelevant in the post-step of both runs.

Case 8. $\mathbb{D}_1 = \mathbb{D}'_1 \mathbf{proc}(y_\alpha[c], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)\mathbf{msg}(\mathbf{send} x^c_\eta y^c_\alpha)\mathbb{D}''_1$ **and**

$$|\mathbb{D}'_1 \mathbf{proc}(y_\alpha[c], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)\mathbf{msg}(\mathbf{send} x^c_\eta y^c_\alpha)\mathbb{D}''_1| \mapsto |\mathbb{D}'_1 \mathbf{proc}(y_{\alpha+1}[c], [x_\eta/w][y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}''_1|$$

We consider sub-cases based on relevancy of process offering along y^c_α.

Subcase 1. $\mathbf{proc}(y_\alpha[c], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)$ **is not relevant.** By definition either $d_1 \sqcup c = c \not\sqsubseteq \xi$ or none of the channels connected to P including y^c_α are relevant. In both cases the negative message $\mathbf{msg}(\mathbf{send} x^c_\eta y^c_\alpha)$ and the continuation process $\mathbf{proc}(y_{\alpha+1}[c], [x_\eta/w][y^c_{\alpha+c}/y^c_\alpha]P@d_1)$ are not relevant either. It is then straightforward to see that

$$\mathbb{D}'_1 \mathbf{proc}(y_{\alpha+1}[c], [x_\eta/w][y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}''_1 \Downarrow \xi =_\xi \mathbb{D}'_1 \mathbb{D}''_1 \Downarrow \xi =_\xi \mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi.$$

Subcase 2. $\mathbf{proc}(y_\alpha[c], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)$ **is relevant.** By definition of relevancy, we get that $c \sqcup d_1 = c \sqsubseteq \xi$ and y^c_α is relevant. This means that $\mathbf{msg}(\mathbf{send} x^c_\eta\, y^c_\alpha)$ and the channel x^c_η are relevant. By assumption that $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}'_2 \mathbf{proc}(y_\alpha[c], w \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)\mathbf{msg}(\mathbf{send} x^c_\eta\, y^c_\alpha)\mathbb{D}''_2.$$

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}'_2 \mathbf{proc}(y_{\alpha+1}[c], [x^c_\eta/w][y^c_{\alpha+1}/y^c_\alpha]P@d_1)\mathbb{D}''_2|\mathbb{F}_2$$

The proof is similar to previous cases.

Case 9. $\mathbb{D}_1 = \mathbb{D}'_1 \mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)\mathbb{D}''_1$**and**

$\mathbf{def}(\Psi'; \Xi' \vdash X = P@\psi' :: x : B'[\psi])$ **and**

$$|\mathbb{D}'_1\mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)\mathbb{D}''_1| \\ \mapsto \\ |\mathbb{D}'_1\mathbf{proc}(x_0[d], \hat{\gamma}(P)@d_2)\mathbf{proc}(y_\alpha[c], [x_0^d/x^d]Q@d_1)\mathbb{D}''_1|$$

where $\gamma = (\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})$ $\hat{\gamma}(\psi) = d$, $\hat{\gamma}(\psi') = d_2$, and $\hat{\gamma}(\Xi') = \Gamma_1$. We consider sub-cases based on relevancy of process offering along y^c_α.

Subcase 1. $\mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)$ **is not relevant.** By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels of this process including y^c_α are relevant.
In both cases, it means that $\mathbf{proc}(x_0[d], \hat{\gamma}(P)@d_2)$ and $\mathbf{proc}(y_\alpha[c], [x_0^d/x^d]Q@d_1)$ are not relevant either. Note that $d_1 \sqsubseteq d_2$ and thus $d_2 \not\sqsubseteq \xi$ if $d_1 \not\sqsubseteq \xi$.

$$\mathbb{D}'_1\mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)\mathbb{D}''_1 \Downarrow \xi =_\xi \mathbb{D}'_1\mathbb{D}''_1 \Downarrow \xi =_\xi \mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$$

Subcase 2. $\mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)$ **is relevant.** By definition of relevancy, we get that $d_1 \sqsubseteq \xi$ and all channels of this process with secrecy levels lower than or equal to the observer level are relevant. By assumption that $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and definition of $=_\xi$:

$$\mathbb{D}_2 = \mathbb{D}'_2\mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)\mathbb{D}''_2.$$

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}'_2\mathbf{proc}(x_0[d], \hat{\gamma}(P)@d_2)\mathbf{proc}(y_\alpha[c], [x_0^d/x^d]Q@d_1)\mathbb{D}''_2|$$

Remark: we can assume that the fresh channel being spawned will be x_0 in both runs. Moreover, the (unique) substitution $\hat{\gamma}$ is the same when stepping both runs.
Note that if x_0^d has secrecy level lower than or equal to the observer level, then taking this step won't change relevancy of any channels. Otherwise some resource of the process may become irrelevant after this step, e.g. x_0^d may block their relevancy path if $d \not\sqsubseteq \xi$ or in the case where $d_2 \not\sqsubseteq \xi$. But this happens to the processes in the both runs and can only change relevant processes/messages in the pre-step to irrelevant processes/messages in the post-step. (similar to the cases 3 and 4 for $\otimes$ and $\multimap$)

Case 10. $\mathbb{D}_1 = \mathbb{D}'_1\mathbf{proc}(y_\alpha[c], F_Y[\gamma]@c)\mathbb{D}''_1$ **and**
$\mathbf{def}(\Psi'; u{:}C[\psi] \vdash F_Y = \mathtt{Fwd}_{C, w^\psi \leftarrow u^\psi}@\psi :: w : C[\psi])$ **and**

$$|\mathbb{D}'_1\mathbf{proc}(y_\alpha[c], F_Y[\gamma]@c)\mathbb{D}''_1| \mapsto |\mathbb{D}'_1\mathbf{proc}(y_\alpha[c], \mathtt{Fwd}_{C, y^c_\alpha \leftarrow z^c_\beta}@d_1)\mathbb{D}''_1|$$

where $\gamma = (\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})$ $\hat{\gamma}(\psi) = c$, $\hat{\gamma}(w) = y_\alpha$, and $\hat{\gamma}(u) = z_\beta$.

The proof of this case is similar to the proof of Case 9(Spawn) by considering sub-cases based on relevancy of process offering along y^c_α.

G.3 Fundamental theorem

Theorem 7 (Fundamental Theorem). *For all security levels ξ, and configurations*

$$\Psi; \Gamma_1 \Vdash \mathbb{D}_1 :: u_\alpha{:}T_1[c_1], \text{ and } \Psi; \Gamma_2 \Vdash \mathbb{D}_2 :: v_\beta{:}T_2[c_2],$$

with $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, $\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi$, *and* $u_\alpha{:}T_1[c_1] \Downarrow \xi = v_\beta{:}T_2[c_2] \Downarrow \xi$ *we have*

$$(\Gamma_1 \Vdash \mathbb{D}_1 :: u_\alpha{:}T_1[c_1]) \equiv_\xi^\Psi (\Gamma_2 \Vdash \mathbb{D}_2 :: v_\beta{:}T_2[c_2]).$$

Proof. Our goal is to show:

For all $\mathbb{D}_1$ and $\mathbb{D}_2$ that are IFC-typed, i.e.,$\Psi; \Gamma_1 \Vdash \mathbb{D}_1 :: u_\alpha{:}T_1[c_1]$ and $\Psi; \Gamma_2 \Vdash \mathbb{D}_2 :: v_\beta{:}T_2[c_2]$, with $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and $\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi = \Gamma$, and $u_\alpha{:}T_1[c_1] \Downarrow \xi = v_\beta{:}T_2[c_2] \Downarrow \xi = K^s$ we have $\forall \mathcal{B}_1 \in \textbf{H-Provider}^\xi(\Gamma_1). \forall \mathcal{B}_2 \in \textbf{H-Provider}^\xi(\Gamma_2). \forall \mathcal{T}_1 \in \textbf{H-Client}^\xi(x_\alpha{:}A_1[c_1]). \forall \mathcal{T}_2 \in \textbf{H-Client}^\xi(y_\beta{:}A_2[c_2])$.

$\forall m. (\mathcal{B}_1|\mathbb{D}_1|\mathcal{T}_1, \mathcal{B}_2|\mathbb{D}|_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m$, and $\forall m. (\mathcal{B}_2|\mathbb{D}|_2\mathcal{T}_2, \mathcal{B}_1|\mathbb{D}|_1\mathcal{T}_1) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m$..

Without loss of generality, we only prove one part of this symmetric relation, i.e.:

For all $\mathbb{D}_1$ and $\mathbb{D}_2$ that are IFC-typed, i.e.,$\Psi; \Gamma_1 \Vdash \mathbb{D}_1 :: u_\alpha{:}T_1[c_1]$ and $\Psi; \Gamma_2 \Vdash \mathbb{D}_2 :: v_\beta{:}T_2[c_2]$, with $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and $\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi = \Gamma$, and $u_\alpha{:}T_1[c_1] \Downarrow \xi = v_\beta{:}T_2[c_2] \Downarrow \xi = K^s$ we have $\forall \mathcal{B}_1 \in \textbf{H-Provider}^\xi(\Gamma_1). \forall \mathcal{B}_2 \in \textbf{H-Provider}^\xi(\Gamma_2). \forall \mathcal{T}_1 \in \textbf{H-Client}^\xi(x_\alpha{:}A_1[c_1]). \forall \mathcal{T}_2 \in \textbf{H-Client}^\xi(y_\beta{:}A_2[c_2])$.

$$\forall m. (\mathcal{B}_1|\mathbb{D}|_1\mathcal{T}_1, \mathcal{B}_2|\mathbb{D}|_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m$$

The goal is equivalent to:

$(\star_1)$ For all m, for all $\mathbb{D}_1$ and $\mathbb{D}_2$ that are IFC-typed, i.e.,$\Psi; \Gamma_1 \Vdash \mathbb{D}_1 :: u_\alpha{:}T_1[c_1]$ and $\Psi; \Gamma_2 \Vdash \mathbb{D}_2 :: v_\beta{:}T_2[c_2]$, with $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and $\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi = \Gamma$, and $u_\alpha{:}T_1[c_1] \Downarrow \xi = v_\beta{:}T_2[c_2] \Downarrow \xi = K^s$ we have $\forall \mathcal{B}_1 \in \textbf{H-Provider}^\xi(\Gamma_1). \forall \mathcal{B}_2 \in \textbf{H-Provider}^\xi(\Gamma_2). \forall \mathcal{T}_1 \in \textbf{H-Client}^\xi(x_\alpha{:}A_1[c_1]). \forall \mathcal{T}_2 \in \textbf{H-Client}^\xi(y_\beta{:}A_2[c_2])$.

$$(\mathcal{B}_1|\mathbb{D}|_1\mathcal{T}_1, \mathcal{B}_2|\mathbb{D}|_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m$$

First, we show that it is enough to prove the following goal.

$(\star_2)$ For all m, for all $\mathbb{D}'_1$ and $\mathbb{D}'_2$ that are IFC-typed, i.e.,$\Psi; \Gamma \Vdash \mathbb{D}'_1 :: K^s$ and $\Psi; \Gamma \Vdash \mathbb{D}'_2 :: K^s$, with $\mathbb{D}'_1 \Downarrow \xi =_\xi \mathbb{D}'_2 \Downarrow \xi$, and $\Gamma \Downarrow \xi = \Gamma$, and $K^s \Downarrow \xi = K^s$ we have

$$(|\mathbb{D}'_1|, |\mathbb{D}'_2|) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m$$

We need to show that from $\star_1$, we get $\star_2$:

Apply a for all introduction on $\star_2$ to get an arbitrary number m, configurations $\mathbb{D}_1$ and $\mathbb{D}_2$, and high providers $\mathcal{B}_1$ and $\mathcal{B}_2$ and high clients $\mathcal{T}_1$ and $\mathcal{T}_2$ satisfying the given assumptions. We build well-typed configurations stel $\mathbb{D}_1'$ and $\mathbb{D}_2'$ with the following steps:

1. Let's assume $\mathcal{T}_1 \neq \cdot$ and $\mathcal{T}_2 \neq \cdot$, i.e., $\mathcal{T}_1 \in \mathsf{Tree}(u_\alpha{:}T_1 \Vdash \_{:}1)$ and $c_1 \not\sqsubseteq \xi$ and $\mathcal{T}_2 \in \mathsf{Tree}(v_\beta{:}T_2 \Vdash \_{:}1)$ and $c_1 \not\sqsubseteq \xi$. We annotate all channels in $\mathcal{T}_1$ with security level c_1 and all processes with running secrecy c_1. We add the same security variable ψ correspondingly to all process variables defined in the signature. We also provide a mapping for the spawns appearing in the process terms such that they map all security variables ψ in the process definition to the security variable c_1. It is straightforward to see that this annotation of $\mathcal{T}_1$ which we call $\mathbb{T}_1$ is IFC-typed. With a similar approach we can build the IFC-typed annotation of $\mathcal{T}_1$ which we call $\mathbb{T}_2$.
2. If $\mathcal{T}_1 = \cdot$ and $\mathcal{T}_2 = \cdot$, then define $\mathbb{T}_1 = \mathbb{T}_2 = \cdot$, and observe that they are trivially IFC-typed.
3. Consider every tree $\mathcal{A}_1 \in \mathcal{B}_1$, we know that $\mathcal{A}_1 \in \mathsf{Tree}(x_\gamma{:}A)$ for some $x_\gamma{:}A[d] \in \Gamma_1$ such that $d \not\sqsubseteq \xi$, we build a security-annotated version of $\mathcal{A}_1$ as $\mathbb{A}_1$, similar to (1), by annotating all channels/running secrecy/substitutions with the level d. From all such annotated trees we build the annotated forest $\mathbb{B}_1$. Similarly, we can build annotated forest $\mathbb{B}_2$. Again it is straightforward to show that both $\mathbb{B}_1$ and $\mathbb{B}_2$ are IFC-typed.

We define $\mathbb{D}_1' = \mathbb{B}_1\mathbb{D}_1\mathbb{T}_1$ and $\mathbb{D}_2' = \mathbb{B}_2\mathbb{D}_2\mathbb{T}_2$. Note that these two configurations are both IFC-typed, and also we have $\Psi; \Gamma \Vdash \mathbb{D}_1' :: K^s$ and $\Psi; \Gamma \Vdash \mathbb{D}_2' :: K^s$. We also know that $\mathbb{D}_1' \Downarrow \xi = \mathbb{D}_2' \Downarrow \xi$. Observe that all relevant nodes occur in $\mathbb{D}_1$ and $\mathbb{D}_2$, for which by assumption we know $\mathbb{D}_1 \Downarrow \xi = \mathbb{D}_2 \Downarrow \xi$. Now we can apply $\star_2$ to get what we want.

It remains to prove ($\star_2$):

($\star_2$) For all m, for all $\mathbb{D}_1$ and $\mathbb{D}_2$ that are IFC-typed, i.e.,$\Psi; \Gamma \Vdash \mathbb{D}_1 :: K^s$ and $\Psi; \Gamma \Vdash \mathbb{D}_2 :: K^s$, with $\mathbb{D}_1 \Downarrow \xi =_\xi \mathbb{D}_2 \Downarrow \xi$, and $\Gamma \Downarrow \xi = \Gamma$, and $K^s \Downarrow \xi = K^s$ we have

$$(|\mathbb{D}_1|; |\mathbb{D}_2|) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m$$

The proof is by induction on the index m.

Base case. $m = 0$**.** We consider arbitrary configurations ($\forall I$ on the goal). By the configuration typing, we know that

$$(|\mathbb{D}_1|; |\mathbb{D}_2|) \in \mathsf{Tree}(|\Gamma| \Vdash |K^s|),$$

which is enough to complete the proof.

Inductive case. $m = m' + 1$**.**

$$(\mathcal{B}_1|\mathbb{D}_1|\mathcal{T}_1; \mathcal{B}_2|\mathbb{D}_2|\mathcal{T}_2) \in \mathsf{Tree}(|\Gamma| \Vdash |K^s|).$$

Consider an arbitrary $\mathbb{D}_1'$ such that $|\mathbb{D}_1| \mapsto^{*\Upsilon;\Theta} |\mathbb{D}_1'|$. By Preservation we know that $\mathbb{D}_1'$ is IFC-typed for the same interface $\Psi; \Gamma \Vdash \mathbb{D}_1' :: K^s$ By Lem. 18, for some $\mathbb{D}_2'$, we get $|\mathbb{D}_2| \mapsto^{*\Upsilon} |\mathbb{D}_2'|$, such that $\Psi; \Gamma \Vdash \mathbb{D}_2' :: K^s$ with $\mathbb{D}_2' \Downarrow \xi = \mathbb{D}_2' \Downarrow \xi$.

We need to show that

($\dagger_1$) $\forall u_\delta' \in \mathbf{Out}(|\Gamma| \Vdash |K|).\, \mathbf{if}\ u_\delta' \in \Upsilon.\, \mathbf{then}\ (|\mathbb{D}_1'|; |\mathbb{D}_2'|) \in \mathcal{V}[\![|\Gamma| \Vdash |K^s|]\!]^m_{\cdot;u_\delta'}$ **and**
($\dagger_2$) $\forall u_\delta' \in \mathbf{In}(|\Gamma| \Vdash |K|).\, \mathbf{if}\ u_\delta' \in \Theta.\, \mathsf{then}\ (|\mathbb{D}_1'|; |\mathbb{D}_2'|) \in \mathcal{V}[\![|\Gamma| \Vdash |K^s|]\!]^m_{u_\delta';\cdot}$.

We prove both $\dagger_1$, and $\dagger_2$. We consider an arbitrary u_δ' and provide a case analysis based on its type. If there is no such u_δ' in the specified set, then the statements trivially holds. There might be a channel in both of these sets, if and only if $\mathbb{D}_1 = \cdot$.

Proof of $\dagger_1$. Consider an arbitrary channel $u_\delta'{:}T \in \mathbf{Out}(|\Gamma| \Vdash |K|^s)$ and assume $u_\delta'{:}T \in \Upsilon$. We consider different cases based on the session type T.

Case 1. $u_\delta'{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:}1[c_1]$, and $\mathbb{D}_1' = \mathbb{D}_1''\mathbf{msg}(\mathbf{close}\, u_\alpha^{c_1})$. By the typing rules and $\mathbb{D}_1' = \mathbb{D}_1''\mathbf{msg}(\mathbf{close}\, u_\alpha^{c_1})$ we know that $\mathbb{D}_1'' = \cdot$, and $\Gamma_1 = \cdot$.

Note that by the definition of relevancy, all processes in $\mathbb{D}_1'$ and $\mathbb{D}_2'$ has to be relevant. As a result,

$$\mathbb{D}_2' = \mathbf{msg}(\mathbf{close}\, u_\alpha^{c_1}).$$

Thus, we satisfy the conditions required by Line 1 of the logical relation to establish

$$\dagger_1 (|\mathbb{D}_1'|; |\mathbb{D}_2'|) \in \mathcal{V}[\![\cdot \Vdash |u_\alpha{:}1[c_1]|]\!]^m_{\cdot;u_\delta'}$$

as needed.

Case 2. $u_\delta'{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in I}[c_1]$, and $\mathbb{D}_1' = \mathbb{D}_1''\mathbf{msg}(u_\alpha^{c_1}.k)$.

By the assumption of the theorem, $\mathbb{D}_1'$ and $\mathbb{D}_2'$ are both relevant, and we have $\mathbb{D}_2' = \mathbb{D}_2''\mathbf{msg}(u_\alpha^{c_1}.k)$. Removing $\mathbf{msg}(u_\alpha^{c_1}.k)$ from $\mathbb{D}_1'$ and $\mathbb{D}_2'$ does not change relevancy of the remaining configuration when $c_1 \sqsubseteq \xi$:

$$\mathbb{D}_1'' \Downarrow \xi = \mathbb{D}_2'' \Downarrow \xi.$$

We can apply the induction hypothesis on the index $m' < m$ to get

$$(|\mathbb{D}_1''|, |\mathbb{D}_2''|) \in \mathcal{E}[\![|\Gamma| \Vdash |u_{\alpha+1}{:}A_k[c_1]|]\!]^{m'}.$$

By line (2) in the definition of $\mathcal{V}$:

$$\dagger_1 (|\mathbb{D}_1''\mathbf{msg}(u_\alpha^{c_1}.k)|, |\mathbb{D}_2''\mathbf{msg}(u_\alpha^{c_1}.k)|) \in \mathcal{V}[\![|\Gamma| \Vdash |u_\alpha{:} \oplus \{\ell : A_\ell\}_{\ell \in I}[c_1]|]\!]^{m'+1}_{\cdot;u_\delta'}.$$

Case 3. $u_\delta'{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:}(A \otimes B)[c_1]$, and

$$\mathbb{D}_1' = \mathbb{D}_1''\mathbb{A}_1\mathbf{msg}(\mathbf{send} x_\beta^{c_1}\, u_\alpha^{c_1})$$

where $\Gamma = \Gamma', \Gamma''$, and $\Psi; \Gamma'' \Vdash \mathbb{A}_1 :: (x_\beta{:}A[c_1])$, and $\Psi; \Gamma' \Vdash \mathbb{D}_1'' :: (u_{\alpha+1}{:}A[c_1])$.

