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 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 CONSESSION.

1) Different recursive calls after branching on a high secrecy channel: In our first example, the sneaky verification process ($\mathsf{Sneaky\_Verifier}_1$) leaks, with the help of an accomplice ($\mathsf{Sneaky\_Partner}_1$), 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. $\mathsf{Sneaky\_Verifier}_1$ and its partner, $\mathsf{Sneaky\_Partner}_1$, differ on how they interact with the attacker right after being spawned: the sneaky verifier sends the label s to the attacker, signaling preceding successful authentication, and its partner sends the label 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 s and f in the $\mathsf{Sneaky\_Verifier}_1$ and $\mathsf{Sneaky\_Partner}_1$, resp., happen before the security token is received. Recursion, however, provides a way of "rolling forward" information learned in one iteration to the next.

$\mathsf{ver} = \mathsf{pin} \multimap \oplus\{succ{:}\mathsf{pin} \otimes \mathsf{ver}, fail{:}\mathsf{pin} \otimes \mathsf{ver}\}$
$\mathsf{pin} = \oplus\{tok_1{:}\mathsf{pin}, \ldots, tok_n{:}\mathsf{pin}\}$
$\mathsf{attacker} = \&\{s{:}\mathsf{attacker}, f{:}\mathsf{attacker}\}$

```
$y{:}\mathsf{attacker}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Verifier}_1 :: x{:}\mathsf{ver}[\mathbf{alice}]$
$x \leftarrow \mathsf{Sneaky\_Verifier}_1 \leftarrow y = (y.s; z \leftarrow \mathbf{recv}\, x$
  $\mathbf{case}\, z\, (tok_j \Rightarrow x.succ; \mathbf{send}\, z\, x; (x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Verifier}_1 \leftarrow y{:}\mathsf{attacker})$
           $\mid tok_{i \neq j} \Rightarrow x.fail; \mathbf{send}\, z\, x; (x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Partner}_1 \leftarrow y{:}\mathsf{attacker})))$

$y{:}\mathsf{attacker}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Partner}_1 :: x{:}\mathsf{ver}[\mathbf{alice}]$
$x \leftarrow \mathsf{Sneaky\_Partner}_1 \leftarrow y = (y.f; z \leftarrow \mathbf{recv}\, x$
  $\mathbf{case}\, z\, (tok_j \Rightarrow x.succ; \mathbf{send}\, z\, x; (x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Verifier}_1 \leftarrow y{:}\mathsf{attacker})$
           $\mid tok_{i \neq j} \Rightarrow x.fail; \mathbf{send}\, z\, x; (x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Partner}_1 \leftarrow y{:}\mathsf{attacker})))$
```

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 Diverge is an example of a process that is supposed to provide an infinite stream of terminating channel references x_1 : term, but instead calls itself recursively without ever interacting with its client.

$\mathsf{term} = 1 \otimes \mathsf{term}$

$\cdot \vdash \mathsf{Diverge} :: x_1{:}\mathsf{term}[\mathbf{guest}]$
$x_1 \leftarrow \mathsf{Diverge} \leftarrow \cdot = (x_1{:}\mathsf{term} \leftarrow \mathsf{Diverge} \leftarrow \cdot)$

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 [1].

A sneaky verifier can exploit the non-reactivity of Diverge to leak information to the attacker. In our second example, the sneaky verifier, $\mathsf{Sneaky\_Verifier}_2$, spawns the diverging process along channel x_1 and continues as its partner $\mathsf{Sneaky\_Partner}_2$. The partner only interacts with the diverging channel x_1 when Alice's PIN provides the incorrect token. As a result, if Alice provides an incorrect token, the process will wait forever for x_1 to send it a terminating channel x_2, and the attacker will receive no further messages. Otherwise, if Alice's token is correct, $\mathsf{Sneaky\_Partner}_2$ continues by recursively calling itself, and right at the beginning of the recursive call, sends the label s to the attacker, signaling the previous successful login.

[1]Farzaneh Derakhshan and Frank Pfenning. Circular proofs as session-typed processes: A local validity condition. CoRR abs/1908.01909 (2019), http://arxiv.org/abs/1908. 01909.

```
$y{:}\mathsf{attacker}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Verifier_2} :: x{:}\mathsf{ver}[\mathbf{alice}]$
$x \leftarrow \mathsf{Sneaky\_Verifier_2} \leftarrow y = ((x_1{:}\mathsf{term}[\mathbf{guest}] \leftarrow \mathsf{Diverge} \leftarrow \cdot);$
                                        $(x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Partner_2} \leftarrow y{:}\mathsf{attacker}, x_1{:}\mathsf{term}))$
```

```
$y{:}\mathsf{attacker}[\mathbf{guest}], x_1{:}\mathsf{term}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Partner_2} :: x{:}\mathsf{ver}[\mathbf{alice}]$
$x \leftarrow \mathsf{Sneaky\_Partner_2} \leftarrow y, x_1 =$
$(y.s; z \leftarrow \mathbf{recv}\, x$ // $y{:}\mathsf{attacker}, z{:}\mathsf{pin}, x_1{:}\mathsf{term} \vdash x{:} \oplus\{succ : \mathsf{pin} \otimes \mathsf{ver}, fail{:}\mathsf{pin} \otimes \mathsf{ver}\}$
  $\mathbf{case}\, z\, (tok_j \Rightarrow x.succ; \mathbf{send}\, z\, x; (x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Partner_2} \leftarrow y{:}\mathsf{attacker}, x_1{:}\mathsf{term})$
          $\mid tok_{i\neq j} \Rightarrow x.fail; \mathbf{send}\, z\, x; x_2 \leftarrow \mathbf{recv}\, x_1; \mathbf{wait}\, x_2;$
              $(x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Partner_2} \leftarrow y{:}\mathsf{attacker}, x_1{:}\mathsf{term})))$
```

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 $b0; b1; b0; b1; \ldots$ and its two mutually recursive partners, Sneaky_Partner$_{\mathsf{b0}}$ and Sneaky_Partner$_{\mathsf{b1}}$.

$\mathsf{bits} = \oplus\{b0 : \mathsf{bits}, b1 : \mathsf{bits}\}$
$\mathsf{cobits} = \&\{b0 : \mathsf{bits}, b1 : \mathsf{bits}\}$
$\mathsf{leak} = \&\{succ : \mathsf{leak}, fail : \mathsf{leak}\}$

```
$x_1{:}\mathsf{bits}[\mathbf{guest}], x_2{:}\mathsf{cobits}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Verifier_3} :: x{:}\mathsf{ver}[\mathbf{alice}]$
$x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2 =$
$\mathbf{case}\, x_1\, (b0 \Rightarrow x_2.b0; z \leftarrow \mathbf{recv} x; \mathbf{case}\, z\ \ (tok_j \Rightarrow x.succ; \mathbf{send}\, z\, x;$
                                    $\mathbf{case}\, x_1 (b0 \Rightarrow x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2,$
                                           $\mid b1 \Rightarrow x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2)$
                                 $\mid tok_{i\neq j} \Rightarrow x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2)$
       $\mid b1 \Rightarrow x_2.b1; z \leftarrow \mathbf{recv} x; \mathbf{case} z\, (tok_j \Rightarrow x.succ; \mathbf{send}\, z\, x;$
                                    $\mathbf{case}\, x_1 (b0 \Rightarrow x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2,$
                                           $\mid b1 \Rightarrow x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2)$
                                 $\mid tok_{i\neq j} \Rightarrow x \leftarrow \mathsf{Sneaky\_Verifier_3} \leftarrow x_1, x_2)))$
```

```
$\cdot \vdash \mathsf{Alternate\_bits} :: x_1{:}\mathsf{bits}[\mathbf{guest}]$
$x_1 \leftarrow \mathsf{Alternate\_bits} \leftarrow \cdot = (x_1.b0; x_2.b1; (x_1{:}\mathsf{bits} \leftarrow \mathsf{Alternate\_bits} \leftarrow \cdot))$
```

```
$x_3{:}\mathsf{leak}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Partner_{b0}} :: x_2{:}\mathsf{cobits}[\mathbf{guest}]$
$x_2 \leftarrow \mathsf{Sneaky\_Partner_{b0}} \leftarrow x_3 = \mathbf{case}\, x_2 (b0 \Rightarrow x_3.succ; x_2 \leftarrow \mathsf{Sneaky\_Partner_{b0}} \leftarrow x_3;$
                                      $\mid b1 \Rightarrow x_3.fail; x_2 \leftarrow \mathsf{Sneaky\_Partner_{b1}} \leftarrow x_3))$
```

```
$x_3{:}\mathsf{leak}[\mathbf{guest}] \vdash \mathsf{Sneaky\_Partner_{b1}} :: x_2{:}\mathsf{cobits}[\mathbf{guest}]$
$x_2 \leftarrow \mathsf{Sneaky\_Partner_{b1}} \leftarrow x_3 = \mathbf{case}\, x_2 (b0 \Rightarrow x_3.fail; x_2 \leftarrow \mathsf{Sneaky\_Partner_{b0}} \leftarrow x_3;$
                                      $\mid b1 \Rightarrow x_3.succ; x_2 \leftarrow \mathsf{Sneaky\_Partner_{b1}} \leftarrow x_3))$
```

At the beginning, the Sneaky_Verifier$_3$ is connected to Alternate_bits along channel x_1, and to Sneaky_Partner$_{\mathsf{b0}}$ along channel x_2. The sneaky verifier starts by relaying the first bit ($b0$) that it receives from x_1 to its accomplice along x_2. 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_Partner$_{\mathsf{b0}}$ 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_Partner$_{\mathsf{b0}}$ 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 x_2 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 IFC type system. However, in contrast to the non-recursive case, causality between actions can no longer 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-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 Ψ contains the constraints $\psi_1 = \mathbf{guest}$ and $\psi = \mathbf{alice}$.

$\Psi; y{:}\mathsf{attacker}[\psi_1] \vdash \mathsf{Sneaky\_Verifier}_1 @\psi' :: x{:}\mathsf{ver}[\psi]$
$x \leftarrow \mathsf{Sneaky\_Verifier}_1 \leftarrow y = (y.s; z \leftarrow \mathbf{recv}\, x$
$\quad \mathbf{case}\, z\, (tok_j \Rightarrow x.succ; \mathbf{send}\, z\, x; x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Verifier}_1 @d_2 \leftarrow y{:}\mathsf{attacker}$
$\qquad\quad \mid tok_{i\neq j} \Rightarrow x.fail; \mathbf{send}\, z\, x; x{:}\mathsf{ver} \leftarrow \mathsf{Sneaky\_Partner}_1 @d_3 \leftarrow y{:}\mathsf{attacker}))@\psi'$

Following the signature typing rule, the security theory Ψ, must satisfy $\Psi \Vdash \mathbf{guest} = \psi_1 \sqsubset \mathbf{alice} = \psi$, $\psi' \sqsubseteq \psi = \mathbf{alice}$. Moreover, by $\&L$, we must have $\Psi \Vdash \psi' \sqsubseteq \psi_1$, to execute $y.s$. By $\multimap R$, we know that the running secrecy of the process after executing $z \leftarrow \mathbf{recv}\, x^{\psi}$ increases to ψ. 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 γ to unify d_2 with ψ' as required by the spawn rule, i.e. $\hat{\gamma}(\Psi) \nVdash d_2 \sqsubseteq \psi_1 = \mathbf{guest} \sqsubset \mathbf{alice} = \psi \sqsubseteq d_2$.

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.

$\mathsf{attacker} = \&\{s{:}\mathsf{attacker}, f{:}\mathsf{attacker}\}$
$\Psi; y{:}\mathsf{attacker}[\psi_1] \vdash \mathsf{SneakyVerifier} @\psi' :: x{:}\mathsf{ver}[\psi]$
$x \leftarrow \mathsf{SneakyVerifier} \leftarrow y = (z \leftarrow \mathbf{recv}\, x;$
$\ \mathbf{case}\, z\, (tok_j \Rightarrow x.succ; y.s; \mathbf{send}\, z\, x;$
$\qquad\qquad x^{[\hat{\gamma}(\psi_1)]} \leftarrow \mathsf{SneakyVerifier}[\gamma]@\hat{\gamma}(\psi') \leftarrow y$
$\qquad \mid tok_{i\neq j} \Rightarrow x.fail; y.f; \mathbf{send}\, z\, x;$
$\qquad\qquad x^{[\hat{\gamma_1}(\psi_1)]} \leftarrow \mathsf{SneakyVerifier}[\gamma_1]@\hat{\gamma_1}(\psi') \leftarrow y))@\psi'$

Assume that the security theory Ψ contains the constraints $\psi_1 = \mathbf{guest}$ and $\psi = \mathbf{alice}$. By Σ_3, Ψ must satisfy $\Psi \Vdash \mathbf{guest} = \psi_1 \sqsubset \mathbf{alice} = \psi$ and $\Psi \Vdash \psi' \sqsubseteq \psi = \mathbf{alice}$. Moreover, by $\&L$, we must have $\Psi \Vdash \psi' \sqsubseteq \psi_1$, to execute $y.s$. By $\multimap R$, we know that the running secrecy of the process after executing $z \leftarrow \mathbf{recv}\, x^{\psi}$ increases to ψ. And by SPAWN, we need to know that the running secrecy $\hat{\gamma}(\psi')$ specified for the callee is at least as high as the caller's, i.e. $\Psi \Vdash \psi \sqsubseteq \hat{\gamma}(\psi')$. Moreover, the caller must satisfy the instantiation of the callee's constraints, i.e., $\Psi \Vdash \gamma : \Psi$. and thus $\Psi \Vdash \hat{\gamma}(\psi') \sqsubseteq \hat{\gamma}(\psi_1) = \mathbf{guest}$ must hold. It is straightforward to observe that there is no possible substitution $\Psi \Vdash \gamma : \Psi$ that satisfies all these requirements, i.e., $\Psi \Vdash \hat{\gamma}(\psi') \sqsubseteq \hat{\gamma}(\psi_1) = \mathbf{guest} \sqsubset \mathbf{alice} = \psi \sqsubseteq \hat{\gamma}(\psi')$.

