Logical Relations for Session-Typed Concurrency, Technical Report

Stephanie Balzer  Farzaneh Derakhshan  Robert Harper  Yue Yao

I. Further Examples

A. Examples of leakage (rejected by \textsc{Consession})

This subsection gives several example programs that leak information by exploiting non-termination and concurrency. None of the examples use secrecy polymorphism to orchestrate the balance between a process’ maximal secrecy and running secrecy and to ensure that a process’ running secrecy stands for an arbitrary iteration of the process. The examples thus do not use secrecy variables nor running secrecy annotations, and hence do not type check in \textsc{Consession}.

1) Different recursive calls after branching on a high secrecy channel: In our first example, the sneaky verification process (\texttt{Sneaky\_Verifier}) leaks, with the help of an accomplice (\texttt{Sneaky\_Partner}), the information of whether Alice’s PIN provides a correct token. If verification of Alice’s PIN is successful, the sneaky verifier calls itself recursively, and if not, it calls its partner. \texttt{Sneaky\_Verifier} and its partner, \texttt{Sneaky\_Partner}, differ on how they interact with the attacker right after being spawned: the sneaky verifier sends the label \texttt{s} to the attacker, signaling preceding successful authentication, and its partner sends the label \texttt{f} to the attacker, signaling preceding unsuccessful authentication. We point out that the code would be perfectly secure without the recursive calls: the sends of label \texttt{s} and \texttt{f} in the \texttt{Sneaky\_Verifier} and \texttt{Sneaky\_Partner}, resp., happen before the security token is received. Recursion, however, provides a way of “rolling forward” information learned in one iteration to the next.

2) Exploiting non-termination: Recursive protocols significantly improve the expressiveness of the language but at the same time degrade liveness. In particular, we lose the guarantee that a process will eventually communicate with its provider or client. The process \texttt{Diverge} is an example of a process that is supposed to provide an infinite stream of terminating channel references \texttt{x}: \texttt{term}, but instead calls itself recursively without ever interacting with its client.

\begin{verbatim}
ver = pin → ⊕\{succ:pin ⊗ ver, fail:pin ⊗ ver\}

pin = ⊕\{tok1:pin, ..., tokn:pin\}

attacker = &\{s:attacker, f:attacker\}

y\_attacker[guest] ⊢ Sneaky\_Verifier1 :: x:ver[alice]

x ← Sneaky\_Verifier1

\{case z \_tok1 \_⇒ x.succ; send z x; (x\_ver ← Sneaky\_Verifier1 ← y\_attacker)
\| tok1,\_j \_⇒ x.fail; send z x; (x\_ver ← Sneaky\_Partner1 ← y\_attacker)\}\}

y\_attacker[guest] ⊢ Sneaky\_Partner1 :: x:ver[alice]

x ← Sneaky\_Partner1

\{case z \_tok1 \_⇒ x.succ; send z x; (x\_ver ← Sneaky\_Verifier1 ← y\_attacker)
\| tok1,\_j \_⇒ x.fail; send z x; (x\_ver ← Sneaky\_Partner1 ← y\_attacker)\}\}

\end{verbatim}

Diverge is only a trivial implementation of a non-reactive process, but there are more ingenious ways to implement such processes, and identifying them was shown to be undecidable[2].

A sneaky verifier can exploit the non-reactivity of \texttt{Diverge} to leak information to the attacker. In our second example, the sneaky verifier, \texttt{Sneaky\_Verifier2}, spawns the diverging process along channel \texttt{x}\textsubscript{1} and continues as its partner \texttt{Sneaky\_Partner2}. The partner only interacts with the diverging channel \texttt{x}\textsubscript{1} when Alice’s PIN provides the incorrect token. As a result, if Alice provides an incorrect token, the process will wait forever for \texttt{x}\textsubscript{1} to send it a terminating channel \texttt{x}\textsubscript{2}, and the attacker will receive no further messages. Otherwise, if Alice’s token is correct, \texttt{Sneaky\_Partner2} continues by recursively calling itself, and right at the beginning of the recursive call, sends the label \texttt{s} to the attacker, signaling the previous successful login.

3) Exploiting concurrency: The following example shows how two attackers can exploit concurrency to infer Alice’s success or failure in verifying her PIN by observing the verifier’s pattern in producing their low-secrecy messages. Process Sneaky Verifier 3 leaks secret information with the help of process Alternate_bits that produces an alternating sequence of bits b0; b1; b0; b1; . . . and its two mutually recursive partners, Sneaky Partnerb0 and Sneaky Partnerb1.

\[
\text{bits} = \oplus \{ b0 : \text{bits}, b1 : \text{bits} \}
\]
\[
\text{cobits} = \&\{ b0 : \text{bits}, b1 : \text{bits} \}
\]
\[
\text{leak} = \&\{ \text{succ} : \text{leak}, \text{fail} : \text{leak} \}
\]

\[
x_1: \text{bits} \vdash \text{Alternate_bits} :: x_1: \text{bits}
\]
\[
x_2 \vdash \text{Sneaky_Partner}_b :: x_2: \text{cobits}
\]
\[
x_3: \text{leak} \vdash \text{Sneaky_Partner}_b :: x_2: \text{cobits}
\]

At the beginning, the Sneaky Verifier 3 is connected to Alternate_bits along channel x1, and to Sneaky Partnerb0 along channel x2. The sneaky verifier starts by relaying the first bit (b0) that it receives from x1 to its accomplice along x2. Next, if the verification of Alice’s PIN is successful it receives one extra bit (b1) from the sequence provided by Alternate_bits without relaying it to its partner and then calls itself recursively. In this case, again, the next bit that it sends to Sneaky Partnerb0 is b0. In fact, as long as the verification is successful, the verifier only sends b0 messages to its partner. As soon as a failure occurs, the verifier skips receiving the extra bit from Alternate_bits and instead relays a b1 label to Sneaky Partnerb0 after its recursive call. With the same argument, we can observe that Sneaky Verifier 3 continues to send b1 labels until the next verification failure, after which it again flips to sending b0. The mutually recursive partners observe these flips in the sequence they receive along x2 and restore the sequence of Alice’s successful and failed logins by building a sequence of type leak.

In all the above examples, the leak is caused by sending to an attacker after receiving along a high-secrecy channel, which is prevented by a flow-sensitive IF system type system. However, in contrast to the non-recursive case, causality between actions can no longer be determined locally, by just considering one iteration of either a sneaky verifier or its partners. The leaks come about because of mutually recursive calls, non-termination of spawned processes, and concurrent communications.

Our solution is to type processes not only by considering one iteration in isolation but considering both the maximal and running secrecy. The running secrecy must thereby be a sound approximation of an arbitrary iteration. To ensure compositionality, we moreover allow process definitions to be polymorphic in their maximal and running secrecy, given restrictions on the relationships between those variables.

To illustrate the use of polymorphic process definitions, we revisit the unsafe example from §I-1-A1 and explain why our type system rejects it. We focus on this example, but the same argument holds for the remaining ones.

Consider the below definition of Sneaky Verifier 1 and assume that the security theory \( \Psi \) contains the constraints \( \psi_1 = \text{guest} \) and \( \psi = \text{alice} \).
Following the signature typing rule, the security theory \( \Psi \), must satisfy \( \Psi \vdash \psi' = \psi_1 \sqcup \text{alice} = \psi \), \( \psi' \sqsubseteq \psi = \text{alice} \). Moreover, by &L, we must have \( \Psi \vdash \psi' \sqsubseteq \psi_1 \), to execute \( y.s \). By \( \rightarrow R \), we know that the running secrecy of the process after executing \( z \leftarrow \text{recv } x^\psi \) increases to \( \psi ' \). And by the spawn rule, we need to know that the running secrecy \( d_2 \) specified for the callee is at least as high as the caller’s, i.e. \( \Psi \vdash \psi \sqsubseteq d_2 \). It is straightforward to observe that there is no possible substitution \( \gamma \) to unify \( d_2 \) with \( \psi' \) as required by the spawn rule, i.e. \( \gamma(\Psi) \not\vdash d_2 \sqsubseteq \psi_1 = \text{guest} \sqcup \text{alice} = \psi \sqsubseteq d_2 \).

**B. Sneaky verifier - revisited.**

We briefly illustrate polymorphic processes on the SneakyVerifier discussed in § II in the main text and show that it is not a well-typed process in CONSESSION.

**attacker = \{ \text{\texttt{sender}, \texttt{move\_ver}\} }**

\[
\Psi; y.\text{\texttt{attacker}}[\psi_1] \vdash \text{\texttt{Sneaky\_Verifier}}[\psi] :: x.\text{\texttt{ver}}[\psi] \\
x \leftarrow \text{\texttt{Sneaky\_Verifier}}[\psi] \leftarrow y = (y.s; z \leftarrow \text{\texttt{recv}} x) \\
case \text{\texttt{z}}(\text{\texttt{tok}}_1) \Rightarrow x.\text{\texttt{succ}}; \text{\texttt{send}} z; x \leftarrow \text{\texttt{Sneaky\_Verifier}}[\psi][\text{\texttt{d}_2}] \leftarrow y.\text{\texttt{attacker}} \\
| \text{\texttt{tok}}_1 \Rightarrow x.\text{\texttt{fail}}; y.f; \text{\texttt{send}} z; x \leftarrow \text{\texttt{Sneaky\_Verifier}}[\psi][\text{\texttt{d}_1}] \leftarrow y.\text{\texttt{attacker}})@\psi' \\
\]

Assume that the security theory \( \Psi \) contains the constraints \( \psi_1 = \text{guest} \) and \( \psi = \text{alice} \). By \( \Sigma_1 \), \( \Psi \) must satisfy \( \Psi \vdash \psi' \sqsubseteq \psi_1 \sqcup \text{alice} = \psi \) and \( \Psi \vdash \psi' \sqsubseteq \psi = \text{alice} \). Moreover, by &L, we must have \( \Psi \vdash \psi' \sqsubseteq \psi_1 \), to execute \( y.s \). By \( \rightarrow R \), we know that the running secrecy of the process after executing \( z \leftarrow \text{recv } x^\psi \) increases to \( \psi ' \). And by Spawn, we need to know that the running secrecy \( \gamma(\psi') \) specified for the callee is at least as high as the caller’s, i.e. \( \Psi \vdash \psi \sqsubseteq \gamma(\psi') \). Moreover, the callee must satisfy the instantiation of the callee’s constraints, i.e., \( \Psi \vdash \gamma: \Psi \), and thus \( \Psi \vdash \gamma(\psi') \sqsubseteq \gamma(\psi_1) = \text{guest} \) must hold. It is straightforward to observe that there is no possible substitution \( \psi \vdash \gamma: \Psi \) that satisfies all these requirements, i.e., \( \Psi \vdash \gamma(\psi') \sqsubseteq \gamma(\psi_1) = \text{guest} \sqcup \text{alice} = \psi \sqsubseteq \gamma(\psi') \).

**C. Secrecy-polymorphic processes to the rescue**

As a bonus to our polymorphic treatment of processes, we get to define generic process definitions. This section completes our banking example by providing generic implementations for a customer and its authorization process that can be spawned for any specific customer of the bank. We add the following section types to our example:

**customer = pin \rightarrow \text{\texttt{auth\_out}} \rightarrow 1**

pin= \{ \text{\texttt{tok}_1: pin, \ldots, tok_n: pin} \}

auth\_out= pin \rightarrow \{ \text{\texttt{succ: account \sqcup auth\_in, fail: pin \sqcup auth\_out}} \}

auth\_in= \text{\texttt{account \rightarrow pin \sqcup auth\_out}}

ver= pin \rightarrow \{ \text{\texttt{succ: pin \sqcup ver, fail: pin \sqcup ver}} \}

account = \{ \text{\texttt{high: account, med: account, low: account}} \}

The bank sets up a new customer by sending the customer an authentication PIN and an authorization process that guards their account.

\[
\Psi; y_1: \text{\texttt{customer}}[\psi_1], u_1: \text{\texttt{pin}}[\psi_1], z_1: \text{\texttt{auth\_out}}[\psi_1], \Psi_0: \psi_1 = \text{\texttt{alice}}, \psi_2 = \text{\texttt{bob}}, \psi_3 = \text{\texttt{bank}}, \psi' = \text{\texttt{guest}}, \psi_0 \\
y_2: \text{\texttt{customer}}[\psi_2], u_2: \text{\texttt{pin}}[\psi_2], z_2: \text{\texttt{auth\_out}}[\psi_2] \vdash \text{\texttt{bank}} :: u_1: [\psi] \\
w \leftarrow \text{\texttt{bank}}[\psi'] \leftarrow y = (\text{\texttt{send}} u_1^{\psi_1} y_1; \text{\texttt{send}} z_1^{\psi_2} y_1; \text{\texttt{send}} z_2^{\psi_3} y_2; \text{\texttt{send}} z_3^{\psi_0} y_2; \text{\texttt{wait}} y_1; \text{\texttt{wait}} y_2; \text{\texttt{close}} w) \\
\Psi_1: \vdash \text{\texttt{New\_Customer}}[\psi_1] :: y.\text{\texttt{customer}}[\psi_1] \\
y \leftarrow \text{\texttt{New\_Customer}} \leftarrow (w^\psi \leftarrow \text{\texttt{recv}} y; z^\psi \leftarrow \text{\texttt{recv}} y; // w^\psi[\psi], z^\text{\texttt{auth\_out}}[\psi] \vdash y_1[\psi]) \\
y^\psi: 1 \leftarrow \text{\texttt{Customer}}[\gamma_1]@\gamma(\psi') \leftarrow u.\text{\texttt{pin}}, z^\text{\texttt{auth\_out}}[\psi]@\psi' \\
\]

where \( \gamma_1: \{ \psi \rightarrow \text{\texttt{bank}}, \psi_1 \rightarrow \text{\texttt{alice}}, \psi_2 \rightarrow \text{\texttt{bob}}, \psi' \rightarrow \text{\texttt{guest}} \} \).

The types auth\_out and auth\_in describe the communication of the authorization process with the customer. The authorization process receives a PIN from the customer and sends it to the verifier to validate the login. If the verifier process signals a successful verification, the authorization goes ahead and logs in the customer by sending a succ label, followed by the account to the customer. If the PIN verification is unsuccessful, the authorization process returns the incorrect PIN to the customer and remains logged out. In the case of a successful verification, the customer remains logged in until it requests a log out by sending back its account to the authorization process; upon such a request, the account sends back the PIN to the customer and logs them out.
where $$\gamma$$.

The account provides a customer’s balance by signaling the corresponding label along $$v\text{account}$$. For example, Alice’s account sends label high to her after being authorized.

$\Psi_2:=\psi=\text{alice}, \psi' \sqsubseteq \psi, \Psi_0$

$\Psi_2:=\psi=\text{alice}, \psi' \sqsubseteq \psi, \Psi_0$

We provide the security theory of each process variable consisting of its essential constraints next to its definition. We leave it to the reader to check that in each spawn, the caller can assert the security theory of their callee. In contrast to Alice’s verifier, her PIN, and her account, the customer and authorization processes do not have any specific information about Alice, e.g., her correct PIN, hard-coded in their code. Thus, we can define them as generic processes that can be spawned for any customer. Their caller only needs to instantiate their secrecy variables with the customer’s correct secrecy level when spawned for a specific customer.
II. ABSTRACT SYNTAX

Fig. 1 defines the abstract syntax of SESSION. Lines without a left-hand side are separated by | from their preceding line.
III. A TYPING SYSTEM FOR SESSION TYPES

a) External and Internal Choice:

\[ \Omega \vdash P :: y:A_k \quad k \in L \] 
\[ \Omega \vdash y.k; P :: y : \{\ell : A_{\ell}\}_{\ell \in L} \] 
\[ \Omega, x.A_k \vdash Q_k :: y:C \quad \forall k \in L \] 
\[ \Omega, x: \{\ell : A_{\ell}\}_{\ell \in L} \vdash \text{case } x(\ell \Rightarrow Q_{\ell})_{\ell \in L} :: y:C \] 
\[ \Omega, x:A_k \vdash Q_k :: y:A_k \quad \forall k \in L \] 
\[ \Omega \vdash \text{case } y(\ell \Rightarrow Q_{\ell})_{\ell \in L} :: y.k; \{\ell : A_{\ell}\}_{\ell \in L} \] 
\[ \Omega, x:A_k \vdash P :: y:C \quad k \in L \] 
\[ \Omega, x: \{\ell : A_{\ell}\}_{\ell \in L} \vdash (x.k; P) :: y:C \]

b) Channel input/output:

\[ \Omega \vdash P :: y:B \] 
\[ \Omega, z:A \vdash \text{send } z; y; P :: y:A \otimes B \] 
\[ \Omega, z:A, x:B \vdash P :: y:C \] 
\[ \Omega, x:A \otimes B \vdash z \leftarrow \text{recv } x; P :: y:C \] 
\[ \Omega \vdash z \leftarrow \text{recv } y; P :: y:A \rightarrow B \] 
\[ \Omega, x:A \rightarrow B \vdash \text{send } z; x; P :: y:C \]

\[ \Omega \vdash P :: y:B \] 
\[ \Omega, z:A \vdash \text{send } z; y; P :: y:A \otimes B \] 
\[ \Omega, z:A, x:B \vdash P :: y:C \] 
\[ \Omega, x:A \otimes B \vdash z \leftarrow \text{recv } x; P :: y:C \] 
\[ \Omega \vdash z \leftarrow \text{recv } y; P :: y:A \rightarrow B \] 
\[ \Omega, x:A \rightarrow B \vdash \text{send } z; x; P :: y:C \]

\[ \Omega_1' \vdash X = P :: x'.A \in \Sigma \quad \Omega_1, x: A \vdash \gamma :: \Omega_1', x': A \quad \Omega_2, x: A \vdash \Sigma \vdash Q :: y:C \] 

\[ \Omega, \Omega_2 \vdash \Sigma \vdash (x \leftarrow X[\gamma] \leftarrow \Omega_1); Q_x :: y:C \] 

\[ \frac{z' : C \vdash F_Y = \text{Fwd}_C \cdot y' \vdash z': y': C \in \Sigma \quad Y = A \in \Sigma \quad z'.C, y'.C \vdash \gamma :: z.C, y.C \} {z: Y \vdash \Sigma \vdash F_Y[\gamma] :: y:C} \]

\[ \vdash (\text{close } y) :: y : \Sigma \] 
\[ \Omega \vdash \Sigma \vdash Q :: y : C \] 
\[ \Omega, x : \Sigma \vdash \text{wait } x; Q :: y : C \]

\[ \frac{Y = A \in \Sigma \quad \Delta \vdash P :: x:Y \} {\Delta \vdash \Sigma \vdash P :: x:Y} \]
\[ \frac{Y = A \in \Sigma \quad \Delta, x: A \vdash \Sigma \vdash P :: z:C \} {\Delta, x: Y \vdash \Sigma \vdash P :: z:C} \]

\[ \vdash (\text{sig} \cdot \ell) \quad \Sigma_1 \] 
\[ \frac{\vdash (\text{sig} \cdot \ell) \quad \Sigma_2 \} {\vdash \Sigma \vdash A \vdash Y = A, \Sigma' \vdash \Sigma_3} \] 
\[ \frac{\vdash \Sigma \vdash P :: x: A \quad \vdash \Sigma \vdash \Sigma' \vdash \Sigma_3 \} {\vdash \Sigma \vdash \Sigma_1 \vdash P :: x: A, \Sigma' \vdash \Sigma_3} \]

\[ \frac{z_{\beta}: A, y_{\alpha+1}: B \vdash \text{send } z_{\beta}, y_{\alpha} :: y_{\alpha}: A \otimes B \} {\Sigma_1} \]
\[ \frac{z_{\beta}: A, x_{\alpha} \rightarrow B \vdash \text{send } z_{\beta}, x_{\alpha} :: x_{\alpha+1}: A \} {\Sigma_3} \]

\[ \frac{k \in L \quad y_{\alpha+1}: A_k \vdash y_{\alpha}, k :: y_{\alpha}: \{\ell : A_{\ell}\}_{\ell \in L}} {\Sigma_2} \]

\[ \frac{k \in L \quad x_{\alpha+1}: A_k \vdash (x_{\alpha}, k; P) :: x_{\alpha+1}: A_k} {\Sigma_3} \]

\[ \vdash (\text{close } y_{\alpha}) :: y_{\alpha} : \Sigma_1 \]
h) Configuration Typing:

\[
\begin{array}{c}
\Delta_0 \vdash C :: \Delta \\
\Delta_0', \Delta, (x_\alpha : A) \vdash \delta :: \Omega', \Omega, (x:A) \\
\Delta_0, \Omega' \vdash P :: (x:A)
\end{array}
\]

\[
\frac{\Delta_0, \Delta_0' \vdash C, \text{proc}(x_\alpha, \delta(P)) :: (x_\alpha : A)}{\Delta_0, \Delta_0' \vdash C, \text{proc}}
\]

\[
\begin{array}{c}
\Delta_0 \vdash C :: \Delta \\
\Delta_0' \vdash M :: (x_\alpha : A)
\end{array}
\]

\[
\frac{\Delta_0, \Delta_0' \vdash C, \text{msg}(M) :: (x_\alpha : A)}{\Delta_0, \Delta_0' \vdash \text{msg}}
\]

\[
\frac{\Delta_0 \vdash C :: \Delta}{\Delta_0, \Delta_0' \vdash C, C_0 :: \Delta, x_\alpha : A}
\]

\[
\frac{\Delta_0 \vdash C :: \Delta}{\Delta_0, \Delta_0' \vdash C, C_0 :: \Delta, x_\alpha : A}
\]

\[
\frac{\Delta_0 \vdash C :: \Delta}{\Delta_0, \Delta_0' \vdash \text{comp}}
\]

Since \( \Sigma \) is fixed, we may drop it in a configuration typing judgment and a process typing judgment, respectively, for brevity.

**Definition 1.** For all \( Y = A \in \Sigma \), we extend \( \Sigma \) by adding the following definition to the signature \( \Sigma \):

\[
x' : A \vdash F_Y = \text{Fwd}_{A,y' \leftarrow x'} :: y' : A \quad Y = A \in \Sigma
\]

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

\[
\begin{align*}
\text{Fwd}_{\{x: A\} \in L, y' \leftarrow x} & := \text{case } x(\ell \Rightarrow y, \ell \in \text{Fwd}_{A,y' \leftarrow x}) & \ell \in L \\
\text{Fwd}_{\{x: A\} \in L, y' \leftarrow x} & := \text{case } y(\ell \Rightarrow x, \ell \in \text{Fwd}_{A,y' \leftarrow x}) & \ell \in L \\
\text{Fwd}_{A,B,y' \leftarrow x} & := w \leftarrow \text{recv } x; \text{send } w; \text{Fwd}_{B,y' \leftarrow x} \\
\text{Fwd}_{A,y' \leftarrow x} & := w \leftarrow \text{recv } y; \text{send } w; \text{Fwd}_{B,y' \leftarrow x} \\
\text{Fwd}_{1,y' \leftarrow x} & := \text{wait } x; \text{close } y \\
\text{Fwd}_{Y,y' \leftarrow x} & := F_Y[x \mapsto x', y \mapsto y'] & Y = A \in \Sigma
\end{align*}
\]

**Lemma 1.** Given the extended signature \( \Sigma \), for all type \( A \), there is a derivation for \( x : A \vdash \text{Fwd}_{A,y' \leftarrow x} :: y : A \).

**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 \( x : C \vdash F_Y := \text{Fwd}_{C,y' \leftarrow x} :: y : A \in \Sigma \). The proof of other cases is straightforward. \( \square \)

**Definition 2** (Well-typed configuration). Well-typed configuration(s) are defined in terms of the judgments \( (D_1, D_2) \in \text{Tree}(\Delta \vdash K) \) and \( D \in \text{Tree}(\Delta \vdash K) \), 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 \( D \in \text{Tree}(\Delta \vdash K) \) as
  \[ \Delta \vdash D :: K \]
- We define \( (D_1, D_2) \in \text{Tree}(\Delta \vdash K) \) as \( D_1 \in \text{Tree}(\Delta \vdash K) \) and \( D_2 \in \text{Tree}(\Delta \vdash K) \)
- We define \( B \in \text{Forest}(\Delta \vdash \Delta') \) as
  \[ \Delta \vdash B :: \Delta' \]
- We define the notation \( T \in B \) meaning \( T \) is a particular tree in a forest of trees \( B \): For \( B \in \text{Forest}(\Delta \vdash \Delta') \), we write \( T \in B \) iff \( B = B'T \), and \( B' \in \text{Forest}(\Delta_1 \vdash \Delta'_1) \) and \( T' \in \text{Tree}(\Delta_2 \vdash K) \) with \( \Delta = \Delta_1, \Delta_2 \) and \( \Delta' = \Delta'_1, K \). \( \diamond \)
is defined globally for an application and consists of concrete security levels \( \iota \). Our running example lattice

\[
\begin{align*}
guest \trianglelefteq \text{alice} \trianglelefteq \text{bank} & \quad \text{guest} \trianglelefteq \text{bob} \trianglelefteq \text{bank}
\end{align*}
\]

is an example of a concrete security lattice. Polymorphic process definitions make use of a security theory \( \Psi \), ranging over security variables \( \psi \) and concrete security levels \( \iota \) from the given concrete security lattice \( \Psi_0 \). At run-time, all security variables occurring in polymorphic processes will be replaced with concrete security levels.

**Definition 3** (Concrete security lattice and security theory). Let \( \Psi_0 \triangleq (\mathcal{L}, \mathcal{E}_0, \sqcup) \) be a concrete join semi-lattice with a partial order \( \mathcal{E}_0 \) over concrete security levels \( \iota, \eta \in \mathcal{L} \) such that \( \mathcal{E}_0 \subseteq \mathcal{E}_0 \) is of the form \( \iota \sqsubseteq \iota' \). We define a security theory \( \Psi \triangleq (\mathcal{V}, \mathcal{E}) \) that augments \( \Psi_0 \) with security variables \( \psi \) and relations \( \mathcal{E} \) over them such that

1. \( \psi \in \mathcal{V} \) is a set of security variables
2. \( c, d \in \mathcal{S} \) is a set of security terms of the security theory \( \Psi \), defined by the grammar
   \[
   c, d ::= c \sqcup d \mid \iota \mid \psi
   \]
3. \( E \in \mathcal{E} \) 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 \( \Psi \). We write

\[\Psi \vdash E\]

if \( E \) is consequence of the lattice theory based on the relations. The judgment is defined in Fig. 2.

**Fig. 2: Inductive definition of \( \Psi \vdash E \).**

IV. 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 \( \Psi_0 \) is defined globally for an application and consists of concrete security levels \( \iota \). Our running example lattice

\[
\begin{align*}
guest \trianglelefteq \text{alice} \trianglelefteq \text{bank} & \quad \text{guest} \trianglelefteq \text{bob} \trianglelefteq \text{bank}
\end{align*}
\]

is an example of a concrete security lattice. Polymorphic process definitions make use of a security theory \( \Psi \), ranging over security variables \( \psi \) and concrete security levels \( \iota \) from the given concrete security lattice \( \Psi_0 \). At run-time, all security variables occurring in polymorphic processes will be replaced with concrete security levels.