Since $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are both IFC-typed, they enjoy the tree invariant. Thus both of them are relevant and since $\mathbb{D}'_1 \Downarrow \xi = \mathbb{D}'_2 \Downarrow \xi$, we have

$$\mathbb{D}'_2 = \mathbb{D}''_2 \mathbb{A}_2 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta \; u^{c_1}_\alpha),$$

such that $\Psi; \Gamma'' \Vdash \mathbb{A}_2 :: (x_\beta{:}A[c_1])$, and $\Psi; \Gamma' \Vdash \mathbb{D}''_2 :: (u_{\alpha+1}{:}A[c_1])$. Moreover, by relevancy of $\mathbb{D}'_i$ (and relevancy of $x^{c_1}_\beta$) we get $\mathbb{D}''_1 \Downarrow \xi = \mathbb{D}''_2 \Downarrow \xi$ and $\mathbb{A}_1 \Downarrow \xi = \mathbb{A}_2 \Downarrow \xi$.
Note that in the particular case with $x_\beta{:}A[c_1] \in \Gamma$, we have $\mathbb{A}_i = \cdot$ and $\Gamma'' = x_\beta{:}A[c_1]$.
We apply the induction hypothesis on the index $m' < m$ for $\Psi; \Gamma'' \Vdash \mathbb{A}_i :: (x_\beta{:}B[c_1])$ to get

$$(|\mathbb{A}_1|, |\mathbb{A}_2|) \in \mathcal{E}[\![|\Gamma''| \Vdash |x_\beta{:}A[c_1]|]\!]^{m'}.$$

and apply the induction hypothesis on the index $m' < m$ for $\Psi; \Gamma' \Vdash \mathbb{D}''_i :: (u_{\alpha+1}{:}A[c_1])$

$$(|\mathbb{D}''_1|, |\mathbb{D}''_1|) \in \mathcal{E}[\![|\Lambda_1| \Vdash |w_\gamma{:}B[c_1]|]\!]^{m'}.$$

By line (4) in the definition of $\mathcal{V}$:

$$\dagger_1(|\mathbb{A}_1 \mathbb{D}''_1 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta \; u^{c_1}_\alpha)|, |\mathbb{A}_2 \mathbb{D}''_1 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta \; u^{c_1}_\alpha)|) \in \mathcal{V}[\![|\Delta| \Vdash |u_\alpha{:}A \otimes B[c_1]|]\!]^{m'+1}_{\cdot;u'_\delta}.$$

Case 4. $\Gamma = \Gamma', x_\gamma{:}\&\{\ell : A_\ell\}_{\ell \in L}[c]$, and $u'_\delta{:}T[c'] = x_\gamma{:}\&\{\ell : A_\ell\}_{\ell \in L}[c]$, and $\mathbb{D}'_1 = \mathbf{msg}(x^c_\gamma.k)\mathbb{D}''_1$.
By the assumption of the theorem, $\mathbb{D}'_2 = \mathbf{msg}(x^c_\gamma.k)\mathbb{D}''_2$. Moreover, $\mathbb{D}''_1 \Downarrow \xi = \mathbb{D}''_2 \Downarrow \xi$. The reason is $c \sqsubseteq \xi$ and thus $x^c_{\gamma+1}$ is relevant in $\mathbb{D}''_i$ and no relevancy changes in the configurations after removing the negative message.
We can apply the induction hypothesis on the smaller index $m' < m$ to get

$$(|\mathbb{D}''_1|; |\mathbb{D}''_2|) \in \mathcal{E}[\![|\Gamma', x_{\gamma+1}{:}A_k[c]| \Vdash |K^s|]\!]^{m'}.$$

By line (8) in the definition of $\mathcal{V}$:

$$\dagger_1(|\mathbf{msg}(x^c_\gamma.k)\,\mathbb{D}''_1|; |\mathbf{msg}(x^c_\gamma.k)\,\mathbb{D}''_2|) \in \mathcal{V}[\![|\Gamma', x_\gamma{:}\&\{\ell{:}A_\ell\}_{\ell \in L}[c]| \Vdash |K^s|]\!]^{m'+1}_{\cdot;u'_\delta}.$$

Case 5. $\Gamma = \Gamma', \Gamma'', x_\gamma{:}(A \multimap B)[c]$, and $u'_\delta{:}T'[c'] = x_\gamma{:}(A \multimap B)[c]$, and

$$\mathbb{D}'_1 = \mathbb{A}_1 \mathbf{msg}(\mathbf{send} y^c_\delta \, x^c_\gamma)\mathbb{D}''_1,$$

such that $\Psi_0; \Gamma'' \Vdash \mathbb{A}_1 :: y_\delta{:}A[c]$ and $\Psi_0; \Gamma', x_{\gamma+1}{:}B[c] \Vdash \mathbb{D}''_1 :: K^s$.
The message $\mathbf{msg}(\mathbf{send} y^c_\delta \, x^c_\gamma)$ is relevant in $\mathbb{D}'_1$. By assumption of the theorem, $\mathbb{D}'_2 = \mathbb{A}_2 \mathbf{msg}(\mathbf{send} y^c_\delta \, x^c_\gamma)\mathbb{D}''_2$, such that $\Psi_0; \Gamma'' \Vdash \mathbb{A}_2 :: y_\delta{:}A[c]$ and $\Psi_0; \Gamma^h_2, \Gamma', x_{\gamma+1}{:}B[c] \Vdash \mathbb{D}''_2 :: K^s$ with Γ^h_2 being the context of all the channels in Γ_2 with security higher than or incomparable to the observable level ξ. Again because of the tree invariant every resource of $\mathbb{A}_2$ has a secrecy less than or equal to the observer. Also, by assumption we know that

$\mathbb{D}'_1 \Downarrow \xi = \mathbb{D}'_2 \Downarrow \xi$. By definition of relevancy, we know that $\mathbf{msg}(\mathbf{send} y^c_\delta\, x^c_\gamma)$ and the tree $\mathbb{A}_1$ connected to it are relevant in $\mathbb{D}_1$ and thus $\mathbb{A}_1$ is equal to $\mathbb{A}_2$, i.e, $\mathbb{A}_1 \Downarrow \xi = \mathbb{A}_2 \Downarrow \xi$.

Removing the tress $\mathbb{A}_1$ and $\mathbb{A}_2$ and the messages from both configurations does not change relevancy of the rest of the configuration since $x^c_{\gamma+1}$ will remain relevant, i.e., $\mathcal{D}''_1 \Downarrow \xi = \mathcal{D}''_2 \Downarrow \xi$.

We can apply the induction hypothesis on the smaller index $m' < m$ and $\mathbb{A}_1 \Downarrow \xi = \mathbb{A}_2 \Downarrow \xi$ and $\mathbb{D}''_1 \Downarrow \xi = \mathbb{D}''_2 \Downarrow \xi$ to get

$$(|\mathbb{A}_1|, |\mathbb{A}_2|) \in \mathcal{E}[\![|\Gamma''| \Vdash |y_\delta{:}A[c]|]\!]^{m'}, \text{ and}$$
$$(|\mathcal{D}''_1|, |\mathcal{D}''_2|) \in \mathcal{E}[\![|\Gamma', x_{\gamma+1}{:}B[c]| \Vdash |K^s|]\!]^{m'}.$$

By line (10) in the definition of $\mathcal{V}$:

$$\dagger_1(|\mathbb{A}_1 \mathbf{msg}(\mathbf{send} y^c_\delta\, x^c_\gamma)\mathbb{D}''_1|;\ |\mathbb{A}_2 \mathbf{msg}(\mathbf{send} y^c_\delta\, x^c_\gamma)\mathbb{D}''_2|) \in \mathcal{V}[\![|\Gamma'_1, x_\gamma{:}A \multimap B[c]| \Vdash |K^s|]\!]^{m'+1}_{\cdot;u'_\delta}.$$

Proof of $\dagger_2$**.** Consider an arbitrary channel $u'_\delta{:}T \in \mathbf{In}(|\Gamma| \Vdash |K|)$ We consider different cases based on the session type T.

Case 1. $u'_\delta{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell \in I}[c_1]$.

We need to apply the induction hypothesis on the index $m' < m$, but first we need to show that the invariant of the induction holds. Consider an arbitrary label $k \in L$.From $\mathbb{D}'_1 \Downarrow \xi = \mathbb{D}'_2 \Downarrow \xi$ and $c_1 \sqsubseteq \xi$, we get

$$\mathbb{D}'_1 \mathbf{msg}(u^{c_1}_\alpha.k) \Downarrow \xi = \mathbb{D}'_2 \mathbf{msg}(u^{c_1}_\alpha.k) \Downarrow \xi.$$

By induction hypothesis

$$(|\mathbb{D}'_1|, |\mathbb{D}'_2|) \in \mathcal{E}[\![|\Gamma| \Vdash |u_{\alpha+1}{:}A_k[c_1]|]\!]^{m'}.$$

By line (3) in the definition of $\mathcal{V}$:

$$\dagger_2(|\mathbb{D}'_1 \mathbf{msg}(u^{c_1}_\alpha.k)|, |\mathbb{D}'_2 \mathbf{msg}(u^{c_1}_\alpha.k)|) \in \mathcal{V}[\![|\Gamma| \Vdash |u_\alpha{:}\&\{\ell : A_\ell\}_{\ell \in I}[c_1]|]\!]^{m'+1}_{u'_\delta;\cdot}.$$

Case 2. $u'_\delta{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:}A \multimap B[c_1]$. By assumption and $c_1 \sqsubseteq \xi$, we know that $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are relevant. Consider an arbitrary channel x_β, which is not a free name in $\Gamma \Vdash K^s$. We know $c_1 \sqsubseteq \xi$, and thus adding a negative message sending $x_\beta[c_1]$ along $u_\alpha[c_1]$ does not change the relevancy of the rest of processes:

$$\mathbb{D}'_1 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta\, u^{c_1}_\alpha) \Downarrow \xi = \mathbb{D}'_2 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta\, u^{c_1}_\alpha) \Downarrow \xi$$

We can apply the induction hypothesis on $m' < m$ to get

$$(|\mathbb{D}'_1 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta\, u^{c_1}_\alpha)|, |\mathbb{D}'_2 \mathbf{msg}(\mathbf{send} x^{c_1}_\beta\, u^{c_1}_\alpha)|) \in \mathcal{E}[\![|\Gamma, x_\beta{:}A[c_1]| \Vdash |u_{\alpha+1}{:}B[c_1]|]\!]^{m'}.$$

By line (5) in the definition of $\mathcal{V}$:

$$\dagger_2(|\mathbb{D}'_1|, |\mathbb{D}'_2|) \in \mathcal{V}[\![|\Gamma| \Vdash |u_\alpha{:}A \multimap B[c_1]|]\!]^{m'+1}_{u'_\delta;\cdot}.$$

Case 3. $\Gamma = \Gamma', x_\gamma{:}1[c]$, and $u'_\delta{:}T'[c'] = x_\gamma{:}1[c]$

We first briefly explain why the invariant of induction holds after we bring the closing message inside $\mathbb{D}'_i$ and remove the channel x^c_γ from Δ, i.e.

$$\mathbf{msg}(\mathbf{close}\, x^c_\gamma)\mathbb{D}'_1 \Downarrow \xi = \mathbf{msg}(\mathbf{close}\, x^c_\gamma)\mathbb{D}'_2 \Downarrow \xi.$$

– If the parent of $\mathbf{msg}(\mathbf{close}\, x^c_\gamma)$ in $\mathbb{D}'_1$ is relevant in $\mathbb{D}'_1$ before bringing the message inside then by the assumption of the theorem, it is the same as the parent of $\mathbf{msg}(\mathbf{close}\, x^c_\gamma)$ in $\mathbb{D}'_2$. If after adding the message $\mathbb{D}'_1$ the parent still remains relevant, it means that it has at least one other relevant channel other than x^c_γ in $\mathbb{D}'_1$ which also exists in $\mathbb{D}'_2$ and will be relevant after adding the message to $\mathbb{D}'_2$. If after adding the message, the message's parent in $\mathbb{D}'_1$ becomes irrelevant, it means that it does not have a relevant path to any other channel in Δ' and K. By the assumption of theorem the parent of the message in $\mathbb{D}'_2$ does not have such path either. The same argument holds for any other node that becomes irrelevant because of adding the closing message to $\mathbb{D}'_i$. As a result the same processes becomes irrelevant in both $\mathbb{D}'_1$ and $\mathbb{D}'_2$ after adding the message to them and the proof of this case is complete. The same argument holds for the case in which the parent of $\mathbf{msg}(\mathbf{close}\, x^c_\gamma)$ in $\mathbb{D}'_2$ before adding the message is relevant.
– Otherwise the parent of $\mathbf{msg}(\mathbf{close}\, x^c_\gamma)$ is irrelevant in $\mathbb{D}'_1$ and $\mathbb{D}'_2$ before adding the message and remains irrelevant after that too. The proof in this case is straightforward.

Now that the invariant holds, we can apply the induction hypothesis on the smaller index $m' < m$:

$$(|\mathbf{msg}(\mathbf{close}\, x^c_\gamma)\mathbb{D}'_1|; |\mathbf{msg}(\mathbf{close}\, x^c_\gamma)\mathbb{D}'_2|) \in \mathcal{E}[\![|\Gamma'| \Vdash |K^s|]\!]^{m'}.$$

By line (6) in the definition of $\mathcal{V}$,

$$\dagger_2\, (|\mathbb{D}'_1|; |\mathbb{D}'_2|) \in \mathcal{V}[\![|\Gamma', x_\gamma{:}1[c]| \Vdash |K^s|]\!]^{m'+1}_{u'_\delta;\cdot}$$

Case 4. $\Gamma = \Gamma', x_\gamma{:} \oplus \{\ell : A_\ell\}_{\ell \in L}[c]$, and $u'_\delta{:}T'[c'] = x_\gamma{:} \oplus \{\ell : A_\ell\}_{\ell \in L}[c]$.
Consider an arbitrary label $k \in L$. We have $\mathbf{msg}(x^c_\gamma.k)\mathbb{D}'_1 \Downarrow \xi = \mathbf{msg}(x^c_\gamma.k)\mathbb{D}'_2 \Downarrow \xi$, since $x^c_{\gamma+1}$ is relevant and the positive messages $\mathbf{msg}(x^c_\gamma.k)$ in both configurations are relevant if and only if their parents are. Thus adding the message does not change relevancy of any other process.
We can apply the induction hypothesis on the smaller index $m' < m$ to get

$$(|\mathbf{msg}(x^c_\gamma.k)\mathbb{D}'_1|; |\mathbf{msg}(x^c_\gamma.k)\mathbb{D}'_2|) \in \mathcal{E}[\![|\Gamma', x_{\gamma+1} : A_k[c]| \Vdash |K^s|]\!]^{m'}.$$

By line (7) in the definition of $\mathcal{V}$:

$$\dagger_2(|\mathbb{D}'_1|; \mathbb{D}'_2) \in \mathcal{V}[\![|\Gamma', x_\gamma{:} \oplus \{\ell : A_\ell\}_{\ell \in L}[c]| \Vdash |K^s|]\!]^{m'+1}_{u'_\delta;\cdot}.$$

Case 5. $\Gamma = \Gamma', x_\gamma{:}(A \otimes B)[c]$, and $u'{:}T'[c'] = x_\gamma{:}(A \otimes B)[c]$.
Consider an arbitraray channel y_η which is not a free name in $\Gamma \Vdash K^s$. We have

$$\mathbf{msg}(\mathbf{send} y^c_\eta\, x^c_\gamma)\mathbb{D}'_1 \Downarrow \xi =_\xi \mathbf{msg}(\mathbf{send} y^c_\eta\, x^c_\gamma)\mathbb{D}'_2 \Downarrow \xi :$$

The quasi-running secrecy of $\mathbf{msg}(\mathbf{send} y^c_\eta\, x^c_\gamma)$ is lower than or equal to the observer level if the quasi-running secrecy of its parent is lower than or equal to the observer level. So the relevancy of the parent of the message and thus the rest of configurations do not change by adding the message to the configuration.
We can apply the induction hypothesis on the smaller index $m' < m$ to get

$$(|\mathbf{msg}(\mathbf{send} y^c_\eta\, x^c_\gamma)\mathbb{D}'_1|; |\mathbf{msg}(\mathbf{send} y^c_\eta\, x^c_\gamma)|) \in \mathcal{E}[\![|\Gamma', y_\eta{:}A[c], x_{\gamma+1}{:}B[c]| \Vdash |K^s|]\!]^{m'}.$$

By line (9) in the definition of $\mathcal{V}$:

$$(|\mathbb{D}'_1|; |\mathcal{D}'_2|) \in \mathcal{V}[\![|\Gamma', x_\gamma{:}A \otimes B[c]| \Vdash |K^s|]\!]^{m'+1}_{u'_\delta;\cdot}.$$

H Diamond property, confluence, and backward/forward closures

This section introduces various supporting lemmas, asserting the diamond property and confluence, as well as forward and backward closure. These lemmas are used in the proofs of Lem. 26 and Corollary 2 introduced in § H, which are in turn instrumental in proving transitivity (see § J) and adequacy (see § K).

H.1 Diamond property, confluence, and minimal sending configuration

Subsequent lemmas rely on the notions of *active* and *produced* processes and messages, which we define next.

Definition 29 (Produced processes and messages). *For each dynamic step in Fig. 9, we say that a process or message is* produced *in the step if it occurs in the post-step (right side of $\mapsto$ notation). For example, the transition step for* Spawn*, produces two processes*

$$\mathbf{proc}(x_0, ([x_0, \Lambda/x, \Lambda']\hat{\gamma}\, P)) \text{ and } \mathbf{proc}(y_\alpha, ([x_0/x]Q)),$$

and the transition step for $\multimap_{\mathsf{snd}}$ produces a message and a process

$$\mathbf{msg}(\mathbf{send}\, x_\beta\, u_\gamma) \text{ and } \mathbf{proc}(y_\alpha, ([u_{\gamma+1}/u_\gamma]P)).$$

◇

Definition 30 (Active processes and messages). *For each dynamic step in Fig. 9, we define the* active *configuration $\mathcal{A}$ as the set of processes and messages occurring in the pre-step (the left side of $\mapsto$ notation). For example, in the transition step for* Spawn*, the active configuration is the single process*

$$\mathbf{proc}(y_\alpha, (x \leftarrow X \leftarrow \Lambda); Q),$$

and in the transition step for $\multimap_{\mathsf{rcv}}$, the active configuration is

$$\mathbf{proc}(y_\alpha, (w \leftarrow \mathbf{recv}\, y_\alpha; P))\, \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha).$$

We define the set of active configurations, i.e. active*, for $\mathcal{D}_1 \mapsto^{*\Upsilon}_{\Delta \Vdash K} \mathcal{D}'_1$ as the union of every step's active configuration. A process is called active in $\mathcal{D}_1 \mapsto^{*\Upsilon} \mathcal{D}'_1$, if it is in the set* active*.*

◇

Lemma 19 (Uniqueness of process productions). *Consider $\Delta \Vdash \mathcal{D} :: K$, and $\mathcal{D} \mapsto^* \mathcal{D}' \mapsto \mathcal{D}_1\mathbf{proc}(x, P)\mathcal{D}_2$, such that $\mathbf{proc}(x, P)$ is produced in the last step. Then process $\mathbf{proc}(x, P)$ does not occur in any of the previous steps $\mathcal{D} \mapsto^* \mathcal{D}'$.*

The same result hold for the production of a message.

Proof. Observe the following invariant in the dynamics

1. In each configuration, there exists at most one process or message offering along a particular generation of x.
2. The offering channel of processes that are not active or produced in the step does not change.
3. The offering channel of a negative message is a fresh generation of a channel (and no process offers along it before the message is received).
4. In the production of $\mathbf{proc}(x_\alpha, P)$, either (a) x^α is freshly generated (in the case of Spawn for the callee), or (b) $\mathbf{proc}(x_\alpha, P)$ replaces another process $\mathbf{proc}(x_\alpha, P')$ that offers along x^α and has a larger process term, i.e. $|P| < |P'|$ (in $1_{\mathsf{rcv}}, \oplus_{\mathsf{rcv}}, \otimes_{\mathsf{rcv}}, \&_{\mathsf{snd}}, \multimap_{\mathsf{snd}}$, and spawn for the continuation of the caller), or (c) x_α is a fresh generation of x (in $\oplus_{\mathsf{snd}}, \otimes_{\mathsf{snd}}, \&_{\mathsf{rcv}}, \multimap_{\mathsf{rcv}}$)

Consider production of a process $\mathbf{proc}(x_\alpha, P)$ and the cases described in 4. If 4.(a) or 4.(c) hold, then by freshness of x_α, such process has not been produced before. It is enough to consider case 4.(b). Assume that there is another occurrence of process $\mathbf{proc}(x_\alpha, P)$ before this production; By the observations 1, 2 and 4 we get to a contradiction: there must be a chain of productions satisfying 4.(b) with decreasing sizes from the earlier $\mathbf{proc}(x_\alpha, P)$ to the later one $\mathbf{proc}(x_\alpha, P)$, which is contradictory.

With a similar reasoning we can prove that if $\Delta \Vdash \mathcal{D} :: K$, and $\mathcal{D} \mapsto^* \mathcal{D}' \mapsto \mathcal{D}_1 \mathbf{msg}(M) \mathcal{D}_2$, such that $\mathbf{msg}(M)$ is produced in the last step and $\overline{u}$ is the name of the message resources, then $\mathbf{msg}(M)$ does not occur in any of $\mathcal{D} \mapsto^* \mathcal{D}'$ steps.

Lemma 20 (Diamond Property). *If $\Delta \Vdash \mathcal{D}_1 :: K$ and $\dagger\ \mathcal{D}_1 \mapsto^{\Upsilon}_{\Delta \Vdash K} \mathcal{D}'_1$ and $\dagger'\ \mathcal{D}_1 \mapsto^{\Upsilon'}_{\Delta \Vdash K} \mathcal{D}''_1$, and $\mathcal{D}'_1 \neq \mathcal{D}''_1$ then there is a configuration $\mathcal{D}$ such that $\star\ \mathcal{D}'_1 \mapsto^{\Upsilon''}_{\Delta \Vdash K} \mathcal{D}$, and $\star'\ \mathcal{D}''_1 \mapsto^{\Upsilon''}_{\Delta \Vdash K} \mathcal{D}$, where $\Upsilon \cup \Upsilon' = \Upsilon''$. The messages produced along the channels $\Upsilon \cap \Upsilon'$ are identical in $\mathcal{D}'_1$ and $\mathcal{D}''_1$ and $\mathcal{D}$.*

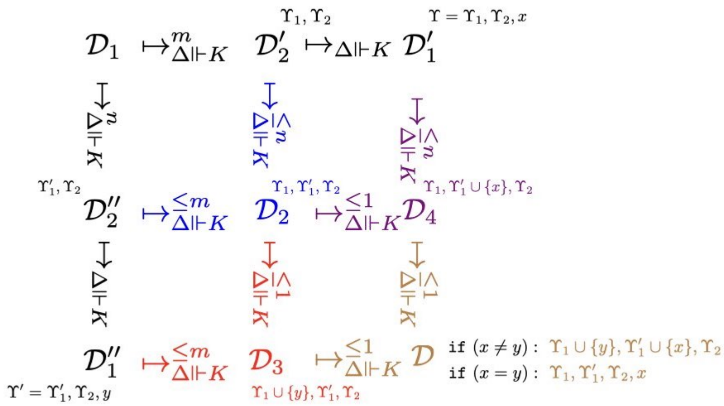
Moreover, every process in $\mathcal{D}'_1$ that is not an active process of $\mathcal{D}_1 \mapsto^{\Theta';\Upsilon'}_{\Delta \Vdash K} \mathcal{D}''_1$ is in $\mathcal{D}$. And every process in $\mathcal{D}''_1$ that is not an active process of $\mathcal{D}_1 \mapsto^{\Theta;\Upsilon}_{\Delta \Vdash K} \mathcal{D}'_1$ is in $\mathcal{D}$.

Proof. The proof is straightforward by cases. The key is to build $\star$ (locally) identical to $\dagger'$, and $\star'$ (locally) identical to $\dagger$.

Lemma 21 (Confluence). *If $\Delta \Vdash \mathcal{D}_1 :: K$ and $\dagger\ \mathcal{D}_1 \mapsto^{m'_{\Upsilon}}_{\Delta \Vdash K} \mathcal{D}'_1$ and $\dagger'\ \mathcal{D}_1 \mapsto^{n'_{\Upsilon'}}_{\Delta \Vdash K} \mathcal{D}''_1$, then there is a configuration $\mathcal{D}$ such that $\star\ \mathcal{D}'_1 \mapsto^{j_{\Upsilon''}}_{\Delta \Vdash K} \mathcal{D}$ for some $j \leq n'$, and $\star'\ \mathcal{D}''_1 \mapsto^{k_{\Upsilon''}}_{\Delta \Vdash K} \mathcal{D}$ for some $k \leq m'$, where $\Upsilon \cup \Upsilon' = \Upsilon''$. The messages produced along the channels $\Upsilon \cap \Upsilon'$ are identical in $\mathcal{D}'_1$ and $\mathcal{D}''_1$ and $\mathcal{D}_1$. The steps in $\star$ are (locally) identical to the steps of $\dagger'$ that do not occur in $\dagger$. And the steps in $\star'$ are (locally) identical to the steps of $\dagger$ that do not occur in $\dagger'$.*

Moreover, every process in $\mathcal{D}'_1$ that is not an active process of $\mathcal{D}_1 \mapsto^{*_{\Upsilon'}}_{\Delta \Vdash K} \mathcal{D}''_1$ is in $\mathcal{D}$. And every process in $\mathcal{D}''_1$ that is not an active process of $\mathcal{D}_1 \mapsto^{*_{\Upsilon}}_{\Delta \Vdash K} \mathcal{D}'_1$ is in $\mathcal{D}$.

Proof. It follows by standard inductions from the diamond property (Lem. 20). The induction is on the pair (n', m'). If $n' = 0$ and $m' = 0$, the proof is straightforward. Similarly, if $n' = 1$ and $m' = 0$ or $n' = 0$ and $m' = 1$ the proof is straightforward. For $n' = 1$ and $m' = 1$, we apply the diamond property (Lem. 20). Assume that $n' = n + 1$ and $m' = m + 1$. We form the following diagram to sketch the structure of the proof.