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 session types to our example:

$\mathsf{customer} = \mathsf{pin} \multimap \mathsf{auth_{out}} \multimap 1$
$\mathsf{pin} = \oplus\{tok_1{:}\mathsf{pin}, \ldots, tok_n{:}\mathsf{pin}\}$
$\mathsf{auth_{out}} = \mathsf{pin} \multimap \oplus\{succ{:}\mathsf{account} \otimes \mathsf{auth_{in}}, fail{:}\mathsf{pin} \otimes \mathsf{auth_{out}}\}$
$\mathsf{auth_{in}} = \mathsf{account} \multimap \mathsf{pin} \otimes \mathsf{auth_{out}}$
$\mathsf{ver} = \mathsf{pin} \multimap \oplus\{succ{:}\mathsf{pin} \otimes \mathsf{ver}, fail{:}\mathsf{pin} \otimes \mathsf{ver}\}$
$\mathsf{account} = \oplus\{high{:}\mathsf{account}, med{:}\mathsf{account}, low{:}\mathsf{account}\}$

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

$\Psi; y_1{:}\,\mathsf{customer}[\psi_1], u_1{:}\,\mathsf{pin}[\psi_1], z_1{:}\mathsf{auth_{out}}[\psi_1], \quad \Psi{:=}\psi_1{=}\mathbf{alice}, \psi_2{=}\mathbf{bob}, \psi{=}\mathbf{bank}, \psi'{=}\mathbf{guest}, \Psi_0$
$\quad y_2{:}\,\mathsf{customer}[\psi_2], u_2{:}\mathsf{pin}[\psi_2], z_2{:}\mathsf{auth_{out}}[\psi_2] \vdash \mathsf{Bank} :: w{:}1[\psi]$
$w \leftarrow \mathsf{Bank}@\psi' \leftarrow y = (\ \mathbf{send}\, u_1^{\psi_1}\, y_1; \mathbf{send}\, z_1^{\psi_1}\, y_1; \mathbf{send} u_2^{\psi_2}\, y_2; \mathbf{send}\, z_2^{\psi_2}\, y_2; \mathbf{wait}\, y_1; \mathbf{wait} y_2; \mathbf{close} w)$

$\Psi_1; \cdot \vdash \mathsf{New\_Customer}@\psi' :: y{:}\,\mathsf{customer}[\psi] \qquad \Psi_1 := \psi' \sqsubseteq \psi, \Psi_0$
$y \leftarrow \mathsf{New\_Customer} \leftarrow \cdot = (\ u^{\psi} \leftarrow \mathbf{recv}\, y; z^{\psi} \leftarrow \mathbf{recv}\, y; \ //\ u{:}\mathsf{pin}[\psi], z{:}\mathsf{auth_{out}}[\psi] \vdash y{:}1[\psi]$
$\qquad\qquad y^{\hat{\gamma}(\psi)}{:}1 \leftarrow \mathsf{Customer}[\gamma_1]@\hat{\gamma}(\psi') \leftarrow u{:}\mathsf{pin}, z{:}\mathsf{auth_{out}})@\psi'$

where $\gamma_1 := \{\psi \mapsto \mathbf{bank}, \psi_1 \mapsto \mathbf{alice}, \psi_2 \mapsto \mathbf{bob}, \psi' \mapsto \mathbf{guest}\}$.

The types $\mathsf{auth_{out}}$ and $\mathsf{auth_{in}}$ 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.

$$\begin{array}{l}
\Psi_1; x{:}\mathsf{ver}[\psi], v{:}\mathsf{account}[\psi] \vdash \mathsf{Auth_{out}}@\psi' :: z{:}\,\mathsf{auth_{out}}[\psi] \qquad \Psi_1 = \psi' \sqsubseteq \psi, \Psi_0\\
z{:}\mathsf{auth_{out}} \leftarrow \mathsf{Auth_{out}} \leftarrow x{:}\mathsf{ver}, v{:}\mathsf{account} = (\ //\ x{:}\mathsf{ver}, v{:}\mathsf{account} \vdash z{:}\mathsf{auth_{out}}\\
\quad w_3^\psi \leftarrow \mathbf{recv}\, z; \mathbf{send} w_3^\psi x;\\
\quad \mathbf{case}\, x\, (succ \Rightarrow z.succ; \mathbf{send} v^\psi z;\ //\ x{:}\mathsf{pin} \otimes \mathsf{ver} \vdash z{:}\mathsf{auth_{in}}\\
\qquad\qquad\qquad\qquad (z^\psi{:}\mathsf{auth_{in}} \leftarrow \mathsf{Auth_{in}}[\gamma_2]@\psi \leftarrow x{:}\mathsf{pin} \otimes \mathsf{ver})\\
\qquad\qquad\quad \mid fail \Rightarrow w_4^\psi \leftarrow \mathbf{recv} x; \mathbf{send} w_4^\psi z;\ //\ x{:}\mathsf{ver}, v{:}\mathsf{account} \vdash z{:}\mathsf{auth_{out}}\\
\qquad\qquad\qquad\qquad (z^\psi{:}\mathsf{auth_{out}} \leftarrow \mathsf{Auth_{out}}[\gamma_2]@\psi \leftarrow x{:}\mathsf{ver}, v{:}\mathsf{account})))@\psi'
\end{array}$$

$$\begin{array}{l}
\Psi_1; x{:}\mathsf{pin} \otimes \mathsf{ver}[\psi] \vdash \mathsf{Auth_{in}}@\psi' :: z{:}\,\mathsf{auth_{in}}[\psi] \qquad \Psi_1 = \psi' \sqsubseteq \psi, \Psi_0\\
z{:}\mathsf{auth_{in}} \leftarrow \mathsf{Auth_{in}} \leftarrow x{:}\mathsf{pin} \otimes \mathsf{ver} = (\\
\quad v^\psi \leftarrow \mathbf{recv}\, z; w_5^\psi \leftarrow \mathbf{recv} x; \mathbf{send} w_5^\psi z;\ //\ x{:}\mathsf{ver}, v{:}\mathsf{account} \vdash z{:}\mathsf{auth_{out}}\}\\
\quad (z^\psi{:}\mathsf{auth_{out}} \leftarrow \mathsf{Auth_{out}}[\gamma_2]@\psi \leftarrow x{:}\mathsf{ver}, v{:}\mathsf{account})))@\psi'
\end{array}$$

$$\begin{array}{l}
\Psi_1; u{:}\mathsf{pin}[\psi], z{:}\mathsf{auth_{out}}[\psi] \vdash \mathsf{Customer}@\psi' :: y{:}\,1[\psi] \qquad \Psi_1 = \psi' \sqsubseteq \psi, \Psi_0\\
y \leftarrow \mathsf{Customer} \leftarrow \cdot = (\ //\ u{:}\mathsf{pin}, z{:}\mathsf{auth_{out}} \vdash y{:}\mathsf{customer}\\
\mathbf{send} u^\psi z; \mathbf{case}\, z\, (succ \Rightarrow v^\psi \leftarrow \mathbf{recv}\, z;\ //\ v{:}\mathsf{account}, z{:}\mathsf{auth_{in}} \vdash y{:}1\\
\qquad \mathbf{case}\, v\, (high \Rightarrow \mathbf{send} v^\psi\, z,\, w_6^\psi \leftarrow \mathbf{recv}\, z; (y^\psi{:}\, 1 \leftarrow \mathsf{Customer}[\gamma_2]\, @\psi \leftarrow w_6{:}\mathsf{pin}, z{:}\mathsf{auth_{out}})\\
\qquad\qquad\quad \mid med \Rightarrow \mathbf{send} v^\psi\, z,\, w_6^\psi \leftarrow \mathbf{recv}\, z; (y^\psi{:}\, 1 \leftarrow \mathsf{Customer}[\gamma_2]\, @\psi \leftarrow w_6{:}\mathsf{pin}, z{:}\mathsf{auth_{out}})\\
\qquad\qquad\quad \mid low \Rightarrow \mathbf{send} v^\psi\, z,\, w_6^\psi \leftarrow \mathbf{recv}\, z; (y^\psi{:}\, 1 \leftarrow \mathsf{Customer}[\gamma_2]\, @\psi \leftarrow w_6{:}\mathsf{pin}, z{:}\mathsf{auth_{out}})\\
\qquad\qquad\qquad\quad \mid fail \Rightarrow\ w_6^\psi \leftarrow \mathbf{recv}\, z; (y^\psi{:}1 \leftarrow \mathsf{Customer}[\gamma_2]\, @\psi \leftarrow w_6{:}\mathsf{pin}, z{:}\mathsf{auth_{out}})))@\psi'
\end{array}$$

where $\gamma_2 := \{\psi \mapsto \psi, \psi' \mapsto \psi\}$.

The account provides a customer's balance by signaling the corresponding label along v:account. For example, Alice's account sends label *high* to her after being authorized.

$$\begin{array}{l}
\Psi_2; \cdot \vdash \mathsf{aAccount}@\psi' :: v{:}\mathsf{account}[\psi] \qquad \Psi_2 := \psi = \mathbf{alice}, \psi' \sqsubseteq \psi, \Psi_0\\
\quad v \leftarrow \mathsf{aAccount} \leftarrow \cdot = (v.high;\\
\qquad v^\psi{:}\mathsf{account} \leftarrow \mathsf{aAccount}[\gamma_3]@\psi' \leftarrow \cdot)@\psi'
\end{array}$$

where $\gamma_3 := \{\psi \mapsto \psi, \psi' \mapsto \psi'\}$. We repeat the definition of Alice's verifier and her pin from the main part of the paper here for convenience.

$$\begin{array}{l}
\Psi_2; \cdot \vdash \mathsf{aVerifier}@\psi' :: x{:}\mathsf{ver}[\psi] \qquad \Psi_2 := \psi = \mathbf{alice}, \psi' \sqsubseteq \psi, \Psi_0\\
x \leftarrow \mathsf{aVerifier} \leftarrow \cdot = (z^\psi \leftarrow \mathbf{recv}\, x\ //\ z{:}\mathsf{pin}[\psi] \vdash x{:} \oplus \{succ : \mathsf{pin}[\psi] \otimes \mathsf{ver}, fail{:}\mathsf{pin}[\psi] \otimes \mathsf{ver}\}[\psi]\\
\quad \mathbf{case}\, z\, (tok_j \Rightarrow x.succ; \mathbf{send}\, z^\psi\, x; (x^\psi{:}\mathsf{ver} \leftarrow \mathsf{aVerifier}[\gamma_2]\, @\psi \leftarrow \cdot)\\
\qquad\qquad \mid tok_{i \neq j} \Rightarrow x.fail; \mathbf{send}\, z^\psi\, x; (x^\psi{:}\mathsf{ver} \leftarrow \mathsf{aVerifier}[\gamma_2]\, @\psi \leftarrow \cdot)))@\psi'
\end{array}$$

$$\begin{array}{l}
\Psi_2; \cdot \vdash \mathsf{apin}@\psi' :: u{:}\mathsf{pin}[\psi] \qquad \Psi_2 := \psi = \mathbf{alice}, \psi' \sqsubseteq \psi, \Psi_0\\
u \leftarrow \mathsf{apin} \leftarrow \cdot = (u.tok_j; (u^\psi{:}\mathsf{pin} \leftarrow \mathsf{apin}[\gamma_3]\, @\psi' \leftarrow \cdot)))@\psi'
\end{array}$$

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.

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

Fig. 1: Abstract syntax of SESSION.

II. Abstract syntax

Fig. 1 defines the abstract syntax of SESSION. Lines without a left-hand side are separated by $\mid$ from their preceding line.

III. A TYPING SYSTEM FOR SESSION TYPES

a) External and Internal Choice:

$$\frac{\Omega \vdash P :: y{:}A_k \qquad k \in L}{\Omega \vdash y.k; P :: y{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}} \oplus R$$

$$\frac{\Omega, x{:}A_k \vdash Q_k :: y{:}C \quad \forall k \in L}{\Omega, x{:} \oplus \{\ell : A_\ell\}_{\ell \in L} \vdash \mathbf{case}\, x(\ell \Rightarrow Q_\ell)_{\ell \in L} :: y{:}C} \oplus L$$

$$\frac{\Omega \vdash Q_k @ c :: y{:}A_k \qquad \forall k \in L}{\Omega \vdash \mathbf{case}\, y(\ell \Rightarrow Q_\ell)_{\ell \in I} :: y{:}\&\{\ell : A_\ell\}_{\ell \in L}} \& R$$

$$\frac{\Omega, x{:}A_k \vdash P :: y{:}C \qquad k \in L}{\Omega, x{:}\&\{\ell : A_\ell\}_{\ell \in I} \vdash (x.k; P) :: y{:}C} \& L$$

b) Channel input/output:

$$\frac{\Omega \vdash P :: y{:}B}{\Omega, z{:}A \vdash \mathbf{send}\, z\, y; P :: y{:}A \otimes B} \otimes R$$

$$\frac{\Omega, z{:}A, x{:}B \vdash P :: y{:}C}{\Omega, x{:}A \otimes B \vdash z \leftarrow \mathbf{recv}\, x; P :: y{:}C} \otimes L$$

$$\frac{\Omega, z{:}A \vdash P :: y{:}B}{\Omega \vdash z \leftarrow \mathbf{recv}\, y; P :: y{:}A \multimap B} \multimap R$$

$$\frac{\Omega, x{:}B \vdash P :: y{:}C}{\Omega, z{:}A, x{:}A \multimap B \vdash \mathbf{send}\, z\, x; P :: y{:}C} \multimap L$$

c) Spawn and process definition:

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

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

d) Termination::

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

e) Silent unfolding::

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

f) Signature checking::

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

g) Message Typing:

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

$$\frac{k \in L}{y_{\alpha+1}{:}A_k \vdash y_\alpha.k :: y_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in L}} \oplus R \qquad \frac{k \in L}{x_\alpha{:}\&\{\ell : A_\ell\}_{\ell \in I} \vdash (x_\alpha.k; P) :: x_{\alpha+1}{:}A_k} \& L \qquad \frac{}{\cdot \vdash \mathbf{close}\, y_\alpha :: y_\alpha{:}1} 1R$$

h) Configuration Typing:

$$\frac{}{x_\alpha{:}A \Vdash \cdot :: (x_\alpha{:}A)}\ \mathbf{emp_1} \qquad \frac{}{\cdot \Vdash \cdot :: (\cdot)}\ \mathbf{emp_2}$$

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

$$\frac{\Delta_0 \Vdash \mathcal{C} :: \Delta \qquad \Delta_0', \Delta \vdash M :: (x_\alpha{:}A)}{\Delta_0, \Delta_0' \Vdash \mathcal{C}, \mathbf{msg}(M) :: (x_\alpha{:}A)}\ \mathbf{msg}$$

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

Since Σ 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 Σ by adding the following definition to the signature Σ:*

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

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

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

Lemma 1. *Given the extended signature Σ, for all type A, there is a derivation for $x : A \vdash_\Sigma \mathtt{Fwd}_{A,y\leftarrow x} :: y : A$.*

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

□

Definition 2 (Well-typed configuration)**.** *Well-typed configuration(s) are defined in terms of the judgments $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $\mathcal{D} \in \mathsf{Tree}(\Delta \Vdash K)$, where K is either of the form $x{:}A$ or _:1 and _ stands for an arbitrary channel name along which no observations are made.*