**Definition 4** (Order-preserving substitution). Let \( \gamma \in \mathcal{V} \rightarrow \mathcal{S} \) 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 \( \Delta \), and security lattices \( \Psi \), is defined by structural induction over the syntactic object, replacing simultaneously all variable occurrences \( \psi_i \) in the object with \( \gamma(\psi_i) \). The function \( \gamma \) and its lifting \( \hat{\gamma} \) must be 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 \( \Psi \vdash \gamma : \Psi' \)

\[
\Psi \vdash \gamma : \Psi' \triangleq \text{if } \Psi' \vdash E, \text{then } \Psi \vdash \hat{\gamma}(E)
\]

Composition \( \gamma \circ \gamma' \) of two substitutions \( \gamma \) and \( \gamma' \) is defined as usual. We observe that if both \( \gamma \) and \( \gamma' \) 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 \textsc{Spawn}). As a result, the security theory \( \Psi' \) of the spawnee must be satisfied by the security theory \( \Psi \) of the spawner, i.e., if \( \Psi' \vdash E \), then \( \Psi \vdash \hat{\gamma}(E) \). Moreover, if the spawner provides a substitution \( \delta \) for \( \Psi_0 \), i.e., \( \Psi_0 \vdash \delta : \Psi \), so does the spawnee, i.e., \( \Psi_0 \vdash \gamma \circ \delta : \Psi' \).
V. A secure typing system

a) External and Internal Choice:

\[
\Gamma, 
\begin{array}{l}
\Psi; \Xi \vdash P @ d_1 :: y: A[k] \quad k \in L \\
\Psi; \Xi \vdash (y'. k; P) @ d_1 :: y, \oplus \{ \ell : A\} \in L [c] \quad \oplus R \\
\Psi; \Xi, x : A[k] \vdash Q_1 @ d_2 :: y : C[c'] \quad \forall k \in L \\
\Psi; \Xi, x : \oplus \{ \ell : A\} \in L [c] \vdash (\text{case } x'(\ell \Rightarrow Q_1) \in L [d_1] :: y : C[c']) \\
\Psi; \Xi, x : \oplus \{ \ell : A\} \in L [c] \vdash x' \quad \forall k \in L \\
\Psi; \Xi, x : A[k] \vdash Q_1 @ d_2 :: y : C[c'] \\
\Psi; \Xi, x : \oplus \{ \ell : A\} \in L [c] \vdash (x'. k; P) @ d_1 :: y : C[c'] \\
\end{array}
\]

\[
\psi = \psi; z_1 : [\Sigma] C[\psi] \vdash \text{Fwd}_C, x' \mapsto \delta_0 : z_1 : [\Sigma] C[\psi] \in \Sigma \\
Y = A \in \Sigma \\
\Psi \vdash \gamma_{sec} : \Psi' \\
\Psi \vdash \gamma_{sec} : \psi = \psi; z : C[\psi], y : C[\psi] \vdash \gamma_{sec} : \psi = z_1 : [\Sigma] C[\psi], x_1 : C[\psi] \\
\Psi; z : C[\psi] \vdash \text{Fwd}_C[(\gamma_{sec}, \gamma_{var})] @c :: y : C[c] \\
\]

D-FWD

Rule SPAWN relies on two substitutions, \( \gamma_{sec} \) which provides a substitution for channel variables to match them up with the definitions in the signature, and \( \gamma_{var} \) which is an order-preserving substitution \( \Psi \vdash \gamma_{sec} : \Psi' \), 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 above invariants for the newly spawned process, by the premise \( \Psi \vdash \gamma_{sec} : \psi \) and allows the newly spawned process to start at least at the spawner’s running secrecy \( d_0 \), by the premise \( \Psi \vdash \gamma_{sec} : \psi \) and \( \gamma_{var} \).

b) Channel input/output:

\[
\begin{array}{l}
\Psi, \Xi \vdash P @ d_1 :: y : B[c] \\
\Psi; \Xi, z : A[c] \vdash (\text{send } y'; P) @ d_1 :: y : A \oplus B[c] \\
\Psi \vdash d_2 = c \cup d_1 \\
\Psi; \Xi, z : A[c], x : B[c] \vdash P @ d_2 :: y : C[c'] \\
\Psi; \Xi, x : A \oplus B[c] \vdash (z \leftarrow \text{recv } x'; P) @ d_1 :: y : C[c'] \\
\Psi; \Xi, z : A[c] \vdash P @ d_1 :: y : B[c] \\
\Psi; \Xi \vdash (z \leftarrow \text{recv } y'; P) @ d_1 :: y : A \rightarrow B[c] \\
\Psi \vdash d_1 \subseteq c \\
\Psi; \Xi, z : A[c], x : A \rightarrow B[c] \vdash (\text{send } z \leftarrow x'; P) @ d_1 :: y : C[c'] \\
\end{array}
\]

c) Definition and spawn:

\[
\begin{array}{l}
\Psi; \Xi \vdash \Sigma X = P @ \psi_0 :: x' : A[\psi] \in \Sigma \\
\Psi \vdash \gamma_{sec} : \Psi' \\
\Psi \vdash \gamma_{sec} : \psi = \psi; z : C[\psi], y : C[\psi] \vdash \gamma_{sec} : \psi = z_1 : [\Sigma] C[\psi], x_1 : C[\psi] \\
\Psi; z : C[\psi] \vdash \text{Fwd}_C[(\gamma_{sec}, \gamma_{var})] @c :: y : C[c] \\
\end{array}
\]

Here is the forward rule we use in the implementation:

\[
\begin{array}{l}
\Psi; \Xi \vdash \Sigma (\text{close } y') @ d_1 :: y : 1[c] \\
\Psi, x : 1[c] \vdash \Sigma (\text{wait } x' ; Q) @ d_1 :: y : \Sigma T[d] \\
\end{array}
\]

1R

1L

d) Termination Rules:

\[
\begin{array}{l}
\Psi; \Xi \vdash \Sigma \text{close } y' @ d_1 :: y : 1[c] \\
\Psi; \Xi, x : 1[c] \vdash \Sigma \text{wait } x' ; Q @ d_1 :: y : \Sigma T[d] \\
\end{array}
\]

1R

1L

e) Silent unfolding:

\[
\begin{array}{l}
Y = A \in \Sigma \\
\Psi; \Xi \vdash \Sigma P :: x : A[c] \\
\Psi; \Xi \vdash \Sigma P :: x : Y[c] \\
\Psi; \Xi, x : A[c] \vdash \Sigma P :: z : C[c'] \\
\Psi; \Xi, x : Y[c] \vdash \Sigma P :: z : C[c'] \\
\end{array}
\]

TVAR_R

TVAR_L
f) Signature checking:
\[
\frac{\vdash_{\Sigma;\psi} (\cdot) \text{ sig} \quad \vdash_{\Sigma;\psi} A \text{ wfdm}}{
\vdash_{\Sigma;\psi} Y = A, \Sigma' \text{ sig} \quad \vdash_{\Sigma;\psi} \Sigma' \text{ sig} \quad \vdash_{\Sigma;\psi} \Sigma \text{ sig}}
\]
\[
\forall i \in \{1 \ldots n\}, \Psi \vdash \psi_i \subseteq \psi \quad \Psi \vdash \psi_0 \subseteq \psi \quad \Psi; y_1; B_1[\psi_1], \ldots, y_n; B_n[\psi_n] \vdash \Sigma \quad P \at \psi_0 :: x:A[\psi] \quad \vdash_{\Sigma;\psi} \Sigma' \text{ sig}
\]
\[
\Psi; y_1; B_1[\psi_1], \ldots, y_n; B_n[\psi_n] \vdash X = P \at \psi_0 :: x:A[\psi], \Sigma' \text{ sig}
\]

h) Configuration Typing:
\[
\frac{\psi_0; z_{\beta}; A[c], \gamma_{\alpha+1}; B[c] \vdash \text{send } z_{\beta} \ x_{\alpha} :: \gamma_{\alpha+1} \otimes B[c]}{\text{R}}
\]
\[
\frac{k \in L \quad \psi_0; \gamma_{\alpha+1}; A[k[c] \vdash \gamma_{\alpha}; k :: \gamma_{\alpha}; \{\ell; A \ell\}_{\ell \in L[c]} \text{R}}{\text{L}}
\]
\[
\frac{\psi_0; \gamma_{\alpha+1}; A[k[c] \vdash \gamma_{\alpha}; k :: \gamma_{\alpha}; \{\ell; A \ell\}_{\ell \in L[c]} \text{R}}{\text{L}}
\]
\[
\frac{\psi_0; \gamma_{\alpha+1}; A[k[c] \vdash \gamma_{\alpha}; k :: \gamma_{\alpha}; \{\ell; A \ell\}_{\ell \in L[c]} \text{R}}{\text{L}}
\]

Since \(\Sigma_0\) and \(\Sigma\) are fixed, we may drop \(\Sigma_0\) and \(\Sigma\) in a configuration typing judgment and a process typing judgment, respectively, for brevity.

**Definition 5.** For all \(Y = A \in \Sigma\), we extend \(\Sigma\) by adding the following IFC-typed forwarder definition to the signature \(\Sigma\):
\[
\psi = \psi; x : A[\psi] \vdash F_A = \text{Fwd}_{A.y \leftarrow x} \psi :: y : A[\psi] \quad Y = A \in \Sigma
\]

Here \(F_Y\) is a specific process variable assigned to the forwarder for process type \(Y\), and \(\text{Fwd}_{A.y \leftarrow x}\) is defined similar to **Def. 7** as
\[
\begin{align*}
\text{Fwd}_{\ell \in L, y \leftarrow x} &::= \text{case } x(\ell \Rightarrow y, \ell; \text{Fwd}_{A_y \leftarrow x})_{\ell \in L} \hfill \\
\text{Fwd}_{L, \ell \in L, y \leftarrow x} &::= \text{case } y(\ell \Rightarrow x, \ell; \text{Fwd}_{A_y \leftarrow x})_{\ell \in L} \\
\text{Fwd}_{A \otimes B, y \leftarrow x} &::= w \leftarrow \text{recv } x; \text{send } w; y; \text{Fwd}_{B, y \leftarrow x} \\
\text{Fwd}_{A \rightarrow B, y \leftarrow x} &::= w \leftarrow \text{recv } y; \text{send } w; x; \text{Fwd}_{B, y \leftarrow x} \\
\text{Fwd}_{1, y \leftarrow x} &::= \text{wait } x; \text{close } y \\
\text{Fwd}_{Y, y \leftarrow x} &::= F_Y[(x' \Rightarrow x, y' \Rightarrow y), (\psi \Rightarrow c)] \quad Y = A \in \Sigma
\end{align*}
\]

**Lemma 2.** Given the extended signature \(\Sigma\), for all type \(A\), there are derivations for
(i) \(\psi; x : A[\psi] \vdash_{\Sigma} \text{Fwd}_{A.y \leftarrow x} \psi :: y : A[\psi]\), and
(ii) \(\psi; x : A[\psi] \vdash_{\Sigma} \text{Fwd}_{A.y \leftarrow x} \psi' :: y : A[\psi]\), when \(\Sigma \vdash \psi' \subseteq \psi\) 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 \(Y \in \Sigma\), the proof is straightforward by the \(D\)-FWD rule since \(\psi = \psi; x : C[\psi] \vdash F_Y := \text{Fwd}_{C, y \leftarrow x} \psi :: y : A[\psi] \in \Sigma\), and the substitutions \(\gamma_{\text{sec}}\) and \(\gamma_{\text{var}}\) 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 \(\psi\) and we
can apply the previous item. Note that by the tree invariant every judgment in our typing derivation satisfies the condition \( \Psi \vdash \psi' \subseteq \psi \).

**Definition 6.** Define security-erasure \( |\Sigma| \) of the signature \( \Sigma \) as

\[
|Y = A, \Sigma'| = Y = A, |\Sigma'|
|\Psi; \Xi \vdash X = P@\psi_0 :: x:A[\psi], \Sigma'| = |\Xi| \vdash X = P :: x:A, |\Sigma'|
\]

**Definition 7.** Define security-erasures \( |\Xi|, |x:A[c]|, |\Gamma|, \) and \( |K^s| \) of the security linear variable context \( \Xi \), security channel variable, security linear channel context \( \Gamma \), and security channel singleton \( K^s \), respectively, as

\[
|\Xi, x:A[c]| \triangleq |\Xi|, x:A
|x:A[c]| \triangleq x:A
|\Gamma, x_\alpha:A[c]| \triangleq |\Gamma|, x_\alpha:A
|-| \triangleq ||
|x_\alpha:A[c]| \triangleq x_\alpha:A
|-1[T]| \triangleq |-1|
\]

**Definition 8.** Define a security-erasure \( |C| \) of the configuration \( C \) as

\[
|\cdot| = .
|C, \text{proc}(x[d], P@d_1)| = |C|, \text{proc}(x, P)
|C, \text{msg}(M)| = |C|, \text{msg}(M)
\]

**Theorem 1.** Every IFC well-typed configuration \( \Psi_0; \Gamma \vdash C :: K^s \) is session-typed \( |C| \in \text{Tree}(|\Gamma| \vdash |K^s|) \).

**Proof.** By induction on the derivation of \( \Psi_0; \Gamma \vdash C :: K^s \).
the configuration typing. For example, stepping the configuration using rule 5).

Definition 10 (Valid configuration). For a well-typed configuration \( \Psi; \Delta \vdash C : \Delta' \), a valid permutation \( \mathcal{P}(C) \) can be derived by simultaneously changing the position of a process or message in \( C \), yielding the permutation \( C' \), i.e., \( \mathcal{P}(C) = C' \), as long as the following conditions are met:

1) For a process \( \text{proc}(z_{\beta}, P) \) in \( C' \) such that \( C' = C_P \text{proc}(z_{\beta}, P)C_2 \) and for all \( y_{\beta} \notin \text{dom}(\Delta) \) that \( P \) is using, there must exist either a process \( \text{proc}(y_{\beta}, P) \) or a message \( \text{msg}(y_{\beta}, \Sigma) \) in \( C_1 \).

2) For a positive message \( \text{msg}(M(z_{\alpha})) \) in \( C' \) such that \( C' = C_2 \text{msg}(M(z_{\alpha}))C_2 \), if \( z_{\alpha} \notin \text{dom}(\Delta) \), there must exist either a process \( \text{proc}(y_{\alpha}, P) \) or a message \( \text{msg}(y_{\alpha}, \Sigma) \) in \( C_1 \). Moreover, for all \( y_{\alpha} \notin \text{dom}(\Delta) \) that \( P \) is using, there must exist either a process \( \text{proc}(y_{\alpha}, P) \) or a message \( \text{msg}(y_{\alpha}, \Sigma) \) in \( C_1 \).
3) For a negative message \( \text{msg}(P(v_β)) \) in \( C' \) such that \( C' = C_1 \text{msg}(P(v_β))C_2 \), if \( v_β \not\in \text{dom}(\Delta) \), there must exist either a process \( \text{proc}(v_β, _) \) or a message \( \text{msg}(w) \) in \( C_1 \). Moreover, for all \( y_α \not\in \text{dom}(\Delta) \) that \( P \) is using, there must exist either a process \( \text{proc}(y_α, _) \) or a message \( \text{msg}(y_α, _) \) in \( C_1 \).

2) Lemmas:

**Lemma 3** (Term variable substitution). The following substitutions are type-preserving and thus admissible:

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

**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. 4.

**Lemma 4** (Making two forests out of one). If \( \Delta \models C C' \vdash \Delta' \), then for some \( \Delta_1 \) we have \( \Lambda_1 \vdash C : \Lambda_1', \Delta_1 \) and \( \Lambda_2, \Delta_1 \vdash \Lambda_2', \Delta' \), 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.

**Theorem 2** (Progress). For any configuration \( C \), if \( C \) is a valid permutation of a configuration \( C'' \) such that \( \Delta \models C'' \vdash \Delta' \), then either \( C \rightarrow C' \) or \( C \) is poised.

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

**Theorem 3** (Preservation). For any configuration \( C \) that is a valid permutation of a configuration \( C'' \) such that \( \Delta \models C'' \vdash \Delta' \), if \( C \rightarrow C' \), then \( C' \) is a valid permutation of a configuration \( C'' \) such that \( \Delta \models C'' \vdash \Delta' \).

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

B. IFC-typed processes

This section proves type safety of CONSESSION.

**Lemma 5** (Term variable substitution). The following substitutions are type-preserving and thus admissible:

1) If \( \Psi; \Xi \vdash \Sigma \text{P@c : x : A}[c'] \) with the tree invariant satisfied, then, for any fresh \( y : A \), we have \( \Psi; \Xi \vdash \Sigma [y/x] \text{P@c : y : A}[c'] \) with the tree invariant still satisfied.
2) If \( \Psi; \Xi, y : B[c''] \vdash \Sigma \text{P@c : x : A}[c'] \) with the tree invariant satisfied, then, for any fresh \( z : B \), we have \( \Psi; \Xi, z : B[d'] \vdash \Sigma [z/y] \text{P@c : x : A}[c'] \) with the tree invariant still satisfied.

**Lemma 6** (Making two forests out of one). If \( \Psi_0; \Gamma \vdash \Sigma C C' \vdash \Gamma' \), then for some \( \Gamma_1 \) we have \( \Psi_0; \Gamma_1' \vdash \Sigma C : \Gamma_1' \), and \( \Psi_0; \Gamma_2' \vdash \Sigma C : \Gamma_2' \), 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 7** (Security variable substitution). If \( \Psi; \Xi \vdash \Sigma \text{P@c : x : B}[d] \) 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 \( \Psi; \Xi \vdash \Sigma \text{P@c : x : B}[d] \). We consider cases for the last step in the derivation.

**Case 1.**

\[
\Psi; \Xi \vdash \Sigma Q_k@c : y : A_k[c], \quad \forall k \in L
\]

\[
\Psi; \Xi \vdash \Sigma \text{(case } y'\langle \ell \Rightarrow Q_\ell \rangle_{\ell \in L})@d_1 : y : \& \{ \ell : A_\ell \}_{\ell \in L}[c] \quad \& R
\]

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
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_{\Sigma} \text{case } y^{\hat{\gamma}(c)}(\ell \Rightarrow \hat{\gamma}(Q_{\ell}))_{\ell \in L} @ \hat{\gamma}(d_{1}) :: y : \&\{\ell : A_{\ell}\}_{\ell \in L} [\hat{\gamma}(c)].\]

By the first part of the tree invariant for the original sequent, we get \(\Psi \vdash d_{1} \subseteq c\), and thus \(\hat{\gamma}(\Psi) \vdash \hat{\gamma}(d_{1}) \subseteq \hat{\gamma}(c)\). 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 \(\gamma\) is defined we get
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi) \vdash_{\Sigma} \text{case } y^{\hat{\gamma}(c)}(\ell \Rightarrow Q_{\ell})_{\ell \in L} @ \hat{\gamma}(d_{1}) :: y : \&\{\ell : A_{\ell}\}_{\ell \in L} [\hat{\gamma}(c)].\]

**Case 2.**
\[\Psi; \Xi, x : A_{k}[c] \vdash_{\Sigma} P @ A_{k}[c] :: y : T[d], k \in L \quad \&\&L\]

By the induction hypothesis on the first premise and definition of the lifting of \(\gamma\) 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 \((\Psi \vdash d_{1} \subseteq c)\) we get \(\hat{\gamma}(\Psi) \vdash \hat{\gamma}(d_{1}) \subseteq \hat{\gamma}(c)\). By the \&\&L rule, we get
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), x : \&\{A_{\ell}\}_{\ell \in L}[\hat{\gamma}(c)] \vdash_{\Sigma} \hat{\gamma}(x^{\hat{\gamma}(c).k}) \cdot \hat{\gamma}(P) @ \hat{\gamma}(d_{1}) :: y : T[\hat{\gamma}(d)],\]

which satisfies the tree invariant since the premise satisfies it. Again by a simple rewrite according to how the lifting of \(\gamma\) is defined we get
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi, x : \&\{A_{\ell}\}_{\ell \in L}[c]) \vdash_{\Sigma} \hat{\gamma}(x^{\hat{\gamma}(c).k}) \cdot \hat{\gamma}(P) @ \hat{\gamma}(d_{1}) :: y : T[\hat{\gamma}(d)].\]

**Case 3.**
\[\Psi; \Xi \vdash_{\Sigma} P @ A[c] :: y : B[d], \quad \otimes R\]

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 \(\hat{\gamma}(\Psi) \vdash \hat{\gamma}(d) \subseteq \hat{\gamma}(c)\). By the \&\&R rule, we get
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), z : A[d] \vdash_{\Sigma} \text{send } z^{\hat{\gamma}(d)} y ; P @ A[c] :: y : (A[d] \otimes B)[d] [\hat{\gamma}(c)],\]

the resulting sequent satisfies the tree invariant. Again by a simple rewrite according to how the lifting of \(\gamma\) is defined we get
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi, z : A[d]) \vdash_{\Sigma} \hat{\gamma}(\text{send } z^{d} y ; P) @ \hat{\gamma}(c) :: y : (A \otimes B)[\hat{\gamma}(d)].\]

**Case 4.**
\[\Psi; \Xi, z : A[d], x : B[d] \vdash_{\Sigma} P @ A[c] :: y : T[d_{1}], \quad \otimes L\]

By the induction hypothesis on the second premise and definition of the lifting of \(\gamma\) we have
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), z : A[\hat{\gamma}(d)] \vdash_{\Sigma} \hat{\gamma}(P) @ \hat{\gamma}(c') :: y : T[\hat{\gamma}(d_{1})],\]

which preserves the tree invariant. Moreover, by applying the substitution on the first premise, we get \(\hat{\gamma}(\Psi) \vdash \hat{\gamma}(c') = \hat{\gamma}(c) \cup \hat{\gamma}(d)\). By the \&\&L rule, we get
\[\hat{\gamma}(\Psi); \hat{\gamma}(\Xi), x : (A \otimes B)[\hat{\gamma}(d)] \vdash_{\Sigma} z^{\hat{\gamma}(d)} \leftarrow \text{recv } x ; \hat{\gamma}(P) @ \hat{\gamma}(c) :: y : T[\hat{\gamma}(d_{1})],\]

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 \(\gamma\).
Case 5. Here is the interesting case!

\[ \Psi' ; \Xi_1 \vdash X = P[\psi_0] :: x':A[\psi] \in \Sigma \quad \Psi \vdash \delta_{\text{sec}} : \Psi' \quad \Psi \vdash \hat{\delta}_{\text{sec}}(\psi) \subseteq d \]

\[ \Psi \vdash d_0 \subseteq \delta_{\text{sec}}(\psi_0) \quad \Xi_1, x:A[\delta_{\text{sec}}(\psi_0)] = (\delta_{\text{var}}, \delta_{\text{var}}) :: \Xi_1' \quad x':A[\psi] \quad \Psi ; \Xi_2, x\vdash A[\delta_{\text{sec}}(\psi)] = (\Sigma; Q@d_0 :: z:C[d]) \]

By the induction hypothesis we have

\[ \ast \hat{\gamma}(\Psi); \hat{\gamma}(\Xi_2), x:A[\hat{\gamma}(\delta_{\text{sec}}(\psi))] \vdash \Sigma \quad \hat{\gamma}(Q)@\hat{\gamma}(d_0) :: y:\hat{\gamma}(T)[\hat{\gamma}(d)] \]

which preserves the tree invariants. By applying \( \gamma \) on the 3rd and 4th premises, we get

\[ \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi)) \subseteq \hat{\gamma}(d), \hat{\gamma}(d_0) \subseteq \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi_0)) \]

By applying the substitution \( \gamma \) on the 2nd premise, we get \( \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\hat{\delta}_{\text{sec}}) :: \Psi' \). And from the 5th premise, we get

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

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

\[ \hat{\gamma}(\Psi); \hat{\gamma}(\Xi_1), \hat{\gamma}(\Xi_2) \vdash (x[\hat{\gamma}(\delta_{\text{sec}}(\psi))] \leftarrow X[\gamma'] \leftarrow \hat{\gamma}(\Xi_1)@\hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi_0))); \hat{\gamma}(Q)@\hat{\gamma}(d_0) :: y : \hat{\gamma}(T)[\hat{\gamma}(d)] \]

To show that this sequent satisfies the tree invariant, we use (a) the fact that the sequent \( \ast \) satisfies the tree invariant and (b) the condition on process definitions in the signature. The condition we imposed on process definitions ensures that \( \Psi' \vdash \psi_0 \subseteq \psi \) and \( \forall y:A[\psi_i] \in \Xi_1, \Psi' \vdash \psi_i \subseteq \psi \).

By \( \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\delta_{\text{sec}}) :: \Psi' \), we can rewrite the above judgment as

\[ \forall y:A[\psi_i] \in \Xi_1, \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi_i)) \subseteq \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi)) \]

By the tree invariant of \( \ast \) (i.e., \( \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi)) \subseteq \hat{\gamma}(d) \))

\[ \forall y:A[\psi_i] \in \Xi_1, \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi_i)) \subseteq \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi)) \subseteq \hat{\gamma}(d) \]

By \( (\delta_{\text{var}}, \hat{\gamma}(\hat{\delta}_{\text{sec}})(\Xi_1')) = \hat{\gamma}(\Xi_1) \)

\[ \forall y:\hat{\gamma}(A)[\psi_i] \in \hat{\gamma}(\Xi_1), \hat{\gamma}(\Psi) \vdash \hat{\gamma}(\psi_i) \subseteq \hat{\gamma}(\hat{\delta}_{\text{sec}}(\psi)) \subseteq \hat{\gamma}(d) \]

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

\[ \square \]

1) Progress: Progress for IFC-typed processes in CONSESSION follows from the progress of SESSION, as IFC typing can be viewed as a refinement typing using the same dynamics as the session typed processes.
Theorem 4 (Preservation). For any configuration C that is a valid permutation of a configuration C' such that Ψ₀; Γ ⊬ C" ::: Γ', if |C| ⊢ C', then there exists a security annotated program C' such that |C'| = C' and is a valid permutation of configuration C" such that Ψ₀; Γ ⊬ C" ::: Γ'. Moreover, the stepping preserves the tree invariant.

Proof. The proof is by considering different cases of |C| ⊢ C'. For each step we provide a security annotated configuration 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 the proof. We provide the detailed proof by inversion for a couple of cases. The rest of the cases are similar.

Case 1. (⊗)

Subcase 1. (send)

\[ C₁ \text{proc}(y₀[c], (send xₛ y₀; P)@d₁) C₂ → C₁ \text{proc}(y₀+1[c], ([y₀+1/y₀] P)@d₁) msg(send xₛ y₀) C₂ \]

By assumption of the theorem: Ψ₀; Γ ⊬ C₁ proc(y₀[c], (send xₛ y₀; P)@d₁) C₂ :: Γ'.

By Lemma 6: Ψ₀; Γᵢ ⊬ C₁ :: Γ₁₁, Γ₁₂, Γ₁₁' and Ψ₀; Γᵢ₁' ⊬ yₜ; A[c] :: yₜ; (A ⊗ B)[c] and Ψ₀; Γᵢ₂ ⊬ C₂ :: Γ₂₁, Γ₂₂.

Where Γ₁₁ = Γ₁₁, Γ₁₂ and Γ₁₁' = Γ₁₁', Γ₁₂'. If xₛ; A[c] ∈ Γ we have Γ₁ = Γ₁₁ and Γ₁₂ = Γ₁₂. If not, xₛ; A[c] ∈ Γ₁₁ then Γ₁ = Γ₁₁ and Γ₁₂ = Γ₁₂.

By inversion on rule proc Ψ₀; Γᵢ₂', Γᵢ₁', xₛ; A[c] :: (send xₛ y₀; P)@d₁ :: : yₜ; (A ⊗ B)[c]. Moreover, (*) ∀uₜ; T[d₂] ∈ Γ₁₁', Γ₁₁, xₛ; A[c]; Ψ₀ ⊬ d₂ ⊆ c.

By substitution of y₀+1 for yₜ:

\[ Ψ₀; Γᵢ₁', Γ₁₁' :: (yₜ+1/yₜ) P@d₁ :: yₜ+1; B[c]. \]

By rule and (*): ↑ Ψ₀; Γᵢ₂'' ⊬ proc(y₀+1[c], [yₜ+1/yₜ] P@d₁) :: yₜ+1; B[c].

By ⊗ and fwd rule Ψ₀; Γᵢ₁' ; A[c], y₀+1; B[c] :: (send xₛ y₀; y₀′)@c :: yₜ; (A ⊗ B)[c].

By msg, * and ↑:

\[ Ψ₀; Γᵢ₂'' ⊬ proc(y₀+1[c], [yₜ+1/yₜ] P@d₁) msg(send xₛ y₀; y₀′) ⊥ y₀+1 :: yₜ; (A ⊗ B)[c]. \]

By configuration typing rules: Ψ₀; Γ ⊬ C₁ proc(y₀+1[c], [yₜ+1/yₜ] P@d₁) msg(send xₛ y₀; y₀′) ⊥ y₀+1 C₂ :: Γ'.

\[ \square \]

Remark: To keep the proofs concise we may use Ψ instead of the term secrecy lattice Ψ₀, whenever we are clearly working with configurations defined in the run-time. Moreover, from now on, we write Ψ₀; Δ ⊬ C :: Δ' also when C is a valid permutation of a configuration C' such that Ψ₀; Δ ⊬ C' :: Δ'.
VII. Recursive Session Logical Relation

This section introduces the recursive session logical relation and supporting definitions.

A. Open configuration transition

**Definition 11.** The configuration $\Delta \models C :: \Delta'$ is ready to send along $\Lambda \subseteq \Delta, \Delta'$ iff for all $y_\alpha : C \in \Lambda$, there is a message $\text{msg}(M)$ in $C$ sending along $y_\alpha$, i.e. $M$ is of the form $y_\alpha.k$, $\text{send} x_\beta y_\alpha$, or $\text{close} y_\alpha$. Similarly, the configuration $\Delta \models C :: \Delta'$ is ready to receive along $\Lambda \subseteq \Delta, \Delta'$ iff for all $y_\alpha : C \in \Lambda$, there is a process $\text{proc}(z_\delta, P)$ in $C$ waiting to receive along $y_\alpha$, i.e. $P$ is of the form $\text{case} y_\alpha(\ell \Rightarrow Q_\ell) \ell \in I, w \leftarrow \text{recv} y_\alpha; Q$, or $\text{wait} y_\alpha; Q$.

**Definition 12.** The set $\text{Out}(\Delta \models K)$, is defined as all channels with the sending semantics in $\Delta, K$, i.e., $y_\alpha \in \text{Out}(\Delta \models K)$ 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 $\text{In}(\Delta \models K)$, is defined as all channels with the receiving semantics in $\Delta, K$, i.e., $y_\alpha \in \text{Out}(\Delta \models K)$ 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 13.** $\text{dom}(\Delta)$ is a set defined inductively as:

\[
\begin{align*}
\text{dom}(\Delta, y_\alpha : A) &= \text{dom}(\Delta) \cup \{y\} \\
\text{dom}(\cdot) &= \emptyset
\end{align*}
\]

**Definition 14** (Open configuration transitions). We provide some notations used in the logical relation and proofs.

- The notation $\rightarrow^*$ refers to taking none or many steps with $\rightarrow$. The notation $\rightarrow^j$ refers to taking $j$ steps with $\rightarrow$.
- We write $D \rightarrow^* D'$ stating that $D \rightarrow^* D'$ and $D'$ is ready to send along $\Gamma$.
- We write $D \rightarrow^* \gamma, \Theta$ stating that $D \rightarrow^* D'$ and $D'$ is ready to send along $\Gamma$ and ready to receive along $\Theta$.