By induction hypothesis, from $\mathcal{D}''_2$ and $\mathcal{D}'_2$, we build $\mathcal{D}_2$ (with the blue steps) that satisfies the required properties. Then, again by induction we build $\mathcal{D}_3$ (with the red steps) and $\mathcal{D}_4$ (with the violet steps). And finally, with a last induction, we build $\mathcal{D}$ (with the brown steps). The diagram depicts how the required properties move along the steps. For each configuration, we put the set of channels that it sends along them near the configuration. In particular, for $\mathcal{D}'_1$, we put $\Upsilon = \Upsilon_1, \Upsilon_2, x$ and for $\mathcal{D}''_1$, we put $\Upsilon' = \Upsilon'_1, \Upsilon_2, y$. The set Υ_2 is in both Υ and Υ', and we have $\Upsilon_1 \cap \Upsilon'_1 = \emptyset$, i.e. we put all the common channels (except possibly x and y) in Υ_2. We assume that the step $\mathcal{D}'_2 \mapsto \mathcal{D}'_1$ produces a message along channel x. In the case that this step does not produce any message we can simply ignore x. Similarly, we assume that the step $\mathcal{D}''_2 \mapsto \mathcal{D}''_1$ produces a message along channel y. In the case that this step does not produce any message we can simply ignore y. At the end, we can build $\mathcal{D}$ with at most $m+1$ steps from $\mathcal{D}''_1$ and at most $n+1$ steps from $\mathcal{D}'_1$. The configuration $\mathcal{D}$ sends messages along the union of Υ and Υ', and by induction the messages along the intersection of $\Upsilon \cap \Upsilon'$ are identical in $\mathcal{D}'_1$ and $\mathcal{D}''_1$ and $\mathcal{D}$. The proof of the second part of the lemma is straightforward by stating the required property for each inductive step and passing them down to $\mathcal{D}$ in the diagram.

As a straightforward corollary to the first part of the confluence lemma, we get that the messages produced along the channels $\Upsilon \cap \Upsilon'$ are identical in $\mathcal{D}'_1$ and $\mathcal{D}''_1$, i.e., identical messages will be produced along the same channels, independent of the non-deterministic path that we take to produce them.

Corollary 1 (Active set independent of non-determinism). *If $\mathcal{D}_1 \mapsto^* \mathcal{D}'_1 \mathcal{A} \mathcal{D}''_1 \mapsto \mathcal{D}^1_1 \mathbf{proc}(x, P) \mathcal{D}^2_1$ and $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2 \mathcal{A}' \mathcal{D}''_2 \mapsto \mathcal{D}^1_2 \mathbf{proc}(x, P) \mathcal{D}^2_2$, where $\mathcal{A}$ and $\mathcal{A}'$ are the active parts that produce $\mathbf{proc}(x, P)$, then $\mathcal{A} = \mathcal{A}'$.*

Proof. This is another corollary to the second part of the confluence lemma (Lem. 21). First observe that $\mathcal{A}$ is not active in $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2\mathcal{A}'\mathcal{D}''_2$: if it is active, we produce $\mathbf{proc}(x, P)$ twice in $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2\mathcal{A}'\mathcal{D}''_2 \mapsto \mathcal{D}^1_2\mathbf{proc}(x, P)\mathcal{D}^2_2$, which contradicts with uniqueness of process productions (Lem. 19). Similarly, $\mathcal{A}'$ is not active in $\mathcal{D}_1 \mapsto^* \mathcal{D}'_1\mathcal{A}\mathcal{D}''_1$. With the second part of the confluence lemma, for some $\mathcal{D}$, we have $\mathcal{D}'_1\mathcal{A}\mathcal{D}''_1 \mapsto^* \mathcal{D}$ and $\mathcal{D}'_2\mathcal{A}'\mathcal{D}''_2 \mapsto^* \mathcal{D}$ such that both $\mathcal{A}$ and $\mathcal{A}'$ occur in $\mathcal{D}$. If $\mathcal{A} \neq \mathcal{A}'$, we can produce $\mathbf{proc}(x, P)$ twice from $\mathcal{D}$ which by Lem. 19 is contradictory. Thus, know that $\mathcal{A} = \mathcal{A}'$.

Note: here we rely on the fact that the steps in Fig. 9 produce the post-steps uniquely from the pre-steps. In particular, in the spawn rule we assume that the fresh channel name is uniquely determined based on the offering channel and the process term of the caller.

Similarly, we can prove that if $\mathcal{D}_1 \mapsto^* \mathcal{D}'_1\mathcal{A}\mathcal{D}''_1 \mapsto \mathcal{D}^1_1\mathbf{msg}(M)\mathcal{D}^2_1$ and $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2\mathcal{A}'\mathcal{D}''_2 \mapsto \mathcal{D}^1_2\mathbf{msg}(M)\mathcal{D}^2_2$, where $\mathcal{A}$ and $\mathcal{A}'$ are the active parts that produce $\mathbf{msg}(M)$, then $\mathcal{A} = \mathcal{A}'$.

In the proof of Corollary 1, we used the fact that the pre-step for each substitution is unique. This can be proved independently by a straightforward observation that if $\mathcal{D}_1 \mapsto^* \mathcal{D}'_1\mathcal{A}\mathcal{D}''_1 \mapsto \mathcal{D}^1_1[x_\beta/y_\alpha]\mathcal{D}^2_1$ and $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2\mathcal{A}'\mathcal{D}''_2 \mapsto \mathcal{D}^1_2[x_\beta/y_\alpha]\mathcal{D}^2_2$, where $\mathcal{A}$ and $\mathcal{A}'$ are forwarding processes that step by renaming the resource y_α to x_β ($[x_\beta/y_\alpha]$), then $\mathcal{A} = \mathcal{A}'$.

Lemma 22 (Building minimal sending configuration).
Consider $\Delta \Vdash \mathcal{D}_2 :: K$, a set of channels $\Upsilon_1 \subseteq \Delta, K$ and $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}'_2$ for some $\Upsilon_2 \supseteq \Upsilon_1$. There exists a set Υ and a configuration $\mathcal{D}''_2$ such that $\Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2$ and $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}''_2$ and $\forall \mathcal{D}^1_2, \Upsilon_3 \supseteq \Upsilon_1.\, \mathsf{if}\, \mathcal{D}_2 \mapsto^{*\Upsilon_3} \mathcal{D}^1_2\, \mathsf{then}\, \mathcal{D}''_2 \mapsto^{*\Upsilon_3} \mathcal{D}^1_2$. We call $\mathcal{D}''_2$ and Υ the minimal sending configuration and the minimal sending set with respect to Υ_1 and $\mathcal{D}_2$, respectively.

Proof. We first provide an algorithm to build $\mathcal{D}''_2$ and Υ based on the transition steps in $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}'_2$ and the set Υ_1 and we show that $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}''_2$ and $\mathcal{D}'_2 \mapsto^{*\Upsilon_2} \mathcal{D}''_2$ and $\Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2$. Then, we prove that if we apply the algorithm on every $\mathcal{D}^1_2$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}^1_2$ and $\Upsilon_1 \subseteq \Upsilon_3$, we build the same $\mathcal{D}''_2$ and Υ that satisfies $\mathcal{D}''_2 \mapsto^{*\Upsilon_3} \mathcal{D}^1_2$.

For every given Υ_1, $\mathcal{D}_2$, and $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon_2$, Algorithm 1 returns a configuration $\mathcal{D}''$ and set Υ and builds the dynamic transition $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}''_2$ in Tns. Observe that the steps of $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}''_2$ are all local transitions of $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$. Thus, by the confluence lemma (Lem. 21), we know that $\mathcal{D}''_2 \mapsto^{*\Upsilon_2} \mathcal{D}$.

It remains to be shown that the $\mathcal{D}''_2$ that Algorithm 1 builds is independent of the choice of $\mathcal{D}$ and Υ_2, and is uniquely identified for each $\mathcal{D}_2$ and Υ_1. By the confluence lemma, we know that for each $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon_2$, the algorithm initialized set M with the same messages. In the first **for** loop, the algorithm collects the generators of the set M in A and the local steps that produces the set M in X_i. By Corollary 1, the set of generators of a set M is the same for each $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$. Similarly, the set of local steps that produces the set M from its generators is the same despite the choice of the configuration $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$. As a result, we collect the same local transition steps in all X_is for every $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$. The local transition steps in all X_is are those

Algorithm 1 Building the minimal sending configuration

Require: A set Υ_1 of channels, configuration $\mathcal{D}_2$, and a configuration $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon_2$.
Ensure: A set Υ, and a configuration $\mathcal{D}_2''$ with $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ and $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2$.
  $S :=$ `the local transition steps in` $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$
  $i := 0$
  $X_0 := \emptyset$
  $M :=$ `the messages in` $\mathcal{D}$ `along` Υ_1
  $A := \emptyset$
  while $(M \neq \emptyset)$ **do**
    for $(s \, \mathrm{in} \, S)$ **do**
      if $(\ \exists p \in post(s).\ p \in M)$ **then** ▷ post(s) is the list on the right-hand side of s.
        $A := A \cup \{pre(s)\}$ ▷ pre(s) is the set of processes on the left-hand side of s.
        $X_i := X_i \cup \{s\}$
      end if
    end for
    $i := i + 1$
    $X_i = \emptyset$
    $M := A$
    $A := \emptyset$
  end while

  $\mathsf{Cfg} := \mathcal{D}_2$
  $\mathsf{Tns} := \cdot$
  $i = i - 1$
  while $(i >= 0)$ **do**
    for (s in X_i) **do**
      Cfgpost := the global post-state when the local step s applies to the configuration Cfg.
      Tns.append(Cfg $\mapsto$ Cfgpost)
      Cfg := Cfgpost
    end for
    $i = i - 1$
  end while
  $\mathcal{D}_2'' := \mathsf{Cfg}$
  $\Upsilon := \mathsf{Send}(\mathcal{D}_2'')$ ▷ Υ is a set of channels along which $\mathcal{D}_2''$ is ready to send.
  return $\mathcal{D}_2'', \mathsf{Tns}, \Upsilon$

with which we construct $\mathcal{D}_2''$ from $\mathcal{D}_2$ and from $\mathcal{D}_2''$ we can uniquely identify the set Υ, and the proof is complete.

H.2 Backward closure

Lemma 23 (Backward closure on the second run). *The second run enjoys backward closure:*

1. *If* $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$ *and for* $\mathcal{D}_2'' \in \mathsf{Tree}(\Delta \Vdash K)$*, we have* $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$ *then* $(\mathcal{D}_1; \mathcal{D}_2'') \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.
2. *If* $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;y_\alpha}$ *and for* $\mathcal{D}_2'' \in \mathsf{Tree}(\Delta \Vdash K)$*, we have* $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$ *with* $\mathcal{D}_2''$ *sending along channel* y_α*, then* $(\mathcal{D}_1; \mathcal{D}_2'') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;y_\alpha}$.
3. *If* $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{y_\alpha;\cdot}$ *and for* $\mathcal{D}_2'' \in \mathsf{Tree}(\Delta \Vdash K)$*, we have* $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$*, then* $(\mathcal{D}_1; \mathcal{D}_2'') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{y_\alpha;\cdot}$.

Proof. We prove the first statement separately and then use it to prove the second and third statements.

1. If $k = 0$, the proof is trivial. Consider $k = m + 1$. By assumption, we have $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1}$. By line (12) of the logical relation we know

   $(\star)\ \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\ \mathsf{if}\ \mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1',\ \mathsf{then}\ \exists \Upsilon_2, \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Upsilon_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Theta_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

   Consider $\mathcal{D}_2''$, for which by the assumption we have $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$. Our goal is to prove $(\mathcal{D}_1; \mathcal{D}_2'') \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1}$. We need to show that

   $(\dagger)\ \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\ \mathsf{if}\ \mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1',\ \mathsf{then}\ \exists \Upsilon_2, \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Upsilon_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Theta_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

   With a $\forall$-Introduction, an if-Introduction on the goal followed by a $\forall$-Elimination and an if-Elimination on the assumption, we get the assumption

   $(\star')\ \exists \Upsilon_2, \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Upsilon_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Theta_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

   and the goal

   $(\dagger')\ \exists \Upsilon_2,\ \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Upsilon_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}\ \mathsf{and}$
   $\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Theta_1.\ \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}^\xi_\Psi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

   We apply an $\exists$-Elimination on the assumption to get Υ_2 and $\mathcal{D}_2'$ that satisfies the conditions and use the same Υ_2 and $\mathcal{D}_2'$ to instantiate the existential quantifier in the goal. Since $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$, we get $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'$, and the proof is complete.
2. The proof is by cases on the row of the logical relation that makes the assumption $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;y_\alpha}$ true. Here we only consider an interesting cases, the proof of other cases is similar.

Row 4. By the conditions of this row, we know that $\Delta = \Delta', \Delta''$, and $K = y_\alpha{:}A \otimes B$. Moreover, we have

$$\mathcal{D}_1 = \mathcal{D}_1' \mathcal{T}_1 \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c) \text{ for } \mathcal{T}_1 \in \mathsf{Tree}_\Psi(\Delta'' \Vdash x_\beta{:}A),$$

$$\mathcal{D}_2 = \mathcal{D}_2' \mathcal{T}_2 \mathbf{msg}(\mathbf{send}, x_\beta^c\, y_\alpha^c) \text{ for } \mathcal{T}_2 \in \mathsf{Tree}_\Psi(\Delta'' \Vdash x_\beta{:}A),$$

$$(\dagger_1)\ (\mathcal{T}_1; \mathcal{T}_2) \in \mathcal{E}_\Psi^\xi[\![\Delta'' \Vdash x_\beta{:}A[c]]\!]^k, \text{ and}$$

$$(\dagger_2)\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{E}_\Psi^\xi[\![\Delta' \Vdash y_{\alpha+1}{:}B]\!]^k$$

These are also the statements that we need to prove when replacing $\mathcal{D}_2$ with $\mathcal{D}_2''$. By the assumption that $\mathcal{D}_2''$ is sesstion-typed and sends along y, uniqueness of channels, and $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$ we get

$$\mathcal{D}_2'' = \mathcal{D}_2^1 \mathcal{T}_2^1 \mathbf{msg}(\mathbf{send}, x_\beta^c\, y_\alpha^c) \text{ for } \mathcal{T}_2^1 \in \mathsf{Tree}_\Psi(\Delta'' \Vdash x_\beta{:}A[c])$$

Moreover, since $\mathcal{T}_2^1$ and $\mathcal{D}_2^1$ are disjoint sub-trees with the common parent $\mathbf{msg}(\mathbf{send}, x_\beta^c\, y_\alpha^c)$ and cannot communicate with each other internally, we have $\mathcal{T}_2^1 \mapsto^* \mathcal{T}_2$ and $\mathcal{D}_2^1 \mapsto^* \mathcal{D}_2'$.
Now we can apply the the result of part (1) of this lemma on $(\dagger_1)$ and $(\dagger_2)$ to get

$$(\dagger_1')\ (\mathcal{T}_1; \mathcal{T}_2^1) \in \mathcal{E}_\Psi^\xi[\![\Delta'' \Vdash x_\beta{:}A[c]]\!]^k, \text{ and}$$

$$(\dagger_2')\ (\mathcal{D}_1'; \mathcal{D}_2^1) \in \mathcal{E}_\Psi^\xi[\![\Delta' \Vdash y_{\alpha+1}{:}B]\!]^k$$

and the proof of this subcase is complete.

3. The proof is by considering the row of the logical relation that ensures the assumption $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]_{y_\alpha;\cdot}^{k+1}$. We provide the detailed proof for an interesting case, the proof of other cases is similar.

Row 5. By the conditions of this row, we know that $K = y_\alpha{:}A \multimap B$. We have $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}_\Psi(\Delta \Vdash y_\alpha{:}A \multimap B)$ and

$$(\dagger)\, \forall\, x_\beta \notin dom(\Delta, K).\, (\mathcal{D}_1 \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c); \mathcal{D}_2 \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)) \in \mathcal{E}_\Psi^\xi[\![\Delta, x_\beta{:}A \Vdash y_{\alpha+1}{:}B]\!]^k$$

These are also the statements that we need to prove when replacing $\mathcal{D}_2$ with $\mathcal{D}_2''$. The first tree statement is straightforward by the assumption that $\mathcal{D}_2''$ is session-typed. Using the local transition steps, we get $\mathcal{D}_2'' \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c) \mapsto^* \mathcal{D}_2 \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)$. We can apply the result of part (1) of this lemma on $(\dagger)$ to get

$$(\dagger')\, \forall\, x_\beta \notin dom(\Delta, K).\, (\mathcal{D}_1 \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c); \mathcal{D}_2'' \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)) \in \mathcal{E}_\Psi^\xi[\![\Delta, x_\beta{:}A \Vdash y_{\alpha+1}{:}B]\!]^k$$

which completes the proof of this case.

Row 9. By the conditions of this row, we know that $\Delta = \Delta', y_\alpha{:}A \otimes B$. We have
$\forall x_\beta \notin dom(\Delta;, y_\alpha{:}A \otimes B, K).\, (\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}_\Psi(\Delta', y_\alpha{:}A \otimes B \Vdash K)$ and

$$(\dagger)\, (\mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c) \mathcal{D}_1; \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c) \mathcal{D}_2) \in \mathcal{E}_\Psi^\xi[\![\Delta', x_\beta{:}A, y_{\alpha+1}{:}B \Vdash K]\!]^k$$

These are also the statements that we need to prove when replacing $\mathcal{D}_2$ with $\mathcal{D}_2''$. The first tree statement is straightforward by the assumption that $\mathcal{D}_2''$ is session-typed. Using the local dynamic steps we get $\mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)\mathcal{D}_2'' \mapsto^* \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)\mathcal{D}_2$. We apply the result of part (1) of this lemma on † to get

$$(\dagger')\, \forall x_\beta \notin dom(\Delta;, y_\alpha{:}A\otimes B, K).\, (\mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)\mathcal{D}_1; \mathbf{msg}(\mathbf{send} x_\beta^c\, y_\alpha^c)\mathcal{D}_2'') \in \mathcal{E}_\Psi^\xi[\![\Delta', x_\beta{:}A, y_{\alpha+1}{:}B \Vdash K]\!]^k$$

H.3 Forward closure

Lemma 24 (Forward closure on the first run). *Consider $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$ and $\mathcal{D}_1 \mapsto^* \mathcal{D}_1''$. We have $(\mathcal{D}_1''; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

Proof. If $k = 0$ the proof is trivial. Consider $k = m+1$. By assumption, we have $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1}$.

By line (12) of the logical relation we get

$\star\ \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\,\mathsf{if}\ \mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, $\mathsf{then}\ \exists \Upsilon_2, \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \mathsf{and}$
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\,\mathbf{if}\ x_\alpha \in \Upsilon_1.\,\mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\,\mathbf{if}\ x_\alpha \in \Theta_1.\,\mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

Consider $\mathcal{D}_1''$, for which by the assumption we have $\mathcal{D}_1 \mapsto^* \mathcal{D}_1''$. Our goal is to prove $(\mathcal{D}_1''; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1}$. We need to show that

$\dagger,\ \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\,\mathsf{if}\ \mathcal{D}_1'' \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, $\mathsf{then}\ \exists \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \mathsf{and}$
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\,\mathbf{if}\ x_\alpha \in \Upsilon_1.\,\mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\ \mathbf{if}\ x_\alpha \in \Theta_1.\,\mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

With a $\forall$-Introduction and an if-Introduction on the goal, we assume $\mathcal{D}_1'' \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$. By assumption of $\mathcal{D}_1 \mapsto^* \mathcal{D}_1''$ we get $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$. We use this to apply $\forall$-Elimination and if- Elimination on the assumption, and get

$\star'\ \exists \Upsilon_2, \mathcal{D}_2'\ \mathsf{such\ that}\ \mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'\ \mathsf{and} \Upsilon_1 \subseteq \Upsilon_2\ \mathsf{and}$
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\,\mathbf{if}\ x_\alpha \in \Upsilon_1.\,\mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\,\mathbf{if}\ x_\alpha \in \Theta_1.\,\mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

Which exactly matches our goal and the proof is complete.

Lemma 25 (Forward closure on the second run with some specific conditions). *Consider $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$ and $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ such that if $\mathcal{D}_1$ sends along the set Υ_1, we have $\Upsilon_1 \subseteq \Upsilon$ and also $\mathcal{D}_2''$ is the minimal configuration built by Lem. 22 given the set Υ_1 and configuration $\mathcal{D}_2$. We have $(\mathcal{D}_1; \mathcal{D}_2'') \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

Proof. If $k = 0$ the proof is trivial. Consider $k = m+1$. By assumption, we have $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1}$. By line (12) of the logical relation we get

$\star\ \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.$ if $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, then $\exists \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Theta_1.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

Consider $\mathcal{D}_2''$, for which by the assumption we have $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$. Our goal is to prove $(\mathcal{D}_1; \mathcal{D}_2'') \in \mathcal{E}[\![\Delta \Vdash K]\!]^{m+1}$. We need to show that

$\dagger\ \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.$ if $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, then $\exists \mathcal{D}_2'$ such that $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and$\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Theta_1.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

With an if-Introduction on the goal followed by an if- Elimination on the assumption, we get the assumption

$\star'\ \exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_2'$ and$\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Theta_1.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

and the goal

$\dagger'\ \exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and$\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Theta_1.\, x(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}_\Psi^\xi[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

We apply an $\exists$-Elimination on the assumption to get $\mathcal{D}_2'$ that satisfies the conditions, i.e., $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$. We use the same $\mathcal{D}_2'$ to instantiate the existential quantifier in the goal, we need to show that $\mathcal{D}_2'' \mapsto^{*\Upsilon_1} \mathcal{D}_2'$. Since $\mathcal{D}_2''$ is the minimal configuration built for Υ_1 and $\mathcal{D}_2$, and $\Upsilon_1 \subseteq \Upsilon_2$, by Lemma 22 we get $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'$, and the proof is complete.

I Moving existential and compositionality

This section introduces two lemmas, Lem. 26 and Corollary 2, which are instrumental in proving transitivity (see § J) and adequacy (see § K).

I.1 Moving existential over universal quantifier

Lemma 26 (Moving existential over universal quantifier). *if we have*

$(\dagger)\ \forall m. \forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.$ if $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, then $\exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\ x_\alpha \in \Theta_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$

then

$\forall\, \Upsilon_1, \Theta_1, \mathcal{D}_1'.$if $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, then $\exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall\, x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\; x_\alpha \in \Upsilon_1.\, \mathbf{then}\; \forall\, k.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;x_\alpha}$ and
$\forall\, x_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\; x_\alpha \in \Theta_1.\, \mathbf{then}\; \forall k.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{x_\alpha;\cdot}$.

Proof. First put $m = 1$ to apply $\forall\,\mathbf{E}$. on the assumption (instantiating $\forall\, m$ only). We get as an assumption

$(\dagger')$ $\forall\, \Upsilon_1, \Theta_1, \mathcal{D}_1'.$ if $\mathcal{D}_1 \mapsto^{*\Upsilon_1;\Theta_1} \mathcal{D}_1'$, then $\exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall\, x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\; x_\alpha \in \Upsilon_1.\, \mathbf{then}\; (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{0+1}_{\cdot;x_\alpha}$ and
$\forall\, x_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\; x_\alpha \in \Theta_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{0+1}_{x_\alpha;\cdot}$.

Next, apply a $\forall\mathbf{I}$. and **if I**. on the goal followed by a corresponding $\forall\mathbf{E}$. and **if E**. on the assumption $(\dagger')$. Now apply $\exists\mathbf{E}$. on the assumption $(\dagger')$ to get $\mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$. Given $\mathcal{D}_2'$, by Lem. 22, we can build the minimal $\mathcal{D}_2''$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ and $\Upsilon_1 \subseteq \Upsilon$. Moreover, we know that for every $\mathcal{D}$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_3} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon_3$, we get $\mathcal{D}_2'' \mapsto^{*\Upsilon_3} \mathcal{D}$.