- *We define $\mathcal{D} \in \mathsf{Tree}(\Delta \Vdash K)$ as*
$$\Delta \Vdash \mathcal{D} :: K$$
- *We define $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ as $\mathcal{D}_1 \in \mathsf{Tree}(\Delta \Vdash K)$ and $\mathcal{D}_2 \in \mathsf{Tree}(\Delta \Vdash K)$*
- *We define $\mathcal{B} \in \mathsf{Forest}(\Delta \Vdash \Delta')$ as*
$$\Delta \Vdash \mathcal{B} :: \Delta'$$
- *We define the notation $\mathcal{T} \in \mathcal{B}$ meaning $\mathcal{T}$ is a particular tree in a forest of trees $\mathcal{B}$: For $\mathcal{B} \in \mathsf{Forest}(\Delta \Vdash \Delta')$, we write $\mathcal{T} \in \mathcal{B}$ iff $\mathcal{B} = \mathcal{B}'\mathcal{T}$, and $\mathcal{B}' \in \mathsf{Forest}(\Delta_1 \Vdash \Delta_1')$ and $\mathcal{T}' \in \mathsf{Tree}(\Delta_2 \Vdash K)$ with $\Delta = \Delta_1, \Delta_2$ and $\Delta' = \Delta_1', K$.*

◇

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

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

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 Ψ_0 is defined globally for an application and consists of concrete security levels ι. Our running example lattice

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

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

Definition 3 (Concrete security lattice and security theory)**.** *Let $\Psi_0 \triangleq \langle \mathcal{L}, \mathcal{E}_0, \sqcup \rangle$ be a concrete join semi-lattice with a partial order $\mathcal{E}_0$ over concrete security levels $\iota, \eta \in \mathcal{L}$ such that $E_0 \in \mathcal{E}_0$ is of the form $\iota \sqsubseteq \iota'$. We define a security theory $\Psi \triangleq \langle \mathcal{V}, \mathcal{E} \rangle$ that augments Ψ_0 with security variables ψ and relations $\mathcal{E}$ over them such that*
- *$\psi \in \mathcal{V}$ is a set of security variables*
- *$c, d \in \mathcal{S}$ is a set of security terms of the security theory Ψ, defined by the grammar*

$$c, d := c \sqcup d \mid \iota \mid \psi$$

- *$E \in \mathcal{E}$ is a set of relations over the elements c, d of the security theory where $E = c \sqsubseteq d$.*

For convenience, we define the projection $\mathcal{E}(\Psi)$ to extract the relations $\mathcal{E}$ of Ψ. We write

$$\Psi \Vdash E$$

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

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

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 Δ, and security lattices Ψ, is defined by structural induction over the syntactic object, replacing simultaneously all variable occurrences ψ_i in the object with $\gamma(\psi_i)$. The function γ and its lifting $\hat{\gamma}$ must be 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 $\Psi \Vdash \gamma : \Psi'$*

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

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

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

V. A SECURE TYPING SYSTEM

a) External and Internal Choice:

$$\frac{\Psi;\Xi \vdash P@d_1 :: y{:}A_k[c] \qquad k \in L}{\Psi;\Xi \vdash (y^c.k; P)@d_1 :: y{:}\oplus\{\ell{:}A_\ell\}_{\ell\in L}[c]}\ \oplus R$$

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

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

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

b) Channel input/output:

$$\frac{\Psi;\Xi \vdash P@d_1 :: y{:}B[c]}{\Psi;\Xi, z{:}A[c] \vdash (\mathbf{send}\, z\, y^c; P)@d_1 :: y{:}A \otimes B[c]}\ \otimes R$$

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

$$\frac{\Psi;\Xi, z{:}A[c] \vdash P@c :: y{:}B[c]}{\Psi;\Xi \vdash (z \leftarrow \mathbf{recv}\, y^c; P)@d_1 :: y{:}A \multimap B[c]}\ R \multimap$$

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

c) Definition and spawn:

$$\frac{\begin{array}{c}\Psi';\Xi'_1 \vdash_\Sigma X = P@\psi_0 :: x'{:}A[\psi] \in \Sigma \qquad \Psi \Vdash \gamma_{\mathsf{sec}} : \Psi' \qquad \Psi \Vdash \hat{\gamma_{\mathsf{sec}}}(\psi) \sqsubseteq d \\ \Psi \Vdash d_0 \sqsubseteq \hat{\gamma_{\mathsf{sec}}}(\psi_0) \qquad \Xi_1, x{:}A[\hat{\gamma_{\mathsf{sec}}}(\psi_0)] \Vdash (\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}}) :: \Xi'_1, x'{:}A[\psi] \qquad \Psi;\Xi_2, x{:}A[\hat{\gamma_{\mathsf{sec}}}(\psi)] \vdash_\Sigma Q@d_0 :: z{:}C[d]\end{array}}{\Psi;\Xi_1,\Xi_2 \vdash_\Sigma ((x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[(\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})] \leftarrow \Xi_1)@\hat{\gamma_{\mathsf{sec}}}(\psi_0); Q_x)@d_0 :: z{:}C[d]}\ \textsc{Spawn}$$

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

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

Here is the forward rule we use in the implementation:

$$\frac{\Psi \Vdash c_1 = c_2}{\Psi; y{:}A[c_1] \vdash_\Sigma (x^{c_2} \leftarrow y^{c_1})@c_0 :: x{:}A[c_2]}\ \textsc{Fwd}$$

d) Termination Rules:

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

e) Silent unfolding::

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

f) Signature checking:

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

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

g) Message Typing:

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

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

$$\frac{}{\Psi_0; \cdot \vdash \mathbf{close}\, y_\alpha :: y_\alpha{:}1[c]}\ 1R$$

h) Configuration Typing:

$$\frac{}{\Psi_0; x{:}A[d] \Vdash \cdot :: (x{:}A[d])}\ \mathbf{emp}_1 \qquad \frac{}{\Psi_0; \cdot \Vdash \cdot :: (\cdot)}\ \mathbf{emp}_2$$

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

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

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

Since Ψ_0 and Σ are fixed, we may drop Ψ_0 and Σ in a configuration typing judgment and a process typing judgment, respectively, for brevity.

Definition 5. *For all $Y = A \in \Sigma$, we extend Σ by adding the followingIFC-typed forwarder definition to the signature Σ:*

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

Here F_Y is a specific process variable assigned to the forwarder process for type variable Y, and $\mathtt{Fwd}_{A, y \leftarrow x}$ is defined similar to Def. 1 as

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

Lemma 2. *Given the extended signature Σ, for all type A, there are derivations for*

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

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

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

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

□

Definition 6. *Define security-erasure $|\Sigma|$ of the signature Σ as*

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

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

$$\begin{array}{lcl} |\Xi, x{:}A[c]| & \overset{\text{def}}{=} & |\Xi|, x{:}A \\ |x{:}A[c]| & \overset{\text{def}}{=} & x{:}A \\[4pt] |\Gamma, x_\alpha{:}A[c]| & \overset{\text{def}}{=} & |\Gamma|, x_\alpha{:}A \\ |\cdot| & \overset{\text{def}}{=} & \cdot \\[4pt] |x_\alpha{:}A[c]| & \overset{\text{def}}{=} & x_\alpha{:}A \\ |\_{:}1[\top]| & \overset{\text{def}}{=} & \_{:}1 \end{array}$$

Definition 8. *Define a security-erasure $|\mathbb{C}|$ of the configuration $\mathbb{C}$ as*

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

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

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

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

Fig. 3: Asynchronous dynamics of SESSION

VI. Progress and Preservation

A. Session-typed processes

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

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

Definition 9 (Poised Configuration). *A configuration $\Delta_1, \Delta_2 \Vdash \mathcal{C}_1, \mathcal{C}_2 :: \Lambda, w{:}A'$ is poised iff either $\mathcal{C}_1, \mathcal{C}_2$ is empty or $\Delta_1 \Vdash \mathcal{C}_1 :: \Lambda$ is poised and $\Delta_2 \Vdash \mathcal{C}_2 :: w{:}A'$ is poised. The configuration $\Delta_2 \Vdash \mathcal{C}_2 :: w{:}A'$ is poised iff it cannot take any steps and at least one of the following conditions hold:*
1) *$\mathcal{C}_2$ is an empty configuration.*
2) *$\mathcal{C}_2 = \mathcal{C}_2'\, \mathbf{msg}(M)\, \mathcal{C}_2''$ such that $\mathbf{msg}(M)$ is a negative message along $y_\alpha \in \Delta_2$, i.e. $y_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell \in L} \Vdash \mathbf{msg}(M) :: y_{\alpha+1}{:}A_k$ or $y_\alpha{:}A \multimap B, z_\beta{:}A \Vdash \mathbf{msg}(M) :: y_{\alpha+1}{:}B$, and both subconfigurations $\mathcal{C}_2'$ and $\mathcal{C}_2''$ are poised.*
3) *$\mathcal{C}_2 = \mathcal{C}_2'\, \mathbf{proc}(x, P)\, \mathcal{C}_2''$ such that $\mathbf{proc}(x, P)$ attempts to receive along a channel $y_\alpha {\in} \Delta_2$, and both subconfigurations $\mathcal{C}_2'$ and $\mathcal{C}_2''$ are poised.*
4) *$\mathcal{C}_2 = \mathcal{C}_2'\, \mathbf{msg}(P)$ such that $\mathbf{msg}(M)$ is a positive message sent along $w_\beta{:}A'$, i.e. $w_{\beta+1}{:}A_k \Vdash \mathbf{msg}(M) :: w_\beta{:}\oplus\{\ell{:}A_\ell\}_{\ell \in L}$ or $w_{\beta+1}{:}B, z_\gamma{:}A \Vdash \mathbf{msg}(M) :: w_\beta{:}A \otimes B$,or $\cdot \Vdash \mathbf{msg}(M) :: w{:}1$, and subconfiguration $\mathcal{C}_2'$ is poised.*
5) *$\mathcal{C}_2 = \mathcal{C}_2'\, \mathbf{proc}(w, P)$ such that $\mathbf{proc}(w, P)$ attempts to receive along $w{:}A'$, and subconfiguration $\mathcal{C}_2'$ is poised.*

◇

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

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

using rule $\otimes_{\mathsf{snd}}$ (see Fig. 3), yields the configuration

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

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

Definition 10 (Valid configuration permutation). *For a well-typed configuration $\Psi_0; \Delta \Vdash \mathcal{C} :: \Delta'$, a valid permutation $\mathcal{P}(\mathcal{C})$ can be derived by simultaneously changing the position of a process or message in $\mathcal{C}$, yielding the permutation $\mathcal{C}'$, i.e., $\mathcal{P}(\mathcal{C}) = \mathcal{C}'$, as long as the following conditions are met:*
1) *For a process $\mathbf{proc}(z_\alpha, P)$ in $\mathcal{C}'$ such that $\mathcal{C}' = \mathcal{C}_1 \mathbf{proc}(z_\alpha, P) \mathcal{C}_2$ and for all $y_\beta \notin \mathsf{dom}(\Delta)$ that P is using, there must exist either a process $\mathbf{proc}(y_\beta, \_)$ or a message $\mathbf{msg}(\_\langle y_\beta \rangle)$ in $\mathcal{C}_1$.*
2) *For a positive message $\mathbf{msg}(M\langle z_\alpha \rangle)$ in $\mathcal{C}'$ such that $\mathcal{C}' = \mathcal{C}_1 \mathbf{msg}(M\langle z_\alpha \rangle) \mathcal{C}_2$, if $z_\alpha \notin \mathsf{dom}(\Delta)$, there must exist either a process $\mathbf{proc}(z_\alpha, \_)$ or a message $\mathbf{msg}(\_\langle z_\alpha \rangle)$ in $\mathcal{C}_1$. Moreover, for all $y \notin \mathsf{dom}(\Delta)$ that P is using, there must exist either a process $\mathbf{proc}(y, \_)$ or a message $\mathbf{msg}(y.\_)$ in $\mathcal{C}_1$.*

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

◇

2) Lemmas:

Lemma 3 (Term variable substitution)**.** *The following substitutions are type-preserving and thus admissible:*
1) *If* $\Delta \vdash_\Sigma P :: x : A$ *then, for any fresh* $y : A$*, we have* $\Delta \vdash_\Sigma [y/x]P :: y : A$*.*
2) *If* $\Delta, y : B \vdash_\Sigma P :: x : A$ *then, for any fresh* $z : B$*, we have* $\Delta, z : B \vdash_\Sigma [z/y]P :: x : A$*.*

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

The next lemma allows us to break up an open forest into two sub-forests, as illustrated in Fig. 4.

Lemma 4 (Making two forests out of one)**.** *If* $\Delta \Vdash \mathcal{C}\,\mathcal{C}' :: \Delta'$*, then for some* Δ_1 *we have* $\Lambda_1 \Vdash \mathcal{C} :: \Lambda_1', \Delta_1$ *and* $\Lambda_2, \Delta_1 \Vdash \mathcal{C}' :: \Lambda_2'$*, where* $\Delta = \Lambda_1, \Lambda_2$ *and* $\Delta' = \Lambda_1', \Lambda_2'$*.*

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

3) Progress:

Theorem 2 (Progress)**.** *For any configuration* $\mathcal{C}$*, if* $\mathcal{C}$ *is a valid permutation of a configuration* $\mathcal{C}''$ *such that* $\Delta \Vdash \mathcal{C}'' :: \Delta'$*, then either* $\mathcal{C} \mapsto \mathcal{C}'$ *or* $\mathcal{C}$ *is poised.*

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

4) Preservation:

Theorem 3 (Preservation)**.** *For any configuration* $\mathcal{C}$ *that is a valid permutation of a configuration* $\mathcal{C}''$ *such that* $\Delta \Vdash \mathcal{C}'' :: \Delta'$*, if* $\mathcal{C} \mapsto \mathcal{C}'$*, then* $\mathcal{C}'$ *is a valid permutation of a configuration* $\mathcal{C}'''$ *such that* $\Delta \Vdash \mathcal{C}''' :: \Delta'$*.*

Proof. The proof is standard and can be 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* $\Psi; \Xi \vdash_\Sigma P@c :: x : A[c']$ *with the tree invariant satisfied, then, for any fresh* $y : A$*, we have* $\Psi; \Xi \vdash_\Sigma [y/x]P@c :: y : A[c']$ *with the tree invariant still satisfied.*
2) *If* $\Psi; \Xi, y : B[c''] \vdash_\Sigma P@c :: x : A[c']$ *with the tree invariant satisfied, then, for any fresh* $z : B$*, we have* $\Psi; \Xi, z : B[c''] \vdash_\Sigma [z/y]P@c :: x : A[c']$ *with the tree invariant still satisfied.*

Lemma 6 (Making two forests out of one)**.** *If* $\Psi_0; \Gamma \Vdash \mathbb{C}\mathbb{C}' :: \Gamma'$*, then for some* Γ_1 *we have* $\Psi_0; \Gamma_1'' \Vdash \mathbb{C} :: \Gamma_1''', \Gamma_1$ *and* $\Psi_0; \Gamma_2'', \Gamma_1 \Vdash \mathbb{C}' :: \Gamma_2'''$*, where* $\Gamma = \Gamma_1'', \Gamma_2''$ *and* $\Gamma' = \Gamma_1''', \Gamma_2'''$*.*

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

Lemma 7 (Security variable substitution)**.** *If* $\Psi; \Xi \vdash_\Sigma P@c :: x : B[d]$ *with the tree invariant satisfied, then for every substitution* $\Psi' \Vdash \gamma : \Psi$*, we have*

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

with the tree invariant satisfied as well.