B. Recursive Session Logical Relation

**Fig. 6** defines the logical relation for intuitionistic linear logic session types with general recursive types.
Fig. 6: Recursive session logical relation
VIII. NONINTERFERENCE

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

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

The fundamental theorem [Thm. 5] is stated in terms of the logical equivalence \( (\Delta_1 \equiv \Delta_2 :: x_\alpha : A_1[c_1]) \equiv^{\Psi_0}_{\Psi_0} (\Delta_2 \equiv \Delta_2 :: y_\beta : A_2[c_2]) \), expressing that two open configurations \( \Delta_1 \) and \( \Delta_2 \) 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 15 (Typing context projections).** Downward projection on security linear contexts and \( K^s \) is defined as follows:

\[
\begin{align*}
\Gamma, x_\alpha : T[c] &\downarrow \xi & \implies & \Gamma \downarrow \xi, x_\alpha : T[c] & \text{if } c \sqsubseteq \xi \\
\Gamma, x_\alpha : T[c] &\downarrow \xi & \implies & \Gamma \downarrow \xi & \text{if } c \not\sqsubseteq \xi \\
\cdot &\downarrow \xi & \implies & \cdot & \\
x_\alpha, T[c] &\downarrow \xi & \implies & x_\alpha : T[c] & \text{if } c \sqsubseteq \xi \\
x_\alpha, T[c] &\downarrow \xi & \implies & 1[T] & \text{if } c \not\sqsubseteq \xi
\end{align*}
\]

**Definition 16 (High provider and High client).**

\[ B \in \text{H-Provider}^\xi(\Gamma, x_\alpha : A[c]) \iff c \not\sqsubseteq \xi \text{ and } B = B' \Gamma \text{ and } B' \in \text{H-Provider}^\xi(\Gamma) \text{ and } T \in \text{Tree}(\cdot \equiv x_\alpha : A), \text{ or } c \subseteq \xi \text{ and } B \in \text{H-Provider}^\xi(\Gamma) \]

\[ T \in \text{H-Client}^\xi(x_\alpha : A[c]) \iff c \not\sqsubseteq \xi \text{ and } T \in \text{Tree}(x_\alpha : A \equiv \_ : 1), \text{ or } c \subseteq \xi \text{ and } B = \cdot \]

**Definition 17 (Equivalence of trees by the logical relation upto the observer level).** We define the relation

\[(\Gamma_1 \equiv \Delta_1 :: x_\alpha : A_1[c_1]) \equiv^{\Psi_0}_{\Psi_0} (\Gamma_2 \equiv \Delta_2 :: y_\beta : A_2[c_2]) \text{ as}
\]

\[
\begin{align*}
\Delta_1 &\equiv \text{Tree}(\Gamma_1 \equiv x_\alpha : A_1) \text{ and } \Delta_2 \equiv \text{Tree}(\Gamma_2 \equiv y_\beta : A_2) \text{ and }
\Gamma_1 \downarrow \xi = \Gamma_2 \downarrow \xi = \Gamma \text{ and } x_\alpha : A_1[c_1] \equiv x_\alpha : A_2[c_2] \equiv K^s \text{ and }
\forall B_1 \in \text{H-Provider}^\xi(\Gamma_1), \forall B_2 \in \text{H-Provider}^\xi(\Gamma_2). \forall T_1 \in \text{H-Client}^\xi(x_\alpha : A_1[c_1]). \forall T_2 \in \text{H-Client}^\xi(y_\beta : A_2[c_2]).
\forall m. (B_1 \Delta_1 T_1, B_2 \Delta_2 T_2) \in \varepsilon[\Gamma] \equiv [K^s]^m, \text{ and }
\forall m. (B_2 \Delta_2 T_2, B_1 \Delta_1 T_1) \in \varepsilon[\Gamma] \equiv [K^s]^m.
\end{align*}
\]

B. Quasi-running secrecy and relevant nodes

The proof of the fundamental theorem [Thm. 5] relies on the notion of a relevant node, introduced in §V of the main text, maintaining the invariant that the relevant nodes of the two program runs execute the same code. The latter is guaranteed by [Lem. 8]. Next we define the notion of a relevant node, expressed locally for an asynchronous semantics (for ease of illustration §V of the main text provides a global and synchronous description). The notion of a relevant node relies on the notion of quasi-running secrecy, also defined below.

**Definition 18 (Quasi-running secrecy).** In an open configuration \( \Psi_0 : \Gamma \equiv \_ : \Gamma' \), 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 \( \text{recv} \) or \( \text{case} \), then its quasi-running secrecy is equal to its running secrecy.
- If the process term is of the form \( \text{case } y^\xi_\alpha(\ell \rightarrow P_\ell)_{\ell \in \ell} \odot d_1 \) or \( x \leftarrow \text{recv } y^\xi_\alpha; P \odot d_1 \), then its quasi-running secrecy is \( d_1 \sqcup c \).
- If the node is a message of a negative type along channel \( y^\xi_\alpha \), its quasi-running secrecy is \( c \).
- If the node is a message of a positive type along channel \( y_\alpha \), 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 19 (Relevant node). Consider a configuration $\Gamma \vdash D : K^*$ and observer level $\xi$. A channel is relevant in $D$ if (1) it has a maximal secrecy level less than or equal to $\xi$, and (2) it is either an observable channel or it shares a process or message with quasi-running secrecy less than or equal to $\xi$ 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 $\xi$, and (2) it has at least one relevant channel. We denote the relevant processes and messages (i.e., nodes) in $D$ by $D \downarrow_\xi$. We write $D_1 \downarrow_\xi =_\xi D_2 \downarrow_\xi$ if the relevant nodes in $D_1$ are identical to those in $D_2$ 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 Section 4. Def. 19 provides us with a local guide to identify whether a process is relevant or not. We note that if $K^*$ is observable, then by the tree invariant, every channel in $D$ 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. 3] crucially relies on this lemma.

Lemma 8 (Keeping relevant nodes in sync). Consider $\Psi_0; \Gamma \vdash D_1 : K^*$ for $i \in \{1, 2\}$ with the relevant nodes in $D_1$ and $D_2$ are identical, i.e., $D_1 \downarrow_\xi = D_2 \downarrow_\xi$, with $\Gamma \downarrow \xi = \Gamma$ and $K^* \downarrow \xi = K^*$. If $|D_1| \rightarrow |D'_1|$, then there exists a $D'_2$ such that $|D_2| \rightarrow |D'_2|$, i.e., $D_2 \downarrow_\xi = D'_2 \downarrow_\xi$. Proof. The proof is by cases on the possible steps of $|D_1| \rightarrow |D'_1|$. In each case we prove that either the step does not change relevancy of any process in $D_1$, or we can step $D_2$ such that the same change of relevancy occurs in $D_2$. Note that in all cases, we get $(D'_1 ; D'_2) \in \textbf{Tree}(|\Gamma| \vdash |K^*|)$ 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. $D_1 = D'_1 \downarrow \text{proc}([y_0 c], c y_0 c, k; P@d_1)D'_1'$ and $|D_1| \downarrow \text{proc}(y_0 c, c y_0 c, k; P@d_1)D'_1 \rightarrow |D'_1| \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_1'\downarrow_\xi$ where $D'_1 = D'_1 \downarrow \text{proc}(y_0 c, c y_0 c, k; P@d_1)D'_1' \rightarrow |D'_1| \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_1'\downarrow_\xi$. We consider subcases based on relevancy of process offering along $y_0 c$:

Subcase 1. proc([y_0 c], c y_0 c, k; P@d_1) is not relevant. By inversion on the typing rules $d_1 \subseteq c$. By definition either $d_1 \not\subseteq \xi$ or none of the channels connected to $P$ including its offering channel $y_0 c$ are relevant. In both cases neither $\text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)$, nor $\text{msg}(y_0 c, k)$ are relevant in the post step. Note that from $d_1 \subseteq c$ and $d_1 \not\subseteq \xi$, we get $c \not\subseteq \xi$. Channel $y_0 c$ is not relevant in the pre-step, and both $y_0 c$ and $y_{a+1}$ are not relevant in pre-step and post-step configurations. Every not relevant resource of proc([y_0 c], c y_0 c, k; P@d_1) will remain irrelevant in the post-step too. In this subcase, it is enough to show $D'_1 \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_1'\downarrow_\xi = \xi D'_1' \downarrow_\xi = \xi D_1 \downarrow_\xi = \xi D_2 \downarrow_\xi$.

To prove this we need two observations:

- Neither $\text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)$ nor $\text{msg}(y_0 c, k)$ are relevant and they will be dismissed by the projection. (As explained above.)
- Replacing $\text{proc}(y_0 c, c y_0 c, k; P@d_1)$ with these two nodes, does not affect relevancy of the rest of processes in $D'_1D'_2'$. Relevancy of processes in $D'_1D'_2'$ remains intact since $y_0 c$ and $y_{a+1}$ are irrelevant. The relevancy of processes in $D'_1'$ 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 $y_{a+1}$ 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 $d_1 \not\subseteq \xi$, and in the pre-step the process has a relevant channel $x : \vee [d]$ as its resource where $d \subseteq \xi$. But by the assumption of the subcase, either $d_1 \not\subseteq \xi$ or $x : \vee [d]$ is not a relevant channel in the pre-step.

This completes the proof.

Subcase 2. proc([y_0 c], c y_0 c, k; P@d_1) is relevant. By assumption $(D_1 \downarrow_\xi =_\xi D_2 \downarrow_\xi)$ we have : $D_2 = D'_2 \downarrow \text{proc}(y_0 c, c y_0 c, k; P@d_1)D'_2'$ and $D_2 \rightarrow D'_2 \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_2'$. and $D_3 = D'_3 \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_3'$. We need to show: $D'_3 \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_3' \downarrow_\xi = _\xi D'_2 \downarrow \text{proc}(y_{a+1} c, [c y_{a+1}/y_0 c]P@d_1)\text{msg}(y_0 c, k)D'_2' \downarrow_\xi$. 

20
• If \( c \not\subseteq \xi \), then \( \text{msg}(y_{\alpha}^c, k) \) is not relevant in both runs, and will be dismissed by the projections. Moreover, in this case \( y_{\alpha}^c \) is not relevant in the pre-step and post-step configurations. Thus the relevancy of processes in \( D'_{1} \) and \( D'_{2} \) will remain intact.

• If \( c \subseteq \xi \), then \( y_{\alpha}^c \) 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 \( y_{\alpha}^c+1 \) 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 invaraint it is always bounded by the max secrecy of the offering channel, \( c \subseteq \xi \). Thus, in the post-step of both runs, \( y_{\alpha}^c+1 \) is relevant. Relevancy of message \( \text{msg}(y_{\alpha}^c, k) \) in the post-steps of the first and second run is determined by the quasi-running secretcies \( (d \text{ and } d') \) of their parents (\( X \) and \( X' \)) in \( D'_{1} \) and \( D'_{2} \). If \( d \subseteq \xi \), 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 \( \text{msg}(y_{\alpha}^c, k) \) are relevant in both runs and \( y_{\alpha}^c \) is relevant in the post-step too. The same holds when \( d' \subseteq \xi \). Otherwise, in both runs the quasi-running secrecy of the parent is higher than or incomparable to the observable \( \text{(the parents are both irrelevant). Thus messages } \text{msg}(y_{\alpha}^c, k) \text{ are not relevant in the post step of both runs, and will be dismissed by the projections. The channel } y_{\alpha}^c \text{ will be irrelevant in the post-step of both runs too. However, this does not affect the processes in } D'_{1} \text{ and } D'_{2} \text{ as the parents of messages } (X \text{ and } X') \text{ are already irrelevant in the pre-step.}

Finally, we need to show that projections of \( D'_{1} \) and \( D'_{2} \) are equal in the post-step too.

Consider the resource of the process that offers along \( y_{\alpha}[c] \). By our writing convention, they are all in \( D'_{1} \) and \( D'_{2} \): (i) Those resources offered by \( D'_{1} \) and \( D'_{2} \) with max secretcies higher than or incomparable to the observer \( \xi \) 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 secretcies lower than the observer \( \xi \) in the pre-step, i.e., a sub-tree \( T_i \) in \( D'_{1} \) offering along a channel \( w[c'] \) where \( c' \subseteq \xi \), are relevant in the pre-step and will remain relevant in the post-step (again, in this case, by the tree invaraint, the quasi-running secrecy of the processes offering along \( y_{\alpha}^c+1 \) cannot increase in the post-step).

**Case 2.** \( D_{1} = D'_{1} \text{proc}(x_{\beta}[d], y_{\alpha}^c, k; P@d_1)D'_{1} \) and

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

We consider subcases based on relevancy of the process offering along \( x_{\beta}^d \):

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

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

In this subcase, our goal is to show

\[
D'_{1} \text{msg}(y_{\alpha}^c, k)\text{proc}(x_{\beta}[d], [y_{\alpha}^c+1/y_{\alpha}^c]P@d_1)D'_{1} \mapsto D'_{1} \text{msg}(y_{\alpha}^c, k)\text{proc}(x_{\beta}[d], [y_{\alpha}^c+1/y_{\alpha}^c]P@d_1)D'_{1}
\]

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

**Subcase 2.** \( \text{proc}(x_{\beta}[d], y_{\alpha}^c, k; P@d_1) \) is relevant. By assumption that \( D_{1} \| \xi = D_{2} \| \xi \), and definition of \( \| \xi \):

\[
D_{2} = D'_{2} \text{proc}(x_{\beta}[d], y_{\alpha}^c, k; P@d_1)D'_{2}
\]

and we have

\[
D_{2} \mapsto D'_{2} \text{msg}(y_{\alpha}^c, k)\text{proc}(x_{\beta}[d], [y_{\alpha}^c+1/y_{\alpha}^c]P@d_1)D'_{2}
\]

It remains to show

\[
D'_{1} \text{msg}(y_{\alpha}^c, k)\text{proc}(x_{\beta}[d], [y_{\alpha}^c+1/y_{\alpha}^c]P@d_1)D'_{1} \mapsto D'_{2} \text{msg}(y_{\alpha}^c, k)\text{proc}(x_{\beta}[d], [y_{\alpha}^c+1/y_{\alpha}^c]P@d_1)D'_{2}
\]

• If \( d \subseteq \xi \), then \( x_{\beta} \) 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 invariant it will be less than or equal to \( d \subseteq \xi \), and thus remains observable. As a result, the relevancy of processes in \( D'_{1} \) remain intact. Moreover, every resource of the processes in \( D'_{1} \) is relevant in the pre-steps and post-steps of both runs.

• If \( d \not\subseteq \xi \), then \( x_{\beta} \) is irrelevant in the pre-step and remains irrelevant in the post-steps of both runs, and thus the relevancy of processes in \( D'_{1} \) remain intact. It remains to show that the projections of \( D'_{1} \) and \( D'_{2} \) in post-steps are still equal.

We first consider the trees offered along \( y_{\alpha}^c \) in both runs. The quasi running secrecy of the negative message \( \text{msg}(y_{\alpha}^c, k) \) is \( c \) in the post-steps.

- If \( c \subseteq \xi \) then the same message exists in both runs, and the tree offered along \( y_{\alpha}^c \) is relevant in the pre-steps and post-steps.
– If \( c \not\subseteq \xi \) then the message is irrelevant in both runs. \( y_c^\alpha \) 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 \( D_1' \) and \( D_2' \) are equal in the post-step too. Consider the resources of the process that offers along \( x_\beta[d] \). By our writing convention, they are all in \( D_1' \) and \( D_2' \). (i) Those resources offered by \( D_1' \) and \( D_2' \) with max secracies higher than or incomparable to the observer \( \xi \) 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 secracies lower than the observer \( \xi \) in the pre-step, i.e., a sub-tree \( T_i \) in \( D_1' \) offering along a channel \( w[c'] \) where \( c' \not\subseteq \xi \), are relevant in the pre-step and thus \( T_1 = T_2 \). 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 \( T_1 = T_2 \) remains relevant in the post-step of both runs. Otherwise, if the quasi-running secracies become nonobservable, \( T_1 = T_2 \) may become irrelevant. However, since \( T_1 = T_2 \) and by the tree invariant it only consists of low-secrecy channels, \( T_1 \) becomes irrelevant in the post-step of the first run iff \( T_2 \) becomes irrelevant in the post-step of the second run. Which completes the proof of this case.

**Case 3.** \( D_1 = D_1' T_1 \| \text{proc}(y_\alpha[c], \text{send}x_\beta y_\alpha^c \oplus d_1) D_1' \) and

\[
|D_1'| T_1 \| \text{proc}(y_\alpha[c], \text{send}x_\beta y_\alpha^c ; P \oplus d_1) D_1'| \rightarrow |D_1'| T_1 \| \text{proc}(y_{\alpha+1}[c], [y_{\alpha+1}/y_\alpha^c] P \oplus d_1) \| \text{msg}(\text{send} x_\beta y_\alpha^c) D_1' |
\]

such that \( \Gamma = \Gamma' T_1 \) and \( \Psi_0; \Gamma \triangleright \rightarrow D_1' \triangleright \rightarrow \Gamma'' \) and \( \Psi_0; \Gamma \triangleright \rightarrow T_1 \triangleright \rightarrow (x_\beta; A[c]) \triangleright \rightarrow \text{proc}(y_\alpha[c], \text{send} x_\beta y_\alpha^c ; P \oplus d_1) \triangleright \rightarrow (y_\alpha; A \otimes B[c]) \).

We consider subcases based on relevancy of process offering along \( y_\alpha^c \):

**Subcase 1.** \( \text{proc}(y_\alpha^c, \text{send} x_\beta y_\alpha^c ; P \oplus d_1) \) is not relevant.

By inversion on the typing rules \( d_1 \subseteq c \). By definition either \( d_1 \not\subseteq \xi \) or none of the channels connected to \( P \) including \( y_\alpha^c \) and \( x_\beta \) are relevant. In both cases, neither \( \text{proc}(y_{\alpha+1}[c], [y_{\alpha+1}/y_\alpha^c] P \oplus d_1) \) nor \( \text{msg}(\text{send} x_\beta y_\alpha^c) \) are relevant.

Channel \( y_\alpha^c \) is irrelevant in the pre-step, and both \( y_\alpha^c \) and \( y_{\alpha+1} \) are irrelevant in pre-step and post-step configurations. In this subcase, our goal is to show

\[
|D_1'| T_1 \| \text{proc}(y_{\alpha+1}[c], [y_{\alpha+1}/y_\alpha^c] P \oplus d_1) \| \text{msg}(\text{send} x_\beta y_\alpha^c) D_1' |
\]

We first prove that the relevancy status of \( T_1 \) remain intact. Note that all channels in \( T_1 \), except \( x_\beta \) 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 \( x_\beta \). We show that the relevancy of \( x_\beta \) remains intact after the step:

- If \( c \not\subseteq \xi \), then \( x_\beta \) 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 \( c \subseteq \xi \), then by \( d_1 \subseteq c \), we have \( d_1 \subseteq \xi \), and thus both \( x_\beta \) and \( y_\alpha^c \) must be irrelevant in the pre-step. In the post-step, the parent of the message has to be irrelevant, otherwise \( y_\alpha^c \) would be relevant in the pre-step. Since the parent of the message is irrelevant, we know that \( x_\beta \) remains irrelevant in the post-step. As a result, the message with three irrelevant channels connected to it is irrelevant in the post-step.

In both cases, the relevancy status of \( T_1 \) remains intact: the tree \( T_1 \) rooted at \( x_\beta \) 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 \( T_1 \), we can prove that the relevancy status of processes in \( D_1' \) and \( D_2' \) remain intact.

**Subcase 2.** \( \text{proc}(y_\alpha^c, \text{send} x_\beta y_\alpha^c ; P \oplus d_1) \) is relevant.

By assumption that \( \mathbb{D}_1 \uplus \xi = \mathbb{D}_2 \uplus \xi \), and definition of \( =_\xi \):

\[
\mathbb{D}_2 = D_2' T_2' \| \text{proc}(y_\alpha[c], \text{send} x_\beta y_\alpha^c ; P \oplus d_1) D_2'
\]

We have

\[
|D_2| \rightarrow |D_2'| \| \text{proc}(y_{\alpha+1}[c], [y_{\alpha+1}/y_\alpha^c] P \oplus d_1) \| \text{msg}(\text{send} x_\beta y_\alpha^c) D_2' |
\]

If \( c \not\subseteq \xi \), then \( \text{msg}(\text{send} x_\beta y_\alpha^c) \) is not relevant in both runs, and will be dismissed by the projections. Moreover, \( y_\alpha^c \) is not relevant in the pre-step and post-step configurations. Thus the relevancy of processes in \( D_1' \) and \( D_2' \) will remain intact. Moreover, in this case \( x_\beta \) is irrelevant in both pre-steps and post-steps. Which means that relevancy status of \( T_1' \) and \( T_2' \) remains intact.

If \( c \subseteq \xi \), then \( y_\alpha^c \) is relevant in both runs. We also know that \( x_\beta \) is relevant in the pre-step and \( T_1' = T_2' \) are relevant in the pre-step. In the post-step, \( y_{\alpha+1} \) is relevant in both runs. Relevancy of messages \( \text{msg}(\text{send} x_\beta y_\alpha^c) \) in the post-steps are determined by the quasi-running secrecy \( d' \) and \( d'' \) of their parents \( (X \) and \( X') \) in \( D_1' \) and \( D_2' \). If \( d' \subseteq \xi \), 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 \( \text{msg}(\text{send} x_\beta y_\alpha^c) \) are relevant in both runs, \( y_\alpha^c \) are relevant in the post-step, and trees \( T_1' = T_2' \) and their offering channels \( x_\beta \) are relevant in the post-steps too. The same holds when \( d' \not\subseteq \xi \).

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 \( \text{msg}(\text{send} x_\beta y_\alpha^c) \) are not relevant in the post step of both runs, and will be
dismissed by the projections. The channels \( y_{\alpha} \) will be irrelevant in the post-step too. However, this does not affect the processes in \( D_1' \) and \( D_2' \) as the parents of messages (\( X \) and \( X' \)) are already irrelevant in the pre-step. The channels \( x_{\beta}' \) both become irrelevant in the post-steps. However, we still have \( T_1' \downarrow \xi = T_2' \downarrow \xi \) as they have the same type and their parents have the same quasi-running secrecy.

We show that projections of \( D_1' \) and \( D_2' \) are equal in the post-step too. The resources with secrecy level higher than or incomparable to the observer level offered along \( D_1' \) and \( D_2' \) 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 \( D_1 \) and \( D_2 \), we need to consider the change in quasi-running secrecy as in \textbf{Case 1. Subcase 2}. Moreover, we need to consider the scenario that a relevant resource (\( w'' \)) in the pre-step loses its relevancy in the post-step because the channel offered along \( x_{\beta}' \) is transferred to the message. This case only happens if \( c' \subseteq \xi \) and thus the trees \( T_1 \) and \( T_2 \) offered along \( w'' \) is present in both runs and \( T_1 = T_2 \). We know that \( w'' \) is irrelevant in the post-step of both runs, and the quasi-running secrecy of the processes using the resource \( w'' \) in both runs are the same.

The relevant and irrelevant processes in \( D_1'' \) remain intact.

\textbf{Case 4.} \( D_1 = D_1'T_1'' \text{proc}(w\eta[c'], \text{send} x_{\beta}' y_{\alpha}' @d_1) D_1'' \) and

\[ |D_1'T_1'' \text{proc}(w\eta[c'], \text{send} x_{\beta}' y_{\alpha}' : P@d_1) D_1''| \rightarrow |D_1'T_1'' \text{msg}(\text{send} x_{\beta}' y_{\alpha}' ) \text{proc}(w\eta[c'], [y_{\alpha}'+1/y_{\alpha}'] P@d_1) D_1''|
\]

such that \( \Gamma = \Gamma_1 \Gamma_2 \) and \( \Gamma_1 \vdash D_1' :: \Gamma', y_{\alpha} : (A \rightarrow B)[c] \) and \( \Gamma_2 \vdash T_1'' :: (x_{\beta} : A)[c] \) and

\[ \Gamma', y_{\alpha} : (A \rightarrow B)[c], x_{\beta} : A[c] \vdash \text{proc}(w\eta[c'], \text{send} x_{\beta}' y_{\alpha}' : P@d_1) :: (w\eta[C][c']). \]

In the case where \( T_1'' \) is empty we have \( x_2 = x_2 : A[c] \). We proceed by considering subcases based on relevancy of the process offering along \( w\eta[c'] \):

\textbf{Subcase 1.} \( \text{proc}(w\eta[c'], \text{send} x_{\beta}' y_{\alpha}' : P@d_1) \) is not relevant. By inversion on the typing rules \( d_1 \not\subseteq c \subseteq c' \). By definition either \( d_1 \not\subseteq \xi \) or none of the channels connected to \( P \) including \( y_{\alpha}' \), and \( x_{\beta}' \) are relevant. In both cases, neither \( \text{msg}(\text{send} x_{\beta}' y_{\alpha}' ) \) nor \( \text{proc}(w\eta[c'], [y_{\alpha}'+1/y_{\alpha}'] P@d_1) \) are relevant. Channel \( w_{\alpha}' \) is irrelevant in the pre-step and post-step configurations.

\textbf{Subcase 2.} \( \text{proc}(w\eta[c'], \text{send} x_{\beta}' y_{\alpha}' : P@d_1) \) is relevant. By assumption that \( D_1 \downarrow \xi = \xi D_2 \downarrow \xi \), and definition of \( \xi \):

\[ D_2 = D_2'T_2'' \text{proc}(w\eta[c'], \text{send} x_{\beta}' y_{\alpha}' : P@d_1) D_2'' \]

We have

\[ |D_2| \rightarrow |D_2'T_2'' \text{msg}(\text{send} x_{\beta}' y_{\alpha}' ) \text{proc}(w\eta[c'], [y_{\alpha}'+1/y_{\alpha}'] P@d_1) D_2''| \]

With the same argument as in \textbf{Case 2. Subcase 2}, we can show that relevancy of \( D_2'' \) remains intact.

For \( T_1'' \), we argue that if \( c \subseteq \xi \), then \( T_1'' \downarrow \xi \) is relevant in the pre-step and remains relevant in the post-step too. If \( c \not\subseteq \xi \), then the relevancy of \( T_1'' \) remain intact from pre-step to post-step. See \textbf{Case 3. Subcase 1}, for a more detailed discussion on transferring a tree via message.

The discussion on relevancy of \( D_1'' \) is similar to the previous cases.

\textbf{Case 5.} \( D_1 = D_1'T_1'' \text{msg}(y_{\alpha} k) \text{proc}(x_{\beta} d, \text{case} y_{\alpha} \ell \Rightarrow P_{\ell} @L@d_1) D_1'' \) and

\[ |D_1'T_1'' \text{msg}(y_{\alpha} k) \text{proc}(x_{\beta} d, \text{case} y_{\alpha} \ell \Rightarrow P_{\ell} @L@d_1) D_1''| \rightarrow |D_1'T_1'' \text{proc}(w_{\beta} d, [y_{\alpha}'+1/y_{\alpha}'] P_{\ell} @L@d_1 \cup c) D_1''| \]

We consider sub-cases based on relevancy of process offering along \( w_{\beta} d \). Observe that \( y_{\alpha}' \) is relevant if an only if \( v'' \) is relevant, since they share a message of secrecy \( c' \).

\textbf{Subcase 1.} \( \text{proc}(x_{\beta} d, \text{case} y_{\alpha} \ell \Rightarrow P_{\ell} @L@d_1) \) is not relevant. By definition either \( d_1 \subseteq c \subseteq \xi \) or none of the channels connected to \( P \) including its offering channel \( y_{\alpha}' \) are relevant. In both cases the messages \( \text{msg}(y_{\alpha} k) \) and the continuation process \( \text{proc}(w_{\beta} d, [y_{\alpha}'+1/y_{\alpha}'] P_{\ell} @L@d_1 \cup c) \) are not relevant either. It is straightforward to see that

\[ D_1'T_1'' \text{proc}(w_{\beta} d, [y_{\alpha}'+1/y_{\alpha}'] P_{\ell} @L@d_1 \cup c) D_1'' \downarrow \xi = \xi D_1 \downarrow \xi = \xi D_2 \downarrow \xi . \]

\textbf{Subcase 2.} \( \text{proc}(x_{\beta} d, \text{case} y_{\alpha} \ell \Rightarrow P_{\ell} @L@d_1) \) is relevant. By definition of relevancy, we get that \( c \cup d_1 \subseteq \xi \) and thus \( y_{\alpha}' \) is relevant. This means that \( \text{msg}(y_{\alpha} k) \) is relevant too. By assumption that \( D_1 \downarrow \xi = \xi D_2 \downarrow \xi \), and definition of \( \xi \):

\[ D_2 = D_2'T_2'' \text{msg}(y_{\alpha} k) \text{proc}(x_{\beta} d, \text{case} y_{\alpha} \ell \Rightarrow P_{\ell} @L@d_1) D_2'' . \]
such that $P_i$ is equal to $P_k$ modulo renaming of some channels with secrecy level higher than or incomparable to the observer. We have

$$\mathcal{D}_2 \mapsto \mathcal{D}_2^{1\text{proc}}(x_2[d], [y_{c+1}/y_{c}] P_k @ (d_1 \cup c)) \mathcal{D}_2'$$

This completes the proof of the subcase as we know that the relevancy of channels in $\mathcal{D}_1'$ and $\mathcal{D}_1''$ remain intact.

**Case 6.** $D_1 = D_1^{1\text{proc}}(y_{a}[c], \text{case } y_{a}'(\ell \Rightarrow P_k) \in L \cup d_1) \text{msg}(y_{a}', k) \mathcal{D}_1''$ and $D_1^{1\text{proc}}(y_{a}[c], \text{case } y_{a}'(\ell \Rightarrow P_k) \in L \cup d_1) \text{msg}(y_{a}', k) \mathcal{D}_1'' \rightarrow D_1^{1\text{proc}}(x_1[c], [y_{c+1}/y_{c}] P_k @ c) \mathcal{D}_1''$

We consider sub-cases based on relevancy of process offering along $y_{a}'$. Observe that $y_{a}'$ is relevant if an only if $x_1$ is relevant, since they share a message of secrecy $c$.

**Subcase 1.** $\text{proc}(y_{a}[c], \text{case } y_{a}'(\ell \Rightarrow P_k) \in L \cup d_1)$ is not relevant. By definition either $d_1 \cup c = c \not\sqsubseteq \xi$ or none of the channels connected to $P$ including its offering channel $y_{a}'$ are relevant. In both cases, means that $y_{c+1}$ is not relevant and $\text{msg}(y_{a}', k)$ is not relevant either. Moreover the continuation process proc$(y_{a+1}[c], [y_{c+1}/y_{c}] P_k @ c)$ won’t be relevant. And $D_2 = D_2^{1\text{proc}}(y_{a+1}[c], [y_{c+1}/y_{c}] P_k @ c) \mathcal{D}_2''$

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

$$\mathcal{D}_2 \mapsto \mathcal{D}_2^{1\text{proc}}(y_{a+1}[c], [y_{c+1}/y_{c}] P_k @ c) \mathcal{D}_2''$$

This completes the proof of the subcase as we know that the relevancy of channels in $\mathcal{D}_1'$ and $\mathcal{D}_1''$ remain intact.