We use this minimal $\mathcal{D}_2''$ to instantiate the existential ($\exists\,\mathbf{I}$.) in the goal, and use $\forall\,\mathbf{I}$. on the goal. In particular, we instantiate k with an arbitrary natural number. The goals are:

$$(\mathcal{D}_1'; \mathcal{D}_2'') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;x_\alpha} \text{ and } (\mathcal{D}_1'; \mathcal{D}_2'') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{x_\alpha;\cdot}.$$

Next, we instantiate the $\forall$ quantifier in the original assumption $(\dagger)$ once again, this time with $m = k$ followed by a $\forall\mathbf{E}$, and **ifE** instantiating the quantifiers with similar $\mathcal{D}_1'$, Υ_1, and Θ_1 as the first time. We get as an assumption:

$(\dagger'')$ $\exists \Upsilon_2, \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall\, x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\; x_\alpha \in \Upsilon_1.\, \mathbf{then}\; (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;x_\alpha}$ and
$\forall\, x_\alpha \in \mathbf{In}(\Delta \Vdash K).\mathbf{if}\; x_\alpha \in \Theta_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{x_\alpha;\cdot}$.

Next, apply $\exists\,\mathbf{I}$. to get a Υ' and $\mathcal{D}'$ that satisfies the conditions, i.e., $\mathcal{D} \mapsto^{*\Upsilon'} \mathcal{D}'$ and $\Upsilon_1 \subseteq \Upsilon'$. Instantiate x_α as those chosen for the goal. We have as assumptions:

$$(\dagger''')\; (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;x_\alpha} \text{ and } (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{x_\alpha;\cdot}.$$

Since $\mathcal{D}_2''$ is the minimal configuration built for Υ_1 and $\mathcal{D}_2$, we know that $\mathcal{D}_2'' \mapsto^{*\Upsilon'} \mathcal{D}'$. We can apply the backward closure results of Lem. 23 to get the goal from the assumptions $(\dagger''')$, and this completes the proof.

I.2 Compositionality

Corollary 2 (Compositionality). $\forall m.\, (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta, u_\alpha{:}T \Vdash K]\!]^m$ *iff for all* $\mathcal{T}_2$ *and* $\mathcal{T}_2$ *s.t.* $\dagger_2\, \forall m.\, (\mathcal{T}_1; \mathcal{T}_2) \in \mathcal{E}[\![\Delta' \Vdash u_\alpha{:}T]\!]^m$ *we have* $\dagger\, \forall k.\, (\mathcal{T}_1\mathcal{D}_1; \mathcal{T}_2\mathcal{D}_2) \in \mathcal{E}[\![\Delta', \Delta \Vdash K]\!]^k$.

Proof. The left to right direction is a corollary of Lemma 27 in which we compose multiple configurations instead of just two. For the right to left direction, we put $\mathcal{T}_i = \cdot$, and $\Delta' = u_\alpha{:}T$, and the rest of the proof is straightforward.

Lemma 27 (Generalized compositionality). *For $i \in \{1,2\}$, and index set I consider session typed tree-shaped configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ such that $n \in I$ and their compositions form session typed tree-shaped configurations $\Delta \Vdash \mathcal{D}_i :: K$, i.e., $\mathcal{D}_i = \{\mathcal{B}_i^n\}_{n\in I}$. If for all $n \in I$, we have $\dagger_n\ \forall m.\, (\mathcal{B}_1^n;\mathcal{B}_2^n) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ then $\dagger\,\forall k.\, (\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

Proof. Our goal is to prove the following:

> For all index set I and all session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ with $n \in I$ such that $\dagger_1\ \forall m.\, (\mathcal{B}_1^n;\mathcal{B}_2^n) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have $\dagger\ \forall k.\, (\{\mathcal{B}_1^n\}_{n\in I};\{\mathcal{B}_2^n\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.

This is equivalent to the following statement which we prove:

> For all natural numbers, k, and for all index set I, and for all session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ such that $\dagger_n\ \forall m.\, (\mathcal{B}_1^n;\mathcal{B}_2^n) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have $\dagger'\ (\{\mathcal{B}_1^n\}_{n\in I};\{\mathcal{B}_2^n\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.

We proceed the proof by an induction on k.

Base case ($k = 0$). The proof is straightforward, since by the definition of the logical relation for session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ with $\Delta \Vdash \{\mathcal{B}_i^n\}_{n\in I} :: K$, we have $(\{\mathcal{B}_1^n\}_{n\in I};\{\mathcal{B}_2^n\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^0$.

Inductive case ($k = k'+1$). Our goal is to prove the following:

> For all index set I, and for all session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ where $n \in I$ such that $\dagger_n\ \forall m.\, (\mathcal{B}_1^n;\mathcal{B}_2^n) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have $\dagger'\ (\{\mathcal{B}_1^n\}_{n\in I};\{\mathcal{B}_2^n\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{k'+1}$

where $\dagger'$ is defined in line (12) of the logical relation as

$$\begin{array}{l}\forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\, \forall j \in \mathbb{N}.\, \textbf{if}\ \{\mathcal{B}_1^n\}_{n\in I} \mapsto^{j\,\Upsilon_1;\Theta_1} \mathcal{D}_1'\ \textsf{then}\ \exists \Upsilon_2, \mathcal{D}_2'\ \textsf{such that}\ \{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\,\Upsilon_2} \mathcal{D}_2'\ \textsf{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \textsf{and}\\ \quad \forall x_\gamma \in \textbf{Out}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Upsilon_1.\, \textbf{then}\ (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}\ \textsf{and}\\ \quad \forall x_\gamma \in \textbf{In}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Theta_1.\, \textbf{then}\ (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

Again, we rewrite the above goal as an equivalent statement as follows:

> For all natural numbers j, for all I, and for all session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ with $n \in I$ such that $\dagger_n\ \forall m.\, (\mathcal{B}_1^n;\mathcal{B}_2^n) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have
>
> $$\begin{array}{l}\forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\ \textbf{if}\ \{\mathcal{B}_1^n\}_{n\in I} \mapsto^{j\,\Upsilon_1;\Theta_1} \mathcal{D}_1'\ \textsf{then}\ \exists \Upsilon_2, \mathcal{D}_2'\ \textsf{such that}\ \{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\,\Upsilon_2} \mathcal{D}_2'\ \textsf{and}\ \Upsilon_1 \subseteq \Upsilon_2\ \textsf{and}\\ \quad \forall x_\gamma \in \textbf{Out}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Upsilon_1.\, \textbf{then}\ (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}\ \textsf{and}\\ \quad \forall x_\gamma \in \textbf{In}(\Delta \Vdash K).\textbf{if}\ x_\gamma \in \Theta_1.\, \textbf{then}\, (\mathcal{D}_1';\mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

We proceed the proof by a nested induction on j.

Base case ($j = 0$). Consider an arbitrary index set I and an arbitrary session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ that satisfy the conditions $\dagger_n$. We need to show that

$$\begin{array}{l}\forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\, \textbf{if}\, \{\mathcal{B}_1^n\}_{n\in I} \mapsto^{0\Upsilon_1;\Theta_1} \mathcal{D}_1' \,\textsf{then}\, \exists \Upsilon_2, \mathcal{D}_2' \,\textsf{such that}\, \{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}_2' \,\textsf{and}\, \Upsilon_1 \subseteq \Upsilon_2 \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\, x_\gamma \in \Upsilon_1.\, \textbf{then}\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if}\, x_\gamma \in \Theta_1.\, (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

Consider an arbitrary Υ_1, Θ_1, ans $\mathcal{D}_1'$, and apply **If I**. on the goal. By the assumption $\{\mathcal{B}_1^n\}_{n\in I} \mapsto^{0\Upsilon_1;\Theta_1} \mathcal{D}_1'$ we know that $\mathcal{D}_1' = \{\mathcal{B}_1^n\}_{n\in I}$, and $\{\mathcal{B}_1^n\}_{n\in I}$ sends along Υ_1 and receives along Θ_1. Our goal is to show the following:

$$\begin{array}{l}\star\ \exists \Upsilon_2, \mathcal{D}_2' \,\textsf{such that}\, \{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}_2' \,\textsf{and}\, \Upsilon_1 \subseteq \Upsilon_2 \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\, x_\gamma \in \Upsilon_1.\, \textbf{then}\, (\{\mathcal{B}_1^n\}_{n\in I}; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\textbf{if}\, x_\gamma \in \Theta_1.\, \textbf{then}\, (\{\mathcal{B}_1^n\}_{n\in I}; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

By the definition of the logical relation, and from assumptions $\dagger_n$ for $n \in I$ we get

$$\begin{array}{l}\dagger_n'\ \forall m. \forall \Upsilon_n, \Theta_n, \mathcal{B}_1^{n'}.\, \textsf{if}\, \mathcal{B}_1^n \mapsto^{*\Upsilon_n;\Theta_n} \mathcal{B}_1^{n'} \,\textsf{then}\, \exists \Upsilon_{n_2}, \mathcal{B}_2^{n'} \,\textsf{such that}\, \mathcal{B}_2^n \mapsto^{*\Upsilon_{n_2}} \mathcal{B}_2^{n'} \,\textsf{and}\, \Upsilon_n \subseteq \Upsilon_{n_2} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if}\, x_\gamma \in \Upsilon_n.\, \textbf{then}\, (\mathcal{B}_1^{n'}; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\, \textbf{if}\, x_\gamma \in \Theta_n.\, \textbf{then}\, (\mathcal{B}_1^{n'}; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

By Lem. 26, we get

$$\begin{array}{l}\dagger_n''\ \forall \Upsilon_n, \Theta_n, \mathcal{B}_1^{n'}.\, \textsf{if}\, \mathcal{B}_1^n \mapsto^{*\Upsilon_n;\Theta_n} \mathcal{B}_1^{n'} \,\textsf{then}\, \exists \Upsilon_{n_2} \mathcal{B}_2^{n'} \,\textsf{such that}\, \mathcal{B}_2^n \mapsto^{*\Upsilon_{n_2}} \mathcal{B}_2^{n'} \,\textsf{and}\, \Upsilon_n \subseteq \Upsilon_{n_2} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if}\, x_\gamma \in \Upsilon_n.\, \textbf{then}\, \forall m.\, (\mathcal{B}_1^{n'}; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\, \textbf{if}\, x_\gamma \in \Theta_n.\, \textbf{then} \forall m.\, (\mathcal{B}_1^{n'}; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

We instantiate the for all quantifier in $\dagger_n''$ by $\mathcal{B}_1^n$ and the sets Υ_n and Θ_n along which $\mathcal{B}_1^n$ sends and receives. Note that by definition we have $\Upsilon_1 \subseteq \bigcup\{\Upsilon_n\}_{n\in I}$ and $\Theta_1 \subseteq \bigcup\{\Theta_n\}_{n\in I}$. We get:

$$\begin{array}{l}\exists \Upsilon_{n_2}, \mathcal{B}_2^{n'} \,\textsf{such that}\, \mathcal{B}_2^n \mapsto^{*\Upsilon_{n_2}} \mathcal{B}_2^{n'} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if}\, x_\gamma \in \Upsilon_n.\, \textbf{then}\, \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\textbf{if}\, x_\gamma \in \Theta_n.\, \textbf{then} \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

By existential elimination, for all $n \in I$, we get a $\mathcal{B}_2^{n'}$ and Υ_{n_2} such that $\mathcal{B}_2 \mapsto^{*\Upsilon_{n_2}} \mathcal{B}_2^{n'}$, and $\Upsilon_n \subseteq \Upsilon_{n_2}$ and

$$\begin{array}{l}\dagger_n'''\ \ \forall x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if}\, x_\gamma \in \Upsilon_n.\, \textbf{then}\, \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \,\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\textbf{if}\, x_\gamma \in \Theta_n.\, \textbf{then} \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n'}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

Note that by definition $\mathbf{Out}(\Delta \Vdash K) \subseteq \bigcup_{n\in I} \mathbf{Out}(\Delta^n \Vdash K^n)$ and $\mathbf{In}(\Delta \Vdash K) \subseteq \bigcup_{n\in I} \mathbf{In}(\Delta^n \Vdash K^n)$.
We apply Lem. 22 to get the minimal sending configurations $\mathcal{B}_2^{n''}$ and set Υ'_n for each given Υ_n and $\mathcal{B}_2^n$. We get $\mathcal{B}_2^{n''}$ such that $\mathcal{B}_2^n \mapsto^{*\Upsilon'_n} \mathcal{B}_2^{n''}$. Since these configurations are minimal, for all $n \in I$, we have $\mathcal{B}_2^{n''} \mapsto^{*\Upsilon_{n_2}} \mathcal{B}_2^{n'}$.
Since $\Upsilon_n \subseteq \Upsilon'_n \subseteq \Upsilon_{n_2}$, we can apply the backward closure (Lem. 23) on $\dagger'''_n$ to get

$\dagger^4_n$ $\forall x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \mathbf{if}\ x_\gamma \in \Upsilon_n.\, \mathbf{then}\ \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n''}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma}$ and
$\forall x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\, \mathbf{if}\ x_\gamma \in \Theta_n.\forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n''}) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}$.

Moreover, we apply forward closure on the second run Lem. 25 on assumptions $\dagger_n$ to get for all $n \in I$:

$$\circ_n\ \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n''}) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$$

We can apply the forward closure on the second run since the conditions of Lem. 25 are satisfied, i.e. $\Upsilon_n \subseteq \Upsilon'_n$ and $\mathcal{B}_2^{n''}$ is the minimal sending configuration with respect to $\mathcal{B}_2^n$ and Υ_n.
We build $\mathcal{D}'_2$ to be $\{\mathcal{B}_2^{n''}\}_{n\in I}$. We have $\{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\Upsilon_2} \{\mathcal{B}_2^{n''}\}_{n\in I}$ with $\Upsilon_1 \subseteq \Upsilon_2$, and $\{\mathcal{B}_2^{n''}\}_{n\in I} \mapsto^{*\Upsilon_3} \{\mathcal{B}_2^{n'}\}_{n\in I}$ with $\Upsilon_2 \subseteq \Upsilon_3$. We, then instantiate the existential quantifier in the goal ($\star$) with Υ_2 and $\mathcal{D}'_2$ that we built for which we know $\{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}'_2$. We need to show

$\star'$ $\forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\gamma \in \Upsilon_1.\, \mathbf{then}\ (\{\mathcal{B}_1^n\}_{n\in I}; \{\mathcal{B}_2^{n''}\}_{n\in I}) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\ x_\gamma \in \Theta_1.\, \mathbf{then}\ (\{\mathcal{B}_1^n\}_{n\in I}; \{\mathcal{B}_2^{n''}\}_{n\in I}) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

Part 1. Consider arbitrary $x_\gamma \in \mathbf{Out}(\Delta \Vdash K)$ and assume $x_\gamma \in \Upsilon_1$. By the structure of the configurations, for some $n \in I$, $x_\gamma \in \Delta^n, K^n$ and thus $x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n)$ and $x_\gamma \in \Upsilon_n$. The goal is to prove

$$\star_1\ (\{\mathcal{B}_1^n\}_{n\in I}; \{\mathcal{B}_2^{n''}\}_{n\in I}) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$$

Part 2. Consider arbitrary $x_\gamma \in \mathbf{In}(\Delta \Vdash K)$ and assume $x_\gamma \in \Theta_1$. By the structure of the configurations, for some $n \in I$, $x_\gamma \in \Delta^n, K^n$ and thus $x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n)$ and $x_\gamma \in \Theta_n$. The goal is to prove

$$\star_2\ (\{\mathcal{B}_1^n\}_{n\in I}; \{\mathcal{B}_2^{n''}\}_{n\in I}) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$$

In both parts, we continue the proof by considering the type of x_γ. The type of x_γ determines whether we need to prove **Part 1.** or **Part 2**. We provide the detailed proof for two interesting cases, the proof of the rest of cases is similar.

Subcase 1. $x_\gamma{:}A \otimes B \in K$. This case corresponds to **Part 1.** of the goal in which we have $x_\gamma \in \mathbf{Out}(\Delta \Vdash x_\gamma{:}A \otimes B)$ and $x_\gamma \in \Upsilon_1$. By the structure of the configuration, there exists a tree $\mathcal{B}_1^\kappa$ that provides the root channel

$K = K^\kappa = x_\gamma{:}A \otimes B$. We use assumption $\dagger_n^4$ for that specific channel ($\dagger_\kappa^4$), we have

$$\dagger_\kappa^5\ \forall m.\, (\mathcal{B}_1^\kappa; \mathcal{B}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m+1}_{\cdot;x_\gamma}$$

First, instantiate the for all quantifier with $m = 0$. By **Row 4.** of the logical relation, we have $\Delta^\kappa = \Delta_1^\kappa, \Delta_2^\kappa$ and for some $y_\beta \in \textbf{chnl}$, we have: $\mathcal{B}_1^\kappa = \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma)$ and $\mathcal{B}_2^{\kappa''} = \mathcal{B}_2^{\kappa'''} \mathcal{A}_2 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma)$.
We want to prove

$$\circ'\, \forall m.\, (\mathcal{A}_1, \mathcal{A}_2) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash y_\beta{:}A]\!]^m \quad \textsf{and} \quad \circ''\forall m.\, (\mathcal{B}_1^{\kappa'}, \mathcal{B}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash x_{\gamma+1}{:}B]\!]^m.$$

Consider an arbitrary m given by $\forall\mathbf{I}$ on the goals $\circ'$ and $\circ''$. Once again, instantiate the quantifier in $\dagger_\kappa^5$, this time with the arbitrary m. Again, we get $\Delta^\kappa = \Delta_1^\kappa, \Delta_2^\kappa$ and for some $y_\beta \in \textbf{chnl}$, we have: $\mathcal{B}_1^\kappa = \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma)$ and $\mathcal{B}_2^{\kappa''} = \mathcal{B}_2^{\kappa'''} \mathcal{A}_2 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma)$. Moreover,

$$(\mathcal{A}_1, \mathcal{A}_2) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash y_\beta{:}A]\!]^m \quad \textsf{and} \quad (\mathcal{B}_1^{\kappa'}, \mathcal{B}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash x_{\gamma+1}{:}B]\!]^m.$$

Since the naming in the configuration is unique, we get the above for the same y_β as we got in the case of $m = 0$ and the proof of $\circ'$ and $\circ''$ is complete.
From this we can prove

$$\begin{aligned}&\mathcal{D}_1 = \{\mathcal{B}_1^n\}_{n\in I} = \{\mathcal{B}_1^n\}_{n\in I \& n\neq\kappa}\ \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma) \text{ and}\\ &\mathcal{D}_2' = \{\mathcal{B}_2^{n''}\}_{n\in I} = \{\mathcal{B}_2^{n''}\}_{n\in I \& n\neq\kappa}\ \mathcal{B}_2^{\kappa'''} \mathcal{A}_2 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma)\end{aligned}$$

First, observe that by the structure of the configuration, there is no tree $\mathcal{B}_1^n$ or $\mathcal{B}_2^{n''}$ using $K^\kappa = x_\gamma{:}A \otimes B$ as its resource, i.e., x_γ is the root. We can break down the resources Δ_1^κ and Δ_2^κ as $\Delta_1^\kappa = \Delta_1^{\kappa'}, \Delta_1^{\kappa''}$ and $\Delta_2^\kappa = \Delta_2^{\kappa'}, \Delta_2^{\kappa''}$, such that $\Delta_1^{\kappa'}$ and $\Delta_2^{\kappa'}$ are in the interface of $\mathcal{D}_1$ and $\mathcal{D}_2'$ and $\Delta_1^{\kappa''}$ and $\Delta_2^{\kappa''}$ are the resources provided by other trees. We can partition $I \backslash \{\kappa\}$ into two disjoint sets I_1 and I_2 such that the configurations $\{\mathcal{B}_1^n\}_{n\in I_1}$ and $\{\mathcal{B}_2^{n''}\}_{n\in I_1}$ provide the resources in $\Delta_1^{\kappa''}$ and configurations $\{\mathcal{B}_1^n\}_{n\in I_2}$ and $\{\mathcal{B}_2^{n''}\}_{n\in I_2}$ provide the resources in $\Delta_2^{\kappa''}$. In other words, we have $\Delta = \Delta_1, \Delta_1^{\kappa'}, \Delta_2, \Delta_2^{\kappa'}$ and $K = x_\gamma{:}A \otimes B$ and

$$\begin{array}{llll} (i) & \Delta_1 \Vdash \{\mathcal{B}_1^n\}_{n\in I_1} :: \Delta_1^{\kappa''} & \Delta_1^{\kappa'}, \Delta_1^{\kappa''} \Vdash \mathcal{B}_1^{\kappa'} :: x_{\gamma+1}{:}B & \Delta_1, \Delta_1^{\kappa'} \Vdash \{\mathcal{B}_1^n\}_{n\in I_1} \mathcal{B}_1^{\kappa'} :: x_{\gamma+1}{:}B \\ & \Delta_1 \Vdash \{\mathcal{B}_2^{n''}\}_{n\in I_1} :: \Delta_1^{\kappa''} & \Delta_1^{\kappa'}, \Delta_1^{\kappa''} \Vdash \mathcal{B}_2^{\kappa'''} :: x_{\gamma+1}{:}B & \Delta_1, \Delta_1^{\kappa'} \Vdash \{\mathcal{B}_2^{n''}\}_{n\in I_1} \mathcal{B}_2^{\kappa'''} :: x_{\gamma+1}{:}B \\ (ii) & \Delta_2 \Vdash \{\mathcal{B}_1^n\}_{n\in I_2} :: \Delta_2^{\kappa''} & \Delta_2^{\kappa'}, \Delta_2^{\kappa''} \Vdash \mathcal{A}_1 :: y_\beta{:}A & \Delta_2, \Delta_2^{\kappa'} \Vdash \{\mathcal{B}_1^n\}_{n\in I_2} \mathcal{A}_1 :: y_\beta{:}A \\ & \Delta_2 \Vdash \{\mathcal{B}_2^{n''}\}_{n\in I_2} :: \Delta_2^{\kappa''} & \Delta_2^{\kappa'}, \Delta_2^{\kappa''} \Vdash \mathcal{A}_2 :: y_\beta{:}A & \Delta_2, \Delta_2^{\kappa'} \Vdash \{\mathcal{B}_2^{n''}\}_{n\in I_2} \mathcal{A}_2 :: y_\beta{:}A \end{array}$$

We also have

$$\begin{aligned}\circ'''\ &\mathcal{D}_1 = \{\mathcal{B}_1^n\}_{n\in I} = \{\mathcal{B}_1^n\}_{n\in I \& n\neq\kappa}\ \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma) = \{\mathcal{B}_1^n\}_{n\in I_1}\, \{\mathcal{B}_1^n\}_{n\in I_2}\, \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma) \text{ and}\\ &\mathcal{D}_2' = \{\mathcal{B}_2^{n''}\}_{n\in I} = \{\mathcal{B}_2^{n''}\}_{n\in I \& n\neq\kappa}\ \mathcal{B}_2^{\kappa'''} \mathcal{A}_2 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma) = \{\mathcal{B}_2^{n''}\}_{n\in I_1} \{\mathcal{B}_2^{n''}\}_{n\in I_2}\ \mathcal{B}_2^{\kappa'''} \mathcal{A}_2 \textbf{msg}(\textbf{send}\, y_\beta x_\gamma)\end{aligned}$$

Recall that earlier we proved

$$\circ'\, \forall m.\, (\mathcal{A}_1, \mathcal{A}_2) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash y_\beta{:}A]\!]^m \quad \textsf{and} \quad \circ''\forall m.\, (\mathcal{B}_1^{\kappa'}, \mathcal{B}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash x_{\gamma+1}{:}B]\!]^m,$$

and we also have for all $n \in I_1 \cup I_2$,

$$\circ_2\ \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n''}) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$$

We can apply the induction hypothesis on the smaller index k', (i), $\circ''$, and $\circ_2$ for those trees indexed in I_1 to get

$$\circ_3\ (\{\mathcal{B}_1^n\}_{n\in I_1}\, \mathcal{B}_1^{\kappa'}; \{\mathcal{B}_2^{n''}\}_{n\in I_1}\, \mathcal{B}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_1, \Delta_1^{\kappa'} \Vdash x_{\gamma+1}{:}B]\!]^{k'}.$$