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

Case 1.

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

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

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

which preserves the invariant.

By the $\&R$ rule, we get

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

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

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

Case 2.

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

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

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

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

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

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

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

Case 3.

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

By the induction hypothesis on the premise we have

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

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

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

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

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

Case 4.

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

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

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

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

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

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

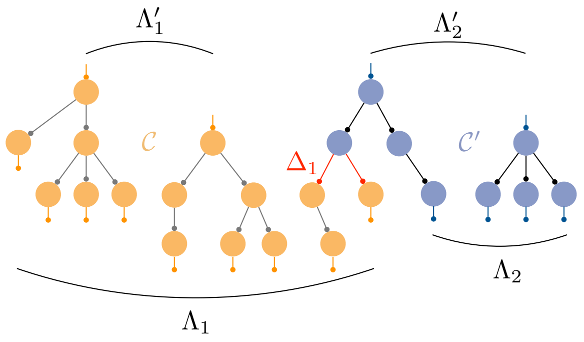

Fig. 4: Schematic illustration of Lem. 4, allowing us to break up an open forest into two open sub-forests.

Case 5. Here is the interesting case!

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

By the induction hypothesis we have

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

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

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

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

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

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

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

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

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

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

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

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

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

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

□

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.

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

Fig. 5: Annotated asynchronous dynamics–proof of preservation.

2) Preservation:

Theorem 4 (Preservation)**.** *For any configuration $\mathbb{C}$ that is a valid permutation of a configuration $\mathbb{C}''$ such that $\Psi_0; \Gamma \Vdash \mathbb{C}'' :: \Gamma'$, if $|\mathbb{C}| \mapsto \mathcal{C}'$, then there exists a security annotated program $\mathbb{C}'$ such that $|\mathbb{C}'| = \mathcal{C}'$ and is a valid permutation of a configuration $\mathbb{C}'''$ such that $\Psi_0; \Gamma \Vdash \mathbb{C}''' :: \Gamma'$. Moreover, the stepping preserves the tree invariant.*

Proof. The proof is by considering different cases of $|\mathbb{C}| \mapsto \mathcal{C}'$. For each step we provide a security annotated configuration $\mathbb{C}'$ and then by inversion on the typing derivations show that it is IFC-typed. For the purpose of presentation, we put security annotations of the post-steps for all possible steps in Fig. 5. We provide the detailed proof by inversion for a couple of cases. The rest of the cases are similar.

Case 1. ($\otimes$)

Subcase 1. (send)

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

By assumption of the theorem: $\Psi_0; \Gamma \Vdash \mathbb{C}_1\mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x^c_\beta\, y_\alpha; P)@d_1)\mathbb{C}_2 :: \Gamma'$.
By Lem. 6: $\Psi_0; \Gamma^1_1 \Vdash \mathbb{C}_1 :: \Gamma_1, \Gamma_2, {\Gamma^1}'_1$ and $\Psi_0; \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c] \Vdash \mathbf{proc}(y_\alpha[c], (\mathbf{send}\, x^c_\beta\, y_\alpha; P)@d_1) :: y_\alpha{:}(A \otimes B)[c]$ and $\Psi_0; \Gamma^1_3, \Gamma_2 \Vdash \mathbb{C}_2 :: \Gamma^{1'}_2$.
Where $\Gamma = \Gamma^1_1, \Gamma^1_2, \Gamma^1_3$ and $\Gamma' = \Gamma^{1'}_1, \Gamma^{1'}_2$. If $x_\beta{:}A[c] \in \Gamma$ we have $\Gamma'_1 = \Gamma_1$ and $\Gamma^{1''}_2, x_\beta{:}A[c] = \Gamma^1_2$, and otherwise $\Gamma'_1, x_\beta{:}A[c] = \Gamma_1$ and $\Gamma^{1''}_2 = \Gamma^1_2$.
By inversion on proc rule $\Psi_0; \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c] \vdash (\mathbf{send}\, x^c_\beta\, y^c_\alpha; P)@d_1 :: y_\alpha{:}(A \otimes B)[c]$. Moreover, $(\star)$ $\forall u_\gamma{:}T[d_2] \in \Gamma^{1''}_2, \Gamma'_1, x_\beta{:}A[c]. \Psi_0 \Vdash d_2 \sqsubseteq c$.
By inversion on $\otimes R$ rule $\Psi_0; \Gamma^{1''}_2, \Gamma'_1 \vdash P@d_1 :: y_\alpha{:}B[c]$.
By substitution of $y_{\alpha+1}$ for y_α:

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

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

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

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

$\square$

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

VII. Recursive session logical relation

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

A. Open configuration transition

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

Definition 12. *The set $\mathbf{Out}(\Delta \Vdash K)$, is defined as all channels with the sending semantics in Δ, K, i.e., $y_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ iff for some positive type T, we have $y_\alpha{:}T \in \Delta$ or for some negative type T, we have $y_\alpha{:}T \in K$.*
Similarly, the set $\mathbf{In}(\Delta \Vdash K)$, is defined as all channels with the receiving semantics in Δ, K, i.e., $y_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ iff for some negative type T, we have $y_\alpha{:}T \in \Delta$ or for some positive type T, we have $y_\alpha{:}T \in K$. ◇

Definition 13. *$dom(\Delta)$ is a set defined inductively as:*

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

◇

Definition 14 (Open configuration transitions)**.** *We provide some notations used in the logical relation and proofs.*
- *The notation $\mapsto^*$ refers to taking none or many steps with $\mapsto$. The notation $\mapsto^j$ refers to taking j steps with $\mapsto$.*
- *We write $\mathcal{D} \mapsto^{*\Upsilon} \mathcal{D}'$ stating that $\mathcal{D} \mapsto^* \mathcal{D}'$ and $\mathcal{D}'$ is ready to send along Υ.*
- *We write $\mathcal{D} \mapsto^{*\Upsilon;\Theta} \mathcal{D}'$ stating that $\mathcal{D} \mapsto^* \mathcal{D}'$ and $\mathcal{D}'$ is ready to send along Υ and ready to receive along Θ.*

◇

B. Recursive Session Logical Relation

Fig. 6 defines the logical relation for intuitionistic linear logic session types with general recursive types.

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

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

Definition 15 (Typing context projections)**.** *Downward projection on security linear contexts and K^s is defined as follows:*

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

◇

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

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

◇

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

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

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

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

◇

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 §IV 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 §IV 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 $\Psi_0; \Gamma \Vdash \mathbb{C} :: \Gamma'$, the quasi-running secrecy of a message or process is determined by its running secrecy, its process term, and the running secrecy of its parent.*

- *If the node is a process with a process term other than* **recv** *or* **case***, then its quasi-running secrecy is equal to its running secrecy.*
- *If the process term is of the form $\mathbf{case}\, y_\alpha^c(\ell \Rightarrow P_\ell)_{\ell \in L}@d_1$ or $x \leftarrow \mathbf{recv}\, y_\alpha^c; P_x@d_1$, then its quasi-running secrecy is $d_1 \sqcup c$.*
- *If the node is a message of a negative type along channel y_α^c, its quasi-running secrecy is c.*
- *If the node is a message of a positive type along channel y_α^c 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 $\Gamma \Vdash \mathbb{D} :: K^s$ and observer level ξ. A channel is relevant in $\mathbb{D}$ if (1) it has a maximal secrecy level less than or equal to ξ, and (2) it is either an observable channel or it shares a process or message with quasi-running secrecy less than or equal to ξ with a relevant channel. (A channel shares a process with another channel if they are siblings or one is the parent of another.) A process or message is relevant if (1) it has quasi-running secrecy less than or equal to ξ, and (2) it has at least one relevant channel. We denote the relevant processes and messages (i.e., nodes) in $\mathbb{D}$ by $\mathbb{D}\Downarrow\xi$. We write $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$ if the relevant nodes in $\mathbb{D}_1$ are identical to those in $\mathbb{D}_2$ up to renaming of channels with higher or incomparable secrecy than the observer.*

◇

We can build the set of all relevant nodes in a configuration by traversing the tree bottom-up as explained in Section 4. Def. 19 provides us with a local guide to identify whether a process is relevant or not. We note that if K^s is observable, then by the tree invariant, every channel in $\mathbb{D}$ is relevant.

Now we can state and prove and maintaining the invariant that the relevant nodes of the two program runs execute the same code. The proof of the fundamental theorem Thm. 5 crucially relies on this lemma.

Lemma 8 (Keeping relevant nodes in sync)**.** *Consider $\Psi_0; \Gamma \Vdash \mathbb{D}_i :: K^s$ for $i \in \{1, 2\}$ with the relevant nodes in $\mathbb{D}_1$ and $\mathbb{D}_2$ are identical, i.e., $\mathbb{D}_1\Downarrow\xi = \mathbb{D}_2\Downarrow\xi$, with $\Gamma \Downarrow \xi = \Gamma$ and $K^s \Downarrow \xi = K^s$. If $|\mathbb{D}_1| \mapsto |\mathbb{D}_1'|$, then there exists a $\mathbb{D}_2'$ such that $|\mathbb{D}_2| \mapsto^{0,1} |\mathbb{D}_2'|$, i.e., $|\mathbb{D}_2|$ steps to $|\mathbb{D}_2'|$ in zero or one step, and $\mathbb{D}_1\Downarrow\xi = \mathbb{D}_2\Downarrow\xi$.*

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

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

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

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

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

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

To prove this we need two observations:

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

This completes the proof.

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

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

and

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

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

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

- If $c \not\sqsubseteq \xi$, then $\mathbf{msg}(y^c_\alpha.k)$ is not relevant in both runs, and will be dismissed by the projections. Moreover, in this case y^c_α is not relevant in the pre-step and post-step configurations. Thus the relevancy of processes in $\mathbb{D}''_1$ and $\mathbb{D}''_2$ will remain intact.
- If $c \sqsubseteq \xi$, then y^c_α 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 $y^c_{\alpha+1}$ 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 invariant it is always bounded by the max secrecy of the offering channel, $c \sqsubseteq \xi$. Thus, in the post-step of both runs, $y^c_{\alpha+1}$ is relevant. Relevancy of message $\mathbf{msg}(y^c_\alpha.k)$ in the post-steps of the first and second run is determined by the quasi-running secrecies (d and d') of their parents (X and X') in $\mathbb{D}''_1$ and $\mathbb{D}''_2$. If $d \sqsubseteq \xi$, then the parent (X) is relevant in the first run and by assumption is identical to a relevant X' in the second run. Thus messages $\mathbf{msg}(y^c_\alpha.k)$ are relevant in both runs and y^c_α is relevant in the post-step too. The same holds when $d' \sqsubseteq \xi$. Otherwise, in both runs the quasi-running secrecy of the parent is higher than or incomparable to the observer (the parents are both irrelevant). Thus messages $\mathbf{msg}(y^c_\alpha.k)$ are not relevant in the post step of both runs, and will be dismissed by the projections. The channel y^c_α will be irrelevant in the post-step of both runs too. However, this does not affect the processes in $\mathbb{D}''_1$ and $\mathbb{D}''_2$ as the parents of messages (X and X') are already irrelevant in the pre-step.

Finally, we need to show that projections of $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are equal in the post-step too.
Consider the resources of the process that offeres along $y_\alpha[c]$. By our writing convention, they are all in $\mathbb{D}'_1$ and $\mathbb{D}'_2$: (i) Those resources offered by $\mathbb{D}'_1$ and $\mathbb{D}'_2$ with max secrecies higher than or incomparable to the observer ξ in the pre-step will remain higher than or incomparable to the observer and thus irrelevant in the post-step too. (ii) Those resources with max secrecies lower than the observer ξ in the pre-step, i.e., a sub-tree $\mathbb{T}_i$ in $\mathbb{D}'_i$ offering along a channel $w[c']$ where $c' \sqsubseteq \xi$, are relevant in the pre-step and will remain relevant in the post-step (again, in this case, by the tree invariant, the quasi-running secrecy of the processes offering along $y^c_{\alpha+1}$ cannot increase in the post-step).
Case 2. $\mathbb{D}_1 = \mathbb{D}'_1\mathbf{proc}(x_\beta[d], y^c_\alpha.k; P@d_1)\mathbb{D}''_1$ **and**

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

We consider subcases based on relevancy of the process offering along x^β_d:
Subcase 1. $\mathbf{proc}(x_\beta[d], y^c_\alpha.k; P@d_1)$ is irrelevant. By inversion on the typing rules, $d_1 \sqsubseteq c \sqsubseteq d$. By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels connected to P including y^c_α and x^d_β are relevant. In both cases, neither $\mathbf{msg}(y^c_\alpha.k)$ nor $\mathbf{proc}(x_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)$ are relevant. Channel x^d_β is irrelevant in the pre-step and post-step configurations.
Channel y^c_α is irrelevant in the pre-step, and both y^c_α and $y^c_{\alpha+1}$ are irrelevant in pre-step and post-step configurations. Every other irrelevant resource of $\mathbf{proc}(x_\beta[d], y^c_\alpha.k; P@d_1)$ will remain irrelevant in the post-step too.
In this subcase, our goal is to show

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

With a same argument as in **Case 1. Subcase 1.**, we can prove that the relevancy of processes in $\mathbb{D}'_1$ and $\mathbb{D}''_1$ remain intact.
Subcase 2. $\mathbf{proc}(x_\beta[d], y^c_\alpha.k; P@d_1)$ is relevant. By assumption that $\mathbb{D}_1\Downarrow\xi =_\xi \mathbb{D}_2\Downarrow\xi$, and definition of $=_\xi$:

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

and we have

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

It remains to show

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

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

– If $c \not\sqsubseteq \xi$ then the message is irrelevant in both runs. y^c_α is irrelevant in the pre-steps of both runs and remain irrelevant in the post-steps too.

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

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

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

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

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

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

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

Channel y^c_α is irrelevant in the pre-step, and both y^c_α and $y^c_{\alpha+1}$ are irrelevant in pre-step and post-step configurations. In this subcase, our goal is to show

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

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

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

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

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

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

We have

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

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

If $c \sqsubseteq \xi$, then y^c_α is relevant in the pre-step in both runs. We also know that x_β is relevant in the pre-step and $\mathbb{T}''_1 = \mathbb{T}''_2$ are relevant in the pre-step. In the post-step, $y^c_{\alpha+1}$ is relevant in both runs. Relevancy of messages $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ in the post-steps are determined by the quasi-running secrecy (d' and d'') of their parents (X and X') in $\mathbb{D}''_1$ and $\mathbb{D}''_2$. If $d' \sqsubseteq \xi$, then the parent (X) is relevant in the first run and by assumption is equal to a relevant X' in the second run. Thus messages $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ are relevant in both runs, y^c_α are relevant in the post-step, and trees $\mathbb{T}''_1 = \mathbb{T}''_2$ and their offering channels x^c_β are relevant in the post-steps too. The same holds when $d' \sqsubseteq \xi$.