**Case 7.** $D_1 = D_1^{1\text{msg}}(\text{send}x_\eta y_{a}', \text{recv}y_{a}; P @ d_1) \mathcal{D}_1''$ and $D_1^{1\text{msg}}(\text{send}x_\eta y_{a}', \text{recv}y_{a}; P @ d_1) \mathcal{D}_1'' \rightarrow D_1^{1\text{proc}}(y_{a}[c], [x_\eta/w][y_{c+1}/y_{c}] P @ (c \cup d_1)) \mathcal{D}_1''$

We consider sub-cases based on relevancy of process offering along $y_{a}'$. Subcase 1. $\text{msg}(y_{a}[c], \text{send}x_\eta y_{a}')$ is not relevant. By definition either $d_1 \cup c = c \not\sqsubseteq \xi$ or none of the channels connected to $P$ including $y_{a}'$ are relevant. In both cases, means that $y_{c+1}$ is not relevant and $\text{msg}(y_{a}', k)$ is not relevant too. By assumption that $D_1 \downarrow \xi = \xi D_2 \downarrow \xi$ and definition of $=\xi$:

$$D_2 = D_2^{1\text{msg}}(y_{a+1}[c], [y_{c+1}/y_{c}] P @ c) \mathcal{D}_2''$$

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 $\mathcal{D}_1'$ 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.** $D_1 = D_1^{1\text{proc}}(y_{a}[c], \text{recv}y_{a}; P @ d_1) \text{msg}(\text{send}x_\eta y_{a}') \mathcal{D}_1''$ and $D_1^{1\text{proc}}(y_{a}[c], \text{recv}y_{a}; P @ d_1) \text{msg}(\text{send}x_\eta y_{a}') \mathcal{D}_1'' \rightarrow D_1^{1\text{proc}}(y_{a+1}[c], [x_\eta/w][y_{c+1}/y_{c}] P @ (c \cup d_1)) \mathcal{D}_1''$

We consider sub-cases based on relevancy of process offering along $y_{a}'$.

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

$$D_1^{1\text{proc}}(y_{a+1}[c], [x_\eta/w][y_{c+1}/y_{c}] P @ d_1) \mathcal{D}_1'' \downarrow \xi = \xi D_1 \downarrow \xi = \xi D_2 \downarrow \xi.$$
Subcase 2. proc\((y_a[c], w \leftarrow \text{recv}v'_0; P@d_1)\) is relevant. By definition of relevancy, we get that \(c \sqcup d_1 = c \sqsubseteq \xi\) and \(y'_a\) is relevant. This means that \(\text{msg}(\text{send}x'_0, y'_0)\) and the channel \(x'_0\) are relevant. By assumption that \(D_1 \vdash \xi = \xi, D_2 \vdash \xi\), and definition of \(=\xi:\)

\[
D_2 = D'_2\\text{proc}(y_a[c], w \leftarrow \text{recv}v'_0; P@d_1)\text{msg}(\text{send}x'_0, y'_0)\\text{D}'_2.
\]

We have

\[
|D_2| \mapsto |D'_2\\text{proc}(y_{a+1}[c], [x'_{a+1}/w][y'_{a+1}/y'_0]P@d_1)\\text{D}'_2|\\mathcal{F}_2
\]

The proof is similar to previous cases.

Case 9. \(D_1 = D'_1\\text{proc}(y_a[c], (x'\gamma\sec(\psi)) \leftarrow X[\gamma] \leftarrow \Gamma_1)@\gamma\sec(\psi'); Q@d_1)\) and

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

\[
|D'_1\\text{proc}(y_a[c], (x'\gamma\sec(\psi)) \leftarrow X[\gamma] \leftarrow \Gamma_1)@\gamma\sec(\psi'); Q@d_1)\\text{D}'_1|
\]

\(\mapsto |D'_2\\text{proc}(x_0[d], \gamma(P)@d_2)\\text{proc}(y_a[c], [x'_{a}/x'd]Q@d_1)\\text{D}'_2|
\]

where \(\gamma = (\gamma\sec, \gamma\var)\) \(\gamma(\psi) = d\), \(\gamma(\psi') = d_2\), and \(\gamma(\Xi) = \Gamma_1\). We consider sub-cases based on relevancy of process offering along \(y'_0\).

Subcase 1. proc\((y_a[c], (x'\gamma\sec(\psi)) \leftarrow X[\gamma] \leftarrow \Gamma_1)@\gamma\sec(\psi'); Q@d_1)\) is not relevant. By definition either \(d_1 \not\sqsubseteq \xi\) or none of the channels of this process including \(y'_0\) are relevant.

In both cases, it means that \(\text{proc}(x_0[d], \gamma(P)@d_2)\) and \(\text{proc}(y_a[c], [x'_{a}/x'd]Q@d_1)\) are not relevant either. Note that \(d_1 \sqsubseteq d_2\) and thus \(d_2 \not\sqsubseteq \xi\) if \(d_1 \not\sqsubseteq \xi\).

\[
D'_1\\text{proc}(y_a[c], (x'\gamma\sec(\psi)) \leftarrow X[\gamma] \leftarrow \Gamma_1)@\gamma\sec(\psi'); Q@d_1)\\text{D}'_1 = \xi, D'_1\\text{D}'_1 \not\subset \xi, D_1 \not\subset \xi, D_2 \not\subset \xi.
\]

Subcase 2. proc\((y_a[c], (x'\gamma\sec(\psi)) \leftarrow X[\gamma] \leftarrow \Gamma_1)@\gamma\sec(\psi'); Q@d_1)\) 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 \(D_1 \vdash \xi = \xi, D_2 \vdash \xi\), and definition of \(=\xi:\)

\[
D_2 = D'_2\\text{proc}(y_a[c], (x'\gamma\sec(\psi)) \leftarrow X[\gamma] \leftarrow \Gamma_1)@\gamma\sec(\psi'); Q@d_1)\\text{D}'_2.
\]

We have

\[
|D_2| \mapsto |D'_2\\text{proc}(x_0[d], \gamma(P)@d_2)\\text{proc}(y_a[c], [x'_{a}/x'd]Q@d_1)\\text{D}'_2|
\]

Remark: we can assume that the fresh channel being spawned will be \(x_0\) in both runs. Moreover, the (unique) substitution \(\gamma\) is the same when stepping both runs.

Note that if \(x'_{a}\) 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. \(x'_{a}\) may block their relevancy path or \(d_1 \not\sqsubseteq \xi\) or in the case where \(d_2 \not\sqsubseteq \xi\). But this happens to the processes in 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 \(\rightarrow\))

Case 10. \(D_1 = D'_1\\text{proc}(y_a[c], F_\gamma[\gamma][c])\text{D}'_1\) and

\[
\text{def}(\Psi; u:C[\psi] \vdash F_\gamma = \text{Fwd}_{C, w \vdash \psi} \Psi :: u : C[\psi])\quad \text{and}
\]

\[
|D'_1\\text{proc}(y_a[c], F_\gamma[\gamma][c])\text{D}'_1| \mapsto |D'_1\\text{proc}(y_a[c], \text{Fwd}_{C, y'_0 \leftarrow z_\beta}@d_1)\text{D}'_1|
\]

where \(\gamma = (\gamma\sec, \gamma\var)\), \(\gamma(\psi) = c\), \(\gamma(u) = y_0\), and \(\gamma(u) = y_0\).

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 \(y'_a\).

\(\square\)

C. Fundamental theorem

Theorem 5 (Fundamental Theorem). For all security levels \(\xi\), and configurations

\[
\Psi; \Gamma_1 \vdash D_1 :: u_a:T_1[c_1], \text{ and } \Psi; \Gamma_2 \vdash D_2 :: v_\beta:T_2[c_2],
\]

with \(D_1 \vdash \xi = \xi, D_2 \vdash \xi, \Gamma_1 \vdash \xi = \Gamma_2 \vdash \xi\), and \(u_a:T_1[c_1] \vdash \xi = v_\beta:T_2[c_2] \vdash \xi\) we have

\[
(\Gamma_1 \vdash D_1 :: u_a:T_1[c_1]) \equiv^\Psi (\Gamma_2 \vdash D_2 :: v_\beta:T_2[c_2]).
\]

Proof. Our goal is to show:
We need to show that from

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

The goal is equivalent to:

Inductive case.

Base case.

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

\[
(\|D_1\|; |D_2|) \in \text{Tree}(|\Gamma| \vdash |K^*|),
\]

which is enough to complete the proof.

**Inductive case.** \( m = m' + 1 \).

\[
(\|B_1\|D_1|T_1|; B_2|D_2|T_2) \in \text{Tree}(|\Gamma| \vdash |K^*|).
\]
Consider an arbitrary $D_1'$ such that $|D_1| \rightarrow^* \tau \cdot |D_1'|$. By Preservation we know that $D_1'$ is IFC-typed for the same interface $\Psi; \Gamma \vdash D_1' :: K^s$ By Lemma 8 for some $D_2$, we get $|D_2| \rightarrow^* \tau \cdot |D_2'|$, such that $\Psi; \Gamma \vdash D_2 :: K^s$ with $D_2' \Downarrow \xi = D_2' \Downarrow \xi$.

We need to show that

$$
\forall u_\delta' \in \text{Out}(\Gamma) \vdash |K^s|, \text{if } u_\delta' \in \mathcal{Y}, \text{then } ((|D_1'|; |D_2'|) \in \mathcal{V}[\Gamma] \vdash |K^s|)_{u_\delta}' \text{ and and}
\forall u_\delta' \in \text{In}(\Gamma) \vdash |K^s|, \text{if } u_\delta' \in \Theta, \text{then } ((|D_1'|; |D_2'|) \in \mathcal{V}[\Gamma] \vdash |K^s|)_{u_\delta}'.
$$

We prove both $\dagger_1$, and $\dagger_2$. We consider an arbitrary $u_\delta'$ and provide a case analysis based on its type. If there is no such $u_\delta'$ in the specified set, then the statements trivially holds. There might be a channel in both of these sets, if and only if $D_1 = \cdot$.

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

**Case 1.** $u_\delta': T'[e'] = K^s = u_\alpha; T_1[c_1] = u_\alpha; \alpha_{c_1}[1]$, and $D_1' = D_1''\text{msg}(\text{close } u_\alpha^{c_1})$.

By the typing rules and $D_1' = D_1''\text{msg}(\text{close } u_\alpha^{c_1})$ we know that $D_1'' = \cdot$, and $\Gamma_1 = \cdot$.

Note that by the definition of relevancy, all processes in $D_1'$ and $D_2'$ has to be relevant. As a result,

$$
D_2' = \text{msg}(\text{close } u_\alpha^{c_1}).
$$

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

$$
\dagger_1 ((|D_1'|; |D_2'|) \in \mathcal{V}[\Gamma] \vdash |u_\alpha;1[c_1]|)_{u_\delta}'.
$$

As needed.

**Case 2.** $u_\delta': T'[e'] = K^s = u_\alpha; T_1[c_1] = u_\alpha; \oplus \{ \ell : A_i \}_{i \in \{c_1\}}$, and $D_1' = D_1''\text{msg}(\text{close } u_\alpha^{c_1}, k)$.

By the assumption of the theorem, $D_1'$ and $D_2'$ are both relevant, and we have $D_2' = D_2''\text{msg}(u_\alpha^{c_1}, k)$. Removing $\text{msg}(u_\alpha^{c_1}, k)$ from $D_1'$ and $D_2'$ does not change relevancy of the remaining configuration when $c_1 \subseteq \xi$:

$$
D_1'' \Downarrow \xi = D_2'' \Downarrow \xi.
$$

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

$$
(|D_1''|, |D_2''|) \in \mathcal{E}[\Gamma] \vdash |u_\alpha; \oplus \{ \ell : A_i \}_{i \in \{c_1\}}|^{m'+1}_{u_\delta}'.
$$

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

$$
\dagger_1 ((|D_1''|, |D_2''|) \in \mathcal{V}[\Gamma] \vdash |u_\alpha; \oplus \{ \ell : A_i \}_{i \in \{c_1\}}|^{m'+1}_{u_\delta}'.
$$

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

$$
D_1' = D_1''A_1\text{msg}(\text{send } x_{\beta_i}^{c_1} u_\alpha^{c_1})
$$

where $\Gamma = \Gamma'$, $\Gamma''$, and $\Psi; \Gamma'' \vdash A_1 :: (x_{\beta_i} : A[c_1])$, and $\Psi; \Gamma' \vdash D_1'' :: (u_\alpha; 1 : A[c_1])$.

Since $D_1'$ and $D_2'$ are both IFC-typed, they enjoy the tree invariant. Thus both of them are relevant and since $D_1' \Downarrow \xi = D_2' \Downarrow \xi$, we have

$$
D_2' = D_2''A_2\text{msg}(\text{send } x_{\beta_i}^{c_1} u_\alpha^{c_1}),
$$

such that $\Psi; \Gamma'' \vdash A_2 :: (x_{\beta_i} : A[c_1])$, and $\Psi; \Gamma' \vdash D_2'' :: (u_\alpha; 1 : A[c_1])$. Moreover, by relevancy of $D_1'$ (and relevancy of $x_{\beta_i}^{c_1}$) we get $D_1'' \Downarrow \xi = D_2'' \Downarrow \xi$ and $A_1 \Downarrow \xi = A_2 \Downarrow \xi$.

Note that in the particular case with $x_{\beta_i} : A[c_1] \in \Gamma$, we have $A_i = \cdot$ and $\Gamma'' = x_{\beta_i} : A[c_1]$.

We apply the induction hypothesis on the index $m' < m$ for $\Psi; \Gamma' \vdash A_1 :: (x_{\beta_i} : B[c_1])$ to get

$$
(|A_1|, |A_2|) \in \mathcal{E}[\Gamma'] \vdash |x_{\beta_i} : A[c_1]|^{m'}.
$$

and apply the induction hypothesis on the index $m' < m$ for $\Psi; \Gamma' \vdash D_1'' :: (u_\alpha; 1 : A[c_1])$

$$
(|D_1'|, |D_2'|) \in \mathcal{E}[\Delta] \vdash |u_\alpha; A \otimes B[c_1]|^{m'+1}_{u_\delta}'.
$$

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

$$
\dagger_1 ((|A_1|, |A_2|) \in \mathcal{E}[\Gamma'] \vdash |x_{\beta_i} : A[c_1]|^{m'}).
$$

**Case 4.** $\Gamma = \Gamma', x_{\gamma_i} : \{ \ell : A_i \}_{i \in \{c_1\}}$, and $u_\delta'; T'[e'] = x_{\gamma_i} : \{ \ell : A_i \}_{i \in \{c_1\}}$, and $D_1' = \text{msg}(x_{\gamma_i}^{c_1}, k)D_2'$.

By the assumption of the theorem, $D_2 = \text{msg}(x_{\gamma_i}^{c_1}, k)D_2'$. Moreover, $D_2'' \Downarrow \xi = D_2' \Downarrow \xi$. The reason is $c_{\gamma_i+1}$ is relevant in $D_2'$ 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
\[ (|D'_1|, |D'_2|) \in \mathcal{E}[|\Gamma', x_{\gamma+1} : A_k[c]| \vdash |K^*|]^{m'} \]
By line (8) in the definition of $\mathcal{V}$:
\[ \vdash_1(|\text{msg}(x'_{\gamma}, k) |D'_1|, |\text{msg}(x'_{\gamma}, k) |D'_2|) \in \mathcal{V}[|\Gamma', x_{\gamma} : \{ \ell : A_k \}_{\ell \in \ell}, |c|] \vdash |K^*|]^{m'+1} \]
**Case 5.** $\Gamma = \Gamma', \Gamma'' x_{\gamma} (A \rightarrow B)[c]$, and $u'_2; T'[c'] = x_{\gamma} (A \rightarrow B)[c]$, and
\[ D'_1 = \mathcal{A}_1 \text{msg}(\text{send}x'_{\gamma}, k) D'_1 \]
such that $\Psi_0; \Gamma'' \vdash \mathcal{A}_1 :: y_0 : A[c]$, and $\Psi_0; \Gamma', x_{\gamma+1} : B[c] \vdash D'_2 :: K^*$. The message $\text{msg}(\text{send}x'_{\gamma})$ is relevant in $D'_1$. By assumption of the theorem, $D'_2 = \mathcal{A}_2 \text{msg}(\text{send}x'_{\gamma}) D'_2$, such that $\Psi_0; \Gamma'' \vdash \mathcal{A}_2 :: y_0 : A[c]$ and $\Psi_0; \Gamma'_2, \Gamma', x_{\gamma+1} : B[c] \vdash D'_2 :: K^*$ with $\Gamma'_2$ being the context of all the channels in $\Gamma_2$ with security higher than or incomparable to the observable level $\xi$. Again because of the tree invariant every resource of $\mathcal{A}_2$ has a secrecy less than or equal to the observer. Also, by assumption we know that $D'_1 \downarrow \xi = D'_2 \downarrow \xi$. By definition of relevancy, we know that $\text{msg}(\text{send}x'_{\gamma})$ and the tree $\mathcal{A}_1$ connected to it are relevant in $D_1$ and thus $\mathcal{A}_1$ is equal to $\mathcal{A}_2$, i.e., $\mathcal{A}_1 \downarrow \xi = \mathcal{A}_2 \downarrow \xi$. Removing the tress $\mathcal{A}_1$ and $\mathcal{A}_2$ and the messages from both configurations does not change relevancy of the rest of the configuration since $x_{\gamma+1}$ will remain relevant, i.e., $D'_1 \downarrow \xi = D'_2 \downarrow \xi$.
We can apply the induction hypothesis on the smaller index $m' < m$ and $\mathcal{A}_1 \downarrow \xi = \mathcal{A}_2 \downarrow \xi$ and $D'_1 \downarrow \xi = D'_2 \downarrow \xi$ to get
\[ (|\mathcal{A}_1|, |\mathcal{A}_2|) \in \mathcal{E}[|\Gamma''| \vdash |y_0 : A[c]|]^{m'}, \text{ and} \\
(|D'_1|, |D'_2|) \in \mathcal{E}[|\Gamma', x_{\gamma+1} : B[c]| \vdash |K^*|]^{m'+1} \]
By line (10) in the definition of $\mathcal{V}$:
\[ \vdash_1(|\mathcal{A}_1 \text{msg}(\text{send}x'_{\gamma}) D'_1|, |\mathcal{A}_2 \text{msg}(\text{send}x'_{\gamma}) D'_2|) \in \mathcal{V}[|\Gamma', x_{\gamma} : A \rightarrow B[c]| \vdash |K^*|]^{m'+1} \]
**Proof of $\vdash_2$.** Consider an arbitrary channel $u'_2; T \in \text{In}[|\Gamma| \vdash |K|]$ We consider different cases based on the session type $T$.
**Case 1.** $u'_2; T'[c'] = K^* = u'_0; T_1[c_1] = u'_0; \{ \ell : A_k \}_{\ell \in \ell}, c_1$. 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 $D'_1 \downarrow \xi = D'_2 \downarrow \xi$ and $c_1 \downarrow \xi$, we get
\[ D'_1 \text{msg}(u'_0, k) \downarrow \xi = D'_2 \text{msg}(u'_0, k) \downarrow \xi \]
By induction hypothesis
\[ (|D'_1|, |D'_2|) \in \mathcal{E}[|\Gamma| \vdash |u'_0+1 : A_k[c_1]|]^{m'} \]
By line (3) in the definition of $\mathcal{V}$:
\[ \vdash_2(|D'_1|, |D'_2|) \in \mathcal{V}[|\Gamma| \vdash |u_0 : \{ \ell : A_k \}_{\ell \in \ell} : c_1]|]^{m'+1} \]
**Case 2.** $u'_2; T'[c'] = K^* = u'_0; T_1[c_1] = u'_0; A \rightarrow B[c_1]$. By assumption and $c_1 \downarrow \xi$, we know that $D'_1$ and $D'_2$ are relevant. Consider an arbitrary channel $x_{\beta}$. which is not a free name in $\Gamma \vdash |K^*|$. We know $c_1 \downarrow \xi$, and thus adding a negative message sending $x_{\beta} \downarrow \xi$ along $u'_0 \downarrow \xi$ does not change the relevancy of the rest of processes:
\[ D'_1 \text{msg}(\text{send}x_{\beta}^{c_1}, u'_0) \downarrow \xi = D'_2 \text{msg}(\text{send}x_{\beta}^{c_1}, u'_0) \downarrow \xi \]
We can apply the induction hypothesis on $m' < m$ to get
\[ (|D'_1|, |D'_2|) \in \mathcal{E}[|\Gamma, x_{\beta} : A[c_1]| \vdash |u_0+1 : B[c_1]|]^{m'} \]
By line (5) in the definition of $\mathcal{V}$:
\[ \vdash_2(|D'_1|, |D'_2|) \in \mathcal{V}[|\Gamma| \vdash |u_0 : A \rightarrow B[c_1]|]^{m'+1} \]
**Case 3.** $\Gamma = \Gamma', x_{\gamma}; [c_1]$, and $u'_2; T'[c'] = x_{\gamma}; [c_1]$
We first briefly explain why the invariant of induction holds after we bring the closing message inside $D'_1$ and remove the channel $x_{\gamma}$ from $\Delta$, i.e.
\[ \text{msg}(\text{close} x_{\gamma}) D'_1 \downarrow \xi = \text{msg}(\text{close} x_{\gamma}) D'_2 \downarrow \xi \]
* If the parent of $\text{msg}(\text{close} x_{\gamma})$ in $D'_1$ is relevant in $D'_2$ before bringing the message inside then by the assumption of the theorem, it is the same as the parent of $\text{msg}(\text{close} x_{\gamma})$ in $D'_2$. If after adding the message $D'_1$ the parent still remains relevant, it means that it has at least one other relevant channel other than $x_{\gamma}$ in $D'_1$ which also exists in $D'_2$ and will be relevant after adding the message to $D'_2$. If after adding the message, the message’s parent in $D'_1$ becomes irrelevant, it means that it does not have a relevant path to
any other channel in $\Delta'$ and $K$. By the assumption of theorem the parent of the message in $D'_2$ does not have such path either. The same argument holds for any other node that becomes irrelevant because of adding the closing message to $D'_1$. As a result the same processes becomes irrelevant in both $D'_1$ and $D'_2$ 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 $\text{msg}(\text{close } x''_c)$ in $D'_2$ before adding the message is relevant.

* Otherwise the parent of $\text{msg}(\text{close } x''_c)$ is irrelevant in $D'_1$ and $D'_2$ 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 $m' < m$:

\[(\text{msg}(\text{close } x''_c)D'_1 : [\text{msg}(\text{close } x''_c)D'_2]) \in E[[\Gamma'] \models |K^*]|^{m'}].\]

By line (6) in the definition of $V$,

\[\downarrow_2(D'_1 : |D'_2|) \in V[[|\Gamma'|, x_{\gamma} : 1[c]] \models |K^*]|^{m'+1}.\]

**Case 4.** $\Gamma = \Gamma', x_{\gamma} : \{\ell : A_{t}\}_{t \in L[c]}$, and $u' : T'[c'] = x_{\gamma} : \{\ell : A_{t}\}_{t \in L[c]}$.

Consider an arbitrary channel $y_{\eta}$ which is not a free name in $\Gamma \models K^*$. We have $\text{msg}(x''_c, k)D'_1 \uplus \xi = \text{msg}(x''_c, k)D'_2 \uplus \xi$, since $x''_c+1$ is relevant and the positive messages $\text{msg}(x''_c, k)$ 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

\[(\text{msg}(x''_c, k)D'_1 : [\text{msg}(x''_c, k)D'_2]) \in E[[\Gamma', x_{\gamma+1} : A_{k}[c]] \models |K^*]|^{m'}].\]

By line (7) in the definition of $V$:

\[\downarrow_2(D'_1 : |D'_2|) \in V[[|\Gamma'|, x_{\gamma} : \{\ell : A_{t}\}_{t \in L[c]} \models |K^*]|^{m'+1}.\]

**Case 5.** $\Gamma = \Gamma', x_{\gamma} : (A \otimes B)[c]$, and $u' : T'[c'] = x_{\gamma} : (A \otimes B)[c]$. Consider an arbitrary channel $y_{\eta}$ which is not a free name in $\Gamma \models K^*$. We have $\text{msg}(\text{send}y''_c x''_c)D'_1 \uplus \xi =_\ell \text{msg}(\text{send}y''_c x''_c)D'_2 \uplus \xi$.

The quasi-running secrecy of $\text{msg}(\text{send}y''_c x''_c)$ 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

\[(\text{msg}(\text{send}y''_c x''_c)D'_1 : [\text{msg}(\text{send}y''_c x''_c)]) \in E[[|\Gamma'|, y_{\eta} : A[c], x_{\gamma+1} : B[c]] \models |K^*]|^{m'}].\]

By line (9) in the definition of $V$:

\[(D'_1 : |D'_2|) \in V[[|\Gamma'|, x_{\gamma} : A \otimes B[c]] \models |K^*]|^{m'+1}.\]
IX. 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 Lemma 16 and Corollary 2 introduced in Section IX which are in turn instrumental in proving transitivity (see § XI) and adequacy (see § XII).

A. 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 20 (Produced processes and messages). For each dynamic step in (Fig. 3) we say that a process or message is produced in the step if it occurs in the post-step (right side of ⇒ notation). For example, the transition step for Spawn, produces two processes

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

and the transition step for $\neg_{\text{snd}}$ produces a message and a process

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

Definition 21 (Active processes and messages). For each dynamic step in (Fig. 3) we define the active configuration $A$ as the set of processes and messages occurring in the pre-step (the left side of ⇒ notation). For example, in the transition step for Spawn, the active configuration is the single process

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

and in the transition step for $\neg_{\text{rev}}$, the active configuration is

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

We define the set of active configurations, i.e., active, for $D \rightarrow D' \rightarrow D$ as the union of every step’s active configuration. A process is called active in $D \rightarrow D'$ if it is in the set active.

Lemma 9 (Uniqueness of process productions). Consider $\Delta \models D :: K$, and $D \rightarrow D' \rightarrow D_1 \text{proc}(x, P)D_2$, such that $\text{proc}(x, P)$ is produced in the last step. Then process $\text{proc}(x, P)$ does not occur in any of the previous steps $D \rightarrow D'$.

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 $\text{proc}(x_\alpha, P)$, either (a) $x_\alpha$ is freshly generated (in the case of Spawn for the callee), or (b) $\text{proc}(x_\alpha, P)$ replaces another process $\text{proc}(x_\alpha, P')$ that offers along $x_\alpha$ and has a larger process term, i.e. $|P| < |P'|$ (in $I_{\text{rev}}, I_{\text{recv}}, I_{\text{rev}}, I_{\text{snd}}, \neg_{\text{snd}}, \text{and spawn}$ for the continuation of the caller), or (c) $x_\alpha$ is a fresh generation of $x$ (in $I_{\text{snd}}, I_{\text{recv}}, I_{\text{rev}}, \neg_{\text{rev}}$)

Consider production of a process $\text{proc}(x_\alpha, P)$ and the cases described in 4. If 4.(a) or 4.(c) hold, then by freshness of $x_\alpha$, such process has not been produced before. It is enough to consider case 4.(b). Assume that there is another occurrence of process $\text{proc}(x_\alpha, P)$ 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 $\text{proc}(x, P)$ to the later one $\text{proc}(x_\alpha, P)$, which is contradictory.

With a similar reasoning we can prove that if $\Delta \models D :: K$, and $D \rightarrow D' \rightarrow D_1, I_{\text{msg}}(M)D_2$, such that $I_{\text{msg}}(M)$ is produced in the last step and $I_{\text{msg}}(M)$ does not occur in any of $D \rightarrow D'$ steps.

Lemma 10 (Diamond Property). If $\Delta \models D :: K$ and $D \rightarrow D_1 \rightarrow D_1' \rightarrow D_1''$ and $\neg_{\text{rev}} D_1 \rightarrow \neg_{\text{rev}} D_1'$ and $\neg_{\text{rev}} D_1 \rightarrow \neg_{\text{rev}} D_1''$ and $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$, then there is a configuration $D'$ such that $\neg_{\text{rev}} D_1'' \rightarrow \neg_{\text{rev}} D_1'$ and $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$, and $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$, where $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$. The messages produced along the channels $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$ are identical in $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$ and $\neg_{\text{rev}} D_1' \rightarrow \neg_{\text{rev}} D_1''$.

Moreover, every process in $D_1'$ that is not an active process of $D_1 \rightarrow D_1''$ is in $D$. And every process in $D_1''$ that is not an active process of $D_1 \rightarrow D_1''$ is in $D$.

Proof. The proof is straightforward by cases. The key is to build $\star$ (locally) identical to $\neg_{\text{rev}}$, and $\star'$ (locally) identical to $\neg_{\text{rev}}$.\[\square\]
Lemma 11 (Confluence). If \( \Delta \vdash D_1 : K \) and \( \uparrow D_1 \rightarrow^{m_1}_{\Delta \vdash K} D'_1 \) and \( \uparrow' D_1 \rightarrow^{m'_{\Delta \vdash K}} D''_1 \), then there is a configuration \( D \) such that \( \star D_1 \rightarrow^{m_1}_{\Delta \vdash K} D \) for some \( j \leq n' \), and \( \star D''_1 \rightarrow^{m'_{\Delta \vdash K}} D \) for some \( k \leq n' \), where \( \Upsilon \sqcup \Upsilon' = \Upsilon'' \). The messages produced along the channels \( \Upsilon \cap \Upsilon' \) are identical in \( D_1' \) and \( D_1'' \), and \( D_1' \). The steps in \( \star \) are (locally) identical to the steps of \( \uparrow' \) that do not occur in \( \uparrow \). And the steps in \( \star' \) are (locally) identical to the steps of \( \uparrow \) that do not occur in \( \uparrow' \).

Moreover, every process in \( D_1'' \) that is not an active process of \( D_1 \rightarrow^{*}_{\Delta \vdash K} D_1' \) is in \( D \) and every process in \( D_1'' \) that is not an active process of \( D_1 \rightarrow^{*}_{\Delta \vdash K} D_1' \) is in \( D \).