Similarly, we can apply the induction hypothesis on the smaller index k', (ii), $\circ'$, and $\circ_2$ for those trees indexed in I_2 to get

$$\circ_3\ (\{\mathcal{B}_1^n\}_{n\in I_2}\, \mathcal{A}_1; \{\mathcal{B}''^n_2\}_{n\in I_2}\, \mathcal{A}_2) \in \mathcal{E}[\![\Delta_1, \Delta_2^{\kappa'} \Vdash y_\beta{:}A]\!]^{k'}.$$

By **Row 4.** of the logical relation, this is enough to establish the goal

$$\star_1\ (\mathcal{D}_1; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$$

Subcase 2. $x_\gamma{:}A \otimes B \in \Delta$, i.e., $\Delta = \Delta', x_\gamma{:}A \otimes B$. This case corresponds to **Part 2.** of the goal in which we have $x_\gamma \in \mathbf{In}(\Delta \Vdash K)$ and $x_\gamma \in \Theta_1$, and the goal is to prove $(\{\mathcal{B}_1^n\}_{n\in I}; \{\mathcal{B}_2^{n''}\}_{n\in I}) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.
By the structure of the configuration, for some index $\kappa \in I$, we have $x_\gamma{:}A \otimes B \in \Delta^\kappa$, i.e. $\Delta^\kappa = \Delta^{\kappa'}, x_\gamma{:}A \otimes B$.
By assumption $\dagger^4_\kappa$, we have

$$\forall\, m.\, (\mathcal{B}_1^\kappa; \mathcal{B}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^{\kappa'}, x_\gamma{:}A \otimes B \Vdash K^\kappa]\!]^{m+1}_{x_\gamma;\cdot}$$

By **Row 9.** of the logical relation, we get:

$$\dagger^5_\kappa\ \forall y_\beta \notin \mathit{dom}(\Delta^{\kappa'}, x_\gamma{:}A{\otimes}B, K^\kappa)\, \forall m.\, (\mathbf{msg}(\mathbf{send}\, y_\beta x_\gamma)\mathcal{B}_1^\kappa; \mathbf{msg}(\mathbf{send}\, y_\beta x_\gamma)\mathcal{B}_2^{\kappa''}) \in \mathcal{E}[\![\Delta^{\kappa'}, y_\beta{:}A, x_{\gamma+1}{:}B \Vdash K^\kappa]\!]^m$$

Moreover, we have

$$\Delta, y_\beta{:}A, x_\gamma{:}B \Vdash \mathbf{msg}(\mathbf{send}\, y_\beta x_\gamma)\{\mathcal{B}_1^n\}_{n\in I} :: K \qquad \Delta, y_\beta{:}A, x_\gamma{:}B \Vdash \mathbf{msg}(\mathbf{send}\, y_\beta x_\gamma)\{\mathcal{B}_2^{n''}\}_{n\in I} :: K$$

Recall that for all $n \in I \backslash \{\kappa\}$ we also have

$$\circ_2\ \forall m.\, (\mathcal{B}_1^n; \mathcal{B}_2^{n''}) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$$

We can apply the induction hypothesis on the smaller index k', $\circ_2$, and $\dagger^5_\kappa$ to get

$$\forall y_\beta \notin \mathit{dom}(\Delta, x_\gamma{:}A{\otimes}B, K).(\mathbf{msg}(\mathbf{send}\, y_\beta x_\gamma)\{\mathcal{B}_1^n\}_{n\in I}; \mathbf{msg}(\mathbf{send}\, y_\beta x_\gamma)\{\mathcal{B}_1^{n''}\}_{n\in I}) \in \mathcal{E}[\![\Delta, y_\beta{:}A, x_{\gamma+1}{:}B \Vdash K]\!]^{k'}.$$

By **Row 9.** of the logical relation, this is enough to establish the goal.

Inductive case ($j = j'+1$). Consider an arbitrary index set I and an arbitrary session-typed configurations $\Delta^n \Vdash \mathcal{B}_i^n :: K^n$ that satisfy the conditions $\dagger_n$. We need to show that

$$\begin{array}{l}\forall \Upsilon_1, \Theta_1, \mathcal{D}'_1.\, \textbf{if}\; \{\mathcal{B}^n_1\}_{n\in I} \mapsto^{j'+1\Upsilon_1;\Theta_1} \mathcal{D}'_1 \;\textsf{then}\; \exists \Upsilon_2, \mathcal{D}'_2 \;\textsf{such that}\; \{\mathcal{B}^n_2\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}'_2 \;\textsf{and}\; \Upsilon_1 \subseteq \Upsilon_2 \;\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\; x_\gamma \in \Upsilon_1.\, \textbf{then}\; (\mathcal{D}'_1;\mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma} \;\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\textbf{if}\; x_\gamma \in \Theta_1.\, \textbf{then}\, (\mathcal{D}'_1;\mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

We apply a $\forall$**I**. and **if I**. on the goal: consider an arbitrary Υ_1, Θ_1, and $\mathcal{D}'_1$ and the first step of $\{\mathcal{B}^n_1\}_{n\in I} \mapsto^{j'+1\Upsilon_1;\Theta_1} \mathcal{D}'_1$. There are two cases to consider:

Case 1. the first step is an internal step but not a communication between the subtrees.

Without loss of generality, let's assume that the communication is internal to $\mathcal{B}^\kappa_1$ for some $\kappa \in I$ and all other trees $\mathcal{B}^n_1$ for $n \neq \kappa$ remain intact, i.e. we have $\{\mathcal{B}^n_1\}_{n\in I} = \{\mathcal{B}^n_1\}_{n\in I_1}\, \mathcal{B}^\kappa_1 \{\mathcal{B}^n_1\}_{n\in I_2}$ and

$$\{\mathcal{B}^n_1\}_{n\in I_1}\, \mathcal{B}^\kappa_1 \{\mathcal{B}^n_1\}_{n\in I_2} \;\mapsto \{\mathcal{B}^n_1\}_{n\in I_1}, \mathcal{B}^{\kappa'}_1 \{\mathcal{B}^n_1\}_{n\in I_2} \;\mapsto^{j'\Upsilon_1;\Theta_1} \mathcal{D}'_1.$$

By forward closure (Lem. 24) on the assumption $\dagger_\kappa$ (i.e.,$\dagger_n$for $n = \kappa$), we get

$$\dagger'_\kappa \;\forall m.\, (\mathcal{B}^{\kappa'}_1;\mathcal{B}_2) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m.$$

Recall that by $\dagger_n$, we also have for $n \neq \kappa$

$$\forall m.\, (\mathcal{B}^n_1;\mathcal{B}^n_2) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m.$$

We can apply the induction hypothesis on the number of steps j' to get:

$$\begin{array}{l}\forall \Upsilon_1, \Theta_1, \mathcal{D}'_1.\, \forall \Upsilon_1.\, \textbf{if}\; \{\mathcal{B}^n_1\}_{n\in I_1}\, \mathcal{B}^{\kappa'}_1 \{\mathcal{B}^n_1\}_{n\in I_2} \mapsto^{j'\Upsilon_1;\Theta_1} \mathcal{D}'_1 \;\textsf{then}\; \exists \Upsilon_2\, \mathcal{D}'_2.\; \{\mathcal{B}^n_2\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}'_2 \;\textsf{and}\; \Upsilon_1 \subseteq \Upsilon_2 \;\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\; x_\gamma \in \Upsilon_1.\, \textbf{then}\; (\mathcal{D}'_1;\mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma} \;\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if}\; x_\gamma \in \Theta_1.\, \textbf{then}\, (\mathcal{D}'_1;\mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

Since $\{\mathcal{B}^n_1\}_{n\in I_1}\, \mathcal{B}^\kappa_1 \{\mathcal{B}^n_1\}_{n\in I_2} \;\mapsto \{\mathcal{B}^n_1\}_{n\in I_1}, \mathcal{B}^{\kappa'}_1 \{\mathcal{B}^n_1\}_{n\in I_2} \;\mapsto^{j'\Upsilon_1;\Theta_1} \mathcal{D}'_1$, we can prove the goal:

$$\begin{array}{l}\exists \Upsilon_2, \mathcal{D}'_2 \;\textsf{such that}\; \{\mathcal{B}^n_2\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}'_2 \;\textsf{and}\; \Upsilon_1 \subseteq \Upsilon_2 \;\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\; x_\gamma \in \Upsilon_1.\, \textbf{then}\; (\mathcal{D}'_1;\mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma} \;\textsf{and}\\ \quad \forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if}\; x_\gamma \in \Theta_1.\, (\mathcal{D}'_1;\mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

which completes the proof of this case.

Case 2. the first step is a communication between two sub-configurations.

Without loss of generality, we assume that the communication is between trees indexed by $\kappa, \lambda \in I$, i.e., $\mathcal{B}^\kappa_1$ offers a resource $u_\alpha{:}T$ and $\mathcal{B}^\lambda_1$ uses the resource, and there is a message available along u_α that is received in the first step. The proof proceeds by case analysis on type of $u^c_\alpha{:}T$. We provide the detailed proof for one case. The proof of the rest of the cases is similar.

Subcase 1. $T = A \otimes B$, i.e., we have

$$\begin{array}{lll}(i) & \Delta^\kappa \Vdash \mathcal{B}^\kappa_1 :: K^\kappa & \textsf{where}\; K^\kappa = u_\alpha{:}A \otimes B\\ & \Delta^\kappa \Vdash \mathcal{B}^\kappa_2 :: K^\kappa & \\ (ii) & \Delta^\lambda \Vdash \mathcal{B}^\lambda_1 :: K^\lambda & \textsf{where}\; \Delta^\lambda = \Delta^{\lambda'}, u_\alpha{:}A \otimes B\\ & \Delta^\lambda \Vdash \mathcal{B}^\lambda_2 :: K^\lambda & \end{array}$$

By the assumptions of this case, we know that there is a message sent along $u^c_\alpha{:}A \otimes B$ in $\mathcal{B}^\kappa_1$ that is received by a process in $\mathcal{B}^\lambda_1$, i.e., $\mathcal{B}^\kappa_1 = \mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)$ and

$$\{\mathcal{B}^n_1\}_{n\in I} = \{\mathcal{B}^n_1\}_{n\in I_1}, \mathcal{B}^\kappa_1 \mathcal{B}^\lambda_1 \{\mathcal{B}^n_1\}_{n\in I_2} = \{\mathcal{B}^n_1\}_{n\in I_1}, \mathcal{B}^{\kappa'}_1 \mathcal{A}_1\, \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mathcal{B}^\lambda_1 \{\mathcal{B}^n_1\}_{n\in I_2} \text{ and}$$
$$\{\mathcal{B}^n_1\}_{n\in I_1}, \mathcal{B}^{\kappa'}_1 \mathcal{A}_1\, \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mathcal{B}^\lambda_1 \{\mathcal{B}^n_1\}_{n\in I_2} \mapsto \{\mathcal{B}^n_1\}_{n\in I_1}, \mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathcal{B}^{\lambda'}_1 \{\mathcal{B}^n_1\}_{n\in I_2} \mapsto^{j'\Upsilon_1;\Theta_1} \mathcal{D}'_1.$$

By $\dagger_\kappa$, Lem. 26, and $\mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mapsto^{0\Upsilon'_\kappa;\Theta'_\kappa} \mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)$, we get

$$\begin{aligned}
&\exists \Upsilon_{\kappa_2} \mathcal{B}^{\kappa'}_2 \text{ such that } \mathcal{B}^\kappa_2 \mapsto^{*\Upsilon_{\kappa_2}} \mathcal{B}^{\kappa'}_2 \text{ and } \Upsilon'_\kappa \subseteq \Upsilon_{\kappa_2}\\
&\;\forall x_\gamma \in \mathbf{Out}(\Delta^\kappa \Vdash u_\alpha{:}A \otimes B).\, \mathbf{if}\ x_\gamma \in \Upsilon'_\kappa.\, \mathbf{then}\ \forall m.\, (\mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha); \mathcal{B}^{\kappa'}_2) \in \mathcal{V}[\![\Delta^\kappa \Vdash u_\alpha{:}A \otimes B]\!]^{m+1}_{\cdot;x_\gamma} \text{ and}\\
&\;\forall x_\gamma \in \mathbf{In}(\Delta^\kappa \Vdash u_\alpha{:}A \otimes B).\, \mathbf{if}\ x_\gamma \in \Theta'_\kappa.\, \mathbf{then}\, \forall m.\, (\mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha); \mathcal{B}^{\kappa'}_2) \in \mathcal{V}[\![\Delta^\kappa \Vdash u_\alpha{:}A \otimes B]\!]^{m+1}_{x_\gamma;\cdot}
\end{aligned}$$

In particular, we know that $u_\alpha{:}A \otimes B \in \mathbf{Out}(\Delta^\kappa \Vdash u_\alpha{:}A \otimes B)$ and $u_\alpha \in \Upsilon'_\kappa$, which gives us:

$$\forall m.\, (\mathcal{B}^{\kappa'}_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha); \mathcal{B}^{\kappa'}_2) \in \mathcal{V}[\![\Delta^\kappa \Vdash u_\alpha{:}A \otimes B]\!]^{m+1}_{\cdot;u_\alpha}$$

By **Row 4.** of the logical relation, we have $\Delta^\kappa = \Delta^\kappa_1, \Delta^\kappa_2$ and

$$\begin{aligned}
&\mathcal{B}^{\kappa'}_2 = \mathcal{B}^{\kappa''}_2 \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha), \text{ and}\\
&\circ' \forall m.\, (\mathcal{A}_1; \mathcal{A}_2) \in \mathcal{E}[\![\Delta^\kappa_2 \Vdash y_\beta{:}A]\!]^m \quad \text{and} \quad \circ''\ \forall m.\, (\mathcal{B}^{\kappa'}_1; \mathcal{B}^{\kappa''}_2) \in \mathcal{E}[\![\Delta^\kappa_1 \Vdash u_{\alpha+1}{:}B]\!]^m.
\end{aligned}$$

Next, we consider $\mathcal{B}^\lambda_1$. By $\dagger_\lambda$, Lem. 26, and $\mathcal{B}^\lambda_1 \mapsto^{0\Upsilon'_\lambda} \mathcal{B}^\lambda_1$, we get

$$\begin{aligned}
&\exists \Upsilon_{\lambda_2} \mathcal{B}^{\lambda'}_2 \text{ such that } \mathcal{B}^\lambda_2 \mapsto^{*\Upsilon_{\lambda_2}} \mathcal{B}^{\lambda'}_2 \text{ and}\\
&\;\forall x_\gamma \in \mathbf{Out}(\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda).\, \mathbf{if}\ x_\gamma \in \Upsilon'_\lambda.\, \mathbf{then}\ \forall m.\, (\mathcal{B}^\lambda_1; \mathcal{B}^{\lambda'}_2) \in \mathcal{V}[\![\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda]\!]^{m+1}_{\cdot;x_\gamma} \text{ and}\\
&\;\forall x_\gamma \in \mathbf{In}(\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda).\, \mathbf{if}\ x_\gamma \in \Theta'_\lambda.\, \forall m.\, (\mathcal{B}^\lambda_1; \mathcal{B}^{\lambda'}_2) \in \mathcal{V}[\![\Delta^\lambda, u_\alpha{:}A \otimes B \Vdash K^\lambda]\!]^{m+1}_{x_\gamma;\cdot}
\end{aligned}$$

In particular, we know that $u_\alpha{:}A \otimes B \in \mathbf{In}(\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda)$ and also $u_\alpha \in \Theta'_\lambda$, which gives us:

$$\forall m.\, (\mathcal{B}^\lambda_1; \mathcal{B}^{\lambda'}_2) \in \mathcal{V}[\![\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda]\!]^{m+1}_{u_\alpha;\cdot}$$

By **Row 9.** of the logical relation for the specific channel y_β (for which by the tree structure, we know $y_\beta \notin \Delta^\lambda, u_\alpha{:}A \otimes B, K^\lambda$) we have

$$\forall m.\, (\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mathcal{B}^\lambda_1; \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mathcal{B}^{\lambda'}_2) \in \mathcal{E}[\![\Delta^\lambda, y_\beta{:}A, u_{\alpha+1}{:}B \Vdash K^\lambda]\!]^m.$$

By forward closure (Lem. 24) and $\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mathcal{B}^\lambda_1 \mapsto \mathcal{B}^{\lambda'}_1$ we have:

$$\circ'''\ \forall m.\, (\mathcal{B}^{\lambda'}_1; \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mathcal{B}^{\lambda'}_2) \in \mathcal{E}[\![\Delta, y_\beta{:}A, u_{\alpha+1}{:}B \Vdash K]\!]^m$$

Put $I' = I, \ell$, where ℓ does not occur in I and define $\mathcal{B}^\ell_1 = \mathcal{A}_1$ and $\mathcal{B}^\ell_2 = \mathcal{A}_2$. By induction on the number of steps, $\circ'$, $\circ''$, $\circ'''$ and $\dagger_n$ for $n \neq \kappa, \lambda$ we get

$\forall \Upsilon_1, \Theta_1, \mathcal{D}_1'.\, \textbf{if}\ \{\mathcal{B}_1^n\}_{n\in I_1}\mathcal{B}_1^{\kappa'}\mathcal{A}_1\mathcal{B}_1^{\lambda'}\{\mathcal{B}_1^n\}_{n\in I_2} \mapsto^{j'\,\Upsilon_1;\Theta_1} \mathcal{D}_1'$
$\quad\textbf{then}\ \exists \Upsilon_2, \mathcal{D}_2'$ such that $\{\mathcal{B}_2^n\}_{n\in I_1}\mathcal{B}_2^{\kappa''}\mathcal{A}_2\mathbf{msg}(\mathbf{send}y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'}\{\mathcal{B}_2^n\}_{n\in I_2} \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Upsilon_1.\, \textbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Theta_1.\, \textbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

By a $\forall\mathbf{E}$. and $\{\mathcal{B}_1^n\}_{n\in I_1}\mathcal{B}_1^{\kappa'}\mathcal{A}_1\mathcal{B}_1^{\lambda'}\{\mathcal{B}_1^n\}_{n\in I_2} \mapsto^{j'\,\Upsilon_1} \mathcal{D}_1'$ we get:

$\exists \Upsilon_2, \mathcal{D}_2'$ such that $\{\mathcal{B}_2^n\}_{n\in I_1}\mathcal{B}_2^{\kappa''}\mathcal{A}_2\mathbf{msg}(\mathbf{send}y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'}\{\mathcal{B}_2^n\}_{n\in I_2} \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Upsilon_1.\, \textbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Theta_1.\, \textbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

We also know that $\{\mathcal{B}_2^n\}_{n\in I} = \{\mathcal{B}_2^n\}_{n\in I_1}\mathcal{B}_2^{\kappa}\mathcal{B}_2^{\lambda}\{\mathcal{B}_2^n\}_{n\in I_2} \mapsto^* \{\mathcal{B}_2^n\}_{n\in I_1}\mathcal{B}_2^{\kappa''}\mathcal{A}_2\mathbf{msg}(\mathbf{send}y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'}\{\mathcal{B}_2^n\}_{n\in I_2}$. We get the following for the same Υ_2 and $\mathcal{D}_2'$:

$\exists \Upsilon_2, \mathcal{D}_2'$ such that $\{\mathcal{B}_2^n\}_{n\in I_1}\mathcal{B}_2^{\kappa''}\mathcal{A}_2\mathbf{msg}(\mathbf{send}y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'}\{\mathcal{B}_2^n\}_{n\in I_2} \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Upsilon_1.\, \textbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if}\ x_\gamma \in \Theta_1.\, \textbf{then}\ (\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

J Logical equivalence

J.1 Equivalence

Lemma 28 (Reflexivity). *For all security levels ξ and configurations $\Delta \Vdash \mathcal{D} :: x_\alpha{:}T$, we have*

$$(\Delta \Vdash \mathcal{D} :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D} :: x_\alpha{:}T).$$

Proof. The proof is straightforward by applying the reflexivity Lemma (Lem. 31) proved in the next section on a trivial lattice that has only one element $\perp$. We annotate all channels and processes with max secrecies and running secrecies equal to $\perp$ and annotate process definitions in the signature all with one secrecy variable ψ. The spawn terms in a process use a substitution that maps ψ to $\perp$. With this translation, all session-typed configurations are also IFC-typed under the trivial lattice.

Lemma 29 (Symmetry). *For all configurations $\mathcal{D}_1$ and $\mathcal{D}_2$, we have $(\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T)$, iff $(\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T)$,*

Proof. The proof is straightforward by the definition of logical equivalence (Def. 1).

Lemma 30 (Transitivity). *For all configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, we have*

if $(\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T)$, and $(\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_3 :: x_\alpha{:}T)$
then $(\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_3 :: x_\alpha{:}T)$.

Proof. The proof follows from Corollary 3 when we instantiate Ψ_0 with the trivial lattice only containing the element $\perp$ and ξ to be $\perp$.

J.2 Noninterference

Lemma 31 (Reflexivity – only for IFC-typed processes). *For all security levels ξ and configurations $\Psi_0; \Gamma \Vdash \mathbb{D} :: x_\alpha{:}T[c]$, we have*

$$(\Gamma \Vdash |\mathbb{D}| :: x_\alpha{:}T[c]) \equiv_\xi^{\Psi_0} (\Gamma \Vdash |\mathbb{D}| :: x_\alpha{:}T[c]).$$

Proof. Corollary of the fundamental theorem (Thm. 7).

Lemma 32 (Symmetry). *For all security levels ξ and configurations $\mathcal{D}_1$ and $\mathcal{D}_2$, we have $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv_\xi^{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2])$, iff $(\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2]) \equiv_\xi^{\Psi_0} (\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1])$.*

Proof. The proof is straightforward by Def. 26

Corollary 3 (Transitivity). *For all security levels ξ, and configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, we have*

if $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv_\xi^{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2])$, and $(\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2]) \equiv_\xi^{\Psi_0} (\Gamma_3 \Vdash \mathcal{D}_3 :: z_\eta{:}T_3[c_3])$
then $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv_\xi^{\Psi_0} (\Gamma_3 \Vdash \mathcal{D}_3 :: z_\eta{:}T_3[c_3])$.

Proof. Consider arbitrary $\mathcal{C}_1$, $\mathcal{F}_1$ and $\mathcal{C}_3$, and $\mathcal{F}_3$ we need to show

$\forall m. (\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1; \mathcal{C}_3\mathcal{D}_3\mathcal{F}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m. (\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1; \mathcal{C}_3\mathcal{D}_3\mathcal{F}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$

By the assumptions, we get $(\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1; \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $(\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2; \mathcal{C}_3\mathcal{D}_3\mathcal{F}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$. The result follows by Lem. 33.

Lemma 33 (Transitivity of the term relation). *If $\dagger_1\, \forall m. (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\dagger_2\, \forall m. (\mathcal{D}_2; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ then $\star\, \forall k. (\mathcal{D}_1; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

Proof. Our goal is to prove for all $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ with $\forall m. (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m. (\mathcal{D}_2; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ then $\forall k. (\mathcal{D}_1; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$. We prove an equivalent statement that says for all k, $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ with $\forall m. (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m. (\mathcal{D}_2; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ then $(\mathcal{D}_1; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$. We proceed by induction on k.

Base case. $(k = 0)$ Consider arbitrary configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. By the assumptions, we know that $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $(\mathcal{D}_2; \mathcal{D}_3) \in \mathsf{Tree}(\Delta \Vdash K)$, which gives us $(\mathcal{D}_1; \mathcal{D}_3) \in \mathsf{Tree}(\Delta \Vdash K)$. It is enough to complete the proof in this case.

Inductive case. $(k = k' + 1)$ Consider arbitrary configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. Our goal is to show

$$\begin{array}{l} \forall\, \mathcal{D}_1'. \forall\, \Upsilon_1. \text{ if } \mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}_1', \text{ then } \exists \mathcal{D}_3' \text{ such that } \mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}_3' \text{ and} \\ \quad \forall\, x_\alpha \in \mathbf{Out}(\Delta \Vdash K). \mathbf{if}\ x_\alpha \in \Upsilon_1. \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot; x_\alpha} \text{ and} \\ \quad \forall\, x_\alpha \in \mathbf{In}(\Delta \Vdash K). (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha; \cdot}. \end{array}$$

Consider an arbitrary $\mathcal{D}_1'$ and Υ_1, and assume $\mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}_1'$. Our goal is to prove

$\exists \mathcal{D}_3'$ such that $\mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}_3'$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha;\cdot}$.