Otherwise, in both runs the quasi-running secrecy of the parent is higher than or incomparable to the observer level (the parents are both irrelevant). Thus messages $\mathbf{msg}(\mathbf{send} x^c_\beta\, y^c_\alpha)$ are not relevant in the post step of both runs, and will be

dismissed by the projections. The channels y^c_α will be irrelevant in the post-step too. However, this does not affect the processes in $\mathbb{D}''_1$ and $\mathbb{D}''_2$ as the parents of messages (X and X') are already irrelevant in the pre-step. The channels x^c_β both become irrelevant in the post-steps. However, we still have $\mathbb{T}''_1 \Downarrow \xi = \mathbb{T}''_2 \Downarrow \xi$ as they have the same type and their parents have the same quasi-running secrecy.
We show that projections of $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are equal in the post-step too. The resources with secrecy level higher than or incomparable to the observer level offered along $\mathbb{D}'_1$ and $\mathbb{D}'_2$ in the pre-step will remain higher than or incomparable to and thus irrelevant in the post-step too. For a relevant resources (w) offered along $\mathbb{D}_1$ and $\mathbb{D}_2$, we need to consider the change in quasi-running secrecy as in **Case 1. Subcase 2.** Moreover, we need to consider the scenario that a relevant resource ($w^{c'}$) in the pre-step loses its relevancy in the post-step because the channel offered along x^c_β is transferred to the message. This case only happens if $c', c \sqsubseteq \xi$ and thus the trees $\mathbb{T}_1$ and $\mathbb{T}_2$ offered along $w^{c'}$ is present in both runs and $\mathbb{T}_1 = \mathbb{T}_2$. We know that $w^{c'}$ is irrelevant in the post-step of both runs, and the quasi-running secrecy of the processes using the resource $w^{c'}$ in both runs are the same.
The relevant and irrelevant processes in $\mathbb{D}''_i$ remain intact.
Case 4. $\mathbb{D}_1 = \mathbb{D}'_1\mathbb{T}''_1\mathbf{proc}(w_\eta[c'], \mathbf{send}x^c_\beta\, y^c_\alpha @d_1)\mathbb{D}''_1$ **and**

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

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

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

In the case where $\mathbb{T}''_1$ is empty we have $\Gamma_2 = x_\beta{:}A[c]$. We proceed by considering subcases based on relevancy of the process offering along $w_\eta[c']$:
Subcase 1. $\mathbf{proc}(w_\eta[c'], \mathbf{send}x^c_\beta\, y^c_\alpha; P@d_1)$ **is not relevant.** By inversion on the typing rules $d_1 \sqsubseteq c \sqsubseteq c'$. By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels connected to P including y^c_α, and x^c_β are relevant. In both cases, neither $\mathbf{msg}(\mathbf{send}x^c_\beta\, y^c_\alpha)$ nor $\mathbf{proc}(w_\eta[c'], [y^c_{\alpha+1}/y^c_\alpha]P@d_1)$ are relevant. Channel $w^{c'}_\eta$ is irrelevant in the pre-step and post-step configurations.
Channel y^c_α is irrelevant in the pre-step, and both y^c_α and $y^c_{\alpha+1}$ are irrelevant in pre-step and post-step configurations. Every other irrelevant resource of the process in the pre-step will remain irrelevant in the post-step too. See **Case 3. Subcase 1.** for the discussion on the relevancy of $\mathbb{T}''_1$.

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

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

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

We have

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

With the same argument as in **Case 2. Subcase 2.** we can show that relevancy of $\mathbb{D}''_i$ remains intact.
For $\mathbb{T}''_i$, we argue that if $c \sqsubset \xi$, then $\mathbb{T}''_1 = \mathbb{T}''_2$ is relevant in the pre-step and remains relevant in the post-step too. If $c \not\sqsubseteq \xi$, then the relevancy of $\mathbb{T}''_i$ remain intact from pre-step to post-step. See **Case 3. Subcase 2.** for a more detailed discussion on transferring a tree via message.
The discussion on relevancy of $\mathbb{D}'_i$ is similar to the previous cases.
Case 5. $\mathbb{D}_1 = \mathbb{D}'_1\mathbf{msg}(y^c_\alpha.k)\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L}@d_1)\mathbb{D}''_1$ **and**

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

We consider sub-cases based on relevancy of process offering along w^d_β. Observe that y^c_α is relevant if an only if v^c is relevant, since they share a message of secrecy c.
Subcase 1. $\mathbf{proc}(x_\beta[d], \mathbf{case}\, y^c_\alpha(\ell \Rightarrow P_\ell)_{\ell \in L}@d_1)$ **is not relevant.** By definition either $d_1 \sqcup c \not\sqsubseteq \xi$ or none of the channels connected to P including its offering channel y^c_α are relevant. In both cases the messages $\mathbf{msg}(y^c_\alpha.k)$ and the continuation process $\mathbf{proc}(w_\beta[d], [y^c_{\alpha+1}/y^c_\alpha]P_k@(d_1 \sqcup c))$ are not relevant either. It is straightforward to see that

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Subcase 1. $\mathbf{proc}(v_\eta[c'], w^c \leftarrow \mathbf{recv} y^c_\alpha; P@d_1)$ **is not relevant.** By definition either $d_1 \sqcup c \not\sqsubseteq \xi$ or none of the channels connected to P including y^c_α are relevant.
In both cases by the definition of quasi-running secrecy we know that neither $\mathbf{msg}(\mathbf{send} x^c_\eta y^c_\alpha)$ nor the continuation process $\mathbf{proc}(v_\eta[c'], [x_\eta/w][y^c_{\alpha+a}/y^c_\alpha]P@c \sqcup d_1)$ are relevant. It is then straightforward to see that

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

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

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

We have

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

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

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

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

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

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

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

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

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

We have

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

The proof is similar to previous cases.

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

$$\begin{array}{c} |\mathbb{D}'_1 \mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)\mathbb{D}''_1| \\ \mapsto \\ |\mathbb{D}'_1 \mathbf{proc}(x_0[d], \hat{\gamma}(P)@d_2)\mathbf{proc}(y_\alpha[c], [x^d_0/x^d]Q@d_1)\mathbb{D}''_1| \end{array}$$

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

Subcase 1. $\mathbf{proc}(y_\alpha[c], (x^{[\hat{\gamma_{\mathsf{sec}}}(\psi)]} \leftarrow X[\gamma] \leftarrow \Gamma_1)@\hat{\gamma_{\mathsf{sec}}}(\psi'); Q@d_1)$ **is not relevant.** By definition either $d_1 \not\sqsubseteq \xi$ or none of the channels of this process including y^c_α are relevant.
In both cases, it means that $\mathbf{proc}(x_0[d], \hat{\gamma}(P)@d_2)$ and $\mathbf{proc}(y_\alpha[c], [x^d_0/x^d]Q@d_1)$ are not relevant either. Note that $d_1 \sqsubseteq d_2$ and thus $d_2 \not\sqsubseteq \xi$ if $d_1 \not\sqsubseteq \xi$.

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

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

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

We have

$$|\mathbb{D}_2| \mapsto |\mathbb{D}'_2 \mathbf{proc}(x_0[d], \hat{\gamma}(P)@d_2)\mathbf{proc}(y_\alpha[c], [x^d_0/x^d]Q@d_1)\mathbb{D}''_2|$$

Remark: we can assume that the fresh channel being spawned will be x_0 in both runs. Moreover, the (unique) substitution $\hat{\gamma}$ is the same when stepping both runs.
Note that if x^d_0 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^d_0 may block their relevancy path if $d \not\sqsubseteq \xi$ or in the case where $d_2 \not\sqsubseteq \xi$. But this happens to the processes in the both runs and can only change relevant processes/messages in the pre-step to irrelevant processes/messages in the post-step. (similar to the cases 3 and 4 for $\otimes$ and $\multimap$)

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

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

where $\gamma = (\gamma_{\mathsf{sec}}, \gamma_{\mathsf{var}})$ $\hat{\gamma}(\psi) = c$, $\hat{\gamma}(w) = y_\alpha$, and $\hat{\gamma}(u) = z_\beta$.
The proof of this case is similar to the proof of Case 9(Spawn) by considering sub-cases based on relevancy of process offering along y^c_α. □

C. Fundamental theorem

Theorem 5 (Fundamental Theorem). *For all security levels ξ, and configurations*

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

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

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

Proof. Our goal is to show:

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

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

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

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

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

The goal is equivalent to:

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

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

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

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

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

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

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

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

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

It remains to prove ($\star_2$):

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

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

The proof is by induction on the index m.

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

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

which is enough to complete the proof.

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

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

Consider an arbitrary $\mathbb{D}'_1$ such that $|\mathbb{D}_1| \mapsto^{*\Upsilon;\Theta} |\mathbb{D}'_1|$. By Preservation we know that $\mathbb{D}'_1$ is IFC-typed for the same interface $\Psi; \Gamma \Vdash \mathbb{D}'_1 :: K^s$ By Lem. 8, for some $\mathbb{D}'_2$, we get $|\mathbb{D}_2| \mapsto^{*\Upsilon} |\mathbb{D}'_2|$, such that $\Psi; \Gamma \Vdash \mathbb{D}'_2 :: K^s$ with $\mathbb{D}'_2 \Downarrow \xi = \mathbb{D}'_2 \Downarrow \xi$.
We need to show that

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

We prove both $\dagger_1$, and $\dagger_2$. We consider an arbitrary u'_δ and provide a case analysis based on its type. If there is no such u'_δ in the specified set, then the statements trivially holds. There might be a channel in both of these sets, if and only if $\mathbb{D}_1 = \cdot$.
Proof of $\dagger_1$. Consider an arbitrary channel $u'_\delta{:}T \in \mathbf{Out}(|\Gamma| \Vdash |K|^s)$ and assume $u'_\delta{:}T \in \Upsilon$. We consider different cases based on the session type T.

Case 1. $u'_\delta{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:}1[c_1]$, and $\mathbb{D}'_1 = \mathbb{D}''_1\mathbf{msg}(\mathbf{close}\, u^{c_1}_\alpha)$.
By the typing rules and $\mathbb{D}'_1 = \mathbb{D}''_1\mathbf{msg}(\mathbf{close}\, u^{c_1}_\alpha)$ we know that $\mathbb{D}''_1 = \cdot$, and $\Gamma_1 = \cdot$.
Note that by the definition of relevancy, all processes in $\mathbb{D}'_1$ and $\mathbb{D}'_2$ has to be relevant. As a result,

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

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

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

as needed.
Case 2. $u'_\delta{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:} \oplus \{\ell{:}A_\ell\}_{\ell \in I}[c_1]$, and $\mathbb{D}'_1 = \mathbb{D}''_1\mathbf{msg}(u^{c_1}_\alpha.k)$.
By the assumption of the theorem, $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are both relevant, and we have $\mathbb{D}'_2 = \mathbb{D}''_2\mathbf{msg}(u^{c_1}_\alpha.k)$. Removing $\mathbf{msg}(u^{c_1}_\alpha.k)$ from $\mathbb{D}'_1$ and $\mathbb{D}'_2$ does not change relevancy of the remaining configuration when $c_1 \sqsubseteq \xi$:

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

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

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

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

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

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

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

where $\Gamma = \Gamma', \Gamma''$, and $\Psi; \Gamma'' \Vdash \mathbb{A}_1 :: (x_\beta{:}A[c_1])$, and $\Psi; \Gamma' \Vdash \mathbb{D}''_1 :: (u_{\alpha+1}{:}A[c_1])$.
Since $\mathbb{D}'_1$ and $\mathbb{D}'_2$ are both IFC-typed, they enjoy the tree invariant. Thus both of them are relevant and since $\mathbb{D}'_1 \Downarrow \xi = \mathbb{D}'_2 \Downarrow \xi$, we have

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

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

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

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

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

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

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

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

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

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

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

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

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

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

such that $\Psi_0; \Gamma'' \Vdash \mathbb{A}_1 :: y_\delta{:}A[c]$ and $\Psi_0; \Gamma', x_{\gamma+1}{:}B[c] \Vdash \mathbb{D}''_1 :: K^s$.
The message $\mathbf{msg}(\mathbf{send} y^c_\delta\, x^c_\gamma)$ is relevant in $\mathbb{D}'_1$. By assumption of the theorem, $\mathbb{D}'_2 = \mathbb{A}_2\mathbf{msg}(\mathbf{send} y^c_\delta\, x^c_\gamma)\mathbb{D}''_2$, such that $\Psi_0; \Gamma'' \Vdash \mathbb{A}_2 :: y_\delta{:}A[c]$ and $\Psi_0; \Gamma^h_2, \Gamma', x_{\gamma+1}{:}B[c] \Vdash \mathbb{D}''_2 :: K^s$ with Γ^h_2 being the context of all the channels in Γ_2 with security higher than or incomparable to the observable level ξ. Again because of the tree invariant every resource of $\mathbb{A}_2$ has a secrecy less than or equal to the observer. Also, by assumption we know that $\mathbb{D}'_1{\Downarrow}\xi = \mathbb{D}'_2{\Downarrow}\xi$. By definition of relevancy, we know that $\mathbf{msg}(\mathbf{send} y^c_\delta\, x^c_\gamma)$ and the tree $\mathbb{A}_1$ connected to it are relevant in $\mathbb{D}_1$ and thus $\mathbb{A}_1$ is equal to $\mathbb{A}_2$, i.e, $\mathbb{A}_1 \Downarrow \xi = \mathbb{A}_2 \Downarrow \xi$.
Removing the tress $\mathbb{A}_1$ and $\mathbb{A}_2$ and the messages from both configurations does not change relevancy of the rest of the configuration since $x^c_{\gamma+1}$ will remain relevant, i.e., $\mathcal{D}''_1{\Downarrow}\xi = \mathcal{D}''_2{\Downarrow}\xi$.
We can apply the induction hypothesis on the smaller index $m' < m$ and $\mathbb{A}_1 \Downarrow \xi = \mathbb{A}_2 \Downarrow \xi$ and $\mathbb{D}''_1 \Downarrow \xi = \mathbb{D}''_2 \Downarrow \xi$ to get

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

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

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

Proof of $\dagger_2$. Consider an arbitrary channel $u'_\delta{:}T \in \mathbf{In}(|\Gamma| \Vdash |K|)$ We consider different cases based on the session type T.
Case 1. $u'_\delta{:}T[c'] = K^s = u_\alpha{:}T_1[c_1] = u_\alpha{:}\&\{\ell{:}A_\ell\}_{\ell\in I}[c_1]$.
We need to apply the induction hypothesis on the index $m' < m$, but first we need to show that the invariant of the induction holds. Consider an arbitrary label $k \in L$.From $\mathbb{D}'_1{\Downarrow}\xi = \mathbb{D}'_2{\Downarrow}\xi$ and $c_1 \sqsubseteq \xi$, we get

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

By induction hypothesis

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

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

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

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

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

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

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

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

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

Case 3. $\Gamma = \Gamma', x_\gamma{:}1[c]$, and $u'_\delta{:}T'[c'] = x_\gamma{:}1[c]$
We first briefly explain why the invariant of induction holds after we bring the closing message inside $\mathbb{D}'_i$ and remove the channel x^c_γ from Δ, i.e.