Proof. It follows by standard inductions from the diamond property (\textbf{Lem. 10}). 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 (\textbf{Lem. 10}). 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 \( D_1'' \) and \( D_2'' \), we build \( D_2 \) (with the blue steps) that satisfies the required properties. Then, again by induction we build \( D_3 \) (with the red steps) and \( D_4 \) (with the violet steps). And finally, with a last induction, we build \( D \) (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 \( D_1' \), we put \( \Upsilon = \Upsilon_1, \Upsilon_2, x \) and for \( D_1'' \), we put \( \Upsilon' = \Upsilon_1', \Upsilon_2, y \). The set \( \Upsilon_2 \) is in both \( \Upsilon \) and \( \Upsilon' \), and we have \( \Upsilon_1 \cap \Upsilon_1' = \emptyset \), i.e., we put all the common channels (except possibly \( x \) and \( y \)) in \( \Upsilon_2 \). We assume that the step \( D_2 \rightarrow D_1' \) 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 \( D_2'' \rightarrow D_1'' \) 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 \( D \) with at most \( m + 1 \) steps from \( D_1'' \) and at most \( n + 1 \) steps from \( D_1' \). The configuration \( D \) sends messages along the union of \( \Upsilon \) and \( \Upsilon' \), and by induction the messages along the intersection of \( \Upsilon \cap \Upsilon' \) are identical in \( D_1' \) and \( D_1'' \) and \( D_1' \). 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 \( D \) 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 \( D_1' \) and \( D_1'' \), 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 \( D_1 \rightarrow^{*} D_1' A D_1'' \rightarrow D_1' \text{proc}(x, P)D_1'' \) and \( D_1 \rightarrow^{*} D_2' A' D_2'' \rightarrow D_2' \text{proc}(x, P)D_2'' \), where \( A \) and \( A' \) are the active parts that produce \( \text{proc}(x, P) \), then \( A = A' \).

Proof. This is another corollary to the second part of the confluence lemma (\textbf{Lem. 11}). First observe that \( A \) is not active in \( D_1 \rightarrow^{*} D_2' A' D_2'' \): if it is active, we produce \( \text{proc}(x, P) \) twice in \( D_1 \rightarrow^{*} D_2' A' D_2'' \rightarrow D_2' \text{proc}(x, P)D_2'' \), which contradicts with uniqueness of process productions (\textbf{Lem. 9}). Similarly, \( A' \) is not active in \( D_1 \rightarrow^{*} D_1' A' D_1'' \). With the second part of the confluence lemma, for some \( D \), we have \( D_1 \rightarrow^{*} D_1' A' D_1'' \rightarrow D_1' \text{proc}(x, P)D_1'' \) such that both \( A \) and \( A' \) occur in \( D \). If \( A \neq A' \), we can produce \( \text{proc}(x, P) \) twice from \( D \) which by \textbf{Lem. 9} is contradictory. Thus, know that \( A = A' \).

Note: here we rely on the fact that the steps in \( \uparrow' \) 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 \( D_1 \rightarrow^{*} D_1' A D_1'' \rightarrow D_1' \text{msg}(M)D_1'' \) and \( D_1 \rightarrow^{*} D_2' A' D_2'' \rightarrow D_2' \text{msg}(M)D_2'' \), where \( A \) and \( A' \) are the active parts that produce \( \text{msg}(M) \), then \( A = A' \).
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 \( D_1 \xrightarrow{\alpha} D'_1 AD''_1 \xrightarrow{\Delta_1[x_\beta/y_\alpha]} D_2 \) and \( D_1 \xrightarrow{\alpha} D'_2 A'D''_2 \xrightarrow{\Delta_2[x_\beta/y_\alpha]} D_2 \), where \( A \) and \( A' \) are forwarding processes that step by renaming the resource \( y_\alpha \) to \( x_\beta \), then \( A = A' \).

**Lemma 12 (Building minimal sending configuration).** Consider \( \Delta \vdash D_2 :: K \), a set of channels \( \Upsilon_1 \subseteq \Delta \), and for some \( \Upsilon_2 \supseteq \Upsilon_1 \). There exists a set \( \Upsilon \) and a configuration \( D'_2 \) such that \( \Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2 \) and \( D_2 \xrightarrow{\alpha} D'_2 \) and \( D'_2 \xrightarrow{\alpha} \Upsilon \) for all \( \Upsilon_2 \geq \Upsilon_1 \). We call \( D'_2 \) and \( \Upsilon \) the minimal sending configuration and the minimal sending set with respect to \( \Upsilon_1 \) and \( \Upsilon_2 \), respectively.

**Proof.** We first provide an algorithm to build \( D'_2 \) and \( \Upsilon \) based on the transition steps in \( D_2 \) and the set \( \Upsilon_1 \) and \( \Upsilon_2 \). By Corollary 1, the set of generators of a set \( \Upsilon \) and a configuration \( D' \) with \( \Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2 \), we build the same \( D'_2 \) and \( \Upsilon \) that satisfies \( D'_2 \xrightarrow{\alpha} \Upsilon \).

**Algorithm 1 Building the minimal sending configuration**

**Require:** A set \( \Upsilon_1 \) of channels, configuration \( D_2 \), and a configuration \( D \) with \( D_2 \xrightarrow{\alpha} \Upsilon \) and \( \Upsilon_1 \subseteq \Upsilon_2 \).

**Ensure:** A set \( \Upsilon \) and a configuration \( D''_2 \) with \( D_2 \xrightarrow{\alpha} \Upsilon \) and \( D''_2 \xrightarrow{\alpha} \Upsilon \) and \( \Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2 \).

\[
S := \text{the local transition steps in } D_2 \xrightarrow{\alpha} \Upsilon \\
i := 0 \\
X_0 := \emptyset \\
M := \text{the messages in } D \text{ along } \Upsilon_1 \\
A := \emptyset \\
\text{while } (M \neq \emptyset) \text{ do} \\
\quad \text{for } (s \text{ in } S) \text{ do} \\
\quad \quad \text{if } (\exists p \in \text{post}(s), p \in M) \text{ then} \\
\quad \quad \quad A := A \cup \{\text{pre}(s)\} \\
\quad \quad \quad X_i := X_i \cup \{s\} \\
\quad \quad \text{end if} \\
\quad \text{end for} \\
\quad i := i + 1 \\
\quad X_i := \emptyset \\
\quad M := A \\
\quad A := \emptyset \\
\text{end while}
\]

\[
\text{Cfg} := D_2 \\
\text{Tns} := \cdot \\
i := i - 1 \\
\text{while } (i > 0) \text{ do} \\
\quad \text{for } (s \text{ in } X_i) \text{ do} \\
\quad \quad \text{Cfgpost} := \text{the global post-state when the local step } s \text{ applies to the configuration } \text{Cfg}. \\
\quad \quad \text{Tns.append(Cfg } \rightarrow \text{Cfgpost) } \\
\quad \quad \text{Cfg} := \text{Cfgpost} \\
\quad \text{end for} \\
\quad i := i - 1 \\
\text{end while}
\]

\[
\text{D}_2'' := \text{Cfg} \\
\Upsilon := \text{Send(D}_2''\rangle \\
\text{return } \text{D}_2'', \text{Tns}, \Upsilon
\]

For every given \( \Upsilon_1, \Upsilon_2, \) and \( \Delta \) with \( D_2 \xrightarrow{\alpha} \Upsilon \) and \( \Upsilon_1 \subseteq \Upsilon_2 \), Algorithm 1 returns a configuration \( D'' \) and set \( \Upsilon \) and builds the dynamic transition \( D_2 \xrightarrow{\alpha} \Upsilon \) \( D'' \) in Tns. Observe that the steps of \( D_2 \xrightarrow{\alpha} \Upsilon \) \( D'' \) are all local transitions of \( D_2 \xrightarrow{\alpha} \Upsilon \). Thus, by the confluence lemma (Lem. 11), we know that \( D'' \xrightarrow{\alpha} \Upsilon \).

It remains to be shown that the \( D''_2 \) that Algorithm 1 builds is independent of the choice of \( D \) and \( \Upsilon_2 \), and is uniquely identified for each \( D_2 \) and \( \Upsilon_1 \). By the confluence lemma, we know that for each \( D \) with \( D_2 \xrightarrow{\alpha} \Upsilon \) 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 \( D \) with \( D_2 \xrightarrow{\alpha} \Upsilon \). Similarly, the set of local steps that produces the set \( M \) from its generators is the same despite the
choice of the configuration $D$ with $D_2 \rightarrow^{xy_2} D$. As a result, we collect the same local transition steps in all $X_i$s for every $D$ with $D_2 \rightarrow^{xy_2} D$. The local transition steps in all $X_i$s are those with which we construct $D''_2$ from $D_2$ and from $D''_2$ we can uniquely identify the set $\bar{Y}$, and the proof is complete.

\section*{B. Backward closure}

\begin{lemma}[Backward closure on the second run] \label{lem:backward}
The second run enjoys backward closure:
\begin{enumerate}
\item If $(D_1; D_2) \in E[\Delta \vdash K]^k$ and for $D''_2 \in \text{Tree}(\Delta \vdash K)$, we have $D''_2 \rightarrow^{xy_2} D_2$ then $(D_1; D''_2) \in E[\Delta \vdash K]^k$.
\item If $(D_1; D_2) \in V[\Delta \vdash K]^k+1$, and for $D''_2 \in \text{Tree}(\Delta \vdash K)$, we have $D''_2 \rightarrow^{xy_2} D_2$ with $D''_2$ sending along channel $y_{a\cdot}$, then $(D_1; D''_2) \in V[\Delta \vdash K]^k+1$.
\item If $(D_1; D_2) \in V[\Delta \vdash K]^k+1$, and for $D''_2 \in \text{Tree}(\Delta \vdash K)$, we have $D''_2 \rightarrow^{xy_2} D_2$, then $(D_1; D''_2) \in V[\Delta \vdash K]^k+1$.
\end{enumerate}
\end{lemma}

\begin{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 $(D_1; D_2) \in E[\Delta \vdash K]^m+1$. By line (12) of the logical relation we know
\[
(*) \forall \mathcal{Y}_1, \Theta_1, D'_{1}, \text{if } D_1 \rightarrow^{xy_1, \Theta_1} D'_1, \text{ then } \exists \mathcal{Y}_2, D'_2 \text{ such that } D_2 \rightarrow^{xy_2} D'_2 \text{ and } \\
\forall x_a \in \text{Out}(\Delta \vdash K), \text{ if } x_a \in \mathcal{Y}_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a} \text{ and } \\
\forall x_a \in \text{In}(\Delta \vdash K), \text{ if } x_a \in \Theta_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a}.
\]

Consider $D''_2$, for which by the assumption we have $D''_2 \rightarrow^{xy_2} D_2$. Our goal is to prove $(D_1; D''_2) \in E[\Delta \vdash K]^m+1$. We need to show that
\[
(\dagger) \forall \mathcal{Y}_1, \Theta_1, D'_{1}, \text{if } D_1 \rightarrow^{xy_1, \Theta_1} D'_1, \text{ then } \exists \mathcal{Y}_2, D'_2 \text{ such that } D''_2 \rightarrow^{xy_2} D'_2 \text{ and } \mathcal{Y}_1 \subseteq \mathcal{Y}_2 \text{ and } \\
\forall x_a \in \text{Out}(\Delta \vdash K), \text{ if } x_a \in \mathcal{Y}_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a} \text{ and } \\
\forall x_a \in \text{In}(\Delta \vdash K), \text{ if } x_a \in \Theta_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a}.
\]

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
\[
(\dagger') \exists \mathcal{Y}_2, D'_2 \text{ such that } D''_2 \rightarrow^{xy_2} D'_2 \text{ and } \\
\forall x_a \in \text{Out}(\Delta \vdash K), \text{ if } x_a \in \mathcal{Y}_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a} \text{ and } \\
\forall x_a \in \text{In}(\Delta \vdash K), \text{ if } x_a \in \Theta_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a}.
\]

and the goal
\[
(\dagger'') \forall \mathcal{Y}_2, D'_2 \text{ such that } D'_2 \rightarrow^{xy_2} D'_2 \text{ and } \\
\forall x_a \in \text{Out}(\Delta \vdash K), \text{ if } x_a \in \mathcal{Y}_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a} \text{ and } \\
\forall x_a \in \text{In}(\Delta \vdash K), \text{ if } x_a \in \Theta_1, \text{ then } (D'_1; D'_2) \in \mathcal{V}_{\bar{D}}[\Delta \vdash K]^{m+1}_{\mathcal{Y}_a}.
\]

We apply an $\exists$-Elimination on the assumption to get $\mathcal{Y}_2$ and $D'_2$ that satisfies the conditions and use the same $\mathcal{Y}_2$ and $D'_2$ to instantiate the existential quantifier in the goal. Since $D''_2 \rightarrow^{xy_2} D_2$, we get $D''_2 \rightarrow^{xy_2} D'_2$, and the proof is complete.

2) The proof is by cases on the row of the logical relation that makes the assumption $(D_1; D_2) \in V[\Delta \vdash K]^{k+1}_{\mathcal{Y}_a}$ true. Here we only consider an interesting case, the proof of other cases is similar.

\textbf{Row 4.} By the conditions of this case, we know that $\Delta = \Delta'$, $\Delta''$, and $K = y_{a\cdot}:A \otimes B$. Moreover, we have
\[
D_1 = D_1' T_1 \text{ msg}(send x_{\beta} y_{a\cdot}) \text{ for } T_1 \in \text{Tree}(\Delta'' \vdash x_{\beta}:A), \\
D_2 = D_2' T_2 \text{ msg}(send x_{\beta} y_{a\cdot}) \text{ for } T_2 \in \text{Tree}(\Delta'' \vdash x_{\beta}:A),
\]

\[
(\dagger_1) (T_1; T_2) \in \mathcal{E}_{\bar{D}}[\Delta'' \vdash x_{\beta}:A[n]]^k, \text{ and } \\
(\dagger_2) (D_1'; D_2') \in \mathcal{E}_{\bar{D}}[\Delta' \vdash y_{a\cdot+1}:B]^k
\]

These are also the statements that we need to prove when replacing $D_2$ with $D''_2$. By the assumption that $D''_2$ is sesssion-typed and sends along $y$, uniqueness of channels, and $D''_2 \rightarrow^{xy_2} D_2$ we get
\[
D''_2 = D_2' T_2' \text{ msg}(send x_{\beta} y_{a\cdot}) \text{ for } T_2' \in \text{Tree}(\Delta'' \vdash x_{\beta}:A[c])
\]

Moreover, since $T_2$ and $D_2'$ are disjoint sub-trees with the common parent $\text{msg}(send, x_{\beta} y_{a\cdot})$ and cannot communicate with each other internally, we have $T_2' \rightarrow^{xy_2} T_2$ and $D_2' \rightarrow^{xy_2} D_2'$.

Now we can apply the the result of part (1) of this lemma on $(\dagger_1)$ and $(\dagger_2)$ to get
\[
(\dagger_1')(T_1; T_2') \in \mathcal{E}_{\bar{D}}[\Delta'' \vdash x_{\beta}:A[n]]^k, \text{ and }
\]

\end{proof}
and the proof of this subcase is complete.

3) The proof is by considering the row of the logical relation that ensures the assumption $(D_1; D_2) \in \mathcal{E}[\Delta \vdash K]_{y_\alpha+1}^{m+1}$. 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 $K = y_\alpha : A \rightarrow B$. We have $(D_1; D_2) \in \text{Tree}_\Phi(\Delta \vdash y_\alpha : A \rightarrow B)$ and

$$(\dagger) \forall x_\beta \notin \text{dom}(\Delta, K). \quad (D_1 \text{msg}(\text{send} x_\beta^\alpha y_\alpha); D_2 \text{msg}(\text{send} x_\beta^\alpha y_\alpha)) \in \mathcal{E}_\Phi[\Delta, x_\beta : A \vdash y_\alpha+1 : B]^{m+1}$$

These are also the statements that we need to prove when replacing $D_2$ with $D_2'$. The first tree statement is straightforward by the assumption that $D_2'$ is session-typed. Using the local transition steps, we get $D_2' \text{msg}(\text{send} x_\beta^\alpha y_\alpha) \rightarrow^*\ D_2' \text{msg}(\text{send} x_\beta^\alpha y_\alpha)$. We can apply the result of part (1) of this lemma on $(\dagger)$ to get

$$(\dagger') \forall x_\beta \notin \text{dom}(\Delta, K). \quad (D_1 \text{msg}(\text{send} x_\beta^\alpha y_\alpha); D_2' \text{msg}(\text{send} x_\beta^\alpha y_\alpha)) \in \mathcal{E}_\Phi[\Delta, x_\beta : A \vdash y_\alpha+1 : B]^{m+1}$$

which completes the proof of this case.

Row 9. By the conditions of this row, we know that $K = \Delta'$. We have $\forall x_\beta \notin \text{dom}(\Delta; y_\alpha : A \otimes B, K). \quad (D_1; D_2) \in \text{Tree}_\Phi(\Delta', y_\alpha : A \otimes B \vdash K)$ and

$$(\dagger) \forall x_\beta \notin \text{dom}(\Delta; y_\alpha : A \otimes B, K). \quad (D_1 \text{msg}(\text{send} x_\beta^\alpha y_\alpha); D_2 \text{msg}(\text{send} x_\beta^\alpha y_\alpha)) \in \mathcal{E}_\Phi[\Delta', x_\beta : A, y_\alpha+1 : B \vdash K]^{m+1}$$

These are also the statements that we need to prove when replacing $D_2$ with $D_2'$. The first tree statement is straightforward by the assumption that $D_2'$ is session-typed. Using the local dynamic steps we get $\text{msg}(\text{send} x_\beta^\alpha y_\alpha)D_2' \rightarrow^*\ \text{msg}(\text{send} x_\beta^\alpha y_\alpha)D_2$. We apply the result of part (1) of this lemma on $(\dagger)$ to get

$$(\dagger') \forall x_\beta \notin \text{dom}(\Delta; y_\alpha : A \otimes B, K). \quad (D_1 \text{msg}(\text{send} x_\beta^\alpha y_\alpha); D_2' \text{msg}(\text{send} x_\beta^\alpha y_\alpha)) \in \mathcal{E}_\Phi[\Delta', x_\beta : A, y_\alpha+1 : B \vdash K]^{m+1}$$

C. Forward closure

Lemma 14 (Forward closure on the first run). Consider $(D_1; D_2) \in \mathcal{E}[\Delta \vdash K]^{m+1}$ and $D_1 \rightarrow^* D_1'$. We have $(D_1'; D_2) \in \mathcal{E}[\Delta \vdash K]^{m+1}$.

Proof. If $k = 0$ the proof is trivial. Consider $k = m+1$. By assumption, we have $(D_1; D_2) \in \mathcal{E}[\Delta \vdash K]^{m+1}$.

By line (12) of the logical relation we get

\begin{align*}
\forall x_\alpha \in \text{Out}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \text{ then } (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{In}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \text{ then } (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{Out}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \text{ then } (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{In}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \text{ then } (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1}
\end{align*}

Consider $D_1'$, for which the assumption we have $D_1 \rightarrow^* D_1'$. Our goal is to prove $(D_1'; D_2) \in \mathcal{E}[\Delta \vdash K]^{m+1}$. We need to show that

\begin{align*}
\forall x_\alpha \in \text{Out}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{In}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{Out}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{In}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1}
\end{align*}

With a $\forall$-Introduction and an if-Introduction on the goal, we assume $D_2'' \rightarrow^* y_1 : \Theta_1 D_1'$. By assumption of $D_1 \rightarrow^* D_1'$ we get $D_1 \rightarrow^* y_1 : \Theta_1 D_1'$. We use this to apply $\forall$-Elimination and if- Elimination on the assumption, and get

\begin{align*}
\forall x_\alpha \in \text{Out}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{In}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{Out}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1} \\
\forall x_\alpha \in \text{In}(\Delta \vdash K). & \quad \text{if } x_\alpha \in \Theta_1 \therefore (D_1; D_2) \in \mathcal{V}_\Phi[\Delta \vdash K]^{m+1}
\end{align*}

Which exactly matches our goal and the proof is complete. \qed

Lemma 15 (Forward closure on the second run with some specific conditions). Consider $(D_1; D_2) \in \mathcal{E}[\Delta \vdash K]^{k}$ and $D_2 \rightarrow^* D_2''$ such that if $D_1$ sends along the set $\Theta_1$, we have $\Theta_1 \subseteq \Theta$ and also $D_2''$ is the minimal configuration built by \text{Lemma} 12 given the set $\Theta_1$ and configuration $D_2$. We have $(D_1; D_2'') \in \mathcal{E}[\Delta \vdash K]^{k}$.

Proof. If $k = 0$ the proof is trivial. Consider $k = m+1$. By assumption, we have $(D_1; D_2) \in \mathcal{E}[\Delta \vdash K]^{m+1}$. By line (12) of the logical relation we get
Consider $D''_2$, for which by the assumption we have $D_2 \mapsto^{*\exists} D''_2$. Our goal is to prove $(D_1; D''_2) \in E[\Delta \vdash K]^{m+1}$. We need to show that

\[
\forall \alpha, \in V_{\Delta}^K[\Delta \vdash K]^{m+1}.
\]

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

\[
\forall x, \in V_{\Delta}^K[\Delta \vdash K]^{m+1}.
\]

and the goal

\[
\forall x, \in V_{\Delta}^K[\Delta \vdash K]^{m+1}.
\]

We apply an $\exists$-Elimination on the assumption to get $D''_2$ that satisfies the conditions, i.e., $D_2 \mapsto^{*\exists} D''_2$ and $\forall x, \in V_{\Delta}^K[\Delta \vdash K]^{m+1}$. We use the same $D''_2$ to instantiate the existential quantifier in the goal, we need to show that $D''_2 \mapsto^{*\exists} D''_2$. Since $D''_2$ is the minimal configuration built for $\forall x, \in V_{\Delta}^K[\Delta \vdash K]^{m+1}$, by Lemma [12] we get $D''_2 \mapsto^{*\exists} D''_2$, and the proof is complete. □
X. Moving existential and compositionality

This section introduces two lemmas, [Lem. 16] and [Corollary 2], which are instrumental in proving transitivity (see §XI), and adequacy (see §XII).

A. Moving existential over universal quantifier

**Lem. 16** (Moving existential over universal quantifier). If we have

\[(\dagger) \forall m. \forall \Gamma_1, \Theta_1, D_1', \text{ if } D_1 \xrightarrow{\forall \xi; \eta_1} D_1', \text{ then } \exists \Gamma_2, D_2' \text{ such that } D_2 \xrightarrow{\exists \xi; \eta_2} D_2' \text{ and } \Theta_1 \subseteq \Gamma_2 \text{ and } \\forall x_\alpha \in \text{Out}(\Delta \vdash K), \text{ if } x_\alpha \in \Gamma_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{m+1} \text{ and } \\forall x_\alpha \in \text{In}(\Delta \vdash K), \text{ if } x_\alpha \in \Theta_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{m+1}.
\]

then

\[\forall \Gamma_1, \Theta_1, D_1', \text{ if } D_1 \xrightarrow{\forall \xi; \eta_1} D_1', \text{ then } \exists \Gamma_2, D_2' \text{ such that } D_2 \xrightarrow{\exists \xi; \eta_2} D_2' \text{ and } \Theta_1 \subseteq \Gamma_2 \text{ and } \\forall x_\alpha \in \text{Out}(\Delta \vdash K), \text{ if } x_\alpha \in \Gamma_1, \text{ then } \forall k. (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1} \text{ and } \\forall x_\alpha \in \text{In}(\Delta \vdash K), \text{ if } x_\alpha \in \Theta_1, \text{ then } \forall k. (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1}.
\]

**Proof.** First put \(m = 1\) to apply \(\forall E\). on the assumption (instantiating \(\forall m\) only). We get as an assumption

\[\forall \Gamma_1, \Theta_1, D_1', \text{ if } D_1 \xrightarrow{\forall \xi; \eta_1} D_1', \text{ then } \exists \Gamma_2, D_2' \text{ such that } D_2 \xrightarrow{\exists \xi; \eta_2} D_2' \text{ and } \Theta_1 \subseteq \Gamma_2 \text{ and } \\forall x_\alpha \in \text{Out}(\Delta \vdash K), \text{ if } x_\alpha \in \Gamma_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{0+1} \text{ and } \\forall x_\alpha \in \text{In}(\Delta \vdash K), \text{ if } x_\alpha \in \Theta_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{0+1}.
\]

Next, apply a \(\forall I\), and if \(I\) on the goal followed by a corresponding \(\exists E\). and if \(E\). on the assumption (\(\dagger\)). Now apply an \(\exists E\). on the assumption (\(\dagger\)) to get \(D_2\) such that \(D_2 \xrightarrow{\forall \xi; \eta_2} D_2'\) and \(\Gamma_1 \subseteq \Gamma_2\). Given \(D_2', \text{ by } \text{Lem. 12}\), we can build the minimal \(D_2''\) such that \(D_2 \xrightarrow{\forall \xi; \eta_2} D_2''\) and \(\Gamma_1 \subseteq \Gamma_2\). Moreover, we know that for every \(D\) such that \(D_2 \xrightarrow{\exists \xi; \eta_2} D\) and \(\Theta_1 \subseteq \Theta_2\), we get

\[\forall x_\alpha \in \text{Out}(\Delta \vdash K), \text{ if } x_\alpha \in \Gamma_1, \text{ then } (D_1'; D_2'') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1} \text{ and } \\forall x_\alpha \in \text{In}(\Delta \vdash K), \text{ if } x_\alpha \in \Theta_1, \text{ then } (D_1'; D_2'') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1}.
\]

We use this minimal \(D_2''\) to instantiate the existential (\(\exists I\)) in the goal, and use \(\forall I\) on the goal. In particular, we instantiate \(k\) with an arbitrary natural number. The goals are:

\[(D_1'; D_2'') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1} \text{ and } (D_1'; D_2'') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1}.
\]

Next, we instantiate the quantifiers in the original assumption (\(\dagger\)) once again, this time with \(m = k\) followed by a \(\forall E\), and if \(E\). instantiating the quantifiers with similar \(D_1', \Gamma_1, \text{ and } \Theta_1\) as the first time. We get as an assumption:

\[\exists \Gamma_2, D_2' \text{ such that } D_2 \xrightarrow{\exists \xi; \eta_2} D_2' \text{ and } \Theta_1 \subseteq \Gamma_2 \text{ and } \forall x_\alpha \in \text{Out}(\Delta \vdash K), \text{ if } x_\alpha \in \Gamma_1, \text{ then } (D_1'; D_2) \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{0+1} \text{ and } \forall x_\alpha \in \text{In}(\Delta \vdash K), \text{ if } x_\alpha \in \Theta_1, \text{ then } (D_1'; D_2) \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{0+1}.
\]

Next, apply \(\exists I\) to get a \(\forall'\) and \(D'\) that satisfies the conditions, i.e., \(\Delta \xrightarrow{\forall' \eta'} \Delta'\) and \(\Theta_1 \subseteq \Gamma_1\). Instantiate \(x_\alpha\) as those chosen for the goal. We have as assumptions:

\[\forall \forall', \exists \Gamma_2, D_2' \text{ such that } D_2 \xrightarrow{\forall \xi; \eta_2} D_2' \text{ and } \Theta_1 \subseteq \Gamma_2 \text{ and } \forall x_\alpha \in \text{Out}(\Delta \vdash K), \text{ if } x_\alpha \in \Gamma_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1} \text{ and } \forall x_\alpha \in \text{In}(\Delta \vdash K), \text{ if } x_\alpha \in \Theta_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \vdash K]_{x_\alpha}^{k+1}.
\]

Since \(D_2''\) is the minimal configuration built for \(\Gamma_1\) and \(D_2\), we know that \(D_2'' \xrightarrow{\forall \xi; \eta_2} D_2'\). We can apply the backward closure results of [Lem. 13] to get the goal from the assumptions (\(\dagger'\)), and this completes the proof.

\[\square\]

B. Compositionality

**Corollary 2** (Compositionality). \(\forall m. (D_1; D_2) \in \mathcal{E}[\Delta, u_\alpha; T \vdash K]_m \iff \forall \forall'. (\exists \forall', D_2') \in \mathcal{E}[\Delta \vdash K]_m \forall \forall'. \exists \forall', D_2' \in \mathcal{E}[\Delta \vdash K]_m \forall \forall'.
\]

**Proof.** The left to right direction is a corollary of Lemma [17] in which we compose multiple configurations instead of just two. For the right to left direction, we put \(\forall' = \ldots\), and \(\Delta' = u_\alpha; T\), and the rest of the proof is straightforward.

\[\square\]

**Lem. 17** (Generalized compositionality). For \(i \in \{1, 2\}\), and index set \(I\) consider session typed tree-shaped configurations \(\Delta^n \vdash \mathcal{B}_i^n \vdash K^n\) such that \(n \in I\) and their compositions form session typed tree-shaped configurations \(\Delta \vdash \mathcal{B}_i \vdash K\), i.e., \(\mathcal{B}_i = \{\mathcal{B}_i^n\}_{n \in I}\). If for all \(n \in I\), we have \(\forall m. (\mathcal{B}_1^n; \mathcal{B}_2^n) \in \mathcal{E}[\Delta^n \vdash K^n]_m\) then \(\forall k. (D_1; D_2) \in \mathcal{E}[\Delta \vdash K]_k\).

**Proof.** Our goal is to prove the following:

For all index set \(I\) and all session-typed configurations \(\Delta^n \vdash \mathcal{B}_i^n \vdash K^n\) with \(n \in I\) such that \(\forall m. (\mathcal{B}_1^n; \mathcal{B}_2^n) \in \mathcal{E}[\Delta^n \vdash K^n]_m\) we have \(\forall k. (\{\mathcal{B}_1^n\}_{n \in I}; \{\mathcal{B}_2^n\}_{n \in I}) \in \mathcal{E}[\Delta \vdash K]_k\).