By assumption $\dagger_1$ and Lem. 26, we have

$\dagger_1'$ $\forall \mathcal{D}_1'.\, \forall \Upsilon_1.$ if $\mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}_1'$, then $\exists \mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}_2'$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

And by assumption $\dagger_2$ and Lem. 26, we have

$\dagger_2'$ $\forall \mathcal{D}_2'.\, \forall \Upsilon_1.$ if $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}_2'$, then $\exists \mathcal{D}_3'$ such that $\mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}_3'$ and
$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m.\, (\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m.\, (\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

We apply $\forall\mathbf{E}$. on $\dagger_1'$ by instantiating the existential quantifiers with $\mathcal{D}_1'$ and Υ_1. We can apply $\mathbf{if E}$. since we know $\mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}_1'$. We get a $\mathcal{D}_2'$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}_2'$. Next, we apply $\forall\mathbf{E}$. on $\dagger_2$ by instantiating the existential quantifiers with $\mathcal{D}_2'$ and Υ_1. We apply $\mathbf{if E}$. as we know $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}_2'$. As a result, we get a $\mathcal{D}_3'$ with $\mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}_3'$ as required by the goal. We need to prove:

$\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha;\cdot}$.

By $\dagger_1'$, and $\dagger_2'$, we have as assumptions

$\dagger_1''$ $\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

and

$\dagger_2''$ $\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m.\, (\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and
$\forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m.\, (\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}$.

There are two parts to prove:

Part 1. Consider an arbitrary $x_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ and assume $x_\alpha \in \Upsilon_1$. Our goal is to show

$$(\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha}$$

And by $\dagger_1''$, and $\dagger_2''$, we have as assumptions

$\dagger_1'''$ $\forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$ and $\dagger_2'''$ $\forall m.\, (\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}$.

We consider cases based on the type of x_α. We provide the detailed proof for a few interesting cases. The proof of other cases is similar.

Case 1. $(K = x_\alpha{:}A \otimes B)$**.** By $\dagger_1'''$ and $\dagger_2'''$we get

$$\begin{array}{llllll} \Delta = \Delta_1, \Delta_2 & \text{and} & \mathcal{D}_1' = \mathcal{D}_1''\mathcal{A}_1\mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) & \text{and} & \mathcal{D}_2' = \mathcal{D}_2''\mathcal{A}_2\mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) \\ & \text{and} & \mathcal{D}_2' = \mathcal{D}_2''\mathcal{A}_1\mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) & \text{and} & \mathcal{D}_3' = \mathcal{D}_3''\mathcal{A}_2\mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) \end{array}$$

Moreover,

$$\begin{array}{ll} \forall m.(\mathcal{D}_1''; \mathcal{D}_2'') \in \mathcal{E}[\![\Delta_1 \Vdash x_{\alpha+1}{:}B]\!]^m & \forall m.(\mathcal{A}_1; \mathcal{A}_2) \in \mathcal{E}[\![\Delta_2 \Vdash y_\beta{:}A]\!]^m \\ \forall m.(\mathcal{D}_2''; \mathcal{D}_3'') \in \mathcal{E}[\![\Delta_1 \Vdash x_{\alpha+1}{:}B]\!]^m & \forall m.(\mathcal{A}_2; \mathcal{A}_3) \in \mathcal{E}[\![\Delta_2 \Vdash y_\beta{:}A]\!]^m \end{array}$$

We apply the induction hypothesis on a smaller observation index k' to get

$$(\mathcal{D}_1''; \mathcal{D}_3'') \in \mathcal{E}[\![\Delta_1 \Vdash x_{\alpha+1}{:}B]\!]^{k'} \qquad (\mathcal{A}_1; \mathcal{A}_3) \in \mathcal{E}[\![\Delta_2 \Vdash y_\beta{:}A]\!]^{k'}$$

Which by **Row 4** of the logical relation is wnough to prove the goal of this subcase, i.e.,

$$(\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot; x_\alpha}$$

Case 2. $(\Delta = x_\alpha{:}A \multimap B, \Delta')$

Part 2. Consider an arbitrary $x_\alpha \in \mathbf{In}(\Delta \Vdash K)$ and assume $x_\alpha \in \Theta_1 x$. Our goal is to show

$$(\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha; \cdot}.$$

By $\dagger_1''$, and $\dagger_2''$, we have as assumptions

$$\dagger_1'''\, \forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha; \cdot} \quad \text{and} \quad \dagger_2'''\, \forall m.\, (\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha; \cdot}.$$

We consider cases based on the type of x_α. We provide the detailed proof for a few interesting cases. The proof of other cases is similar.

Case 1. $(\Delta = \Delta', x_\alpha{:}A \otimes B)$ By $\dagger_1'''$ and $\dagger_2'''$we get for all $y_\beta \notin dom(\Delta, x_\alpha : A \otimes B, K)$.

$$\begin{array}{l} \forall m.(\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{D}_1'; \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{D}_2') \in \mathcal{E}[\![\Delta, y_\beta : A, x_{\alpha+1} : B \Vdash K]\!]^m \\ \forall m.(\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{D}_2'; \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{D}_3') \in \mathcal{E}[\![\Delta, y_\beta : A, x_{\alpha+1} : B \Vdash K]\!]^m \end{array}$$

We apply the induction hypothesis on a smaller observation index k' to get

$$\forall y_\beta \notin dom(\Delta, x_\alpha : A \otimes B, K).\, (\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{D}_1'; \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{D}_3') \in \mathcal{E}[\![\Delta, y_\beta : A, x_{\alpha+1} : B \Vdash K]\!]^{k'}$$

Which by **Row 9** of the logical relation is wnough to prove the goal of this subcase, i.e.,

$$(\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha; \cdot}$$

Case 2. $(K = x_\alpha{:}A \multimap B)$

K Adequacy

To show that the RSLR is adequate, we prove that configurations related by the logical relation are bisimilar and vice versa. To facilitate this proof, we first give the definition of *asynchronous bisimilarity* [53] (Def. 33). The definition relies on the labeled transition system resulting from our RSLR displayed in Fig. 16 and on weak transitions and free names defined next.

Internal transition $\xrightarrow{\tau}$ *defined as:*

$\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ iff $\mathcal{D}_1 \mapsto \mathcal{D}_1'$

Actions $\xrightarrow{\overline{y_\alpha}\, q}$, $\xrightarrow{\mathbf{L}\, y_\alpha\, q}$ *and* $\xrightarrow{\mathbf{R}\, y_\alpha\, q}$ *defined as below when* $y \in \mathbf{fn}(\mathcal{D}_1)$*:*

$$(1)\ \mathcal{D}_1 \mathbf{msg}(\mathbf{close}\, y_\alpha)\, \mathcal{D}_2 \xrightarrow{\overline{y_\alpha}\, \mathbf{close}} \mathcal{D}_1 \mathcal{D}_2$$

$$(2)\mathcal{D}_1\, \mathbf{msg}(y_\alpha.k)\mathcal{D}_2 \xrightarrow{\overline{y_\alpha}\, k} \mathcal{D}_1 \mathcal{D}_2$$

$$(3)\mathcal{D}_1\, \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)\mathcal{D}_2 \xrightarrow{\overline{y_\alpha}\, x_\beta} \mathcal{D}_1 \mathcal{D}_2$$

$$(4)\ \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{wait}\, y_\alpha; P)\mathcal{D}_2 \xrightarrow{\mathbf{L}\, y_\alpha\, \mathbf{close}} \mathbf{msg}(\mathbf{close}\, y_\alpha)\, \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{wait}\, y_\alpha; P)\, \mathcal{D}_2$$

$$(5)\ \mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I})\, \mathcal{D}_2 \xrightarrow{\mathbf{L}\, y_\alpha\, k} \mathbf{msg}(y_\alpha.k)\mathcal{D}_1 \mathbf{proc}(z_\delta, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I})\mathcal{D}_2$$

$$(6)\mathcal{D}_1\, \mathbf{proc}(z_\delta, w \leftarrow \mathbf{recv}\, y_\alpha)\mathcal{D}_2 \xrightarrow{\mathbf{L}\, y_\alpha\, x_\beta} \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)\mathcal{D}_1\, \mathbf{proc}(z_\delta, w \leftarrow \mathbf{recv}\, y_\alpha)\, \mathcal{D}_2$$

$$(7)\mathcal{D}_1\, \mathbf{proc}(y_\alpha, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I})\mathcal{D}_2 \xrightarrow{\mathbf{R}\, y_\alpha\, k} \mathcal{D}_1\, \mathbf{proc}(y_\alpha, \mathbf{case}\, y_\alpha\, (\ell \Rightarrow P_\ell)_{\ell \in I})\, \mathcal{D}_2 \mathbf{msg}(y_\alpha.k)$$

$$(8)\mathcal{D}_1 \mathbf{proc}(y_\alpha, w \leftarrow \mathbf{recv}\, y_\alpha)\, \mathcal{D}_2 \xrightarrow{\mathbf{R}\, y_\alpha\, x_\beta} \mathcal{D}_1 \mathbf{proc}(y_\alpha, w \leftarrow \mathbf{recv}\, y_\alpha)\mathcal{D}_2\, \mathbf{msg}(\mathbf{send}\, x_\beta\, y_\alpha)$$

Fig. 16: Labeled transition system induced by RSLR.

Definition 31 (Free names of a configuration). *For* $\Delta \Vdash \mathcal{D} :: \Delta'$*, we define* $\mathsf{fn}(\mathcal{D})$ *as* $dom(\Delta, \Delta')$*.*

Definition 32 (Weak transition relations).

(1) $\Rightarrow$ *is the reflexive and transitive closure of* $\xrightarrow{\tau}$*.*

(2) $\xRightarrow{\alpha}$ *is* $\Rightarrow\xrightarrow{\alpha}$*.*

Definition 33 (Asynchronous bisimilarity). *Asynchronous bisimilarity is the largest symmetric relation such that whenever* $\mathcal{D}_1 \approx_a \mathcal{D}_2$*, we have*

1. $(\tau - \mathtt{step})$ *if* $\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathcal{D}_2'$,
2. (output) *if* $\mathcal{D}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathcal{D}_2'$
3. $(\mathsf{left\ input})$ *for all* $q \notin \mathsf{fn}(\mathcal{D}_1)$*., if* $\mathcal{D}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}_2'$,
4. $(\mathsf{right\ input})$ *for all* $q \notin \mathsf{fn}(\mathcal{D}_1)$*., if* $\mathcal{D}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'.\, \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathcal{D}_2' \mathbf{msg}(x_\alpha.q)$,

where $\mathbf{msg}(x_\alpha.q)$ *is defined as* $\mathbf{msg}(\mathbf{close}\, x_\alpha)$ *if* $q = \mathbf{close}$*,* $\mathbf{msg}(x_\alpha.k)$ *if* $q = k$*, and* $\mathbf{msg}(\mathbf{send}\, z_\delta\, x_\alpha)$ *if* $q = z_\delta$*.*

Definition 34 (High provider and High client). *We repeat the definition of high provider and high client configurations (§ G.1) here.*

$$\cdot \in \mathbf{H\text{-}Provider}^{\xi}(\cdot)$$

$$\begin{array}{lll} \mathcal{B} \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma, x_\alpha{:}A[c]) & \text{iff} & c \not\sqsubseteq \xi \text{ and } \mathcal{B} = \mathcal{B}'\mathcal{T} \text{ and } \mathcal{B}' \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma) \text{ and } \mathcal{T} \in \mathsf{Tree}(\cdot \Vdash x_\alpha{:}A), \mathbf{or} \\ & & c \sqsubseteq \xi \text{ and } \mathcal{B} \in \mathbf{H\text{-}Provider}^{\xi}(\Gamma) \end{array}$$

$$\begin{array}{lll} \mathcal{T} \in \mathbf{H\text{-}Client}^{\xi}(x_\alpha{:}A[c]) & \text{iff} & c \not\sqsubseteq \xi \text{ and } \mathcal{T} \in \mathsf{Tree}(x_\alpha{:}A \Vdash \_ : 1), \mathbf{or} \\ & & c \sqsubseteq \xi \text{ and } \mathcal{B} = \cdot \end{array}$$

◇

Definition 35. *For* $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$*,* $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$ *we define*

$$\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1] \approx_a^{\xi} \Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2] \text{ as}$$

$\Gamma_1 \Downarrow \xi = \Gamma_2, \Downarrow \xi$ and $y_\beta{:}A_2[c_2] \Downarrow \xi = x_\alpha{:}A_1[c_1] \Downarrow \xi$ and
$\forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]), \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2])$.
$\mathcal{B}_1\mathcal{D}_1\mathcal{T}_1 \approx_a \mathcal{B}_2\mathcal{D}_2\mathcal{T}_2$.

Corollary 4. *For all* $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$ *and* $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$*, we have* $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv_{\xi}^{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ *iff* $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \approx_a^{\xi} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$*.*

Proof. The proof is straightforward by considering Def. 26, Def. 35, and the following corollary (Corollary 5).

Corollary 5. *For all* $(\mathcal{D}_1, \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$*, we have* $\forall m.(\mathcal{D}_1, \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ *and* $\forall m.(\mathcal{D}_2, \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ *iff* $\mathcal{D}_1 \approx_a \mathcal{D}_2$*.*

Proof. It is a corollary of the following lemma (Lem. 34)

Lemma 34. *Consider a pair of session-typed forests* $(\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{Forest}(\Delta \Vdash \Delta')$ *consisting of multiple session-typed trees indexed in the set* I*, i.e.,* $\mathcal{C}_i = \{\mathcal{C}_i^j\}_{j \in I}$ *for* $i \in \{1, 2\}$*, and* $(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$ *with* $\Delta = \{\Delta^j\}_{j \in I}$ *and* $\Delta' = \{K^j\}_{j \in I}$*. We have:*

$$\forall j \in I. \forall m. \, (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m \text{ and } (\mathcal{C}_2^j, \mathcal{C}_1^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m \text{ iff } \mathcal{C}_1 \approx_a \mathcal{C}_2.$$

Proof. The proof consists of two parts.

(1) Soundness. We want to prove for any arbitrary pair of forests $\mathcal{C}_1 = \{\mathcal{C}_1^j\}_{j \in I}$ and $\mathcal{C}_2 = \{\mathcal{C}_2^j\}_{j \in I}$ such that $\forall j \in I. (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, if $\forall j \in I. \forall m.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ and $\forall j \in I. \forall m.(\mathcal{C}_2^j, \mathcal{C}_1^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ then $\{\mathcal{C}_1^j\}_{j \in I} \approx_a \{\mathcal{C}_2^j\}_{j \in I}$.

We proceed the proof by coinduction on the generating function of the bisimilarity. Consider an arbitrary $\mathcal{C}_1 = \{\mathcal{C}_1^j\}_{j \in I}$ and $\mathcal{C}_2 = \{\mathcal{C}_2^j\}_{j \in I}$ such that $\forall j \in I. (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, and assume $\forall j \in I. \forall m.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash$

$K^j]\!]^m$. Our goal is to show $\{\mathcal{C}_1^j\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$. We prove it by showing the items (1)-(4) Def. 33 by assuming $\forall j \in I.\forall m.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ only. The symmetric relation can be established with the other assumption $\forall j \in I.\forall m.(\mathcal{C}_2^j, \mathcal{C}_1^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ We continue by establishing items (1)-(4) in Def. 33:

1. Assume $\mathcal{C}_1 \xrightarrow{\tau} \mathcal{C}_1'$, then by the definition of $\tau-$transition, we have $\mathcal{C}_1 \mapsto \mathcal{C}_1'$. In other words we have $\{\mathcal{C}_1^j\}_{j\in I} \mapsto \{\mathcal{C}_1^{j'}\}_{j\in I}$ for $\mathcal{C}_1' = \{\mathcal{C}_1^{j'}\}_{j\in I}$. To establish item (1) in the definition, it is enough to prove that $\{\mathcal{C}_1^{j'}\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$. Since we unfolded the coinductive definition of the bisimilarity's generating function once, we can apply the coinductive case if the assumptions are satisfied, i.e., we need to prove that each tree in the post-step forest is session-typed and the pairs are related by the logical relation.
   By assumption $\forall j \in I.\forall m.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ and by the forward closure lemma (Lem. 24), we get $\forall j \in I.\forall m.(\mathcal{C}_1^{j'}, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$. This along with type-preservation of session-typed programs, is enough to get $\{\mathcal{C}_1^{j'}\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$ by coinduction.
2. Assume $\mathcal{C}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{C}_1'$, then by Fig. 16 we have $\mathcal{C}_1 = \mathcal{C}_1^1\mathbf{msg}(x_\alpha.q)\,\mathcal{C}_1^2$, with $\mathcal{C}_1' = \mathcal{C}_1^1, \mathcal{C}_1^2$.
   This means that for some $\kappa \in I$, we have either $\mathcal{C}_1^\kappa = \mathcal{C}_1^{\kappa'}\mathbf{msg}(x_\alpha.q)$ or $\mathcal{C}_1^\kappa = \mathbf{msg}(x_\alpha.q)\mathcal{C}_1^{\kappa'}$, and $\mathcal{C}_1' = \{\mathcal{C}_1^j\}_{j\in I-\{\kappa\}}\mathcal{C}_1^{\kappa'}$. In particular, we know that $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^\kappa) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m$, and $\mathcal{C}_1^\kappa \mapsto^{0\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^\kappa$, such that $x_\alpha \in \Upsilon_1^\kappa$. By the definition of the term interpretation and Lem. 26, we get

   $$\begin{array}{l}\forall \Upsilon_1^\kappa, \Theta_1^\kappa, \mathcal{C}_1^{\kappa'}.\mathsf{if}\ \mathcal{C}_1^\kappa \mapsto^{*\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^{\kappa'},\ \mathsf{then}\ \exists \Upsilon_2^\kappa, \mathcal{C}_2^{\kappa'}\ \mathsf{such\ that}\ \mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'}\ \mathsf{and}\ \Upsilon_1^\kappa \subseteq \Upsilon_2^\kappa\ \mathsf{and} \\ \quad \forall x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa).\,\mathbf{if}\ x_\alpha \in \Upsilon_1^\kappa.\,\mathbf{then}\ \forall k.\,(\mathcal{C}_1^{\kappa'}; \mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{\cdot;x_\alpha}\ \mathsf{and} \\ \quad \forall x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa).\mathbf{if}\ x_\alpha \in \Theta_1^\kappa.\,\mathbf{then}\ \forall k.\,(\mathcal{C}_1^{\kappa'}; \mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{x_\alpha;\cdot}.\end{array}$$

   We can instantiate the $\forall$ quantifiers and apply an **if**$-$Elimination rule to get $\mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'}$ for some $\Upsilon_2^\kappa \supseteq \Upsilon_1^\kappa$ and some $\mathcal{C}_2^{\kappa'}$. Since $x_\alpha \in \Upsilon_1^\kappa$, we get by the definition of the logical relation $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^{\kappa'}) \in \mathcal{V}^\xi_{\Psi_0}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m+1}_{\cdot;x_\alpha}$. This (depending on the type of the channel x_α) gives us $\mathcal{C}_2^{\kappa'} = \mathcal{C}_{\kappa_2''}\mathbf{msg}(x_\alpha.q)$ or $\mathcal{C}_2^{\kappa'} = \mathbf{msg}(x_\alpha.q)\mathcal{C}_{\kappa_2''}$. Note that if q is a channel name, $\mathcal{C}_2^{\kappa''}$ is a forest, rather than a tree, and if q is the label **close**, the configuration $\mathcal{C}_2^{\kappa''}$ is empty.
   Put $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_2^{\kappa''}$. To establish item (2) in the definition, it is enough to prove that $\mathcal{C}_1' \approx_a \mathcal{C}_2'$. Since we unfolded the coinductive definition of the bisimilarity's generating function once, we can apply the coinductive case if the assumptions are satisfied, i.e., we need to prove that each tree in the post-step forest is session-typed and the pairs are related by the logical relation.
   The proof proceeds by cases on the type of x_α. Here we only provide one interesting case, the proof of other cases is similar.
   Subcase 1. $K^\kappa = x_\alpha{:}A \otimes B \in \Delta'$**.** We have $\Delta^\kappa = \Delta_1^\kappa, \Delta_2^\kappa$ and $\mathcal{C}_1^\kappa = \mathcal{C}_1^{\kappa'}\mathbf{msg}(\mathbf{send}\,y_\delta\,x_\alpha)$ with $\mathcal{C}_1^{\kappa'} = \mathcal{C}_1^{\kappa''}\mathcal{A}_1$. Moreover, we have $\mathcal{C}_2^{\kappa'} = \mathcal{C}_2^{\kappa''}\mathbf{msg}(\mathbf{send}\,y_\delta\,x_\alpha)$ with

$\mathcal{C}_2^{\kappa''} = \mathcal{C}_2^{\kappa'''}\mathcal{A}_2$. And $\forall m.(\mathcal{C}_1^{\kappa''};\mathcal{C}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash x_{\alpha+1}{:}B]\!]^m$ and $\forall m.(\mathcal{A}_1;\mathcal{A}_2) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash y_\delta{:}A]\!]^m$.
Recall that all other trees in the forest are still related, i.e. $\forall j \in I - \{\kappa\}$, we have $\forall m.(\mathcal{C}_1^j;\mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ This is enough to apply the coinductive argument and get $\mathcal{C}_1' \approx_a \mathcal{C}_2'$ where $\mathcal{C}_1' = \{\mathcal{C}_1^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_1^{\kappa'}$ and $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_2^{\kappa''}$.

3. Assume for an arbitrary q, we have $\mathcal{C}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{C}_1'$. By Fig. 16, there is a process $\mathbf{proc}(z_\beta, P_{x_\alpha}) \in \mathcal{C}_1$ which is waiting to receive a message along x_α. In particular, for some tree $\mathcal{C}_1^\kappa$ for $\kappa \in I$, we have $\mathbf{proc}(z_\beta, P_{x_\alpha}) \in \mathcal{C}_1^\kappa$.
We know that $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^\kappa) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m$, and $\mathcal{C}_1^\kappa \mapsto^{0\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^\kappa$, such that $x_\alpha \in \Theta_1^\kappa$. By the definition of the term interpretation and Lem. 26, we get

$\forall\, \Upsilon_1^\kappa, \Theta_1^\kappa, \mathcal{C}_1^{\kappa'}.\text{if } \mathcal{C}_1^\kappa \mapsto^{*\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^{\kappa'}, \text{ then } \exists \Upsilon_2^\kappa, \mathcal{C}_2^{\kappa'} \text{ such that } \mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'} \text{ and } \Upsilon_1^\kappa \subseteq \Upsilon_2^\kappa \text{ and}$
$\quad \forall\, x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa).\, \mathbf{if}\ x_\alpha \in \Upsilon_1^\kappa.\, \mathbf{then}\ \forall\, k.\, (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{\cdot;x_\alpha} \text{ and}$
$\quad \forall\, x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa).\mathbf{if}\ x_\alpha \in \Theta_1^\kappa.\, \mathbf{then}\ \forall k.\, (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{x_\alpha;\cdot}.$

We can instantiate the $\forall$ quantifiers and apply an **if**−Elimination rule to get $\mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'}$ for some $\Upsilon_2^\kappa \supseteq \Upsilon_1^\kappa$ and some $\mathcal{C}_2^{\kappa'}$. Since $x_\alpha \in \Theta_1^\kappa$, we get by the definition of the logical relation $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^{\kappa'}) \in \mathcal{V}^\xi_{\Psi_0}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m+1}_{x_\alpha;\cdot}$.
We put $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_2^{\kappa'}$.
The proof proceeds by cases on the type of x_α. Here we only provide one interesting case, the proof of other cases is similar.
Subcase 1. $x_\alpha{:}A \otimes B \in \Delta^\kappa \subseteq \Delta$ **(i.e.,** $\Delta^\kappa = \Delta^{\kappa'}, x_\alpha{:}A \otimes B$**)** In this case, we know that $\mathcal{C}_1^\kappa = \mathcal{C}_1^{\kappa_1}\, \mathbf{proc}(z_\delta, w \leftarrow \mathbf{recv}\, x_\alpha)\, \mathcal{C}_1^{\kappa_2}$ and q is a channel y_δ which is fresh in both $\mathcal{C}_1^\kappa$ and $\mathcal{C}_2^{\kappa'}$. Moreover, we have $\mathcal{C}_1^{\kappa'} = \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_1^\kappa$, and thus $\mathcal{C}_1' = \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_1$.
By the logical relation, we get $\forall m.(\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_1^\kappa; \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_2^{\kappa'}) \in \mathcal{E}[\![\Delta^{\kappa'}, y_\delta{:}A, x_{\alpha+1}{:}B \Vdash K^\kappa]\!]^m$.
Recall that all other trees in the forest are still related, i.e. $\forall j \in I - \{\kappa\}$, we have $\forall m.(\mathcal{C}_1^j;\mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ This is enough to apply the coinductive argument and get $\mathcal{C}_1' \approx_a \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_2'$ as required.