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

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

- Otherwise the parent of $\mathbf{msg}(\mathbf{close}\, x_\gamma^c)$ is irrelevant in $\mathbb{D}_1'$ and $\mathbb{D}_2'$ before adding the message and remains irrelevant after that too. The proof in this case is straightforward.

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

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

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

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

Case 4. $\Gamma = \Gamma', x_\gamma{:} \oplus \{\ell : A_\ell\}_{\ell \in L}[c]$, and $u_\delta'{:}T'[c'] = x_\gamma{:} \oplus \{\ell : A_\ell\}_{\ell \in L}[c]$.
Consider an arbitrary label $k \in L$. We have $\mathbf{msg}(x_\gamma^c.k)\mathbb{D}_1'\Downarrow\xi = \mathbf{msg}(x_\gamma^c.k)\mathbb{D}_2'\Downarrow\xi$, since $x_{\gamma+1}^c$ is relevant and the positive messages $\mathbf{msg}(x_\gamma^c.k)$ 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

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

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

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

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

$$\mathbf{msg}(\mathbf{send} y_\eta^c\, x_\gamma^c)\mathbb{D}_1'\Downarrow\xi =_\xi \mathbf{msg}(\mathbf{send} y_\eta^c\, x_\gamma^c)\mathbb{D}_2'\Downarrow\xi :$$

The quasi-running secrecy of $\mathbf{msg}(\mathbf{send} y_\eta^c\, x_\gamma^c)$ 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

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

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

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

$\square$

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 Lem. 16 and Corollary 2 introduced in § 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 $\mapsto$ notation). For example, the transition step for* Spawn, *produces two processes*

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

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

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

◇

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

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

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

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

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

◇

Lemma 9 (Uniqueness of process productions). *Consider $\Delta \Vdash \mathcal{D} :: K$, and $\mathcal{D} \mapsto^* \mathcal{D}' \mapsto \mathcal{D}_1\mathbf{proc}(x, P)\mathcal{D}_2$, such that $\mathbf{proc}(x, P)$ is produced in the last step. Then process $\mathbf{proc}(x, P)$ does not occur in any of the previous steps $\mathcal{D} \mapsto^* \mathcal{D}'$.*

The same result hold for the production of a message.

Proof. Observe the following invariant in the dynamics

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

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

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

Lemma 10 (Diamond Property). *If $\Delta \Vdash \mathcal{D}_1 :: K$ and $\dagger\ \mathcal{D}_1 \mapsto^{\Upsilon}_{\Delta \Vdash K} \mathcal{D}'_1$ and $\dagger'\ \mathcal{D}_1 \mapsto^{\Upsilon'}_{\Delta \Vdash K} \mathcal{D}''_1$, and $\mathcal{D}'_1 \neq \mathcal{D}''_1$ then there is a configuration $\mathcal{D}$ such that $\star\ \mathcal{D}'_1 \mapsto^{\Upsilon''}_{\Delta \Vdash K} \mathcal{D}$, and $\star'\ \mathcal{D}''_1 \mapsto^{\Upsilon''}_{\Delta \Vdash K} \mathcal{D}$, where $\Upsilon \cup \Upsilon' = \Upsilon''$. The messages produced along the channels $\Upsilon \cap \Upsilon'$ are identical in $\mathcal{D}'_1$ and $\mathcal{D}''_1$ and $\mathcal{D}$.*

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

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

Lemma 11 (Confluence)**.** *If $\Delta \Vdash \mathcal{D}_1 :: K$ and $\dagger\, \mathcal{D}_1 \mapsto^{m'_\Upsilon}_{\Delta \Vdash K} \mathcal{D}'_1$ and $\dagger'\, \mathcal{D}_1 \mapsto^{n'_{\Upsilon'}}_{\Delta \Vdash K} \mathcal{D}''_1$, then there is a configuration $\mathcal{D}$ such that $\star\, \mathcal{D}'_1 \mapsto^{j_{\Upsilon''}}_{\Delta \Vdash K} \mathcal{D}$ for some $j \leq n'$, and $\star'\, \mathcal{D}''_1 \mapsto^{k_{\Upsilon''}}_{\Delta \Vdash K} \mathcal{D}$ for some $k \leq m'$, where $\Upsilon \cup \Upsilon' = \Upsilon''$. The messages produced along the channels $\Upsilon \cap \Upsilon'$ are identical in $\mathcal{D}'_1$ and $\mathcal{D}''_1$ and $\mathcal{D}_1$. The steps in $\star$ are (locally) identical to the steps of $\dagger'$ that do not occur in $\dagger$. And the steps in $\star'$ are (locally) identical to the steps of $\dagger$ that do not occur in $\dagger'$.*

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

Proof. It follows by standard inductions from the diamond property (Lem. 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 (Lem. 10). Assume that $n' = n + 1$ and $m' = m + 1$. We form the following diagram to sketch the structure of the proof.

$$\begin{tikzcd}
 & & \Upsilon_1, \Upsilon_2 & \Upsilon = \Upsilon_1, \Upsilon_2, x & \\
 & \mathcal{D}_1 \arrow[r, mapsto, "m", "\Delta \Vdash K"'] \arrow[d, mapsto, "n", "\Delta \Vdash K"'] & \mathcal{D}'_2 \arrow[r, mapsto, "\Delta \Vdash K"'] \arrow[d, mapsto, "\leq n", "\Delta \Vdash K"'] & \mathcal{D}'_1 \arrow[d, mapsto, "\leq n", "\Delta \Vdash K"'] & \\
\Upsilon'_1, \Upsilon_2 & \mathcal{D}''_2 \arrow[r, mapsto, "\leq m", "\Delta \Vdash K"'] \arrow[d, mapsto, "\Delta \Vdash K"'] & \mathcal{D}_2 \arrow[r, mapsto, "\leq 1", "\Delta \Vdash K"'] \arrow[d, mapsto, "\leq 1", "\Delta \Vdash K"'] & \mathcal{D}_4 \arrow[d, mapsto, "\leq 1", "\Delta \Vdash K"'] & \\
 & & \Upsilon_1, \Upsilon'_1, \Upsilon_2 & \Upsilon_1, \Upsilon'_1 \cup \{x\}, \Upsilon_2 & \\
 & \mathcal{D}''_1 \arrow[r, mapsto, "\leq m", "\Delta \Vdash K"'] & \mathcal{D}_3 \arrow[r, mapsto, "\leq 1", "\Delta \Vdash K"'] & \mathcal{D} & \texttt{if } (x \neq y) : \ \Upsilon_1 \cup \{y\}, \Upsilon'_1 \cup \{x\}, \Upsilon_2 \\
 & \Upsilon' = \Upsilon'_1, \Upsilon_2, y & \Upsilon_1 \cup \{y\}, \Upsilon'_1, \Upsilon_2 & & \texttt{if } (x = y) : \ \Upsilon_1, \Upsilon'_1, \Upsilon_2, x
\end{tikzcd}$$

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

□

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

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

Proof. This is another corollary to the second part of the confluence lemma (Lem. 11). First observe that $\mathcal{A}$ is not active in $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2 \mathcal{A}' \mathcal{D}''_2$: if it is active, we produce $\mathbf{proc}(x, P)$ twice in $\mathcal{D}_1 \mapsto^* \mathcal{D}'_2 \mathcal{A}' \mathcal{D}''_2 \mapsto \mathcal{D}^1_2 \mathbf{proc}(x, P) \mathcal{D}^2_2$, which contradicts with uniqueness of process productions (Lem. 9). Similarly, $\mathcal{A}'$ is not active in $\mathcal{D}_1 \mapsto^* \mathcal{D}'_1 \mathcal{A} \mathcal{D}''_1$. With the second part of the confluence lemma, for some $\mathcal{D}$, we have $\mathcal{D}'_1 \mathcal{A} \mathcal{D}''_1 \mapsto^* \mathcal{D}$ and $\mathcal{D}'_2 \mathcal{A}' \mathcal{D}''_2 \mapsto^* \mathcal{D}$ such that both $\mathcal{A}$ and $\mathcal{A}'$ occur in $\mathcal{D}$. If $\mathcal{A} \neq \mathcal{A}'$, we can produce $\mathbf{proc}(x, P)$ twice from $\mathcal{D}$ which by Lem. 9 is contradictory. Thus, know that $\mathcal{A} = \mathcal{A}'$.

Note: here we rely on the fact that the steps in Fig. 3 produce the post-steps uniquely from the pre-steps. In particular, in the spawn rule we assume that the fresh channel name is uniquely determined based on the offering channel and the process term of the caller. □

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

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

Lemma 12 (Building minimal sending configuration)**.** *Consider $\Delta \Vdash \mathcal{D}_2 :: K$, a set of channels $\Upsilon_1 \subseteq \Delta, K$ and $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ for some $\Upsilon_2 \supseteq \Upsilon_1$. There exists a set Υ and a configuration $\mathcal{D}_2''$ such that $\Upsilon_1 \subseteq \Upsilon \subseteq \Upsilon_2$ and $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ and $\forall \mathcal{D}_2^1, \Upsilon_3 \supseteq \Upsilon_1.$ if $\mathcal{D}_2 \mapsto^{*\Upsilon_3} \mathcal{D}_2^1$ then $\mathcal{D}_2'' \mapsto^{*\Upsilon_3} \mathcal{D}_2^1$. We call $\mathcal{D}_2''$ and Υ the minimal sending configuration and the minimal sending set with respect to Υ_1 and $\mathcal{D}_2$, respectively.*

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

Algorithm 1 Building the minimal sending configuration

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

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

For every given Υ_1, $\mathcal{D}_2$, and $\mathcal{D}$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon_2$, Algorithm 1 returns a configuration $\mathcal{D}''$ and set Υ and builds the dynamic transition $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ in Tns. Observe that the steps of $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ are all local transitions of $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}$. Thus, by the confluence lemma (Lem. 11), we know that $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}$.

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

B. Backward closure

Lemma 13 (Backward closure on the second run)**.** *The second run enjoys backward closure:*

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

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

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

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

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

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

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

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

and the goal

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

We apply an $\exists$-Elimination on the assumption to get Υ_2 and $\mathcal{D}_2'$ that satisfies the conditions and use the same Υ_2 and $\mathcal{D}_2'$ to instantiate the existential quantifier in the goal. Since $\mathcal{D}_2'' \mapsto^* \mathcal{D}_2$, we get $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'$, and the proof is complete.

2) The proof is by cases on the row of the logical relation that makes the assumption $(\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k+1}_{\cdot;y_\alpha}$ true. Here we only consider an interesting cases, the proof of other cases is similar.
Row 4. By the conditions of this row, we know that $\Delta = \Delta', \Delta''$, and $K = y_\alpha{:}A \otimes B$. Moreover, we have

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

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

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

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

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

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

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

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

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

and the proof of this subcase is complete.

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

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

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

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

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

which completes the proof of this case.

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

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

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

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

□

C. Forward closure

Lemma 14 (Forward closure on the first run)**.** *Consider $(\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$ and $\mathcal{D}_1 \mapsto^* \mathcal{D}_1''$. We have $(\mathcal{D}_1'';\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

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

By line (12) of the logical relation we get

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

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

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

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

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

Which exactly matches our goal and the proof is complete. □

Lemma 15 (Forward closure on the second run with some specific conditions)**.** *Consider $(\mathcal{D}_1;\mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$ and $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ such that if $\mathcal{D}_1$ sends along the set Υ_1, we have $\Upsilon_1 \subseteq \Upsilon$ and also $\mathcal{D}_2''$ is the minimal configuration built by Lem. 12 given the set Υ_1 and configuration $\mathcal{D}_2$. We have $(\mathcal{D}_1;\mathcal{D}_2'') \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

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

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

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

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

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

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

and the goal

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

We apply an $\exists$-Elimination on the assumption to get $\mathcal{D}_2'$ that satisfies the conditions, i.e., $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$. We use the same $\mathcal{D}_2'$ to instantiate the existential quantifier in the goal, we need to show that $\mathcal{D}_2'' \mapsto^{*\Upsilon_1} \mathcal{D}_2'$. Since $\mathcal{D}_2''$ is the minimal configuration built for Υ_1 and $\mathcal{D}_2$, and $\Upsilon_1 \subseteq \Upsilon_2$, by Lemma 12 we get $\mathcal{D}_2'' \mapsto^{*\Upsilon_2} \mathcal{D}_2'$, 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

Lemma 16 (Moving existential over universal quantifier)**.** *if we have*

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

then

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

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

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

Next, apply a $\forall\mathbf{I}.$ and $\mathbf{if}\,\mathbf{I}.$ on the goal followed by a corresponding $\forall\mathbf{E}.$ and $\mathbf{if}\,\mathbf{E}.$ on the assumption ($\dagger'$). Now apply an $\exists\mathbf{E}.$ on the assumption ($\dagger'$) to get $\mathcal{D}_2'$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_2} \mathcal{D}_2'$ and $\Upsilon_1 \subseteq \Upsilon_2$. Given $\mathcal{D}_2'$, by Lem. 12, we can build the minimal $\mathcal{D}_2''$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon} \mathcal{D}_2''$ and $\Upsilon_1 \subseteq \Upsilon$. Moreover, we know that for every $\mathcal{D}$ such that $\mathcal{D}_2 \mapsto^{*\Upsilon_3} \mathcal{D}$ and $\Upsilon_1 \subseteq \Upsilon_3$, we get $\mathcal{D}_2'' \mapsto^{*\Upsilon_3} \mathcal{D}$.

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

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

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

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

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

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

Since $\mathcal{D}_2''$ is the minimal configuration built for Υ_1 and $\mathcal{D}_2$, we know that $\mathcal{D}_2'' \mapsto^{*\Upsilon'} \mathcal{D}'$. We can apply the backward closure results of Lem. 13 to get the goal from the assumptions ($\dagger'''$), and this completes the proof.