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 \( \Delta^n \vdash B^n_1 : K^n \) such that 
\[ \forall n. (B^n_1; B^n_2) \in \mathcal{E}[\Delta^n \vdash K^n]^m \]
we have \( \forall x. (\{B^n_1\}_n \in I; \{B^n_2\}_n \in I) \in \mathcal{E}[\Delta \vdash K]^k \).

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 
\[ \Delta^n \vdash B^n_1 : K^n \] 
with \( \Delta \) and \( \{B^n_1\}_n \in I : K^n \), we have \( \forall x. (\{B^n_1\}_n \in I; \{B^n_2\}_n \in I) \in \mathcal{E}[\Delta \vdash K]^0 \).

**Inductive case (\( k + 1 \)).** Our goal is to prove the following:
For all index set \( I \), and for all session-typed configurations \( \Delta^n \vdash B^n_1 : K^n \) where \( n \in I \) such that 
\[ \forall n. (B^n_1; B^n_2) \in \mathcal{E}[\Delta^n \vdash K^n]^m \] 
we have \( \forall x. (\{B^n_1\}_n \in I; \{B^n_2\}_n \in I) \in \mathcal{E}[\Delta \vdash K]^{k+1} \). 

where \( x \) is defined in line (12) of the logical relation as
\[ \forall \gamma \in \Omega(x \in \Delta \vdash K); \text{if } x \in \Theta_1 \text{, then } (D^n_1; D^n_2) \in \mathcal{V}[\Delta \vdash K]^{k+1} \] 

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 \( \Delta^n \vdash B^n_1 : K^n \) with \( n \in I \) such that 
\[ \forall n. (B^n_1; B^n_2) \in \mathcal{E}[\Delta^n \vdash K^n]^m \] 
we have
\[ \forall \gamma \in \Omega(x \in \Delta \vdash K); \text{if } x \in \Theta_1 \text{, then } (D^n_1; D^n_2) \in \mathcal{V}[\Delta \vdash K]^{k+1} \] 

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 \( \Delta^n \vdash B^n_1 : K^n \) that satisfy the conditions \( \forall n \).
We need to show that
\[ \forall \gamma \in \Omega(x \in \Delta \vdash K); \text{if } x \in \Theta_1 \text{, then } (D^n_1; D^n_2) \in \mathcal{V}[\Delta \vdash K]^{k+1} \] 

Consider an arbitrary \( \gamma, \Theta_1, \) ans \( D^n_1 \), and apply If, on the goal. By the assumption \( (B^n_1) \in I \) we know that 
\( \forall n. (B^n_1; B^n_2) \in \mathcal{E}[\Delta^n \vdash K^n]^m \) and \( \forall n. (B^n_1; B^n_2) \in \mathcal{E}[\Delta \vdash K]^{k+1} \). 

By the definition of the logical relation, and from assumptions \( \forall n \) for \( n \in I \) we get
\[ \forall \gamma \in \Omega(x \in \Delta \vdash K); \text{if } x \in \Theta_1 \text{, then } (B^n_1; B^n_2) \in \mathcal{V}[\Delta \vdash K]^{k+1} \] 

We instantiate the forall quantifier in \( \forall n \) by \( B^n_1 \) and the sets \( \gamma \) and \( \Theta_1 \) along which \( B^n_1 \) sends and receives. Note that by definition we have \( \forall n. (B^n_1; B^n_2) \in \mathcal{E}[\Delta \vdash K]^{k+1} \). 

By existential elimination, for all \( n \in I \), we get a \( B^n_2 \) and \( \gamma \) such that \( B^n_2 \vdash K^n \), and \( \forall n \in I \) and 
\[ \forall x. (\{B^n_1\}_n \in I; \{B^n_2\}_n \in I) \in \mathcal{E}[\Delta \vdash K]^{k+1} \] 

Note that by definition $\text{Out}(\Delta \parallel K) \subseteq \bigcup_{n \in I} \text{Out}(\Delta^n \parallel K^n)$ and $\text{In}(\Delta \parallel K) \subseteq \bigcup_{n \in I} \text{In}(\Delta^n \parallel K^n)$.

We apply Lemma 12 to get the minimal sending configurations $B_2^\prime$ and set $\Upsilon_n$ for each given $\Upsilon_n$. We get $B_2^\prime$ such that $B_2^\prime \Rightarrow \gamma x_2 B_2^\prime$. Since these configurations are minimal, for all $n \in I$, we have $B_2^\prime \Rightarrow \gamma x_2 B_2^\prime$.

Since $\Upsilon_n \subseteq \Upsilon_{n+1}$, we can apply the backward closure [Lemma 13] on $\downarrow x_2^\prime$ to get

$$\downarrow x_2^\prime \forall x_2 \in \text{Out}(\Delta^n \parallel K^n).$$

Moreover, we apply forward closure on the second run [Lemma 15] on assumptions $\uparrow x_2^\prime$ to get for all $n \in I$:

$$\forall x_2 \in \text{Out}(\Delta^n \parallel K^n).$$

We can apply the forward closure on the second run since the conditions of Lemma 15 are satisfied, i.e. $\Upsilon_n \subseteq \Upsilon_{n+1}$ and $B_2^\prime$ is the minimal sending configuration with respect to $B_2^\prime$ and $\Upsilon_n$.

We build $D_2^\prime$ to be $\{B_2^\prime\}_{n \in I}$. We have $\{B_2^\prime\}_{n \in I} \Rightarrow \gamma x_2 \{B_2^\prime\}_{n \in I}$ with $\Upsilon_1 \subseteq \Upsilon_2$, and $\{B_2^\prime\}_{n \in I} \Rightarrow \gamma \Upsilon 2 \{B_2^\prime\}_{n \in I}$ with $\Upsilon_2 \subseteq \Upsilon_3$. We, then, instantiate the existential quantifier in the goal $(\ast)$ with $\Upsilon_2$ and $D_2^\prime$ that we built for which we know $\{B_2^\prime\}_{n \in I} \Rightarrow \gamma x_2 D_2^\prime$. We need to show

$$\forall x_2 \in \text{Out}(\Delta^n \parallel K^n).$$

Part 1. Consider arbitrary $x_2 \in \text{Out}(\Delta^n \parallel K^n)$ and assume $x_2 \in \Upsilon_1$. By the structure of the configurations, for some $n \in I$, $x_2 \in \Delta^n, K^n$ and thus $x_2 \in \text{Out}(\Delta^n \parallel K^n)$ and $x_2 \in \Upsilon_n$. The goal is to prove

$$\forall x_2 \in \text{Out}(\Delta^n \parallel K^n).$$

Part 2. Consider arbitrary $x_2 \in \text{In}(\Delta \parallel K^n)$ and assume $x_2 \in \Upsilon_1$. By the structure of the configurations, for some $n \in I$, $x_2 \in \Delta^n, K^n$ and thus $x_2 \in \text{In}(\Delta^n \parallel K^n)$ and $x_2 \in \Upsilon_n$. The goal is to prove

$$\forall x_2 \in \text{In}(\Delta \parallel K^n).$$

In both parts, we continue the proof by considering the type of $x_2$. The type of $x_2$ 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.** $x_2 : A \otimes B \in K$. This case corresponds to Part 1 of the goal in which we have $x_2 \in \text{Out}(\Delta \parallel x_2 : A \otimes B)$ and $x_2 \in \Upsilon_1$. By the structure of the configuration, there exists a tree $B_2^\gamma$ that provides the root channel $K = K^\gamma = x_2 : A \otimes B$.

We use assumption $\downarrow x_2^\prime$ for that specific channel $(\downarrow x_2^\prime)$, we have

$$\downarrow x_2^\prime \forall m. (B_2^\prime ; B_2^\prime) \in \text{Out}(\Delta^m \parallel K^m)$$

First, instantiate the forall quantifier with $m = 0$. By Row 4, of the logical relation, we have $\Delta^\gamma = \Delta^\gamma, \Delta^\gamma$ and for some $y_\beta \in \text{chnl}$, we have: $B_1^\prime = B_1^\gamma A_1 msg(send y_\beta x_\gamma)$ and $B_2^\prime = B_2^\gamma A_2 msg(send y_\beta x_\gamma)$.

We want to prove

$$\exists! m. (A_1, A_2) \in \text{Out}(\Delta^m \parallel \gamma y_\beta : A^m) \ \text{and} \ \exists! m. (B_1^\gamma, B_2^\gamma) \in \text{Out}(\Delta^m \parallel \gamma x_\gamma + 1 : B^m).$$

Consider an arbitrary $m$ given by $\forall I$ on the goals $\exists! m. (A_1, A_2)$, $\exists! m. (B_1^\gamma, B_2^\gamma)$, once again, instantiate the quantifier in $\downarrow x_2^\prime$, this time with the arbitrary $m$. Again, we get $\Delta^\gamma = \Delta^\gamma, \Delta^\gamma$ and for some $y_\beta \in \text{chnl}$, we have: $B_1^\prime = B_1^\gamma A_1 msg(send y_\beta x_\gamma)$ and $B_2^\prime = B_2^\gamma A_2 msg(send y_\beta x_\gamma)$. Moreover, $\exists! m. (A_1, A_2) \in \text{Out}(\Delta^m \parallel \gamma y_\beta : A^m)$ and $\exists! m. (B_1^\gamma, B_2^\gamma) \in \text{Out}(\Delta^m \parallel \gamma x_\gamma + 1 : B^m)$.

Since the namings in the configuration is unique, we get the above for the same $y_\beta$ as we got in the case of $m = 0$ and the proof of $\exists! m.$ and $\exists! m.$ is complete.

From this we can prove

$$D_1 = \{B_1^\prime\}_{n \in I} = \{B_1^n\}_{n \in I \land n \neq \kappa} B_1^\gamma A_1 msg(send y_\beta x_\gamma)$$

and

$$D_2 = \{B_2^\prime\}_{n \in I} = \{B_2^n\}_{n \in I \land n \neq \kappa} B_2^\gamma A_2 msg(send y_\beta x_\gamma).$$

First, observe that by the structure of the configuration, there is no tree $B_1^\gamma$ or $B_2^\gamma$ using $K^\gamma = x_\gamma : A \otimes B$ as its resource, i.e., $x_\gamma$ is the root. We can break down the resources $\Delta^\gamma$ and $\Delta^\gamma$ as $\Delta^\gamma = \Delta^\gamma, \Delta^\gamma$ and $\Delta^\gamma = \Delta^\gamma, \Delta^\gamma$, such that $\Delta^\gamma$ and $\Delta^\gamma$ are in the interface of $D_1$ and $D_2$ and $\Delta^\gamma$ and $\Delta^\gamma$ are the resources provided by other trees. We can partition $I \setminus \{\kappa\}$ into two disjoint sets $I_1$ and $I_2$ such that the configurations $\{B_1^\prime\}_{n \in I_1}$ and $\{B_2^\prime\}_{n \in I_2}$ provide the resources in $\Delta^\gamma$.
and configurations \( \{B^n_1\}_{n \in I_1} \) and \( \{B''^n_2\}_{n \in I_2} \) provide the resources in \( \Delta''_2 \). In other words, we have \( \Delta = \Delta_1, \Delta'_1, \Delta_2, \Delta'_2 \) and \( K = x_\gamma; A \otimes B \) and

\[
\begin{align*}
  (i) \quad & \Delta_1 \vdash \{B^n_1\}_{n \in I_1} :: \Delta''_1 \\
  & \Delta_1 \vdash \{B''^n_2\}_{n \in I_2} :: \Delta''_2 \\
  & \Delta_1 \vdash \{B''^n_2\}_{n \in I_2} :: \Delta''_1 \vdash x_{\gamma+1}; B \\
  & \Delta_1 \vdash \{B'_1\}_{n \in I_1} \vdash x_{\gamma+1}; B \\
  & \Delta_1 \vdash \{B''_2\}_{n \in I_2} :: \Delta''_1 \vdash x_{\gamma+1}; B \\
  (ii) \quad & \Delta_2 \vdash \{B^n_1\}_{n \in I_1} :: \Delta''_2 \\
  & \Delta_2 \vdash \{B''^n_2\}_{n \in I_2} :: \Delta''_2 \\
  & \Delta_2 \vdash \{B''^n_2\}_{n \in I_2} :: \Delta''_2 \vdash A :: y_\beta; A \\
  & \Delta_2 \vdash \{B'_1\}_{n \in I_1} \vdash A :: y_\beta; A \\
  & \Delta_2 \vdash \{B''_2\}_{n \in I_2} :: \Delta''_2 \vdash A :: y_\beta; A
\end{align*}
\]

We also have

\[
\begin{align*}
  o'' \quad & D_1 = \{B^n_1\}_{n \in I_1} = \{B^n_1\}_{n \in I_1} \in \mathbb{L} B(y_{\beta}; A)A_{1}\text{msg}(\text{send} y_{\beta}; x_{\gamma}) = \{B^n_1\}_{n \in I_1} \{B^n_2\}_{n \in I_2} \{B''^n_2\}_{n \in I_2} A_1 \text{msg}(\text{send} y_{\beta}; x_{\gamma}) \}
\end{align*}
\]

Recall that earlier we proved

\[
\begin{align*}
  o' \forall m. (A_1, A_2) \in \mathcal{E}[\Delta''_2 \vdash x_{\gamma+1}; B] \\
  \text{and also we have for all } n \in I_1 \cup I_2, \quad o_2 \forall m. (B''^n_2) \in \mathcal{E}[\Delta''_2 \vdash x_{\gamma+1}; B]
\end{align*}
\]

We can apply the induction hypothesis on the smaller index \( k' \), (i), \( o'' \), and \( o_2 \) for those trees indexed in \( I_1 \) to get

\[
\begin{align*}
  o_3 \forall m. (B''^n_2) \in \mathcal{E}[\Delta''_2 \vdash x_{\gamma+1}; B]
\end{align*}
\]

Similarly, we can apply the induction hypothesis on the smaller index \( k' \), (ii), \( o' \), and \( o_2 \) for those trees indexed in \( I_2 \) to get

\[
\begin{align*}
  o_3 \forall m. (B''^n_2) \in \mathcal{E}[\Delta''_2 \vdash x_{\gamma+1}; B]
\end{align*}
\]

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

\[
\begin{align*}
  *\{D_1; D'_2\} \in \mathcal{V}[\Delta \vdash K]^{k''+1}
\end{align*}
\]

Subcase 2. \( x_\gamma; A \otimes B \in \Delta \), i.e., \( \Delta = \Delta' \). This case corresponds to Part 2 of the goal in which we have \( x_\gamma \in \text{In}(\Delta \vdash K) \) and \( x_\gamma \in \Theta_1 \), and the goal is to prove \( \{B''^n_2\}_{n \in I_2} \in \mathcal{V}[\Delta \vdash K]^{k''+1} \).

By the structure of the configuration, for some index \( \kappa \in I \), we have \( x_\gamma; A \otimes B \in \Delta', \text{ i.e. } \Delta' = \Delta'' \). There exist \( *\{D_1; D'_2\} \in \mathcal{V}[\Delta' \vdash K]^{m+1} \).

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

\[
\begin{align*}
  o'' \forall m. (B''^n_2) \in \mathcal{E}[\Delta'' \vdash x_{\gamma+1}; A \otimes B] \vdash K^m
\end{align*}
\]

Moreover, we have

\[
\begin{align*}
  \Delta, y_\beta; A, x_\gamma; B \vdash \text{msg}(y_\beta; x_\gamma) \{B'_1\}_{n \in I_1} :: K \\
  \Delta, y_\beta; A, x_\gamma; B \vdash \text{msg}(y_\beta; x_\gamma) \{B''^n_2\}_{n \in I_2} :: K
\end{align*}
\]

Recall that for all \( n \in I \setminus \{\kappa\} \) we also have

\[
\begin{align*}
  o_2 \forall m. (B''^n_2) \in \mathcal{E}[\Delta'' \vdash K^m]
\end{align*}
\]

We can apply the induction hypothesis on the smaller index \( k' \), \( o_2 \), and \( o'' \) to get

\[
\begin{align*}
  \forall m. (B''^n_2) \in \mathcal{E}[\Delta'' \vdash K^m]
\end{align*}
\]

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

Inductive case \( j' = j' + 1 \). Consider an arbitrary index set \( J \) and an arbitrary session-typed configurations \( \Delta' \vdash K' \vdash K^n \) that satisfy the conditions \( \bar{\tau} \). We need to show that

\[
\begin{align*}
  \forall \Theta_1, \Theta_2, D'_1, \text{ if } \{B'_1\}_{n \in I} \mapsto_{\tau_1 \Theta_1, \Theta_2} D'_1 \text{ then } \exists \Theta_2, D'_2 \text{ such that } \{B''^n_2\}_{n \in I} \mapsto_{\tau_2 \Theta_2} D'_2 \text{ and } \Theta_1 \subseteq \Theta_2 \text{ and } \\
  \forall x_{\gamma} \in \text{Out}(\Delta \vdash K) \text{ if } x_{\gamma} \in \Theta_1 \text{ then } \{D'_1; D'_2\} \in \mathcal{V}[\Delta \vdash K]^m \text{ and } \\
  \forall x_{\gamma} \in \text{In}(\Delta \vdash K) \text{ if } x_{\gamma} \in \Theta_1 \text{ then } \{D'_1; D'_2\} \in \mathcal{V}[\Delta \vdash K]^k.
\end{align*}
\]

We apply a \( \forall \Theta_1 \) and if \( \Theta_1 \). on the goal: consider an arbitrary \( \Theta_1, \Theta_2, \) and \( D'_1 \) and the first step of \( \{B'_1\}_{n \in I} \mapsto_{\tau_1 \Theta_1, \Theta_2} D'_1 \).

There are two cases to consider:

Case 1. the first step is an internal step but not a communication between the sub-trees.
Without loss of generality, let’s assume that the communication is internal to $B^n_1$ for some $k \in I$ and all other trees $B^n_i$ for $n \neq \kappa$ remain intact, i.e. we have $\{B^n_1\}_{n \in I} = \{B^n_i\}_{n \in I} = \{B^n_i\}_{n \in I}$ and

$$\{B^n_1\}_{n \in t_1} B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \{B^n_1\}_{n \in t_1}, B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \gamma_1. \Phi_1 \cdot D_1'. \quad \text{By forward closure (Lem. 14) on the assumption } \gamma_1 (i.e., \gamma_1 \text{ for } n = \kappa), \text{ we get}$$

$$\gamma_1 \forall m. (B^n_1; B^n_2) \in \mathcal{E}[\Delta^n \models K^n]^m.$$ 

Recall that by $\gamma_1$, we also have for $n \neq \kappa$

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

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

$$\forall Y_1, \Theta_1, D_1'. \forall Y_1. \text{ if } \{B^n_1\}_{n \in t_1} B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \{B^n_1\}_{n \in t_1}, D_1' \text{ then } \exists Y_2 D_2'. \{B^n_2\}_{n \in t} \Rightarrow \gamma_2 D_2 \text{ and } Y_1 \subseteq Y_2 \text{ and}$$

$$\forall x, y \in \text{Out}(\Delta \models K). \text{ if } x, y \in Y_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \models K]^x_{y+1} \text{ and}$$

$$\forall x, y \in \text{In}(\Delta \models K). \text{ if } x, y \in Y_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \models K]^x_{y+1}.$$ 

Since $\{B^n_1\}_{n \in t_1} B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \{B^n_1\}_{n \in t_1}, B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \gamma_1. \Phi_1 \cdot D_1'$, we can prove the goal:

$$\exists Y_2, \Theta_2 \text{ such that } \{B^n_2\}_{n \in t} \Rightarrow \gamma_2 D_2 \text{ and } Y_1 \subseteq Y_2 \text{ and}$$

$$\forall x, y \in \text{Out}(\Delta \models K). \text{ if } x, y \in Y_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \models K]^x_{y+1} \text{ and}$$

$$\forall x, y \in \text{In}(\Delta \models K). \text{ if } x, y \in Y_1, \text{ then } (D_1'; D_2') \in \mathcal{V}[\Delta \models K]^x_{y+1}.$$ 

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., $B^n_1$ offers a resource $u_1; T$ and $B^n_2$ uses the resource, and there is a message available along $u_1$ that is received in the first step. The proof proceeds by case analysis on type of $u_1; T$. 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.,

$$\begin{align*}
(i) & \quad \Delta^\kappa \vdash B^n_1 :: K^\kappa & \text{ where } K^\kappa = u_1; A \otimes B \\
(ii) & \quad \Delta^\lambda \vdash B^n_2 :: K^\lambda & \text{ where } K^\lambda = \Delta^\lambda, u_1; A \otimes B \\
& \quad \Delta^\lambda \vdash B^n_2 :: K^\lambda \\
\end{align*}$$

By the assumptions of this case, we know that there is a message sent along $u_1; A \otimes B$ in $B^n_1$ that is received by a process in $B^n_1$, i.e., $B^n_1 = B^n_1 A_1 \text{msg}(\text{send}_y u_1)$ and

$$\begin{align*}
\{B^n_1\}_{n \in t} = \{B^n_1\}_{n \in t_1} B^n_1 \{B^n_1\}_{n \in t_2} = \{B^n_1\}_{n \in t_1}, B^n_1, A_1 \text{msg}(\text{send}_y u_1) B^n_1 \{B^n_1\}_{n \in t_2} \text{ and} \\
\{B^n_1\}_{n \in t_1}, B^n_1, A_1 \text{msg}(\text{send}_y u_1) B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \{B^n_1\}_{n \in t_1}, B^n_1, A_1 \text{msg}(\text{send}_y u_1) B^n_1 \{B^n_1\}_{n \in t_2} \Rightarrow \gamma_1. \Phi_1 \cdot D_1'. \end{align*}$$

By $\gamma_1$, Lem. 16 and $B^n_1, A_1 \text{msg}(\text{send}_y u_1) \Rightarrow \gamma_1 B^n_1, A_1 \text{msg}(\text{send}_y u_1)$, we get

$$\exists \gamma_2, B^n_2' \text{ such that } B^n_2 \Rightarrow \gamma_2 B^n_2' \text{ and } \gamma_2 \subseteq \gamma_2.$$ 

$$\forall x \in \text{Out}(\Delta \models u_1; A \otimes B), \text{ if } x \in \gamma_2, \text{ then } \forall m. (B^n_1, A_1 \text{msg}(\text{send}_y u_1); B^n_2') \in \mathcal{V}[\Delta^\kappa \models u_1; A \otimes B^x_{m+1}] \text{ and}$$

$$\forall x \in \text{In}(\Delta \models u_1; A \otimes B). \text{ if } x \in \gamma_2, \text{ then } \forall m. (B^n_1, A_1 \text{msg}(\text{send}_y u_1); B^n_2') \in \mathcal{V}[\Delta^\kappa \models u_1; A \otimes B^x_{m+1}]$$

In particular, we know that $u_1; A \otimes B \in \text{Out}(\Delta \models u_1; A \otimes B)$ and $u_1 \in \gamma_2$, which gives us:

$$\forall m. (B^n_1, A_1 \text{msg}(\text{send}_y u_1); B^n_2') \in \mathcal{V}[\Delta^\kappa \models u_1; A \otimes B^x_{m+1}]$$

By Row 4. of the logical relation, we have $\Delta^\kappa = \Delta^\kappa_1, \Delta^\kappa_2$ and

$$\begin{align*}
& \quad B^n_1 = B^n_1 A_1 \text{msg}(\text{send}_y u_1), \text{ and} \\
& \quad \forall m. (A_1; A_2) \in \mathcal{E}[\Delta^\kappa \models y_1: y_2]^m \text{ and } \forall m. (B^n_1; B^n_2) \in \mathcal{E}[\Delta^\kappa \models u_1; A \otimes B]^m. 
\end{align*}$$

Next, we consider $B^n_1$. By $\gamma_1$, Lem. 16 and $B^n_1 \Rightarrow \gamma_1 B^n_1$, we get

$$\exists \gamma_2, B^n_2' \text{ such that } B^n_2 \Rightarrow \gamma_2 B^n_2' \text{ and}$$

$$\forall x \in \text{Out}(\Delta^\lambda \models u_1; A \otimes B \models K^\lambda), \text{ if } x \in \gamma_2, \text{ then } \forall m. (B^n_1; B^n_2) \in \mathcal{V}[\Delta^\lambda \models u_1; A \otimes B \models K^\lambda]^x_{m+1} \text{ and}$$

$$\forall x \in \text{In}(\Delta^\lambda \models u_1; A \otimes B \models K^\lambda). \text{ if } x \in \gamma_2, \text{ then } \forall m. (B^n_1; B^n_2) \in \mathcal{V}[\Delta^\lambda \models u_1; A \otimes B \models K^\lambda]^x_{m+1}.$$ 

40
In particular, we know that $u_\alpha : A \otimes B \in \text{In}(\Delta^\lambda, u_\alpha : A \otimes B \vdash K^\lambda)$ and also $u_\alpha \in \Theta^\lambda_1$, which gives us:

$$\forall m. (B_1^\lambda; B_2^\lambda) \in V[\Delta^\lambda, u_\alpha : A \otimes B \vdash K^\lambda]_{u_\alpha}^{m+1}.$$ 

By Row 9, of the logical relation for the specific channel $y_\beta$ (for which by the tree structure, we know $y_\beta \notin \Delta^\lambda, u_\alpha : A \otimes B, K^\lambda$) we have

$$\forall m. (\text{msg}(\text{send}_{y_\beta} u_\alpha)B_1^\lambda; \text{msg}(\text{send}_{y_\beta} u_\alpha)B_2^\lambda) \in E[\Delta^\lambda, y_\beta : A, u_{\alpha+1} : B \vdash K^\lambda]_m.$$ 

By forward closure [Lem. 14] and $\text{msg}(\text{send}_{y_\beta} u_\alpha)B_1^\lambda \mapsto B_1^\lambda$ we have:

$$\circ'' \forall m. (B_1^\lambda; \text{msg}(\text{send}_{y_\beta} u_\alpha)B_2^\lambda) \in E[\Delta, y_\beta : A, u_{\alpha+1} : B \vdash K]_m$$ 

Put $I' = I, \ell$, where $\ell$ does not occur in $I$ and define $B_1^\kappa = A_1$ and $B_2^\kappa = A_2$. By induction on the number of steps, $\circ'$, $\circ''$, $\circ'''$ and $\circ''''$ for $n \neq \kappa, \lambda$ we get

$$\forall \gamma_1, \Theta_1, D_1'. \text{ if } \{B_1^n\}_{n \in I_1}, B_1^\kappa A_1 B_1^\lambda \{B_1^n\}_{n \in I_2} \mapsto^{\gamma_1, m_1, D_1'}$$

$$\text{then } \exists \gamma_2, D_2' \text{ such that } \{B_2^n\}_{n \in I_1}, B_2^\kappa A_2 \text{msg}(\text{send}_{y_\beta} u_\alpha)B_2^\lambda \{B_2^n\}_{n \in I_2} \mapsto^{\gamma_2, D_2'} \text{ and } \gamma_1 \subseteq \gamma_2$$

$$\forall x_\gamma \in \text{Out}(\Delta \vdash K), \text{ if } x_\gamma \in \Theta_1, \text{ then } (D_1'; D_2') \in V[\Delta \vdash K]_{\gamma_1}^{k+1}$$

$$\forall x_\gamma \in \text{In}(\Delta \vdash K), \text{ if } x_\gamma \in \Theta_1, \text{ then } (D_1'; D_2') \in V[\Delta \vdash K]_{\gamma_1}^{k+1}$$

By a $\forall E$. and $\{B_1^n\}_{n \in I_1}, B_1^\kappa A_1 B_1^\lambda \{B_1^n\}_{n \in I_2} \mapsto^{\gamma_1, \Theta_1} D_1'$ we get:

$$\exists \gamma_2, D_2' \text{ such that } \{B_2^n\}_{n \in I_1}, B_2^\kappa A_2 \text{msg}(\text{send}_{y_\beta} u_\alpha)B_2^\lambda \{B_2^n\}_{n \in I_2} \mapsto^{\gamma_2, D_2'} \text{ and } \gamma_1 \subseteq \gamma_2$$

$$\forall x_\gamma \in \text{Out}(\Delta \vdash K), \text{ if } x_\gamma \in \Theta_1, \text{ then } (D_1'; D_2') \in V[\Delta \vdash K]_{\gamma_1}^{k+1}$$

We also know that $\{B_2^n\}_{n \in I} = \{B_2^n\}_{n \in I_1}, B_2^\kappa A_2 \text{msg}(\text{send}_{y_\beta} u_\alpha)B_2^\lambda \{B_2^n\}_{n \in I_2} \mapsto^{\gamma_2, D_2'} \text{ and } \gamma_1 \subseteq \gamma_2$$

$$\forall x_\gamma \in \text{Out}(\Delta \vdash K), \text{ if } x_\gamma \in \Theta_1, \text{ then } (D_1'; D_2') \in V[\Delta \vdash K]_{\gamma_1}^{k+1}.$$
XI. LOGICAL EQUIVALENCE

A. Equivalence

**Lemma 18** (Reflexivity). For all security levels \( \xi \) and configurations \( \Delta \vDash D \vdash x_\alpha ; T \), we have

\( (\Delta \vDash D \vdash x_\alpha ; T) \equiv (\Delta \vDash D \vdash x_\alpha ; T) \).

**Proof.** The proof is straightforward by applying the reflexivity Lemma 21 proved in the next section on a trivial lattice that has only one element \( \bot \). We annotate all channels and processes with max secracies and running secracies equal to \( \bot \) and annotate process definitions in the signature all with one secrecy variable \( \psi \). The spawn terms in a process use a substitution that maps \( \psi \) to \( \bot \). With this translation, all session-typed configurations are also IFC-typed under the trivial lattice. \( \square \)

**Lemma 19** (Symmetry). For all configurations \( D_1 \) and \( D_2 \), we have \((\Delta \vDash D_1 \vdash x_\alpha ; T) \equiv (\Delta \vDash D_2 \vdash x_\alpha ; T)\), iff \((\Delta \vDash D_2 \vdash x_\alpha ; T) \equiv (\Delta \vDash D_1 \vdash x_\alpha ; T)\).