4. Assume for an arbitrary q, we have $\mathcal{C}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{C}_1'$. The proof is similar to the previous case:
By Fig. 16, there is a process $\mathbf{proc}(x_\alpha, P_{x_\alpha}) \in \mathcal{C}_1$ which is waiting to receive a message along x_α. In particular, for some tree $\mathcal{C}_1^\kappa$ for $\kappa \in I$, we have $\mathbf{proc}(x_\alpha, P) \in \mathcal{C}_1^\kappa$.
We know that $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^\kappa) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m$, and $\mathcal{C}_1^\kappa \mapsto^{0\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^\kappa$, such that $x_\alpha \in \Theta_1^\kappa$. By the definition of the term interpretation and Lem. 26, we get

$\forall\, \Upsilon_1^\kappa, \Theta_1^\kappa, \mathcal{C}_1^{\kappa'}.\text{if } \mathcal{C}_1^\kappa \mapsto^{*\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^{\kappa'}, \text{ then } \exists \Upsilon_2^\kappa, \mathcal{C}_2^{\kappa'} \text{ such that } \mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'} \text{ and } \Upsilon_1^\kappa \subseteq \Upsilon_2^\kappa \text{ and}$
$\quad \forall\, x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa).\, \mathbf{if}\ x_\alpha \in \Upsilon_1^\kappa.\, \mathbf{then}\ \forall\, k.\, (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{\cdot;x_\alpha} \text{ and}$
$\quad \forall\, x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa).\mathbf{if}\ x_\alpha \in \Theta_1^\kappa.\, \mathbf{then}\ \forall k.\, (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{x_\alpha;\cdot}.$

We can instantiate the $\forall$ quantifiers and apply an **if**−Elimination rule to get $\mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'}$ for some $\Upsilon_2^\kappa \supseteq \Upsilon_1^\kappa$ and some $\mathcal{C}_2^{\kappa'}$. Since $x_\alpha \in \Theta_1^\kappa$, we get

by the definition of the logical relation $\forall m.(\mathcal{C}_1^{\kappa}, \mathcal{C}_2^{\kappa'}) \in \mathcal{V}^{\xi}_{\Psi_0}[\![\Delta^{\kappa} \Vdash K^{\kappa}]\!]^{m+1}_{x_\alpha;\cdot}$.
We put $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j \in I-\{\kappa\}}, \mathcal{C}_2^{\kappa'}$.
The proof proceeds by cases on the type of x_α. Here we only provide one interesting case, the proof of other cases is similar.

Subcase 1. $K^{\kappa} = x_\alpha{:}A \multimap B \in \Delta'$. In this case, we know that $\mathcal{C}_1^{\kappa} = \mathcal{C}_1^{\kappa_1}\, \mathbf{proc}(x_\alpha, w \leftarrow \mathbf{recv}\, x_\alpha)\, \mathcal{C}_1^{\kappa_2}$ and q is a channel y_δ which is not free in $\mathcal{C}_1^{\kappa}$ (and by the typing not in $\mathcal{C}_2^{\kappa'}$ either). Moreover, we have $\mathcal{C}_1^{\kappa'} = \mathcal{C}_1^{\kappa}\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$, and thus $\mathcal{C}_1' = \mathcal{C}_1\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$.
By the logical relation, we get $\forall m.(\mathcal{C}_1^{\kappa}\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha); \mathcal{C}_2^{\kappa'}\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)) \in \mathcal{E}[\![\Delta^{\kappa}, y_\delta{:}A \Vdash x_{\alpha+1}{:}B]\!]^m$.
Recall that all other trees in the forest are still related, i.e. $\forall j \in I-\{\kappa\}$, we have $\forall m.(\mathcal{C}_1^j; \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ This is enough to apply the coinductive argument and get $\mathcal{C}_1' \approx_a \mathcal{C}_2'\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$ as required.

(2) Completeness. We want to prove for any arbitrary pair of forests $\mathcal{C}_1 = \{\mathcal{C}_1^j\}_{j\in I}$ and $\mathcal{C}_2 = \{\mathcal{C}_2^j\}_{j\in I}$ such that $\forall j \in I.\, (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, if $\{\mathcal{C}_1^j\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$ then $\forall j \in I. \forall m.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$.
We instead prove an equivalent statement that says

> For any natural number m and any arbitrary pair of forests $\mathcal{C}_1 = \{\mathcal{C}_1^j\}_{j\in I}$ and $\mathcal{C}_2 = \{\mathcal{C}_2^j\}_{j\in I}$ such that $\forall j \in I.\, (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, if $\{\mathcal{C}_1^j\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$ then $\forall j \in I.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$.

We proceed by induction on m.

Base case. $(m = 0)$ The proof is straightforward by the definition of logical relation and the fact that the configurations are session-typed.

Inductive case. $(m = m'+1)$ Consider an arbitrary $\kappa \in I$. By the definition of the term interpretation, we need to prove

$$\begin{array}{l}
\forall \Upsilon_1^{\kappa}, \Theta_1^{\kappa}, \mathcal{C}_1^{\kappa}.\mathsf{if}\ \mathcal{C}_1^{\kappa'} \mapsto^{*\Upsilon_1^{\kappa};\Theta_1^{\kappa}} \mathcal{C}_1^{\kappa'},\ \mathsf{then}\ \exists \Upsilon_2^{\kappa}, \mathcal{C}_2^{\kappa'}\ \mathsf{such\ that}\ \mathcal{C}_2^{\kappa} \mapsto^{*\Upsilon_2^{\kappa}} \mathcal{C}_2^{\kappa'}\ \mathsf{and}\ \Upsilon_1^{\kappa} \subseteq \Upsilon_2^{\kappa}\ \mathsf{and} \\
\quad \forall\, x_\alpha \in \mathbf{Out}(\Delta^{\kappa} \Vdash K^{\kappa}).\,\mathbf{if}\ x_\alpha \in \Upsilon_1^{\kappa}.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'}; \mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^{\kappa} \Vdash K^{\kappa}]\!]^{m'+1}_{\cdot;x_\alpha}\ \mathsf{and} \\
\quad \forall\, x_\alpha \in \mathbf{In}(\Delta^{\kappa} \Vdash K^{\kappa}).\mathbf{if}\ x_\alpha \in \Theta_1^{\kappa}.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'}; \mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^{\kappa} \Vdash K^{\kappa}]\!]^{m'+1}_{x_\alpha;\cdot}.
\end{array}$$

Consider an arbitrary Υ_1^{κ}, Θ_1^{κ}, and $\mathcal{C}_1^{\kappa'}$, and assume $\mathcal{C}_1^{\kappa'} \mapsto^{*\Upsilon_1^{\kappa};\Theta_1^{\kappa}} \mathcal{C}_1^{\kappa'}$, we need to prove

$$\begin{array}{l}
\exists \Upsilon_2^{\kappa}, \mathcal{C}_2^{\kappa'}\ \mathsf{such\ that}\ \mathcal{C}_2^{\kappa} \mapsto^{*\Upsilon_2^{\kappa}} \mathcal{C}_2^{\kappa'}\ \mathsf{and}\ \Upsilon_1^{\kappa} \subseteq \Upsilon_2^{\kappa}\ \mathsf{and} \\
\quad \forall\, x_\alpha \in \mathbf{Out}(\Delta^{\kappa} \Vdash K^{\kappa}).\,\mathbf{if}\ x_\alpha \in \Upsilon_1^{\kappa}.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'}; \mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^{\kappa} \Vdash K^{\kappa}]\!]^{m'+1}_{\cdot;x_\alpha}\ \mathsf{and} \\
\quad \forall\, x_\alpha \in \mathbf{In}(\Delta^{\kappa} \Vdash K^{\kappa}).\mathbf{if}\ x_\alpha \in \Theta_1^{\kappa}.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'}; \mathcal{C}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^{\kappa} \Vdash K^{\kappa}]\!]^{m'+1}_{x_\alpha;\cdot}.
\end{array}$$

By local transition steps, we know that $\mathcal{C}_1 \mapsto^* \mathcal{C}_1'$, where $\mathcal{C}_1' = \{\mathcal{C}_1^{\kappa}\}_{j\in I-\{\kappa\}}\mathcal{C}_1^{\kappa'}$. By $\mathcal{C}_1 \approx_a \mathcal{C}_2$, we can apply the clause for τ-transitions in Def. 33 for zero or multiple times, to get $\mathcal{C}_2 \mapsto^* \mathcal{C}_2'$ such that $\mathcal{C}_1' \approx_a \mathcal{C}_2'$. Consider the channels in the sets Υ_1^{κ} and Θ_1^{k}. There are three cases:

A. By item (2) of Def. 33 and $\mathcal{C}_1' \approx_a \mathcal{C}_2'$, we know that for all $x_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ such that $x_\alpha \in \Upsilon_1^{\kappa}$, we have $\mathcal{C}_2' \mapsto^* \mathcal{C}_2^{x_s}$ such that $\mathcal{C}_2^{x_s}$ sends along the channel x_α and $\mathcal{C}'_1 = \mathcal{C}_1^1\mathbf{msg}(x_\alpha.q)\mathcal{C}_1^2$ and $\mathcal{C}_2^{x_s} = \mathcal{C}_2^{x'_s}\mathbf{msg}(x_\alpha.q)\mathcal{C}_2^{x''_s}$, and we have $\mathcal{C}_1^1\mathcal{C}_1^2 \approx_a \mathcal{C}_2^{x'_s}\mathcal{C}_2^{x''_s}$.

B. By item (3) of Def. 33 and $\mathcal{C}'_1 \approx_a \mathcal{C}'_2$, we know that for all $x_\alpha \in \mathbf{In}(\Delta \Vdash \cdot)$ such that $x_\alpha \in \Theta_1^\kappa$, we have $\mathcal{C}'_2 \mapsto^* \mathcal{C}_2^{x_r}$ such that $\mathbf{msg}(x_\alpha.q)\mathcal{C}'_1 \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{C}_2^{x_r}$.

C. By item (4) of Def. 33 and $\mathcal{C}'_1 \approx_a \mathcal{C}'_2$, we know that for all $x_\alpha \in \mathbf{In}(\cdot \Vdash K)$ such that $x_\alpha \in \Theta_1^\kappa$, we have $\mathcal{C}'_2 \mapsto^* \mathcal{C}_2^{x_r}$ such that $\mathcal{C}'_1\mathbf{msg}(x_\alpha.q) \approx_a \mathcal{C}_2^{x_r}\mathbf{msg}(x_\alpha.q)$.

Apply the confluence lemma (Lem. 21) on (i) $\mathcal{C}'_2 \mapsto^* \mathcal{C}_2^{x_s}$ for all $x_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ such that $x_\alpha \in \Upsilon_1^\kappa$ and (ii) $\mathcal{C}'_2 \mapsto^* \mathcal{C}_2^{x_r}$ for all $x_\alpha \in \mathbf{In}(\Delta \Vdash K)$ such that $x_\alpha \in \Theta_1^\kappa$ to build $\mathcal{C}''_2$. In particular, by the confluence lemma, we get $\mathcal{C}'_2 \mapsto^{*\Upsilon_2} \mathcal{C}''_2$ where $\Upsilon_1^\kappa \subseteq \Upsilon_2$, and (i') $\mathcal{C}_2^{x_s} \mapsto^* \mathcal{C}''_2$ for all $x_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ such that $x_\alpha \in \Upsilon_1^\kappa$ and (ii) $\mathcal{C}_2^{x_r} \mapsto^* \mathcal{C}''_2$ for all $x_\alpha \in \mathbf{In}(\Delta \Vdash K)$ such that $x_\alpha \in \Theta_1^\kappa$.

More precisely, by the forest structure, we have $\mathcal{C}''_2 = \{\mathcal{C}_2^{\kappa''}\}_{j\in I-\{\kappa\}}\mathcal{C}_2^{\kappa''}$, with $\mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa''}$ with $\Upsilon_1^\kappa \subseteq \Upsilon_2^\kappa$. We use Υ_2^κ and $\mathcal{C}_2^{\kappa''}$ to instantiate the existential quantifier in the goal. We need to prove that

$$\forall x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa).\,\mathbf{if}\, x_\alpha \in \Upsilon_1^\kappa.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m'+1}_{\cdot;x_\alpha} \text{ and}$$
$$\forall x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa).\mathbf{if}\, x_\alpha \in \Theta_1^\kappa.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m'+1}_{x_\alpha;\cdot}.$$

By the forward closure lemma for the bisimulation (Lem. 35), we get

A'. By item [A.] we get, for all $x_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ such that $x_\alpha \in \Upsilon_1^\kappa$, we have $\mathcal{C}'_2 \mapsto^* \mathcal{C}''_2$ such that $\mathcal{C}''_2$ sends along the channel x_α, and we have $\mathcal{C}'_1 = \mathcal{C}_1^1\mathbf{msg}(x_\alpha.q)\mathcal{C}_1^2$ and $\mathcal{C}''_2 = \mathcal{C}^{x'_s}\mathbf{msg}(x_\alpha.q)\mathcal{C}^{x''_s}$ and $\mathcal{C}_1^1\mathcal{C}_1^2 \approx_a \mathcal{C}^{x'_s}\mathcal{C}^{x''_s}$.

B'. By item B., the confluence lemma, and the forward closure lemma for the bisimulation (Lem. 35), we get: for all $x_\alpha \in \mathbf{In}(\Delta \Vdash \cdot)$ such that $x_\alpha \in \Theta_1^\kappa$, we have $\mathcal{C}'_2 \mapsto^* \mathcal{C}''_2$ such that $\mathbf{msg}(x_\alpha.q)\mathcal{C}'_1 \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{C}''_2$.

C'. By item C., the confluence lemma, and the forward closure lemma for the bisimulation (Lem. 35), we get: for all $x_\alpha \in \mathbf{In}(\cdot \Vdash K)$ such that $x_\alpha \in \Theta_1^\kappa$, we have $\mathcal{C}'_2 \mapsto^* \mathcal{C}''_2$ such that $\mathcal{C}'_1\mathbf{msg}(x_\alpha.q) \approx_a \mathcal{C}''_2\mathbf{msg}(x_\alpha.q)$.

There are two parts we need to prove:

Part 1. $\forall x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa).\,\mathbf{if}\, x_\alpha \in \Upsilon_1^\kappa.\,\mathbf{then}\ (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m'+1}_{\cdot;x_\alpha}$

Assume an arbitrary $x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa)$ with $x_\alpha \in \Upsilon_1$. We consider cases based on the type of x_α. Here we provide the detailed proof for one case, the proof of other cases is similar.

Subcase 1. $K^\kappa = x_\alpha{:}A \otimes B$. We need to prove $(\mathcal{C}_1^{\kappa'},\mathcal{C}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^\kappa \Vdash x_\alpha{:}A \otimes B]\!]^{m'+1}_{\cdot;x_\alpha}$. We first establish $\mathcal{C}_1^{\kappa'} = \mathcal{C}_1^{\kappa''}\mathcal{A}_1\mathbf{msg}(\mathbf{send}\,y_\beta\,x_\alpha)$ and $\mathcal{C}_2^{\kappa''} = \mathcal{C}_2^{\kappa'''}\mathcal{A}_2\mathbf{msg}(\mathbf{send}\,y_\beta\,x_\alpha)$, using item [A'] above and well-typedness of programs. Next, we apply the inductive hypothesis to show $(\mathcal{A}_1;\mathcal{A}_2) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash y_\beta{:}A]\!]^{m'}$ and also $(\mathcal{C}_1^{\kappa''};\mathcal{C}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash x_{\alpha+1}{:}B]\!]^{m'}$ with $\Delta^\kappa = \Delta_1^\kappa, \Delta_2^\kappa$. We get this by configuration typing, $\mathcal{C}_1^1\mathcal{C}_1^2 \approx_a \mathcal{C}^{x'_s}\mathcal{C}^{x''_s}$ given in item [A'] above and the fact that both $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{C}_1^{\kappa''}$ and $\mathcal{C}_2^{\kappa'''}$ are separate trees in the forests $\mathcal{C}_1^1\mathcal{C}_1^2$ and $\mathcal{C}^{x'_s}\mathcal{C}^{x''_s}$.

Part 2. $\forall x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa).\mathbf{if}\, x_\alpha \in \Theta_1^\kappa.\,\mathbf{then}\, (\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''}) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m'+1}_{x_\alpha;\cdot}$. Assume an arbitrary $x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa)$ with $x_\alpha \in \Theta_1$. We consider cases based on the type of x_α. Here we provide the detailed proof for one case, the proof of other cases is similar.

Subcase 1. $\Delta^\kappa = \Delta_1^\kappa x_\alpha{:}A\otimes B$. We need to prove $(\mathcal{C}_1^{\kappa'},\mathcal{C}_2^{\kappa''}) \in \mathcal{V}[\![\Delta_1^\kappa, x_\alpha{:}A\otimes B \Vdash K^\kappa]\!]^{m'+1}_{x_\alpha;\cdot}$. We apply the inductive hypothesis on $\mathbf{msg}(\mathbf{send}\, y_\beta\, x_\alpha)\mathcal{C}_1' \approx_a \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{C}_2''$ which is given by item [B'] above to get $(\mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{C}_1^{\kappa'}, \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{C}_2^{\kappa''}) \in \mathcal{E}[\![\Delta_1^\kappa, y_\beta{:}A, x_{\alpha+1}{:}B \Vdash K^\kappa]\!]^{m'}$ which completes the proof of this case.

Lemma 35 (Forward closure). *For all $\mathcal{D}_1, \mathcal{D}_2$ if $\mathcal{D}_1 \approx_a \mathcal{D}_2$ and $\mathcal{D}_2 \Rightarrow \mathcal{D}_2'$, then $\mathcal{D}_1 \approx_a \mathcal{D}_2'$.*

Proof. The proof is by coinduction on the generative function of the bisimulation $(\mathcal{D}_1 \approx_a \mathcal{D}_2')$. We consider four cases required to establish $\mathcal{D}_1 \approx_a \mathcal{D}_2'$.

1. $(\tau - \mathtt{step})$ if $\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ then by the assumption $\mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2''$ and $\mathcal{D}_1' \approx_a \mathcal{D}_2''$. This gives us $\mathcal{D}_2 \mapsto^* \mathcal{D}_2''$. We also know by assumption that $\mathcal{D}_2 \mapsto^* \mathcal{D}_2'$. By the confluence lemma(Lem. 21), we can build a $\mathcal{D}$ such that $\mathcal{D}_2'' \mapsto^* \mathcal{D}$ and $\mathcal{D}_2' \mapsto^* \mathcal{D}$. By coinduction on $\mathcal{D}_1' \approx_a \mathcal{D}_2''$ having the assumption $\mathcal{D}_2'' \mapsto^* \mathcal{D}$, we get $\mathcal{D}_1' \approx_a \mathcal{D}$, which along with $\mathcal{D}_2' \xRightarrow{\tau} \mathcal{D}$ is enough to get the result.
2. (output) if $\mathcal{D}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}_1'$ then by the assumption $\mathcal{D}_2 \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}_2^4$ and $\mathcal{D}_1' \approx_a \mathcal{D}_2^4$. This gives us $\mathcal{D}_2 \mapsto^* \mathcal{D}_2''$, and $\mathcal{D}_2'' \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}_2'''$, and $\mathcal{D}_2''' \mapsto^* \mathcal{D}_2^4$. We also know by assumption that $\mathcal{D}_2 \mapsto^* \mathcal{D}_2'$. By the confluence lemma(Lem. 21), we can build a $\mathcal{D}$ such that $\mathcal{D}_2'' \mapsto^* \mathcal{D}$ and $\mathcal{D}_2' \mapsto^* \mathcal{D}$. Moreover, for some $\mathcal{D}'$ we have $\mathcal{D} \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}'$, such that $\mathcal{D}_2''' \mapsto^* \mathcal{D}'$. Recall that we also have $\mathcal{D}_2''' \mapsto^* \mathcal{D}_2^4$. Again, we apply the confluence lemma(Lem. 21) to get a $\mathcal{D}''$ such that $\mathcal{D}_2^4 \mapsto^* \mathcal{D}''$ and $\mathcal{D}' \mapsto^* \mathcal{D}''$. By coinduction on $\mathcal{D}_1' \approx_a \mathcal{D}_2^4$ having the assumption $\mathcal{D}_2^4 \mapsto^* \mathcal{D}''$, we get $\mathcal{D}_1' \approx_a \mathcal{D}''$. Moreover, we have $\mathcal{D}_2' \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}''$, i.e., $\mathcal{D}_2' \mapsto^* \mathcal{D}$, and $\mathcal{D} \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}'$, and $\mathcal{D}' \mapsto^* \mathcal{D}''$. This is enough to get the result.
3. $(\mathsf{left\,input})$ Consider an arbitrary q which is not free in $\mathcal{D}_1$, and assume $\mathcal{D}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{D}_1'$, then $\exists \mathcal{D}_2''.\mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2''$ and $\mathcal{D}_1' \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}_2''$,. This gives us $\mathcal{D}_2 \mapsto^* \mathcal{D}_2''$. We also know by assumption that $\mathcal{D}_2 \mapsto^* \mathcal{D}_2'$. By the confluence lemma(Lem. 21), we can build a $\mathcal{D}$ such that $\mathcal{D}_2'' \mapsto^* \mathcal{D}$ and $\mathcal{D}_2' \mapsto^* \mathcal{D}$. This also gives us $\mathbf{msg}(x_\alpha.q)\mathcal{D}_2'' \mapsto^* \mathbf{msg}(x_\alpha.q)\mathcal{D}$. By coinduction on $\mathcal{D}_1' \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}_2''$ having the assumption $\mathbf{msg}(x_\alpha.q)\mathcal{D}_2'' \mapsto^* \mathbf{msg}(x_\alpha.q)\mathcal{D}$, we get $\mathcal{D}_1' \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}$. Moreover, we have $\mathcal{D}_2' \Rightarrow \mathcal{D}$, i.e., $\mathcal{D}_2' \mapsto^* \mathcal{D}$. This is enough to complete the proof of this case.
4. $(\mathsf{right\,input})$ for all $q \notin \mathsf{fn}(\mathcal{D}_1)$, if $\mathcal{D}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{D}_1'$ then $\exists \mathcal{D}_2'.\mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ and $\mathcal{D}_1' \approx_a \mathcal{D}_2'\mathbf{msg}(x_\alpha.q)$. The proof is similar to the previous case.