□

B. Compositionality

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

Proof. The left to right direction is a corollary of Lemma 17 in which we compose multiple configurations instead of just two. For the right to left direction, we put $\mathcal{T}_i = \cdot$, and $\Delta' = u_\alpha{:}T$, and the rest of the proof is straightforward. □

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

Proof. Our goal is to prove the following:

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

This is equivalent to the following statement which we prove:

For all natural numbers, k, and for all index set I, and for all session-typed configurations $\Delta^n \Vdash \mathcal{B}^n_i :: K^n$ such that $\dagger_n\ \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^n_2) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have $\dagger'\ (\{\mathcal{B}^n_1\}_{n\in I}; \{\mathcal{B}^n_2\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.

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 $\Delta^n \Vdash \mathcal{B}^n_i :: K^n$ with $\Delta \Vdash \{\mathcal{B}^n_i\}_{n\in I} :: K$, we have $(\{\mathcal{B}^n_1\}_{n\in I}; \{\mathcal{B}^n_2\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^0$.

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

For all index set I, and for all session-typed configurations $\Delta^n \Vdash \mathcal{B}^n_i :: K^n$ where $n \in I$ such that $\dagger_n\ \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^n_2) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have $\dagger'\ (\{\mathcal{B}^n_1\}_{n\in I}; \{\mathcal{B}^n_2\}_{n\in I}) \in \mathcal{E}[\![\Delta \Vdash K]\!]^{k'+1}$

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

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

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

For all natural numbers j, for all I, and for all session-typed configurations $\Delta^n \Vdash \mathcal{B}^n_i :: K^n$ with $n \in I$ such that $\dagger_n\ \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^n_2) \in \mathcal{E}[\![\Delta^n \Vdash K^n]\!]^m$ we have

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

We proceed the proof by a nested induction on j.

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

$$\begin{array}{l}\forall \Upsilon_1, \Theta_1,\, \mathcal{D}'_1.\, \textbf{if } \{\mathcal{B}^n_1\}_{n\in I} \mapsto^{0\Upsilon_1;\Theta_1} \mathcal{D}'_1 \text{ then } \exists \Upsilon_2, \mathcal{D}'_2 \text{ such that } \{\mathcal{B}^n_2\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}'_2 \text{ and } \Upsilon_1 \subseteq \Upsilon_2 \text{ and}\\ \forall\, x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \textbf{if } x_\gamma \in \Upsilon_1.\, \textbf{then } (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma} \text{ and}\\ \forall\, x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \textbf{if } x_\gamma \in \Theta_1.\, (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}.\end{array}$$

Consider an arbitrary Υ_1, Θ_1, ans $\mathcal{D}'_1$, and apply **If I.** on the goal. By the assumption $\{\mathcal{B}^n_1\}_{n\in I} \mapsto^{0\Upsilon_1;\Theta_1} \mathcal{D}'_1$ we know that $\mathcal{D}'_1 = \{\mathcal{B}^n_1\}_{n\in I}$, and $\{\mathcal{B}^n_1\}_{n\in I}$ sends along Υ_1 and receives along Θ_1. Our goal is to show the following:

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

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

$$\begin{array}{l}\dagger'_n\, \forall m. \forall \Upsilon_n, \Theta_n, \mathcal{B}^{n'}_1.\, \text{if } \mathcal{B}^n_1 \mapsto^{*\Upsilon_n;\Theta_n} \mathcal{B}^{n'}_1 \text{ then } \exists \Upsilon_{n_2}, \mathcal{B}^{n'}_2 \text{ such that } \mathcal{B}^n_2 \mapsto^{*\Upsilon_{n_2}} \mathcal{B}^{n'}_2 \text{ and } \Upsilon_n \subseteq \Upsilon_{n_2} \text{ and}\\ \forall\, x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if } x_\gamma \in \Upsilon_n.\, \textbf{then } (\mathcal{B}^{n'}_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \text{ and}\\ \forall\, x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\, \textbf{if } x_\gamma \in \Theta_n.\, \textbf{then } (\mathcal{B}^{n'}_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

By Lem. 16, we get

$$\begin{array}{l}\dagger''_n\ \forall \Upsilon_n, \Theta_n, \mathcal{B}^{n'}_1.\, \text{if } \mathcal{B}^n_1 \mapsto^{*\Upsilon_n;\Theta_n} \mathcal{B}^{n'}_1 \text{ then } \exists \Upsilon_{n_2} \mathcal{B}^{n'}_2 \text{ such that } \mathcal{B}^n_2 \mapsto^{*\Upsilon_{n_2}} \mathcal{B}^{n'}_2 \text{ and } \Upsilon_n \subseteq \Upsilon_{n_2} \text{ and}\\ \forall\, x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if } x_\gamma \in \Upsilon_n.\, \textbf{then } \forall m.\, (\mathcal{B}^{n'}_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \text{ and}\\ \forall\, x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\, \textbf{if } x_\gamma \in \Theta_n.\, \textbf{then} \forall m.\, (\mathcal{B}^{n'}_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

We instantiate the forall quantifier in $\dagger''_n$ by $\mathcal{B}^n_1$ and the sets Υ_n and Θ_n along which $\mathcal{B}^n_1$ sends and receives. Note that by definition we have $\Upsilon_1 \subseteq \bigcup\{\Upsilon_n\}_{n\in I}$ and $\Theta_1 \subseteq \bigcup\{\Theta_n\}_{n\in I}$. We get:

$$\begin{array}{l}\exists \Upsilon_{n_2}, \mathcal{B}^{n'}_2 \text{ such that } \mathcal{B}^n_2 \mapsto^{*\Upsilon_{n_2}} \mathcal{B}^{n'}_2 \text{ and}\\ \forall\, x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if } x_\gamma \in \Upsilon_n.\, \textbf{then } \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \text{ and}\\ \forall\, x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\textbf{if } x_\gamma \in \Theta_n.\, \textbf{then} \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

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

$$\begin{array}{l}\dagger'''_n\ \ \forall\, x_\gamma \in \mathbf{Out}(\Delta^n \Vdash K^n).\, \textbf{if } x_\gamma \in \Upsilon_n.\, \textbf{then } \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{\cdot;x_\gamma} \text{ and}\\ \forall\, x_\gamma \in \mathbf{In}(\Delta^n \Vdash K^n).\textbf{if } x_\gamma \in \Theta_n.\, \textbf{then} \forall m.\, (\mathcal{B}^n_1; \mathcal{B}^{n'}_2) \in \mathcal{V}[\![\Delta^n \Vdash K^n]\!]^{m+1}_{x_\gamma;\cdot}.\end{array}$$

Note that by definition $\mathbf{Out}(\Delta \Vdash K) \subseteq \bigcup_{n\in I} \mathbf{Out}(\Delta^n \Vdash K^n)$ and $\mathbf{In}(\Delta \Vdash K) \subseteq \bigcup_{n\in I} \mathbf{In}(\Delta^n \Vdash K^n)$.
We apply Lem. 12 to get the minimal sending configurations $\mathcal{B}_2^{n''}$ and set Υ'_n for each given Υ_n and $\mathcal{B}_2^n$. We get $\mathcal{B}_2^{n''}$ such that $\mathcal{B}_2^n \mapsto^{*\Upsilon'_n} \mathcal{B}_2^{n''}$. Since these configurations are minimal, for all $n \in I$, we have $\mathcal{B}_2^{n''} \mapsto^{*\Upsilon_{n_2}} \mathcal{B}_2^{n'}$.
Since $\Upsilon_n \subseteq \Upsilon'_n \subseteq \Upsilon_{n_2}$, we can apply the backward closure (Lem. 13) on $\dagger_n'''$ to get

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

Moreover, we apply forward closure on the second run Lem. 15 on assumptions $\dagger_n$ to get for all $n \in I$:

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

We can apply the forward closure on the second run since the conditions of Lem. 15 are satisfied, i.e. $\Upsilon_n \subseteq \Upsilon'_n$ and $\mathcal{B}_2^{n''}$is the minimal sending configuration with respect to $\mathcal{B}_2^n$ and Υ_n.
We build $\mathcal{D}'_2$ to be $\{\mathcal{B}_2^{n''}\}_{n\in I}$. We have $\{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\Upsilon_2} \{\mathcal{B}_2^{n''}\}_{n\in I}$ with $\Upsilon_1 \subseteq \Upsilon_2$, and $\{\mathcal{B}_2^{n''}\}_{n\in I} \mapsto^{*\Upsilon_3} \{\mathcal{B}_2^{n'}\}_{n\in I}$ with $\Upsilon_2 \subseteq \Upsilon_3$. We, then instantiate the existential quantifier in the goal $(\star)$ with Υ_2 and $\mathcal{D}'_2$ that we built for which we know $\{\mathcal{B}_2^n\}_{n\in I} \mapsto^{*\Upsilon_2} \mathcal{D}'_2$. We need to show

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

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

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

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

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

In both parts, we continue the proof by considering the type of x_γ. The type of x_γ determines whether we need to prove **Part 1.** or **Part 2**. We provide the detailed proof for two interesting cases, the proof of the rest of cases is similar.
Subcase 1. $x_\gamma{:}A \otimes B \in K$. This case corresponds to **Part 1.** of the goal in which we have $x_\gamma \in \mathbf{Out}(\Delta \Vdash x_\gamma{:}A \otimes B)$ and $x_\gamma \in \Upsilon_1$. By the structure of the configuration, there exists a tree $\mathcal{B}_1^\kappa$ that provides the root channel $K = K^\kappa = x_\gamma{:}A \otimes B$. We use assumption $\dagger_n^4$ for that specific channel ($\dagger_\kappa^4$), we have

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

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

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

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

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

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

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

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

and configurations $\{\mathcal{B}_1^n\}_{n\in I_2}$ and $\{\mathcal{B}_2^{n''}\}_{n\in I_2}$ provide the resources in $\Delta_2^{\kappa''}$. In other words, we have $\Delta = \Delta_1, \Delta_1^{\kappa'}, \Delta_2, \Delta_2^{\kappa'}$ and $K = x_\gamma{:}A \otimes B$ and

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

We also have

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

Recall that earlier we proved

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

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

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

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

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

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

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

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

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

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

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

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

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

Moreover, we have

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

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

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

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

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

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

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

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

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

Case 1. the first step is an internal step but not a communication between the sub-trees.

Without loss of generality, let's assume that the communication is internal to $\mathcal{B}_1^\kappa$ for some $\kappa \in I$ and all other trees $\mathcal{B}_1^n$ for $n \neq \kappa$ remain intact, i.e. we have $\{\mathcal{B}_1^n\}_{n\in I} = \{\mathcal{B}_1^n\}_{n\in I_1}\, \mathcal{B}_1^\kappa \{\mathcal{B}_1^n\}_{n\in I_2}$ and

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

By forward closure (Lem. 14) on the assumption $\dagger_\kappa$ (i.e.,$\dagger_n$for $n = \kappa$), we get

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

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

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

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

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

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

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

which completes the proof of this case.

Case 2. the first step is a communication between two sub-configurations. Without loss of generality, we assume that the communication is between trees indexed by $\kappa, \lambda \in I$, i.e., $\mathcal{B}_1^\kappa$ offers a resource $u_\alpha{:}T$ and $\mathcal{B}_1^\lambda$ uses the resource, and there is a message available along u_α that is received in the first step. The proof proceeds by case analysis on type of $u_\alpha^c{:}T$. We provide the detailed proof for one case. The proof of the rest of the cases is similar.

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

$$\begin{array}{lll}(i) & \Delta^\kappa \Vdash \mathcal{B}_1^\kappa :: K^\kappa & \text{where } K^\kappa = u_\alpha{:}A \otimes B\\ & \Delta^\kappa \Vdash \mathcal{B}_2^\kappa :: K^\kappa & \\ (ii) & \Delta^\lambda \Vdash \mathcal{B}_1^\lambda :: K^\lambda & \text{where } \Delta^\lambda = \Delta^{\lambda'}, u_\alpha{:}A \otimes B\\ & \Delta^\lambda \Vdash \mathcal{B}_2^\lambda :: K^\lambda & \end{array}$$

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

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

By $\dagger_\kappa$, Lem. 16, and $\mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha) \mapsto^{0\,\Upsilon_\kappa';\Theta_\kappa'} \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)$, we get

$$\begin{array}{l}\exists \Upsilon_{\kappa_2} \mathcal{B}_2^{\kappa'} \;\text{such that}\; \mathcal{B}_2^\kappa \mapsto^{*\Upsilon_{\kappa_2}} \mathcal{B}_2^{\kappa'} \;\text{and}\; \Upsilon_\kappa' \subseteq \Upsilon_{\kappa_2}\\ \forall x_\gamma \in \mathbf{Out}(\Delta^\kappa \Vdash u_\alpha{:}A \otimes B).\, \mathbf{if}\; x_\gamma \in \Upsilon_\kappa'.\, \mathbf{then}\; \forall m.\, (\mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha); \mathcal{B}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash u_\alpha{:}A \otimes B]\!]^{m+1}_{\cdot;x_\gamma} \;\text{and}\\ \forall x_\gamma \in \mathbf{In}(\Delta^\kappa \Vdash u_\alpha{:}A \otimes B).\, \mathbf{if}\; x_\gamma \in \Theta_\kappa'.\, \mathbf{then}\, \forall m.\, (\mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha); \mathcal{B}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash u_\alpha{:}A \otimes B]\!]^{m+1}_{x_\gamma;\cdot}\end{array}$$

In particular, we know that $u_\alpha{:}A \otimes B \in \mathbf{Out}(\Delta^\kappa \Vdash u_\alpha{:}A \otimes B)$ and $u_\alpha \in \Upsilon_\kappa'$, which gives us:

$$\forall m.\, (\mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha); \mathcal{B}_2^{\kappa'}) \in \mathcal{V}[\![\Delta^\kappa \Vdash u_\alpha{:}A \otimes B]\!]^{m+1}_{\cdot;u_\alpha}$$

By **Row 4.** of the logical relation, we have $\Delta^\kappa = \Delta_1^\kappa, \Delta_2^\kappa$ and

$$\begin{array}{l}\mathcal{B}_2^{\kappa'} = \mathcal{B}_2^{\kappa''} \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha), \;\text{and}\\ \circ' \forall m.\, (\mathcal{A}_1; \mathcal{A}_2) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash y_\beta{:}A]\!]^m \qquad \text{and} \qquad \circ''\; \forall m.\, (\mathcal{B}_1^{\kappa'}; \mathcal{B}_2^{\kappa''}) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash u_{\alpha+1}{:}B]\!]^m.\end{array}$$