**Proof.** The proof is straightforward by the definition of triple equality. \( \square \)

**Lemma 20** (Transitivity). For all configurations \( D_1, D_2, \) and \( D_3 \), we have

if \( (\Delta \vDash D_1 \vdash x_\alpha ; T) \equiv (\Delta \vDash D_2 \vdash x_\alpha ; T) \), and \( (\Delta \vDash D_2 \vdash x_\alpha ; T) \equiv (\Delta \vDash D_3 \vdash x_\alpha ; T) \)

then \( (\Delta \vDash D_1 \vdash x_\alpha ; T) \equiv (\Delta \vDash D_3 \vdash x_\alpha ; T) \).

**Proof.** The proof follows from **Corollary 3** when we instantiate \( \Psi_0 \) with the trivial lattice only containing the element \( \bot \) and \( \xi \) to be \( \bot \). \( \square \)

B. Noninterference

**Lemma 21** (Reflexivity – only for IFC-typed processes). For all security levels \( \xi \) and configurations \( \Psi_0; \Gamma \vdash D \vdash x_\alpha ; T[c] \), we have

\( (\Gamma \vdash \|D\| \vdash x_\alpha ; T[c]) \equiv_\Psi^0 (\Gamma \vdash \|D\| \vdash x_\alpha ; T[c]) \).

**Proof.** Corollary of the fundamental theorem (Thm. 5). \( \square \)

**Lemma 22** (Symmetry). For all security levels \( \xi \) and configurations \( D_1 \) and \( D_2 \), we have \((\Gamma_1 \vdash D_1 \vdash x_\alpha ; T_1[c_1]) \equiv_\Psi^0 (\Gamma_2 \vdash D_2 \vdash y_\beta ; T_2[c_2])\), iff \((\Gamma_2 \vdash D_2 \vdash y_\beta ; T_2[c_2]) \equiv_\Psi^0 (\Gamma_1 \vdash D_1 \vdash x_\alpha ; T_1[c_1])\).

**Proof.** The proof is straightforward by **Def. 17**. \( \square \)

**Corollary 3** (Transitivity). For all security levels \( \xi \), and configurations \( D_1, D_2, \) and \( D_3 \), we have

if \( (\Gamma_1 \vdash D_1 \vdash x_\alpha ; T_1[c_1]) \equiv_\Psi^0 (\Gamma_2 \vdash D_2 \vdash y_\beta ; T_2[c_2]) \), and \( (\Gamma_2 \vdash D_2 \vdash y_\beta ; T_2[c_2]) \equiv_\Psi^0 (\Gamma_3 \vdash D_3 \vdash z_\gamma ; T_3[c_3]) \)

then \( (\Gamma_3 \vdash D_3 \vdash z_\gamma ; T_3[c_3]) \equiv_\Psi^0 (\Gamma_1 \vdash D_1 \vdash x_\alpha ; T_1[c_1]) \).

**Proof.** Consider arbitrary \( C_1 \), \( F_1 \) and \( C_3 \), and \( F_3 \) we need to show

\[ \forall m. (C_1 D_1 F_1; C_3 D_3 F_3) \in E[\Delta \vDash K]^m \] and \[ \forall m. (C_1 D_1 F_1; C_3 D_3 F_3) \in E[\Delta \vDash K]^m \]

By the assumptions, we get \((C_1 D_1 F_1; C_2 D_2 F_2) \in E[\Delta \vDash K]^m \) and \((C_2 D_2 F_2; C_3 D_3 F_3) \in E[\Delta \vDash K]^m \). The result follows by **Lem. 23** \( \square \)

**Lemma 23** (Transitivity of the term relation). If \( \Gamma_1 \vdash \Gamma m. (D_1; D_2) \in E[\Delta \vDash K]^m \) and \( \exists \Gamma \forall m. (D_2; D_3) \in E[\Delta \vDash K]^m \) then \( \exists \Gamma \forall k. (D_1; D_3) \in E[\Delta \vDash K]^k \).

**Proof.** Our goal is to prove for all \( D_1, D_2, \) and \( D_3 \) with \( \forall m. (D_1; D_2) \in E[\Delta \vDash K]^m \) and \( \forall m. (D_2; D_3) \in E[\Delta \vDash K]^m \) then \( \forall k. (D_1; D_3) \in E[\Delta \vDash K]^k \). We prove an equivalent statement that says for all \( k, D_1, D_2, \) and \( D_3 \) with \( \forall m. (D_1; D_2) \in E[\Delta \vDash K]^m \) and \( \forall m. (D_2; D_3) \in E[\Delta \vDash K]^m \) then \( (D_1; D_3) \in E[\Delta \vDash K]^k \). We proceed by induction on \( k \).

**Base case.** \( k = 0 \) Consider arbitrary configurations \( D_1, D_2, \) and \( D_3 \). By the assumptions, we know that \((D_1; D_2) \in Tree(\Delta \vDash K)\) and \((D_2; D_3) \in Tree(\Delta \vDash K)\), which gives us \((D_1; D_3) \in Tree(\Delta \vDash K)\). It is enough to complete the proof in this case.

**Inductive case.** \( k = k' + 1 \) Consider arbitrary configurations \( D_1, D_2, \) and \( D_3 \). Our goal is to show

\[ \forall \alpha \exists D'_1. \forall T_1. if D_1 \rightarrow^{\alpha \times 1} D'_1, then \exists D'_3 \vdash D_3 \rightarrow^{\alpha \times 1} D'_3 \text{ and} \]
\[ \forall x_\alpha \in Out(\Delta \vDash K), if x_\alpha \in T_1, then (D'_1; D'_3) \in \nu[\Delta \vDash K]_{x_\alpha}^{k' + 1} \]
\[ \forall x_\alpha \in In(\Delta \vDash K). (D'_1; D'_3) \in \nu[\Delta \vDash K]_{x_\alpha}^{k' + 1}. \]

Consider an arbitrary \( D'_1 \) and \( T_1 \), and assume \( D_1 \rightarrow^{\alpha \times 1} D'_1 \). Our goal is to prove
\[\exists D'_3 \text{ such that } D_3 \rightarrow^{r_1} D'_3 \text{ and} \]
\[\forall x_\alpha \in \text{Out}(\Delta \models K), \text{ if } x_\alpha \in \Upsilon_1, \text{ then } (D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{k'\alpha + 1} \text{ and} \]
\[\forall x_\alpha \in \text{In}(\Delta \models K), (D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{k'\alpha + 1}. \]

By assumption \( \dagger_1 \) and Lemma 16 we have
\[\forall D'_1, \forall \Upsilon_1, \text{ if } D_1 \rightarrow^{r_1} D'_1, \text{ then } \exists D'_2 \text{ such that } D_2 \rightarrow^{r_1} D'_2 \text{ and} \]
\[\forall x_\alpha \in \text{Out}(\Delta \models K), \text{ if } x_\alpha \in \Upsilon_1, \text{ then } \forall m.(D'_1; D'_2) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1} \text{ and} \]
\[\forall x_\alpha \in \text{In}(\Delta \models K), \forall m.(D'_1; D'_2) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1}. \]

And by assumption \( \dagger_2 \) and Lemma 16 we have
\[\forall D'_2, \forall \Upsilon_1, \text{ if } D_2 \rightarrow^{r_1} D'_2, \text{ then } \exists D'_3 \text{ such that } D_3 \rightarrow^{r_1} D'_3 \text{ and} \]
\[\forall x_\alpha \in \text{Out}(\Delta \models K), \text{ if } x_\alpha \in \Upsilon_1, \text{ then } \forall m.(D'_2; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1} \text{ and} \]
\[\forall x_\alpha \in \text{In}(\Delta \models K), \forall m.(D'_2; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1}. \]

We apply \( \forall E. \) on \( \dagger_1 \) by instantiating the existential quantifiers with \( D'_1 \) and \( \Upsilon_1 \). We can apply \( \text{if } E. \) since we know \( D_1 \rightarrow^{r_1} D'_1 \). We get a \( D'_2 \) with \( D_2 \rightarrow^{r_1} D'_2 \). Next, we apply \( \forall E. \) on \( \dagger_2 \) by instantiating the existential quantifiers with \( D'_2 \) and \( \Upsilon_1 \). We apply \( \text{if } E. \) as we know \( D_2 \rightarrow^{r_1} D'_2 \). As a result, we get a \( D'_3 \) with \( D_3 \rightarrow^{r_1} D'_3 \) as required by the goal. We need to prove:
\[\forall x_\alpha \in \text{Out}(\Delta \models K), \text{ if } x_\alpha \in \Upsilon_1, \text{ then } (D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{k'\alpha + 1} \text{ and} \]
\[\forall x_\alpha \in \text{In}(\Delta \models K), (D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{k'\alpha + 1}. \]

By \( \dagger_1' \) and \( \dagger_2' \) we have as assumptions
\[\forall x_\alpha \in \text{Out}(\Delta \models K), \text{ if } x_\alpha \in \Upsilon_1, \text{ then } \forall m.(D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1} \text{ and} \]
\[\forall x_\alpha \in \text{In}(\Delta \models K), \forall m.(D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1}. \]

and
\[\forall x_\alpha \in \text{Out}(\Delta \models K), \text{ if } x_\alpha \in \Upsilon_1, \text{ then } \forall m.(D'_2; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1} \text{ and} \]
\[\forall x_\alpha \in \text{In}(\Delta \models K), \forall m.(D'_2; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1}. \]

There are two parts to prove:

**Part 1.** Consider an arbitrary \( x_\alpha \in \text{Out}(\Delta \models K) \) and assume \( x_\alpha \in \Upsilon_1 \). Our goal is to show
\[(D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{k'\alpha + 1} \]

And by \( \dagger_1'' \) and \( \dagger_2'' \) we have as assumptions
\[\forall m.(D'_1; D'_2) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1} \text{ and} \]
\[\forall m.(D'_2; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{m + 1}. \]

We consider cases based on the type of \( x_\alpha \). 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
\[\Delta = \Delta_1, \Delta_2 \quad \text{and} \quad D'_1 = D'_1'A_1 \text{msg}(send_{y_\beta} x_\alpha) \quad \text{and} \quad D'_2 = D''_2'A_2 \text{msg}(send_{y_\beta} x_\alpha) \]
\[\text{and} \quad D'_3 = D''_2'A_2 \text{msg}(send_{y_\beta} x_\alpha). \]

Moreover,
\[\forall m.(D''_1; D''_2) \in \mathcal{E}[\Delta_1 \models x_{\alpha+1}; B]^{m} \quad \text{and} \quad \forall m.(A_1; A_2) \in \mathcal{E}[\Delta_2 \models y_\beta; A]^{m}, \]
\[\forall m.(A_2; A_3) \in \mathcal{E}[\Delta_2 \models y_\beta; A]^{m}. \]

We apply the induction hypothesis on a smaller observation index \( k' \) to get
\[(D'_1'; D'_2') \in \mathcal{E}[\Delta_1 \models x_{\alpha+1}; B]^{k'} \quad (A_1; A_2) \in \mathcal{E}[\Delta_2 \models y_\beta; A]^{k'} \]

Which by Row 4 of the logical relation is enough to prove the goal of this subcase, i.e.,
\[(D'_1'; D'_3') \in \mathcal{V}[\Delta \models K]_{x_{\alpha}}^{k'\alpha + 1}. \]

**Case 2.** \((\Delta = x_\alpha.A \rightarrow B, \Delta')\)

**Part 2.** Consider an arbitrary \( x_\alpha \in \text{In}(\Delta \models K) \) and assume \( x_\alpha \in \Theta_1 x. \) Our goal is to show
\[(D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha}^{k'\alpha + 1}. \]
By \(\vdash_1''\) and \(\vdash_2''\), we have as assumptions
\[
\vdash_1''' \forall m. (D'_1; D'_2) \in \mathcal{V}[\Delta \models K]_{m+1}^n \quad \text{and} \quad \vdash_2''' \forall m. (D'_2; D'_3) \in \mathcal{V}[\Delta \models K]_{m+1}^n.
\]
We consider cases based on the type of \(x_\alpha\). 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 \(\vdash_1'''\) and \(\vdash_2'''\) we get for all \(y_\beta \notin \text{dom}(\Delta, x_\alpha : A \otimes B, K)\).
\[
\forall m. (\text{msg}(\text{send}_{y_\beta} u_\alpha) D'_1; \text{msg}(\text{send}_{y_\beta} x_\alpha) D'_3) \in \mathcal{E}[\Delta, y_\beta : A, x_\alpha+1 : B \models K]^m
\]
\[
\forall m. (\text{msg}(\text{send}_{y_\beta} u_\alpha) D'_2; \text{msg}(\text{send}_{y_\beta} x_\alpha) D'_3) \in \mathcal{E}[\Delta, y_\beta : A, x_\alpha+1 : B \models K]^m
\]
We apply the induction hypothesis on a smaller observation index \(k'\) to get
\[
\forall y_\beta \notin \text{dom}(\Delta, x_\alpha : A \otimes B, K). (\text{msg}(\text{send}_{y_\beta} u_\alpha) D'_1; \text{msg}(\text{send}_{y_\beta} x_\alpha) D'_3) \in \mathcal{E}[\Delta, y_\beta : A, x_\alpha+1 : B \models K]^{k'}
\]
Which by **Row 9** of the logical relation is enough to prove the goal of this subcase, i.e.,
\[
(D'_1; D'_3) \in \mathcal{V}[\Delta \models K]_{x_\alpha+1}^{k'+1}
\]

**Case 2.** (\(K = x_\alpha : A \rightarrow B\))
Internal transition $\xrightarrow{\tau}$ defined as:

$D_1 \xrightarrow{\tau} D'_1$ if $D_1 \not\xrightarrow{\alpha} D'_1$

Actions $\xrightarrow{\text{close}_y}$, $\xrightarrow{\text{close}}$, $\xrightarrow{k}$, and $\xrightarrow{x\beta}$ defined as below when $y \in \text{fn}(D_1)$:

1. $D_1 \xrightarrow{\text{msg}(\text{close}_y)} D_2$
2. $D_1 \xrightarrow{\text{msg}(y, k)} D_2$
3. $D_1 \xrightarrow{\text{msg}(x\beta, y)} D_2$
4. $D_1 \xrightarrow{\text{proc}(z\delta, \text{wait}_y)} D_2$
5. $D_1 \xrightarrow{\text{proc}(z\delta, \text{case}_y \ell \Rightarrow P_\ell)} D_2$
6. $D_1 \xrightarrow{\text{proc}(z\delta, w \leftarrow \text{recv}_y)} D_2$
7. $D_1 \xrightarrow{\text{proc}(y, \text{case}_y \ell \Rightarrow P_\ell)} D_2$
8. $D_1 \xrightarrow{\text{proc}(y, w \leftarrow \text{recv}_y)} D_2$

Fig. 7: The transition rules. An overline indicates that an outgoing message, an otherwise the message is incoming.

XII. Adequacy

Definition 22 (Free names of a configuration). For $\Delta \vdash D :: \Delta'$, we define $\text{fn}(D)$ as $\text{dom}(\Delta, \Delta')$.

Definition 23 (Weak transition relations).

1. $\Rightarrow$ is the reflexive and transitive closure of $\xrightarrow{\tau}$.
2. $\xrightarrow{\alpha} \Rightarrow$ is $\Rightarrow \alpha$.

Definition 24 (Asynchronous bisimilarity). Asynchronous bisimilarity is the largest symmetric relation such that whenever $D_1 \not\xrightarrow{\alpha} D_2$, we have

1. (τ-step) if $D_1 \xrightarrow{\tau} D'_1$ then $\exists D'_2, D_2 \xrightarrow{\tau} D'_2$ and $D'_1 \not\xrightarrow{\alpha} D'_2$,
2. (output) if $D_1 \xrightarrow{\text{close}_y} D'_1$ then $\exists D'_2, D_2 \xrightarrow{\text{close}} D'_2$ and $D'_1 \not\xrightarrow{\alpha} D'_2$,
3. (left input) for all $q \notin \text{fn}(D_1)$, if $D_1 \xrightarrow{\text{L}_{x\alpha} q} D'_1$ then $\exists D'_2, D_2 \xrightarrow{\text{L}_{x\alpha} q} D'_2$ and $D'_1 \not\xrightarrow{\alpha} D'_2$,
4. (right input) for all $q \notin \text{fn}(D_1)$, if $D_1 \xrightarrow{\text{R}_{x\alpha} q} D'_1$ then $\exists D'_2, D_2 \xrightarrow{\text{R}_{x\alpha} q} D'_2$ and $D'_1 \not\xrightarrow{\alpha} D'_2$

where $\text{msg}(x, q)$ is defined as $\text{msg}(\text{close}_x)$ if $q = \text{close}$, $\text{msg}(x, k)$ if $q = k$, and $\text{msg}(\text{send}_x z, x)$ if $q = z$.

Definition 25 (High provider and High client). We repeat the definition of high provider and high client configurations §VIII-A here.

$B \in \text{H-Provider}^\xi(\cdot)$ iff $c \not\in \xi$ and $B = B' T$ and $B' \in \text{H-Provider}^\xi(\Gamma)$ and $T \in \text{Tree}(\cdot \vdash x, A)$, or $c \subseteq \xi$ and $B = \text{H-Provider}^\xi(\Gamma)$

$\mathcal{T} \in \text{H-Client}^\xi(x, A)$ iff $c \not\subseteq \xi$ and $\mathcal{T} \in \text{Tree}(x, A \vdash - : 1)$, or $c \subseteq \xi$ and $B = \cdot$.

\(45\)
Definition 26. For $D_1 \in \text{Tree}([\Gamma_1] \vdash x_\alpha;A_1)$, $D_2 \in \text{Tree}([\Gamma_2] \vdash y_\beta;A_2)$ we define
\[
\Gamma_1 \vdash D_1 :: x_\alpha;A_1[c_1] \overset{\xi}{\Rightarrow} \Gamma_2 \vdash D_2 :: y_\beta;A_2[c_2]
\]
\[
\Gamma_1 \vdash \xi = \Gamma_2, \downarrow \xi \text{ and } y_\beta;A_2[c_2] \downarrow \xi = x_\alpha;A_1[c_1] \downarrow \xi \text{ and } \forall B_1 \in \text{H-Provider}(\Gamma_1), B_2 \in \text{H-Provider}(\Gamma_2), T_1 \in \text{H-Client}(x_\alpha;A_1[c_1]), T_2 \in \text{H-Client}(y_\beta;A_2[c_2]).
\]

Corollary 4. For all $D_1 \in \text{Tree}([\Gamma_1] \vdash x_\alpha;A_1)$ and $D_2 \in \text{Tree}([\Gamma_2] \vdash y_\beta;A_2)$, we have $(\Gamma_1 \vdash D_1 :: x_\alpha;A_1[c_1]) \equiv^\Psi (\Gamma_2 \vdash D_2 :: y_\beta;A_2[c_2])$.

Proof. The proof is straightforward by considering Def. 17 and the following corollary (Corollary 5).

Corollary 5. for all $(D_1, D_2) \in \text{Tree}(\Delta \vdash K)$, we have $\forall m.(D_1, D_2) \in \mathcal{E}[\Delta \vdash K]^m$ and $\forall m.(D_2, D_1) \in \mathcal{E}[\Delta \vdash K]^m$ iff $D_1 \approx_d D_2$.

Proof. It is a corollary of the following lemma (Lem. 24).

Lemma 24. Consider a pair of session-typed forests $(C_1, C_2)$ Forest$(\Delta \vdash \Delta')$ consisting of multiple session-typed trees indexed in the set $I$, i.e., $C_i = (C_i')_{j \in I}$ for $i \in \{1, 2\}$, and $(C_1', C_2') \in \text{Tree}(\Delta' \vdash K')$ with $\Delta = \{\Delta_j\}_{j \in I}$ and $\Delta' = \{K_j\}_{j \in I}$.

We have:
\[
\forall j \in I.\forall m.(C_j, C_j') \in \mathcal{E}[\Delta_j \vdash K_j]^m \text{ and } (C_2', C_1') \in \mathcal{E}[\Delta' \vdash K']^m \text{ iff } C_1 \approx_a C_2.
\]

Proof. The proof consists of two parts.

1) Soundness. We want to prove for any arbitrary pair of forests $C_1 = \{C_1'\}_{j \in I}$ and $C_2 = \{C_2'\}_{j \in I}$ such that $\forall j \in I. (C_1', C_2') \in \text{Tree}(\Delta' \vdash K')$, if $\forall j \in I.\forall m.(C_j, C_j') \in \mathcal{E}[\Delta_j \vdash K_j]^m$ and $\forall j \in I.\forall m.(C_j', C_2') \in \mathcal{E}[\Delta_j \vdash K_j]^m$ then $\{C_1'\}_{j \in I} \approx_a \{C_2'\}_{j \in I}$.

We prove the proof by coinduction on the generating function of the bisimilarity. Consider an arbitrary $C_1 = \{C_1'\}_{j \in I}$ and $C_2 = \{C_2'\}_{j \in I}$ such that $\forall j \in I. (C_1', C_2') \in \text{Tree}(\Delta' \vdash K')$, and assume $\forall j \in I.\forall m.(C_j, C_j') \in \mathcal{E}[\Delta_j \vdash K_j]^m$. Our goal is to show $\{C_1'\}_{j \in I} \approx_a \{C_2'\}_{j \in I}$. We prove it by showing the items (1)-(4) by assuming $\forall j \in I.\forall m.(C_j, C_j') \in \mathcal{E}[\Delta_j \vdash K_j]^m$ only. The symmetric relation can be established with the other assumption $\forall j \in I.\forall m.(C_j', C_2') \in \mathcal{E}[\Delta_j \vdash K_j]^m$. We continue by establishing items (1)-(4) in Def. 24.

1) Assume $C_1 \overset{\tau}{\Rightarrow} C_1'$, then by the definition of $\tau$-transition, we have $C_1 \Rightarrow C_1'$. In other words we have $\{C_1'\}_{j \in I} \Rightarrow \{C_1'\}_{j \in I}$ for $C_1' = \{C_1'\}_{j \in I}$. To establish item (1) in the definition, it is enough to prove that $\{C_1'\}_{j \in I} \approx_a \{C_2'\}_{j \in I}$. Since we unfolded the coniductive 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 $\forall j \in I.\forall m.(C_j, C_j') \in \mathcal{E}[\Delta_j \vdash K_j]^m$ and by the forward closure lemma (Lem. 14), we get $\forall j \in I.\forall m.(C_j, C_j') \in \mathcal{E}[\Delta_j \vdash K_j]^m$. This along with type-preservation of session-typed programs, is enough to get $\{C_1'\}_{j \in I} \approx_a \{C_2'\}_{j \in I}$ by coinduction.

2) Assume $C_1 \overset{\tau}{\Rightarrow} C_1'$, then by Fig. 7, we have $C_1 = C_1' \msg(x_{\alpha}, q)C_2'$, with $C_1' = C_1'C_2'$. This means that for some $\kappa \in I$, we have either $C_1' = C_1' \msg(x_{\alpha}, q)$ or $C_1' = \msg(x_{\alpha}, q)C_2'$, and $C_1' = \{C_1'\}_{j \in I - \{\kappa\}}C_1'$.

In particular, we know that $\forall m.(C_1, C_2') \in \mathcal{E}[\Delta \vdash K_{\kappa}]^m$, and $C_1' \overset{0_{\kappa};\tau}{\Rightarrow} C_1'$, such that $x_{\alpha} \in \mathcal{T}_1$. By the definition of the term interpretation and Lem. 16 we get
\[
\forall \mathcal{T}_1, \Theta_1, \mathcal{C}_1', \mathcal{C}_2', \text{if } C_1' \overset{0_{\kappa};\tau}{\Rightarrow} C_2', \text{then } \exists \mathcal{T}_2, \mathcal{C}_2' \text{ such that } C_2' \overset{0_{\kappa};\tau}{\Rightarrow} C_2' \text{ and } \mathcal{T}_1 \subseteq \mathcal{T}_2 \text{ and }
\forall x_{\alpha} \in \mathcal{O}(\Delta \vdash K_{\kappa}), \text{if } x_{\alpha} \in \mathcal{T}_1, \text{then } \forall k. (C_1' ; C_2') \in \mathcal{V}[\Delta_{k} \vdash K_{k}]^m_{x_{\alpha}} \text{ and }
\forall x_{\alpha} \in \mathcal{I}(\Delta \vdash K_{\kappa}), \text{if } x_{\alpha} \in \mathcal{T}_1, \text{then } \forall k. (C_1' ; C_2') \in \mathcal{V}[\Delta_{k} \vdash K_{k}]^m_{x_{\alpha}}.
\]

We can instantiate the $\forall$ quantifiers and apply an if–Elimination rule to get $C_2' \overset{0_{\kappa};\tau}{\Rightarrow} C_2'$ for some $\mathcal{T}_2 \supseteq \mathcal{T}_1$ and some $C_2'$. Since $x_{\alpha} \in \mathcal{T}_1$, we get by the definition of the logical relation $\forall m.(C_1', C_2') \in \mathcal{V}_{\phi}(\Delta \vdash K_{\kappa})^m_{x_{\alpha}}$. This (depending on the type of the channel $x_{\alpha}$) gives us $C_2' = C_2' \msg(x_{\alpha}, q)$ or $C_2' = \msg(x_{\alpha}, q)C_2'$. Note that if $q$ is a channel name, $C_2'$ is a forest, rather than a tree, and if $q$ is the label close, the configuration $C_2''$ is empty.

Put $C_2' = \{C_2'\}_{j \in I - \{\kappa\}}C_2''$. To establish item (2) in the definition, it is enough to prove that $C_1' \approx_a C_2'$. Since we unfolded the coniductive 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 $x_{\alpha}$. 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 $\Delta^\kappa = \Delta^\kappa_1 \cup \Delta^\kappa_2$ and $C^\kappa_1 = C^\kappa_1'$ msg$(\text{send } y_\delta x_\alpha)$ with $C^\kappa_1' = C^\kappa_1'' A_1$. Moreover, we have $C^\kappa_2 = C^\kappa_2' \text{ msg}(\text{send } y_\delta x_\alpha)$ with $C^\kappa_2' = C^\kappa_2'' A_2$. And $\forall m.(C^\kappa_1 ; C^\kappa_2') \in \mathcal{E}[\Delta^\kappa_1 \vdash x_{\alpha + 1}; B]^m$ and $\forall m.(A_1 ; A_2) \in \mathcal{E}[\Delta^\kappa_2 \vdash y_\delta : A]^m$.

Recall that all other trees in the forest are still related, i.e. $\forall j \in I - \{\kappa\}$, we have $\forall m.(C^\kappa_j ; C^\kappa_j) \in \mathcal{E}[\Delta^\kappa_j \vdash K_j]^m$ This is enough to apply the coinductive argument and get $C^\kappa_1 \approx C^\kappa_2$ where $C^\kappa_1 = \{C^\kappa_1_j\}_{j \in I - \{\kappa\}}, C^\kappa_1'$ and $C^\kappa_2 = \{C^\kappa_2_j\}_{j \in I - \{\kappa\}}, C^\kappa_2'$. 

3) Assume for an arbitrary $\kappa$, we have $C_1 \xrightarrow{L_{x_\alpha} q} C'_1$. By Fig. 7, there is a process $\text{proc}(x_\beta, P_\alpha x_\alpha) \in C_1$ which is waiting to receive a message along $x_\alpha$. In particular, for some tree $C^\kappa_1$ for $\kappa \in I$, we have $\text{proc}(x_\beta, P_\alpha x_\alpha) \in C^\kappa_1$. 

We know that $\forall m.(C^\kappa_1, C^\kappa_2) \in \mathcal{E}[\Delta^\kappa \vdash K^\kappa_1]^m$, and $C^\kappa_1 \rightarrow^{\nu \tau_1 \tau_2} C^\kappa_1$, such that $x_\alpha \in \Theta^\kappa_1$. By the definition of the term interpretation and Lem. 16 we get

$$\forall \tau^\kappa_1, \Theta^\kappa_1, C^\kappa_1' . \text{ if } C^\kappa_1' \rightarrow^{\nu \tau_1 \tau_2} C^\kappa_1' \text{, then } \exists \tau^\kappa_2, C^\kappa_2' \text{ such that } C^\kappa_2' \rightarrow^{\nu \tau_1 \tau_2} C^\kappa_2' \text{ and } \forall \tau^\kappa_3 \subseteq \tau^\kappa_2$$

We can instantiate the $\forall$ quantifiers and apply an $\text{if}$–Elimination rule to get $C^\kappa_1 \rightarrow^{\nu \tau_1 \tau_2} C^\kappa_2'$ for some $\tau^\kappa_2 \subseteq \tau^\kappa_1$ and some $C^\kappa_2'$. Since $x_\alpha \in \Theta^\kappa_1$, we get by the definition of the logical relation $\forall m.(C^\kappa_1, C^\kappa_2) \in \mathcal{E}[\Delta^\kappa \vdash K^\kappa_1]^m$.

The proof proceeds by cases on the type of $x_\alpha$. 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., $\Delta^\kappa = \Delta^\kappa_1, x_\alpha : A \otimes B$) In this case, we know that $C^\kappa_1 = C^\kappa_1 \text{ proc}(x_\beta, w \leftarrow \text{recv } x_\alpha) C^\kappa_1$ and $q$ is a channel $y_\delta$ which is fresh in both $C^\kappa_1$ and $C^\kappa_2$. Moreover, we have $C^\kappa_1 = \text{ msg}(\text{send } y_\delta x_\alpha) C^\kappa_1$, and thus $C^\kappa_1 = \text{ msg}(\text{send } y_\delta x_\alpha) C^\kappa_1$.

By the logical relation, we get $\forall m.(\text{msg}(\text{send } y_\delta x_\alpha) C^\kappa_1, \text{msg}(\text{send } y_\delta x_\alpha) C^\kappa_2) \in \mathcal{E}[\Delta^\kappa \vdash y_\delta : A, x_\alpha + 1 ; B \vdash K^\kappa_2]^m$.