L Biorthogonality

L.1 Equivalence

Definition 36 (Session-typed environment). *A session-typed environment with the interface $\Delta \Vdash K$, is of the form $\mathcal{C}[\,]\mathcal{F}$, such that for some Δ' and K', we have $\Delta' \Vdash \mathcal{C} :: \Delta$ and $K \Vdash \mathcal{F} :: K'$.*

Definition 37 (Program- and environment- relations). *A session program-relation is a binary relation between session-typed programs, i.e., open configurations of the form $\Delta \Vdash \mathcal{D} :: K$. Given the interface $\Delta \Vdash K$, we write $PRel(\Delta \Vdash K)$ for the set of all program relations that relate programs $\Delta \Vdash \mathcal{D} :: K$.*

A session environment-relation is a binary relation between session-typed environments. Given the interface $\Delta \Vdash K$, we write $ERel(\Delta \Vdash K)$ for the set of all environment relations that relate environments $\mathcal{C}[\,]\mathcal{F}$ with the interface $\Delta \Vdash K$.

Definition 38 (The $(\_)^\top$ operation on relations). *Given any interface $\Delta \Vdash K$ and $r \in PRel(\Delta \Vdash K)$, we define $r^\top \in ERel(\Delta \Vdash K)$ by*

$$(\mathcal{C}_1[\,]\mathcal{F}_1, \mathcal{C}_2[\,]\mathcal{F}_2) \in r^\top \text{ iff } \forall (\mathcal{D}_1, \mathcal{D}_2) \in r. (\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_a \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2);$$

and given any $s \in ERel(\Delta \Vdash K)$, we define $s^\top \in PRel(\Delta \Vdash K)$ by

$$(\mathcal{D}_1, \mathcal{D}_2) \in s^\top \text{ iff } \forall (\mathcal{C}_1[\,]\mathcal{F}_1, \mathcal{C}_2[\,]\mathcal{F}_2) \in s. (\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_a \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2);$$

As explained in Pitts [48], just by virtue of how these definitions are defined, we get that the operation is a Galois connection, which is inflationary, and idempotent.

Definition 39.
- *Define the relation $\Delta \Vdash \mathcal{D}_1 :: K \equiv \Delta \Vdash \mathcal{D}_2 :: K$ as $(\mathcal{D}_1, \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $\forall m. (\mathcal{D}_1, \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]$ and $\forall m. (\mathcal{D}_2, \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]$.*
- *Define the relation $\Delta \dashv \mathcal{C}_1[\,]\mathcal{F}_1 \dashv K \equiv \Delta \dashv \mathcal{C}_2[\,]\mathcal{F}_2 \dashv K$ as (i) $\exists K'$ such that $K \Vdash \mathcal{F}_1 :: K' \equiv K \Vdash \mathcal{F}_2 :: K'$ and (ii) $\exists \Delta'$ such that $(\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{Forest}(\Delta' \Vdash \Delta)$ and $\forall \mathcal{T}_1 \in \mathcal{C}_1$, and $\forall \mathcal{T}_2 \in \mathcal{C}_2$ with $(\mathcal{T}_1, \mathcal{T}_2) \in \mathsf{Tree}(\Delta'_1 \Vdash K'')$, we have $\Delta'_1 \Vdash \mathcal{T}_1 :: K'' \equiv \Delta'_1 \Vdash \mathcal{T}_2 :: K''$.*

Theorem 8 ($\top\top$-closure). *Consider $(\mathcal{D}_1, \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$, we have*

$$(\Delta \Vdash \mathcal{D}_1 :: K) \equiv (\Delta \Vdash \mathcal{D}_2 :: K)$$

$$\textbf{iff}$$

$$\forall \mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2. \text{ s.t. } (\Delta \dashv \mathcal{C}_1[\,]\mathcal{F}_1 \dashv K) \equiv (\Delta \dashv \mathcal{C}_2[\,]\mathcal{F}_2 \dashv K) \text{ we have }$$
$$\forall m. \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_a \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2$$

Proof. There are two directions to consider:

Left to Right: The result is straightforward by the compositionality result(Corollary 2) and the soundness result (Lem. 34) and the definition of equivalence $\equiv$ for programs and environments (Def. 39).

Right to Left: Consider $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{F}_1 = \mathcal{F}_2 = \cdot$. By Lem. 31 and Def. 39, we know that $(\Delta \dashv \cdot [\ ] \cdot \dashv K) \equiv (\Delta \dashv \cdot [\ ] \cdot \dashv K)$. From this we get $\forall m. \mathcal{D}_1 \approx_a \mathcal{D}_2$, and by the completeness lemma (Lem. 34) we get $\forall m. (\mathcal{D}_1, \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m. (\mathcal{D}_2, \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$. By Def. 39, we have $\Delta \Vdash \mathcal{D}_1 :: K \equiv \Delta \Vdash \mathcal{D}_2 :: K$, which completes the proof.

This result is enough to prove that our equivalence relation is $\top\top$-closed, i.e., following the results presented in the results presented in Pitts [48], it shows $r = s^{\top}$, when r and s defined as our logical equivalence.

L.2 Noninterference

Definition 40 (Session-typed environment). *A session-typed environment with the interface $\Delta \Vdash K$, is of the form $\mathcal{C}[\ ]\mathcal{F}$, such that for some Δ' and K', we have $\Delta' \Vdash \mathcal{C} :: \Delta$ and $K \Vdash \mathcal{F} :: K'$.*

Definition 41 (Program- and environment- relations). *A security session program-relation is a binary relation between session-typed programs, i.e., open configurations of the form $\Delta \Vdash \mathcal{D} :: K$. Given a security level $\xi \in \Psi_0$, and two interfaces $\Gamma_1 \Vdash x_\alpha{:}A_1[c_1]$ and $\Gamma_2 \Vdash y_\beta{:}A_2[c_2]$ we write $PRel^{\xi}(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$ for the set of all security program relations that relate programs $|\Gamma_1| \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1$ and $|\Gamma_2| \Vdash \mathcal{D}_2 :: y_\beta{:}A_2$.*

A **low-security** *session environment-relation is a binary relation between low security session-typed environments. Given two interfaces $\Gamma_1 \Vdash x_\alpha{:}A_1[c_1]$ and $\Gamma_2 \Vdash y_\beta{:}A_2[c_2]$, we write $ERel^{\xi}(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$ for the set of all environment relations that relate low security session-typed environments $\mathcal{C}_1[\ ]\mathcal{F}_1$ with the interface $|\Gamma_1 \Downarrow \xi| \Vdash |x_\alpha{:}A_1[c_1] \Downarrow \xi|$ and $\mathcal{C}_2[\ ]\mathcal{F}_2$ with the interface $|\Gamma_2 \Downarrow \xi| \Vdash |y_\beta{:}A_2[c_2] \Downarrow \xi|$.*

Definition 42 (The $(\_)^{\top}$ operation on relations). *Given two interfaces $\Gamma_1 \Vdash x_\alpha{:}A_1[c_1]$ and $\Gamma_2 \Vdash y_\beta{:}A_2[c_2]$ and the observer security level $\xi \in \Psi_0$ and $r \in PRel^{\xi}(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$, we define $r^{\top} \in ERel^{\xi}(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$ by*

$$\begin{gathered}(\mathcal{C}_1[\ ]\mathcal{F}_1, \mathcal{C}_2[\ ]\mathcal{F}_2) \in r^{\top} \text{ iff } \forall (\mathcal{D}_1, \mathcal{D}_2) \in r. \\ \forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]),\ \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]). \\ \mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1 \approx_a \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2\end{gathered}$$

and given any $s \in ERel(\Delta \Vdash K)$, we define $s^{\top} \in PRel(\Delta \Vdash K)$ by

$$\begin{gathered}(\mathcal{D}_1, \mathcal{D}_2) \in s^{\top} \text{ iff } \forall (\mathcal{C}_1[\ ]\mathcal{F}_1, \mathcal{C}_2[\ ]\mathcal{F}_2) \in s. \\ \forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]),\ \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]). \\ \mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1 \approx_a \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2\end{gathered}$$

As explained in Pitts [48], just by virtue of how these definitions are defined, we get that the operation is a Galois connection, which is inflationary, and idempotent.

Definition 43 (Equivalence of forests by the logical relation upto the observer level). *We write* $\Gamma'_1 \Vdash \mathcal{C}_1 :: \Gamma_1 \equiv^{\xi}_{\Psi_0} \Gamma'_2 \Vdash \mathcal{C}_2 :: \Gamma_2$ *iff*

(i) **(Both forests are well-typed.)** $\mathcal{C}_1 \in \mathsf{Forest}(|\Gamma'_1| \Vdash |\Gamma_1|)$ *and* $\mathcal{C}_2 \in \mathsf{Forest}(|\Gamma'_2| \Vdash |\Gamma_2|)$

(ii) **(Their observable interface is the same.)** $\Gamma'_1 \Downarrow \xi = \Gamma'_2 \Downarrow \xi$ *and* $\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi$, *and*

(iii) **(Each tree in the 1st forest with an observable interface has a counterpart in the 2nd forest which is equivalent to it.)** *for every* $\mathcal{T}_1 \in \mathcal{C}_1$ *such that* $\mathcal{T}_1 \in \mathsf{Tree}(|\Gamma''_1| \Vdash |x_\alpha{:}A_1[c_1]|)$, *and* $\Gamma''_1 \subseteq \Gamma'_1$ *and* $x_\alpha{:}A_1[c_1] \in \Gamma_1$ *and* $(\Gamma''_1, x_\alpha{:}A_1[c_1]) \Downarrow \xi \neq \cdot$ *there exists a* $\mathcal{T}_2 \in \mathcal{C}_2$ *such that* $\mathcal{T}_2 \in \mathsf{Tree}(|\Gamma''_2| \Vdash |y_\beta{:}A_2[c_2]|)$, *and* $\Gamma''_2 \subseteq \Gamma'_2$ *and* $y_\beta{:}A_2[c_2] \in \Gamma_2$ *for* $\Gamma''_1 \Downarrow \xi = \Gamma''_2 \Downarrow \xi$ *and* $y_\beta{:}A_2[c_2] \Downarrow \xi = x_\alpha{:}A_1[c_1] \Downarrow \xi$, *and*

(iv) **(Each tree in the 2nd forest with an observable interface has a counterpart in the 1st forest which is equivalent to it.)** *vice versa.*

Definition 44. *Define the relation*

$$\Gamma_1 \dashv \mathcal{C}_1[\ ]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1] \equiv^{\xi}_{\Psi_0} \Gamma_2 \dashv \mathcal{C}_2[\ ]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2] \ \text{ as}$$

(i) **(The observable interface of the environments are the same.)** *We have* $\Gamma = \Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi$ *and* $K^s = x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi$, *and*

(ii) **(The configurations $\mathcal{C}_1$ and $\mathcal{C}_2$ offer channels annotated as observable and use the same channels and are equivalent for any observable (low-confidentiality) annotation of their resource channel.)** *for some* Δ, *we have* $(\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{Forest}(\Delta \Vdash |\Gamma|)$. *In particular, if* $\Gamma = \cdot$, *we have* $\mathcal{C}_1 = \mathcal{C}_2 = \cdot$.
Moreover, for every Γ'_1 *and* Γ'_2 *such that* $|\Gamma'_1| = |\Gamma'_2| = \Delta$, *and* $\forall w_\eta{:}A[c] \in \Gamma'_1.\ c \sqsubseteq \xi$ *and* $\forall w_\eta{:}A[c] \in \Gamma'_2.\ c \sqsubseteq \xi$ *we have* $\Gamma'_1 \Vdash \mathcal{C}_1 :: \Gamma \equiv^{\xi}_{\Psi_0} \Gamma'_2 \Vdash \mathcal{C}_2 :: \Gamma$.

(iii) **(The configurations $\mathcal{F}_1$ and $\mathcal{F}_2$ use channels annotated as observable, offer the same channels, and are equivalent for any observable (low-confidentiality) annotation of their offering channels.)** *If* $K^s = \_{:}1[\top]$, *then* $\mathcal{F}_1 = \mathcal{F}_2 = \cdot$. *Otherwise, for some* $w_\eta{:}A$, *we have* $(\mathcal{F}_1, \mathcal{F}_2) \in \mathsf{Tree}(K^s \Vdash w_\eta{:}A)$. *Moreover,* $\forall c, c' \sqsubseteq \xi$, *we have* $K^s \Vdash \mathcal{F}_1 :: w_\eta{:}A[c] \equiv^{\xi}_{\Psi_0} K^s \Vdash \mathcal{F}_2 :: w_\eta{:}A[c']$ *and.*

Lemma 36 (Compositionality of $\equiv^{\xi}_{\Psi_0}$). *If*

(i) $(\Gamma'_1 \Vdash \mathcal{C}_1 :: \Gamma) \equiv^{\xi}_{\Psi_0} (\Gamma'_2 \Vdash \mathcal{C}_2 :: \Gamma)$ with $\Gamma = \Gamma_1 \Downarrow_\xi = \Gamma_2 \Downarrow_\xi$ and
(ii) $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ and
(iii) $(K^s \Vdash \mathcal{F}_1 :: K^s_1) \equiv^{\xi}_{\Psi_0} (K^s \Vdash \mathcal{F}_2 :: K^s_2)$ with $K^s = x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi$ then

$$(\Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: K^s_1) \equiv^{\xi}_{\Psi_0} (\Gamma'_2, \Gamma^h_2 \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: K^s_2),$$

where Γ^h_1 *is the set of all channels* $w_\eta{:}C[d] \in \Gamma_1$ *with* $d \not\sqsubseteq \xi$ *and* Γ^h_2 *is the set of all channels* $w_\eta{:}C[d] \in \Gamma_2$ *with* $d \not\sqsubseteq \xi$.

Proof. By the configuration typing and definition of equivalence (Def. 26) and (i), (ii), (iii), we know that $\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \in \mathsf{Tree}(|\Gamma'_1| \Vdash |K^s_1|)$ and $\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 \in \mathsf{Tree}(|\Gamma'_2| \Vdash |K^s_2|)$. Moreover, by definition of equivalence (Def. 26) and (i), (iii) we get $\Gamma' = (\Gamma'_1, \Gamma^h_1) \Downarrow \xi = (\Gamma'_2, \Gamma^h_1) \Downarrow \xi$ and $K^{s'} = K^s_1 \Downarrow \xi = K^s_2 \Downarrow \xi$.

Assume arbitrary configurations $\mathcal{B}_1 \in \textbf{H-Provider}^\xi(\Gamma'_1)$, $\mathcal{B}^h_1 \in \textbf{H-Provider}^\xi(\Gamma^h_1)$, $\mathcal{B}_2 \in \textbf{H-Provider}^\xi(\Gamma'_2)$, $\mathcal{B}^h_2 \in \textbf{H-Provider}^\xi(\Gamma^h_2)$, $\mathcal{T}_1 \in \textbf{H-Client}^\xi(K^s_1)$, and $\mathcal{T}_2 \in \textbf{H-Client}^\xi(K^s_2)$. Our goal is to prove

$$\forall m.\, (\mathcal{B}_1\mathcal{B}^h_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1, \mathcal{B}_2\mathcal{B}^h_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma'| \Vdash |K^{s'}|]\!]^m, \text{ and}$$
$$\forall m.\, (\mathcal{B}_2\mathcal{B}^h_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2, \mathcal{B}_1\mathcal{B}^h_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1) \in \mathcal{E}[\![|\Gamma'| \Vdash |K^{s'}|]\!]^m.$$

By assumptions (i)-(iii) we have:

(i') Note that for any trees $\mathcal{A}''_1 \in \mathcal{C}_1$ and $\mathcal{A}''_2 \in \mathcal{C}_2$ they have an observable offering channel occuring in Γ. Assume that $\mathcal{C}_1$ as a forest includes n separate trees. By the previous observation, we know that $\mathcal{C}_2$ also consists of n separate trees. For any $j \leq n$ consider the tree $\mathcal{A}^j_1 \in \mathcal{C}_1$ consider the corresponding tree $\mathcal{A}^j_2 \in \mathcal{C}_2$ that exists by Def. 43 and for which we have $(\dagger_j)$ $\Gamma^{j'}_1 \Vdash \mathcal{A}^j_1 :: w^j_\eta{:}A^j[c^j] \equiv^\xi_{\Psi_0} \Gamma^{j'}_2 \Vdash \mathcal{A}^j_2 :: w^j_\eta{:}A^j[c^j]$.

From $\dagger_j$, we get $\Gamma^{j'} = \Gamma^{j'}_1 = \Gamma^{j'}_2$ and for arbitrary configurations $\mathcal{T}^j_1 \in \textbf{H-Client}^\xi(\Gamma^{j'}_1)$, and $\mathcal{T}^j_2 \in \textbf{H-Client}^\xi(\Gamma^{j'}_2)$, we have

$$\forall m.\, (\mathcal{B}^j_1\mathcal{C}_1, \mathcal{B}^j_2\mathcal{C}_2) \in \mathcal{E}[\![|\Gamma^{j'} \Downarrow \xi| \Vdash w^j_\eta{:}A^j]\!]^m, \text{ and}$$
$$\forall m.\, (\mathcal{B}^j_2\mathcal{C}_2, \mathcal{B}^j_1\mathcal{C}_1) \in \mathcal{E}[\![|\Gamma^{j'} \Downarrow \xi| \Vdash w^j_\eta{:}A^j]\!]^m.$$

Observe that $\Gamma'_1 = \{\Gamma^{j'}_1\}_{j \leq n}$ and $\Gamma'_2 = \{\Gamma^{j'}_2\}_{j \leq n}$, and $\Gamma = \{w^j_\eta{:}A^j[c^j]\}_{j \leq n}$.

(ii') By Def. 26, we know that for arbitrary configurations $\mathcal{B}^h_1 \in \textbf{H-Provider}^\xi(\Gamma_1)$, and $\mathcal{B}^d_2 \in \textbf{H-Provider}^\xi(\Gamma_2)$, and $\mathcal{T}^d_1 \in \textbf{H-Client}^\xi(x_\alpha{:}A_1[c_1])$, and $\mathcal{T}^d_2 \in \textbf{H-Client}^\xi(y_\beta{:}A_2[c_2])$.

$$\forall m.\, (\mathcal{B}^d_1\mathcal{D}_1\mathcal{T}^d_1, \mathcal{B}^d_2\mathcal{D}_2\mathcal{T}^d_2) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m, \text{ and}$$
$$\forall m.\, (\mathcal{B}^d_2\mathcal{D}_2\mathcal{T}^d_2, \mathcal{B}^d_1\mathcal{D}_1\mathcal{T}^d_1) \in \mathcal{E}[\![|\Gamma| \Vdash |K^s|]\!]^m.$$

(iii') By Def. 26, if $c_1 \sqsubseteq \xi$ and $c_2 \sqsubseteq \xi$, we have for any $\mathcal{T}_1 \in \textbf{H-Client}^\xi(K^s_1)$, and $\mathcal{T}_2 \in \textbf{H-Client}^\xi(K^s_2)$

$$\forall m.\, (\mathcal{F}_1\mathcal{T}^h_1, \mathcal{F}_2\mathcal{T}^h_2) \in \mathcal{E}[\![|K^s| \Vdash |K^{s'}|]\!]^m, \text{ and}$$
$$\forall m.\, (\mathcal{F}_2\mathcal{T}^h_2, \mathcal{F}_1\mathcal{T}^h_1) \in \mathcal{E}[\![|K^s| \Vdash |K^{s'}|]\!]^m,$$

It is straightforward to show that given (i'-iii'), and several applications of the compositionality lemma Corollary 2 we get the goal.

Theorem 9 ($\top\top$-closure). *Consider* $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash |x_\alpha{:}A_1[c_1]|)$ *and* $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash |y_\beta{:}A_2[c_2]|)$ *and a given observer level* $\xi \in \Psi_0$. *We have*

$$(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$$

iff

$$\forall \mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2.\, \text{s.t.}\, (\Gamma_1 \dashv \mathcal{C}_1[\,]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \dashv \mathcal{C}_2[\,]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2]) \text{ we have}$$
$$\forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]),\ \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]).$$
$$\mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1 \approx_a \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2$$

Proof. There are two directions to consider:

Left to Right: Consider $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ and arbitrary $\mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2$ such that we have

$$(\Gamma_1 \dashv \mathcal{C}_1[\,]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \dashv \mathcal{C}_2[\,]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2]).$$

By Def. 44, we get that $\Gamma = \Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi$ and $K^s = x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi$. Moreover for some Δ, we have $(\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{Forest}(\Delta \Vdash |\Gamma|)$ and for any low secrecy annotations of Δ as Γ'_1 and Γ'_2 we get $\Gamma'_1 \Vdash \mathcal{C}_1 :: \Gamma \equiv^{\xi}_{\Psi_0} \Gamma'_2 \Vdash \mathcal{C}_2 :: \Gamma$. Also if $K^s = \_{:}1[\top]$, we have $\mathcal{F}_1 = \mathcal{F}_2 = \cdot$, and otherwise $(\mathcal{F}_1, \mathcal{F}_2) \in \mathsf{Tree}(K^s \Vdash w_\eta{:}A)$ for some $w_\eta{:}A$, for any low secrecy annotations (with $c_3, c_4 \sqsubseteq \xi$) of $w_\eta{:}A$ as $w_\eta{:}A[c_3]$ and $w_\eta{:}A[c_4]$ we get $K^s \Vdash \mathcal{C}_1 :: w_\eta{:}A[c_3] \equiv^{\xi}_{\Psi_0} K^s \Vdash \mathcal{C}_2 :: w_\eta{:}A[c_4]$.
Our goal is to prove:

$$\forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^{\xi}(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^{\xi}(x_\alpha{:}A_1[c_1]),\ \mathcal{T}_2 \in \mathsf{H\text{-}Client}^{\xi}(y_\beta{:}A_2[c_2]).$$
$$\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_a \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2.$$

By compositionality of equivalence upto relation (Lem. 36), we get $\Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: x_\alpha{:}A_1[c_1] \equiv^{\xi}_{\Psi_0} \Gamma'_2, \Gamma^h_2 \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: y_\beta{:}A_2[c_2]$ in the case where $K^s := \_{:}1[\top]$, i.e., $c_1, c_2 \not\sqsubseteq \xi$, and otherwise we get $\Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: w_\eta{:}A[c_3] \equiv^{\xi}_{\Psi_0} \Gamma'_2, \Gamma^h_2 \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: w_\eta{:}A[c_4]$ where Γ^h_1 is the set of all channels $z_\gamma{:}C[d] \in \Gamma_1$ with $d \not\sqsubseteq \xi$ and Γ^h_2 is the set of all channels $z_\gamma{:}C[d] \in \Gamma_2$ with $d \not\sqsubseteq \xi$.
By the soundness theorem for equivalence relation (Corollary 4), we get $\Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: x_\alpha{:}A_1[c_1] \approx^{\xi}_a \Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: y_\beta{:}A_2[c_2]$ in the first case where $K^s = \_{:}1[\top]$ and otherwise $\Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: w_\eta{:}A[c_3] \approx^{\xi}_a \Gamma'_1, \Gamma^h_1 \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: w_\eta{:}A[c_4]$. In the first case, the proof is complete by Def. 35 and the observation that all channels in Γ'_1 and Γ'_2 are annotated with channels of secrecy $d' \sqsubseteq \xi$. In the second case, the proof is complete by Def. 35 and observing that all channels in Γ'_1 and Γ'_2 are annotated with channels of secrecy $d' \sqsubseteq \xi$ and that $c_1, c_2 \sqsubseteq \xi$ and $c_3, c_4 \sqsubseteq \xi$.

Right to Left: Assume $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{F}_1 = \mathcal{F}_2 = \cdot$. By Lem. 31 and Def. 39, we know that $(\Gamma_1 \dashv \cdot[\,]\cdot \dashv x_\alpha{:}A_1[c_1]) \equiv (\Gamma_2 \dashv \cdot[\,]\cdot \dashv y_\beta{:}A_1[c_2])$. From this we get $\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1] \approx^{\xi}_a \Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2]$. Now, we can apply the completeness lemma (Corollary 4) to get $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\xi}_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$.

This result is enough to prove that our equivalence relation is $\top\top$-closed, i.e., following the results presented in the results presented in Pitts [48], it shows $r = s^{\top}$, when r and s defined as our logical equivalence.