We can instantiate the $\forall$ quantifiers and apply an $\text{if}$–Elimination rule to get $C^\kappa_1 \rightarrow^{\nu \tau_1 \tau_2} C^\kappa_2'$ for some $\tau^\kappa_2 \subseteq \tau^\kappa_1$ and some $C^\kappa_2'$. Since $x_\alpha \in \Theta^\kappa_1$, we get by the definition of the logical relation $\forall m.(C^\kappa_1, C^\kappa_2) \in \mathcal{E}[\Delta^\kappa \vdash K^\kappa_1]^m$.

The proof proceeds by cases on the type of $x_\alpha$. Here we only provide one interesting case, the proof of other cases is similar.

Subcase 1. $K^\kappa = x_\alpha : A \rightarrow B \in \Delta'$. In this case, we know that $C^\kappa_1 = C^\kappa_1 \text{ proc}(x_\beta, w \leftarrow \text{recv } x_\alpha) C^\kappa_1$ and $q$ is a channel $y_\delta$ which is not free in $C^\kappa_1$ (and by the typing not in $C^\kappa_2$ either). Moreover, we have $C^\kappa_1 = \text{ msg}(\text{send } y_\delta x_\alpha)$, and thus $C^\kappa_1 = \text{ msg}(\text{send } y_\delta x_\alpha)$.

By the logical relation, we get $\forall m.(\text{msg}(\text{send } y_\delta x_\alpha) C^\kappa_1, \text{msg}(\text{send } y_\delta x_\alpha) C^\kappa_2) \in \mathcal{E}[\Delta^\kappa \vdash y_\delta : A \vdash x_\alpha + 1 ; B]^m$.

We instead prove an equivalent statement that says

For any natural number $m$ and any arbitrary pair of forests $C_1 = \{C^\kappa_1\}_{j \in I}$ and $C_2 = \{C^\kappa_2\}_{j \in I}$ such that $\forall j \in I. (C^\kappa_1, C^\kappa_2) \in \text{Tree}(\Delta^\kappa \vdash K^\kappa)$, if $\{C^\kappa_1\}_{j \in I} \approx_a \{C^\kappa_2\}_{j \in I}$ then $\forall j \in I. (C^\kappa_1, C^\kappa_2) \in \mathcal{E}[\Delta^\kappa \vdash K^\kappa]^m$.

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
Consider an arbitrary $\Gamma^i_k, \Theta^i_k$, and $C^i_k$, and assume $C^i_k \Rightarrow \tau C^i_k$ we need to prove

$$\exists \exists Y^i_1, C^i_2 \text{ such that } C^i_2 \Rightarrow \tau Y^i_2 \text{ and } \forall x \in \text{Out}(\Delta \vdash K). \text{if } x \in \Gamma^i_1 \text{ then } (C^i_1; C^i_2) \in \mathcal{V}[\Delta \vdash K]^m_{x+a} \text{ and } \forall x \in \text{In}(\Delta \vdash K). \text{if } x \in \Theta^i_1 \text{ then } (C^i_1; C^i_2) \in \mathcal{V}[\Delta \vdash K]^m_{x+a}.$$

By local transition steps, we know that $C^i_1 \Rightarrow \tau C^i_1'$, where $C^i_1' = \{c^i_1\}_{t \leftarrow \{\tau\}}^i C^i_1$. By $C^i_1 \equiv \Theta^i_1$, we can apply the clause for $\tau$-transitions in Def. 24 for zero or multiple times, to get $C^i_2 \Rightarrow \tau C^i_2'$ such that $C^i_1' \Rightarrow \tau C^i_2'$. Consider the channels in the sets $\Theta^i_1$ and $\Theta^i_2$. There are three cases:

A. By item (2) of Def. 24 and $C^i_1 \equiv \Theta^i_2$, we know that for all $x \in \text{Out}(\Delta \vdash K)$ such that $x \in \Gamma^i_1$, we have $C^i_2 \Rightarrow \tau C^i_2'$ such that $C^i_2'$ sends along the channel $x$ and $C^i_2' = \mathcal{C}_1 \mathcal{M}(x) \mathcal{C}_1^C \mathcal{C}_2$ and $C^i_2'' = \mathcal{C}_2 \mathcal{M}(x) \mathcal{C}_2^C \mathcal{C}_2''$. We have $C^i_1 C^i_2 \equiv \Theta^i_2$, $C^i_2' C^i_2''$.

B. By item (3) of Def. 24 and $C^i_1 \equiv \Theta^i_2$, we know that for all $x \in \text{In}(\Delta \vdash \cdot)$ such that $x \in \Theta^i_1$, we have $C^i_2 \Rightarrow \tau C^i_2''$ such that $\mathcal{C}_1 \mathcal{M}(x) \mathcal{C}_1^C \mathcal{C}_2$.

C. By item (4) of Def. 24 and $C^i_1 \equiv \Theta^i_2$, we know that for all $x \in \text{In}(\Delta \vdash K)$ such that $x \in \Theta^i_1$, we have $C^i_2 \Rightarrow \tau C^i_2''$ such that $\mathcal{C}_1 \mathcal{M}(x) \mathcal{C}_1^C \mathcal{C}_2$.

Apply the confluence lemma (Lem. 11) on (i) $C^i_2 \Rightarrow \tau C^i_2''$ for all $x \in \text{Out}(\Delta \vdash K)$ such that $x \in \Theta^i_1$, (ii) $C^i_2 \Rightarrow \tau C^i_2''$ for all $x \in \text{In}(\Delta \vdash K)$ such that $x \in \Theta^i_1$, and (iii) $C^i_2 \Rightarrow \tau C^i_2''$ for all $x \in \text{In}(\Delta \vdash K)$ such that $x \in \Theta^i_1$. More precisely, by the forest structure, we have $C^i_2'' = \{c_2''\}_{t \leftarrow \{\tau\}^i}^j C^i_2''$, with $C^i_2 \Rightarrow \tau C^i_2''$ with $Y^i_2 \subseteq Y^i_3$. We use $Y^i_2$ and $C^i_2''$ to instantiate the existential quantifier in the goal. We need to prove that

$$\forall x \in \text{Out}(\Delta \vdash K), \text{if } x \in \Gamma^i_1 \text{ then } (C^i_1; C^i_2'') \in \mathcal{V}[\Delta \vdash K]^m_{x+a} \text{ and } \forall x \in \text{In}(\Delta \vdash K), \text{if } x \in \Theta^i_1 \text{ then } (C^i_1; C^i_2'') \in \mathcal{V}[\Delta \vdash K]^m_{x+a}.$$
**Proof.** The proof is by coinduction on the generative function of the bisimulation \((D_1 \approx_a D'_2)\). We consider four cases required to establish \(D_1 \approx_a D'_2\).

1) **(|- step)** if \(D_1 \xrightarrow{\tau} D'_1\) then by the assumption \(D_2 \xrightarrow{\tau} D''_2\) and \(D'_1 \approx_a D''_2\). This gives us \(D_2 \xrightarrow{\tau} D''_2\) and \(D'_1 \approx_a D''_2\). We also know by assumption that \(D_2 \xrightarrow{\tau} D''_2\). By the confluence lemma (Lem. 11), we can build a \(D\) such that \(D''_2 \xrightarrow{\tau} D\) and \(D'_2 \xrightarrow{\tau} D\). By coinduction on \(D'_1 \approx_a D''_2\) having the assumption \(D''_2 \xrightarrow{\tau} D\), we get \(D'_1 \approx_a D\), which along with \(D'_2 \xrightarrow{\tau} D\) is enough to get the result.

2) **(output)** if \(D_1 \xrightarrow{a.q} D'_1\) then by the assumption \(D_2 \xrightarrow{a.q} D''_2\) and \(D'_1 \approx_a D''_2\). This gives us \(D_2 \xrightarrow{a.q} D''_2\) and \(D'_1 \approx_a D''_2\). We also know by assumption that \(D_2 \xrightarrow{a.q} D''_2\). By the confluence lemma (Lem. 11), we can build a \(D\) such that \(D''_2 \xrightarrow{a.q} D\) and \(D'_2 \xrightarrow{a.q} D\). Moreover, for some \(D'\) we have \(D \xrightarrow{a.q} D'\), such that \(D''_2 \xrightarrow{a.q} D'\). Recall that we also have \(D''_2 \xrightarrow{a.q} D''_2\). Again, we apply the confluence lemma (Lem. 11) to get a \(D''\) such that \(D''_2 \xrightarrow{a.q} D''\) and \(D' \xrightarrow{a.q} D''\). By coinduction on \(D'_1 \approx_a D''_2\) having the assumption \(D''_2 \xrightarrow{a.q} D''\), we get \(D'_1 \approx_a D''\). Moreover, we have \(D'_2 \xrightarrow{a.q} D''\), i.e., \(D'_2 \xrightarrow{a.q} D\) and \(D \xrightarrow{a.q} D'\), and \(D' \xrightarrow{a.q} D''\). This is enough to get the result.

3) **(left input)** Consider an arbitrary \(q\) which is not free in \(D_1\), and assume \(D_1 \xrightarrow{L \alpha \cdot q} D'_1\), then \(\exists D''_2, D_2 \xrightarrow{\tau} D''_2\) and \(D'_1 \approx_a \text{msg}(x.a.q)D''_2\). This gives us \(D_2 \xrightarrow{\tau} D''_2\) and \(D'_1 \approx_a \text{msg}(x.a.q)D''_2\). By the confluence lemma (Lem. 11), we can build a \(D\) such that \(D''_2 \xrightarrow{\tau} D\) and \(D'_2 \xrightarrow{\tau} D\). This also gives us \(\text{msg}(x.a.q)D''_2 \xrightarrow{\tau} \text{msg}(x.a.q)D\). By coinduction on \(D'_1 \approx_a \text{msg}(x.a.q)D''_2\) having the assumption \(\text{msg}(x.a.q)D''_2 \xrightarrow{\tau} \text{msg}(x.a.q)D\), we get \(D'_1 \approx_a \text{msg}(x.a.q)D\). Moreover, we have \(D'_2 \xrightarrow{\tau} D\), i.e., \(D'_2 \xrightarrow{\tau} D\). This is enough to complete the proof of this case.

4) **(right input)** for all \(q \notin \text{fn}(D_1)\), if \(D_1 \xrightarrow{R \alpha \cdot q} D'_1\) then \(\exists D''_2, D_2 \xrightarrow{\tau} D''_2\) and \(D'_1 \approx_a D''_2 \text{msg}(x.a.q)\). The proof is similar to the previous case.

\[\square\]
XIII. BIORTHOGONALITY

A. Equivalence

**Definition 27** (Session-typed environment). A session-typed environment with the interface $\Delta \vdash K$, is of the form $C[, ]F$, such that for some $\Delta'$ and $K'$, we have $\Delta' \vdash C :: \Delta$ and $K \vdash F :: K'$.

**Definition 28** (Program- and environment- relations). A session program-relation is a binary relation between session-typed programs, i.e., open configurations of the form $\Delta \vdash D :: K$. Given the interface $\Delta \vdash K$, we write $PRel(\Delta \vdash K)$ for the set of all program relations that relate programs $\Delta \vdash D :: K$.

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 $C[, ]F$ with the interface $\Delta \vdash K$.

**Definition 29** (The $\top^T$ operation on relations). Given any interface $\Delta \vdash K$ and $r \in PRel(\Delta \vdash K)$, we define $r^T \in ERel(\Delta \vdash K)$ by

$$(C_1[, ]F_1, C_2[, ]F_2) \in r^T \iff \forall (D_1, D_2) \in r.(C_1D_1F_1 \approx_a C_2D_2F_2);$$

and given any $s \in ERel(\Delta \vdash K)$, we define $s^T \in PRel(\Delta \vdash K)$ by

$$(D_1, D_2) \in s^T \iff \forall (C_1[, ]F_1, C_2[, ]F_2) \in s.(C_1D_1F_1 \approx_a C_2D_2F_2);$$

As explained in [1], just by virtue of how these definitions are defined, we get that the operation is a Galois connection, which is inflationary, and idempotent.

**Definition 30.**

- Define the relation $\Delta \vdash_D D_1 :: K \equiv \Delta \vdash D_2 :: K$ as $(D_1, D_2) \in \text{Tree}(\Delta \vdash K)$ and $\forall m. (D_1, D_2) \in \mathcal{E}[\Delta \vdash K]$ and $\forall m.(D_2, D_1) \in \mathcal{E}[\Delta \vdash K]$.

- Define the relation $\Delta \vdash C_1[, ]F_1 \vdash K \equiv \Delta \vdash C_2[, ]F_2 \vdash K$ as (i) $\exists K'$ such that $K \vdash F_1 :: K' \equiv K \vdash F_2 :: K'$ and (ii) $\Delta' \vdash C_1, C_2 \in \text{Forest}(\Delta' \vdash \top)$ and $\forall T_1 \in C_1, \forall T_2 \in C_2$ with $(T_1, T_2) \in \text{Tree}(\Delta' \vdash K''), we have $\Delta' \vdash T_1 :: K'' \equiv \Delta' \vdash T_2 :: K'''$.

**Theorem 6** ($\top$-$\top$-closure). Consider $(D_1, D_2) \in \text{Tree}(\Delta \vdash K)$, we have

$$\quad (\Delta \vdash_D D_1 :: K) \equiv (\Delta \vdash D_2 :: K)$$

iff

$$\forall C_1, C_2, F_1, F_2, \text{s.t. } (\Delta \vdash C_1[, ]F_1 \vdash K) \equiv (\Delta \vdash C_2[, ]F_2 \vdash K) \text{ we have }$$

$$\forall m. C_1D_1F_1 \approx_a C_2D_2F_2.$$

**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. 24) and the definition of equivalence $\equiv$ for programs and environments (Def. 30).

**Right to Left:** Consider $C_1 = C_2 = F_1 = F_2 = \cdot$. By Lem. 21 and Def. 30 we know that $(\Delta \vdash [\cdot \vdash \cdot] \vdash K) \equiv (\Delta \vdash [\cdot \vdash \cdot] \vdash K)$. By Def. 30 we have $\Delta \vdash C_1D_1 :: K \equiv \Delta \vdash C_2D_2 :: K$, 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 [1], it shows $r = s^T$, when $r$ and $s$ defined as our logical equivalence.

B. Noninterference

**Definition 31** (Session-typed environment). A session-typed environment with the interface $\Delta \vdash K$, is of the form $C[, ]F$, such that for some $\Delta'$ and $K'$, we have $\Delta' \vdash C :: \Delta$ and $K \vdash F :: K'$.

**Definition 32** (Program- and environment- relations). A security session program-relation is a binary relation between session-typed programs, i.e., open configurations of the form $\Delta \vdash D :: K$. 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 program relations that relate programs $\Gamma_1 \vdash D_1 :: x_\alpha.A_1$ and $\Gamma_2 \vdash D_2 :: y_\beta.A_2$.

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 $C_1[, ]F_1$ with the interface $\Gamma_1 \downarrow \xi \vdash x_\alpha.A_1[c_1] \downarrow \xi$ and $C_2[, ]F_2$ with the interface $\Gamma_2 \downarrow \xi \vdash y_\beta.A_2[c_2] \downarrow \xi$.  

50
Definition 35. Define the relation
\[ \Gamma_1 \vdash C_1[|F_1'|, x_{\alpha}:A_1[c_1], \Gamma_2 \vdash y_{\beta}:A_2[c_2]] \Rightarrow \exists \delta_0. \Gamma_1 \vdash C_1[|F_2'|, y_{\beta}:A_2[c_2]] ~\text{as}\]

(i) (The observable interface of the environments are the same.) We have \( \Gamma = \Gamma_1 \downarrow \xi = \Gamma_2 \downarrow \xi \) and \( K^x = x_{\alpha}:A_1[c_1] \downarrow \xi = y_{\beta}:A_2[c_2] \downarrow \xi \), and

(ii) (The configurations \( C_1 \) and \( C_2 \) 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 \( \Delta \), we have \( \langle C_1, C_2 \rangle \in \mathsf{Forest}(\Delta \vdash |\Gamma|) \). In particular, if \( \Delta = \varepsilon \), we have \( \Gamma_1 = \Gamma_2 = \varepsilon \).

Moreover, for every \( \Gamma_1' \) and \( \Gamma_2' \) such that \( |\Gamma_1'| = |\Gamma_2'| = \Delta \), and \( \forall w_{\eta}:A[c] \in \Gamma_1' \), \( c \subseteq \xi \) and \( \forall w_{\eta}:A[c] \in \Gamma_2' \), \( c \subseteq \xi \) we have \( \Gamma_1 \vdash C_1 \vdash \varepsilon \subseteq \Gamma_2 \vdash \varepsilon \).

(iii) (The configurations \( F_1 \) and \( F_2 \) use channels annotated as observable, offer the same channels, and are equivalent for any observable (low-confidentiality) annotation of their offering channels.) If \( K^x = \varepsilon[|\Gamma|] \), then \( F_1 = F_2 = \varepsilon \). Otherwise, for some \( w_{\eta}:A \), we have \( (F_1, F_2) \in \mathsf{Tree}(K^x \vdash w_{\eta}:A) \). Moreover, \( \forall c, c' \subseteq \xi \), we have \( \forall K^x \vdash F_1 \vdash w_{\eta}:A[c] \equiv \exists \delta_0. K^x \vdash F_2 \vdash w_{\eta}:A[c'] \) and.

\[ \square \]

Lemma 26 (Compositionality of \( \equiv_{\delta_0} \)). If

(i) \( \Gamma_1' \vdash C_1 \vdash \Gamma \) \( \equiv_{\delta_0} \) \( \Gamma_2' \vdash C_2 \vdash \Gamma \) with \( \Gamma = \Gamma_1' \downarrow \xi = \Gamma_2' \downarrow \xi \) and

(ii) \( \Gamma_1 \vdash D_1 \vdash x_{\alpha}:A_1[c_1] \) \( \equiv_{\delta_0} \) \( \Gamma_2 \vdash D_2 \vdash y_{\beta}:A_2[c_2] \) and

(iii) \( K^x \vdash F_1 \vdash K_1 \) \( \equiv_{\delta_0} \) \( K^x \vdash F_2 \vdash K_2 \) with \( K^x = x_{\alpha}:A_1[c_1] \downarrow \xi = y_{\beta}:A_2[c_2] \downarrow \xi \) then

\[ \Gamma_1' \downarrow \xi \vdash C_1 D_1 F_1 \downarrow \xi \vdash K_1 \] \( \equiv_{\delta_0} \) \( \Gamma_2' \downarrow \xi \vdash C_2 D_2 F_2 \downarrow \xi \vdash K_2 \]

where \( \Gamma_1' \) is the set of all channels \( w_{\eta}:C[d] \in \Gamma_1 \) with \( d \nsubseteq \xi \) and \( \Gamma_1' \) is the set of all channels \( w_{\eta}:C[d] \in \Gamma_2 \) with \( d \nsubseteq \xi \).

Proof. By the configuration typing and definition of equivalence \( \equiv_{\delta_0} \) and (i), (ii), (iii), we know that \( C_1 D_1 F_1 \in \mathsf{Tree}(\Gamma_1' \vdash |K_1'|) \) and \( C_2 D_2 F_2 \in \mathsf{Tree}(\Gamma_2' \vdash |K_2'|) \). Moreover, by definition of equivalence \( \equiv_{\delta_0} \) and (i), (iii) we get \( \Gamma' = (\Gamma_1', \Gamma_2') \downarrow \xi = (\Gamma_2', \Gamma_2') \downarrow \xi \) and \( K'^x = K_1' \downarrow \xi = K_2' \downarrow \xi \).

Assume arbitrary configurations \( B_1 \in \mathsf{HostProvider}(\Gamma_1'), B_2 \in \mathsf{HostProvider}(\Gamma_2'), B_2 \in \mathsf{HostProvider}(\Gamma_2'), B_2 \in \mathsf{HostProvider}(\Gamma_2'), B_2 \in \mathsf{HostProvider}(\Gamma_2'), \) \( \Gamma_1 \in \mathsf{HostClient}(\Gamma_1'), \) and \( \Gamma_2 \in \mathsf{HostClient}(\Gamma_2') \). Our goal is to prove

\[ \forall m. (B_1 B_2 B_1 C_1 D_1 F_1 T_1, B_2 B_2 C_2 D_2 F_2 T_2) \in \mathcal{E}[\Gamma'] \vdash |K^x| \uparrow m \text{, and } \forall m. (B_2 B_2 B_2 C_2 D_2 F_2 T_2, B_1 B_1 C_1 D_1 F_1 T_1) \in \mathcal{E}[\Gamma'] \vdash |K^x| \uparrow m. \]
By assumptions (i)-(iii) we have:

(i') Note that for any trees $A_i' \in C_1$ and $A_j'' \in C_2$ they have an observable offering channel occurring in $\Gamma$. Assume that $C_1$ as a forest includes $n$ separate trees. By the previous observation, we know that $C_2$ also consists of $n$ separate trees. For any $j \leq n$ consider the tree $A_i' \in C_1$ consider the corresponding tree $A_j'' \in C_2$ that exists by Def. 34 and for which we have $(\epsilon_j) \Gamma_i' \triangleright A_i' : w_{\eta_j}: A'_j[\epsilon_j] \equiv \delta \Gamma_j' \triangleright A_j'' : w_{\eta_j}: A'_j[\epsilon_j]$.

From (i), we get $\Gamma_i' = \Gamma_i^1 = \Gamma_i^2$ and for arbitrary configurations $T_1^j \in H-Client^x(\Gamma_i^1)$, and $T_2^j \in H-Client^x(\Gamma_i^2)$, we have

$$\forall m. (B_1^1C_1, B_2^1C_2) \in E[\Gamma_i' \downarrow \xi || w_{\eta_j}: A'_j[\epsilon_j]]^m$$

$$\forall m. (B_1^2C_1, B_2^2C_1) \in E[\Gamma_i' \downarrow \xi || w_{\eta_j}: A'_j[\epsilon_j]]^m$$

Observe that $\Gamma_i' = \{\Gamma_i^j\}'_{j \leq n}$ and $\Gamma_i^2 = \{\Gamma_i^j\}'_{j \leq n}$, and $\Gamma_i = \{w_{\eta_j}: A'_j[\epsilon_j]\}'_{j \leq n}$.

(ii') By Def. 17 we know that for arbitrary configurations $B_i^1 \in H-Provider^x(\Gamma_1)$, and $B_i^2 \in H-Provider^x(\Gamma_2)$, and $T_i^1 \in H-Client^x(x_{\alpha}: A_1[c_1])$, and $T_i^2 \in H-Client^x(y_{\beta}: A_2[c_2])$.

$$\forall m. (B_1^1D_1, T_1^1, B_2^1D_2, T_2^d) \in E[\Gamma_i \downarrow |K_1^1]| || K_1^1]|^m$$

$$\forall m. (B_1^2D_2, T_1^1, B_2^2D_2, T_2^d) \in E[\Gamma_i \downarrow |K_1^1]| || K_1^1]|^m$$

(iii') if $c_1 \equiv \xi$ and $c_2 \equiv \xi$, we have for any $T_1 \in H-Client^x(K_1^1)$, and $T_2 \in H-Client^x(K_2^1)$

$$\forall m. (F_1 T_1^1, F_2 T_2^d) \in E[|K_1^1| || |K_1^1|]^m$$

$$\forall m. (F_2 T_2^d, F_1 T_1^1) \in E[|K_1^1| || |K_1^1|]^m$$

It is straightforward to show that given (i'-iii'), and several applications of the compositionality lemma Corollary 2 we get the goal.

**Theorem 7 (TT-closure).** Consider $D_1 \in Tree(|\Gamma_i| \triangleright |x_{\alpha}: A_1[c_1]|)$ and $D_2 \in Tree(|\Gamma_i| \triangleright |y_{\beta}: A_2[c_2]|)$ and a given observer level $\xi \in \Psi_0$. We have

$$(\Gamma_i \triangleright D_1 \triangleright x_{\alpha}: A_1[c_1]) \equiv \delta \Gamma_i \triangleright D_2 \triangleright y_{\beta}: A_2[c_2])$$

iff

$$\forall C_1, C_2, F_1, F_2 \ s.t. (\Gamma_i \triangleright C_1[|F_1 \triangleright x_{\alpha}: A_1[c_1]|] \equiv \delta \Gamma_i \triangleright C_2[|F_2 \triangleright y_{\beta}: A_2[c_2]|]$$

we have

$$\forall B_1 \in H-Provider^x(\Gamma_1), B_2 \in H-Provider^x(\Gamma_2), T_1 \in H-Client^x(x_{\alpha}: A_1[c_1]), T_2 \in H-Client^x(y_{\beta}: A_2[c_2]).$$

Then, $B_1C_1D_1C_1F_1 \approx B_2C_2D_2F_2T_2$.

**Proof.** There are two directions to consider:

**Left to Right:** Consider $(\Gamma_i \triangleright D_1 \triangleright x_{\alpha}: A_1[c_1]) \equiv \delta \Gamma_i \triangleright D_2 \triangleright y_{\beta}: A_2[c_2])$ and arbitrary $C_1, C_2, F_1, F_2$ such that we have

$$(\Gamma_i \triangleright C_1[|F_1 \triangleright x_{\alpha}: A_1[c_1]|] \equiv \delta \Gamma_i \triangleright C_2[|F_2 \triangleright y_{\beta}: A_2[c_2]|]$$

By Def. 35, we get that $\Gamma_i = \Gamma_i \downarrow \xi = \Gamma_i \downarrow \xi$ and $K^1 = x_{\alpha}: A_1[c_1] \downarrow \xi = y_{\beta}: A_2[c_2] \downarrow \xi$. Moreover for some $\Delta$, we have $(C_1, C_2) \in Forest(\Delta \triangleright |\Gamma_i|)$ and for any low secency annotations of $\Delta$ as $\Gamma_i^1$ and $\Gamma_i^2$ we get $\Gamma_i = C_1 \equiv \delta \Gamma_i \triangleright C_2 \equiv \delta \Gamma_i \triangleright C_2$.

Also if $K^1 = \epsilon_1[\tau]$, we have $F_1 = \tau$, and otherwise $(F_1, F_2) \in Tree(K^1 \triangleright |w_{\eta}: A|$) for some $w_{\eta}A$, for any low secency annotations (with $c_3, c_4 \equiv \xi$) of $w_{\eta}A$ as $w_{\eta}: A[c_3]$ and $w_{\eta}: A[c_4]$ we get $K^1 \triangleright C_1 \equiv w_{\eta}: A[c_3] \equiv \delta \Gamma_i K^1 \triangleright C_2 \equiv w_{\eta}: A[c_4]$.

Our goal is to prove:

$$\forall B_1 \in H-Provider^x(\Gamma_1), B_2 \in H-Provider^x(\Gamma_2), T_1 \in H-Client^x(x_{\alpha}: A_1[c_1]), T_2 \in H-Client^x(y_{\beta}: A_2[c_2]).$$

$$C_1 D_1 F_1 \approx C_2 D_2 F_2.$$

By compositionality of equivalence up to relation (Lem. 26), we get $\Gamma_i^1, \Gamma_i^2 \triangleright C_1 D_1 F_1 \triangleright x_{\alpha}: A_1[c_1] \equiv \delta \Gamma_i \triangleright C_2 D_2 F_2 \triangleright y_{\beta}: A_2[c_2]$ in the case where $K^1 = \epsilon_1[\tau]$, i.e., $c_3, c_4 \not\equiv \xi$, and otherwise we get $\Gamma_i^1, \Gamma_i^2 \triangleright C_1 D_1 F_1 \triangleright w_{\eta}: A[c_1] \equiv \delta \Gamma_i \triangleright C_2 D_2 F_2 \triangleright w_{\eta}: A[c_4]$ where $\Gamma_i^1$ is the set of all channels $z_{\gamma}: C[d] \in \Gamma_1$ with $d \not\equiv \xi$ and $\Gamma_i^2$ is the set of all channels $z_{\gamma}: C[d] \in \Gamma_2$ with $d \not\equiv \xi$.

By the soundness theorem for equivalence relation (Corollary 4), we get $\Gamma_i^1, \Gamma_i^2 \triangleright C_1 D_1 F_1 \triangleright x_{\alpha}: A_1[c_1] \approx \delta \Gamma_i \triangleright C_2 D_2 F_2 \triangleright y_{\beta}: A_2[c_2]$ in the first case where $K^1 = \epsilon_1[\tau]$ and otherwise $\Gamma_i^1, \Gamma_i^2 \triangleright C_1 D_1 F_1 \triangleright w_{\eta}: A[c_1] \approx \delta \Gamma_i \triangleright C_2 D_2 F_2 \triangleright w_{\eta}: A[c_4]$. In the first case, the proof is complete by Def. 26 and the observation that all channels in $\Gamma_i^1$ and $\Gamma_i^2$ are annotated with channels of secrecy $d' \equiv \xi$ and that $c_1, c_2 \equiv \xi$ and $c_3, c_4 \equiv \xi$. In the second case, the proof is complete by Def. 26 and observing that all channels in $\Gamma_i^1$ and $\Gamma_i^2$ are annotated with channels of secrecy $d' \equiv \xi$ and that $c_1, c_2 \equiv \xi$ and $c_3, c_4 \equiv \xi$.
Right to Left: Assume \( C_1 = C_2 = \mathcal{F}_1 = \mathcal{F}_2 = \cdot \). By [Lemma 21] and [Definition 30] we know that \((\Gamma_1 \vdash \cdot \cdot x_\alpha : A_1[c_1]) \equiv (\Gamma_2 \vdash \cdot \cdot y_\beta : A_2[c_2])\). From this we get \( \Gamma_1 \vDash D_1 :: x_\alpha : A_1[c_1] \approx_\alpha \Gamma_2 \vDash D_2 :: y_\beta : A_2[c_2] \). Now, we can apply the completeness lemma (Corollary 4) to get \((\Gamma_1 \vDash D_1 :: x_\alpha : A_1[c_1]) \equiv \Psi_0 (\Gamma_2 \vDash D_2 :: y_\beta : A_2[c_2])\).

This result is enough to prove that our equivalence relation is \( \top \cdot \top \)-closed, i.e., following the results presented in the results presented in [1], it shows \( r = s^T \), when \( r \) and \( s \) defined as our logical equivalence.

References