Next, we consider $\mathcal{B}_1^\lambda$. By $\dagger_\lambda$, Lem. 16, and $\mathcal{B}_1^\lambda \mapsto^{0\,\Upsilon_\lambda'} \mathcal{B}_1^\lambda$, we get

$$\begin{array}{l}\exists \Upsilon_{\lambda_2} \mathcal{B}_2^{\lambda'} \;\text{such that}\; \mathcal{B}_2^\lambda \mapsto^{*\Upsilon_{\lambda_2}} \mathcal{B}_2^{\lambda'} \;\text{and}\\ \forall x_\gamma \in \mathbf{Out}(\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda).\, \mathbf{if}\; x_\gamma \in \Upsilon_\lambda'.\, \mathbf{then}\; \forall m.\, (\mathcal{B}_1^\lambda; \mathcal{B}_2^{\lambda'}) \in \mathcal{V}[\![\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda]\!]^{m+1}_{\cdot;x_\gamma} \;\text{and}\\ \forall x_\gamma \in \mathbf{In}(\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda).\, \mathbf{if}\; x_\gamma \in \Theta_\lambda'.\, \forall m.\, (\mathcal{B}_1^\lambda; \mathcal{B}_2^{\lambda'}) \in \mathcal{V}[\![\Delta^\lambda, u_\alpha{:}A \otimes B \Vdash K^\lambda]\!]^{m+1}_{x_\gamma;\cdot}\end{array}$$

In particular, we know that $u_\alpha{:}A \otimes B \in \mathbf{In}(\Delta^{\lambda'}, u_\alpha{:}A \otimes B \Vdash K^\lambda)$ and also $u_\alpha \in \Theta'_\lambda$, which gives us:

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

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

$$\forall m.\, (\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_1^\lambda; \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'}) \in \mathcal{E}[\![\Delta^\lambda, y_\beta{:}A, u_{\alpha+1}{:}B \Vdash K^\lambda]\!]^m.$$

By forward closure (Lem. 14) and $\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_1^\lambda \mapsto \mathcal{B}_1^{\lambda'}$ we have:

$$\circ'''\ \forall m.\, (\mathcal{B}_1^{\lambda'}; \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'}) \in \mathcal{E}[\![\Delta, y_\beta{:}A, u_{\alpha+1}{:}B \Vdash K]\!]^m$$

Put $I' = I, \ell$, where ℓ does not occur in I and define $\mathcal{B}_1^\ell = \mathcal{A}_1$ and $\mathcal{B}_2^\ell = \mathcal{A}_2$. By induction on the number of steps, $\circ'$, $\circ''$, $\circ'''$ and $\dagger_n$ for $n \neq \kappa, \lambda$ we get

$\forall \Upsilon_1, \Theta_1, \mathcal{D}'_1.\, \mathbf{if}\ \{\mathcal{B}_1^n\}_{n \in I_1} \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathcal{B}_1^{\lambda'} \{\mathcal{B}_1^n\}_{n \in I_2} \mapsto^{j'\,\Upsilon_1;\Theta_1} \mathcal{D}'_1$
$\quad\quad \mathbf{then}\, \exists \Upsilon_2, \mathcal{D}'_2$ such that $\{\mathcal{B}_2^n\}_{n \in I_1} \mathcal{B}_2^{\kappa''} \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'} \{\mathcal{B}_2^n\}_{n \in I_2} \mapsto^{*\Upsilon_2} \mathcal{D}'_2$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall\, x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\gamma \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall\, x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\, x_\gamma \in \Theta_1.\, \mathbf{then}\ (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

By a $\forall\mathbf{E}$. and $\{\mathcal{B}_1^n\}_{n \in I_1} \mathcal{B}_1^{\kappa'} \mathcal{A}_1 \mathcal{B}_1^{\lambda'} \{\mathcal{B}_1^n\}_{n \in I_2} \mapsto^{j'\,\Upsilon_1} \mathcal{D}'_1$ we get:

$\exists \Upsilon_2, \mathcal{D}'_2$ such that $\{\mathcal{B}_2^n\}_{n \in I_1} \mathcal{B}_2^{\kappa''} \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'} \{\mathcal{B}_2^n\}_{n \in I_2} \mapsto^{*\Upsilon_2} \mathcal{D}'_2$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall\, x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\gamma \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall\, x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\, x_\gamma \in \Theta_1.\, \mathbf{then}\ (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

We also know that $\{\mathcal{B}_2^n\}_{n \in I} = \{\mathcal{B}_2^n\}_{n \in I_1} \mathcal{B}_2^\kappa \mathcal{B}_2^\lambda \{\mathcal{B}_2^n\}_{n \in I_2} \mapsto^* \{\mathcal{B}_2^n\}_{n \in I_1} \mathcal{B}_2^{\kappa''} \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'} \{\mathcal{B}_2^n\}_{n \in I_2}$. We get the following for the same Υ_2 and $\mathcal{D}'_2$:

$\exists \Upsilon_2, \mathcal{D}'_2$ such that $\{\mathcal{B}_2^n\}_{n \in I_1} \mathcal{B}_2^{\kappa''} \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{B}_2^{\lambda'} \{\mathcal{B}_2^n\}_{n \in I_2} \mapsto^{*\Upsilon_2} \mathcal{D}'_2$ and $\Upsilon_1 \subseteq \Upsilon_2$ and
$\forall\, x_\gamma \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\gamma \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\gamma}$ and
$\forall\, x_\gamma \in \mathbf{In}(\Delta \Vdash K).\, \mathbf{if}\, x_\gamma \in \Theta_1.\, \mathbf{then}\ (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\gamma;\cdot}$.

□

XI. Logical equivalence

A. Equivalence

Lemma 18 (Reflexivity). *For all security levels ξ and configurations $\Delta \Vdash \mathcal{D} :: x_\alpha{:}T$, we have*

$$(\Delta \Vdash \mathcal{D} :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D} :: x_\alpha{:}T).$$

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

Lemma 19 (Symmetry). *For all configurations $\mathcal{D}_1$ and $\mathcal{D}_2$, we have $(\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T)$, iff $(\Delta \Vdash \mathcal{D}_2 :: x_\alpha{:}T) \equiv (\Delta \Vdash \mathcal{D}_1 :: x_\alpha{:}T)$,*

Proof. The proof is straightforward by the definition of triple equality. □

Lemma 20 (Transitivity). *For all configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, we have*

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

Proof. The proof follows from Corollary 3 when we instantiate Ψ_0 with the trivial lattice only containing the element $\perp$ and ξ to be $\perp$. □

B. Noninterference

Lemma 21 (Reflexivity – only for IFC-typed processes). *For all security levels ξ and configurations $\Psi_0; \Gamma \Vdash \mathbb{D} :: x_\alpha{:}T[c]$, we have*

$$(\Gamma \Vdash |\mathbb{D}| :: x_\alpha{:}T[c]) \equiv_\xi^{\Psi_0} (\Gamma \Vdash |\mathbb{D}| :: x_\alpha{:}T[c]).$$

Proof. Corollary of the fundamental theorem (Thm. 5). □

Lemma 22 (Symmetry). *For all security levels ξ and configurations $\mathcal{D}_1$ and $\mathcal{D}_2$, we have $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1]) \equiv_\xi^{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2])$, iff $(\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}T_2[c_2]) \equiv_\xi^{\Psi_0} (\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}T_1[c_1])$.*

Proof. The proof is straightforward by Def. 17 □

Corollary 3 (Transitivity). *For all security levels ξ, and configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, we have*

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

Proof. Consider arbitrary $\mathcal{C}_1$, $\mathcal{F}_1$ and $\mathcal{C}_3$, and $\mathcal{F}_3$ we need to show

$$\forall m.\, (\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1; \mathcal{C}_3\mathcal{D}_3\mathcal{F}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m\ \mathsf{and}\ \forall m.\, (\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1; \mathcal{C}_3\mathcal{D}_3\mathcal{F}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$$

By the assumptions, we get $(\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1; \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $(\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2; \mathcal{C}_3\mathcal{D}_3\mathcal{F}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$. The result follows by Lem. 23. □

Lemma 23 (Transitivity of the term relation). *If $\dagger_1\, \forall m.\, (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\dagger_2\, \forall m.\, (\mathcal{D}_2; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ then $\star\, \forall k.\, (\mathcal{D}_1; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$.*

Proof. Our goal is to prove for all $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ with $\forall m.\, (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m.\, (\mathcal{D}_2; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ then $\forall k.\, (\mathcal{D}_1; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$. We prove an equivalent statement that says for all k, $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ with $\forall m.\, (\mathcal{D}_1; \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m.\, (\mathcal{D}_2; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ then $(\mathcal{D}_1; \mathcal{D}_3) \in \mathcal{E}[\![\Delta \Vdash K]\!]^k$. We proceed by induction on k.

Base case. ($k = 0$) Consider arbitrary configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. By the assumptions, we know that $(\mathcal{D}_1; \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$ and $(\mathcal{D}_2; \mathcal{D}_3) \in \mathsf{Tree}(\Delta \Vdash K)$, which gives us $(\mathcal{D}_1; \mathcal{D}_3) \in \mathsf{Tree}(\Delta \Vdash K)$. It is enough to complete the proof in this case.

Inductive case. ($k = k' + 1$) Consider arbitrary configurations $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. Our goal is to show

$$\begin{array}{l}\forall \mathcal{D}_1'.\, \forall \Upsilon_1.\ \mathsf{if}\ \mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}_1',\ \mathsf{then}\ \exists \mathcal{D}_3'\ \mathsf{such\ that}\ \mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}_3'\ \mathsf{and}\\ \forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\ x_\alpha \in \Upsilon_1.\, \mathbf{then}\ (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot; x_\alpha}\ \mathsf{and}\\ \forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, (\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha; \cdot}.\end{array}$$

Consider an arbitrary $\mathcal{D}_1'$ and Υ_1, and assume $\mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}_1'$. Our goal is to prove

$$\begin{array}{l}\exists \mathcal{D}'_3 \text{ such that } \mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}'_3 \text{ and} \\ \forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, (\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha} \text{ and} \\ \forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, (\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha;\cdot}.\end{array}$$

By assumption $\dagger_1$ and Lem. 16, we have

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

And by assumption $\dagger_2$ and Lem. 16, we have

$$\begin{array}{l}\dagger'_2\, \forall \mathcal{D}'_2.\, \forall \Upsilon_1.\, \text{if } \mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}'_2, \text{ then } \exists \mathcal{D}'_3 \text{ such that } \mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}'_3 \text{ and} \\ \forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m.\, (\mathcal{D}'_2; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha} \text{ and} \\ \forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m.\, (\mathcal{D}'_2; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.\end{array}$$

We apply $\forall\mathbf{E}$. on $\dagger'_1$ by instantiating the existential quantifiers with $\mathcal{D}'_1$ and Υ_1. We can apply $\mathbf{if E}$. since we know $\mathcal{D}_1 \mapsto^{*\Upsilon_1} \mathcal{D}'_1$. We get a $\mathcal{D}'_2$ with $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}'_2$. Next, we apply $\forall\mathbf{E}$. on $\dagger_2$ by instantiating the existential quantifiers with $\mathcal{D}'_2$ and Υ_1. We apply $\mathbf{if E}$. as we know $\mathcal{D}_2 \mapsto^{*\Upsilon_1} \mathcal{D}'_2$. As a result, we get a $\mathcal{D}'_3$ with $\mathcal{D}_3 \mapsto^{*\Upsilon_1} \mathcal{D}'_3$ as required by the goal. We need to prove:

$$\begin{array}{l}\forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, (\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha} \text{ and} \\ \forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, (\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha;\cdot}.\end{array}$$

By $\dagger'_1$, and $\dagger'_2$, we have as assumptions

$$\begin{array}{l}\dagger''_1\ \forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m. (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha} \text{ and} \\ \quad \forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m. (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.\end{array}$$

and

$$\begin{array}{l}\dagger''_2\ \forall x_\alpha \in \mathbf{Out}(\Delta \Vdash K).\, \mathbf{if}\, x_\alpha \in \Upsilon_1.\, \mathbf{then}\, \forall m.\, (\mathcal{D}'_2; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha} \text{ and} \\ \quad \forall x_\alpha \in \mathbf{In}(\Delta \Vdash K).\, \forall m.\, (\mathcal{D}'_2; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.\end{array}$$

There are two parts to prove:

Part 1. Consider an arbitrary $x_\alpha \in \mathbf{Out}(\Delta \Vdash K)$ and assume $x_\alpha \in \Upsilon_1$. Our goal is to show

$$(\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha}$$

And by $\dagger''_1$, and $\dagger''_2$, we have as assumptions

$$\dagger'''_1 \forall m. (\mathcal{D}'_1; \mathcal{D}'_2) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha} \quad \text{and} \quad \dagger'''_2 \forall m.\, (\mathcal{D}'_2; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{\cdot;x_\alpha}.$$

We consider cases based on the type of x_α. We provide the detailed proof for a few interesting cases. The proof of other cases is similar.

Case 1. ($K = x_\alpha{:}A \otimes B$). By $\dagger'''_1$ and $\dagger'''_2$we get

$$\begin{array}{lll}\Delta = \Delta_1, \Delta_2 & \text{and} \quad \mathcal{D}'_1 = \mathcal{D}''_1 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) & \text{and} \quad \mathcal{D}'_2 = \mathcal{D}''_2 \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) \\ & \text{and} \quad \mathcal{D}'_2 = \mathcal{D}''_2 \mathcal{A}_1 \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha) & \text{and} \quad \mathcal{D}'_3 = \mathcal{D}''_3 \mathcal{A}_2 \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\end{array}$$

Moreover,

$$\begin{array}{ll}\forall m. (\mathcal{D}''_1; \mathcal{D}''_2) \in \mathcal{E}[\![\Delta_1 \Vdash x_{\alpha+1}{:}B]\!]^m & \forall m. (\mathcal{A}_1; \mathcal{A}_2) \in \mathcal{E}[\![\Delta_2 \Vdash y_\beta{:}A]\!]^m \\ \forall m. (\mathcal{D}''_2; \mathcal{D}''_3) \in \mathcal{E}[\![\Delta_1 \Vdash x_{\alpha+1}{:}B]\!]^m & \forall m. (\mathcal{A}_2; \mathcal{A}_3) \in \mathcal{E}[\![\Delta_2 \Vdash y_\beta{:}A]\!]^m\end{array}$$

We apply the induction hypothesis on a smaller observation index k' to get

$$(\mathcal{D}''_1; \mathcal{D}''_3) \in \mathcal{E}[\![\Delta_1 \Vdash x_{\alpha+1}{:}B]\!]^{k'} \qquad (\mathcal{A}_1; \mathcal{A}_3) \in \mathcal{E}[\![\Delta_2 \Vdash y_\beta{:}A]\!]^{k'}$$

Which by **Row 4** of the logical relation is wnough to prove the goal of this subcase, i.e.,

$$(\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{\cdot;x_\alpha}$$

Case 2. ($\Delta = x_\alpha{:}A \multimap B, \Delta'$)

Part 2. Consider an arbitrary $x_\alpha \in \mathbf{In}(\Delta \Vdash K)$ and assume $x_\alpha \in \Theta_1 x$. Our goal is to show

$$(\mathcal{D}'_1; \mathcal{D}'_3) \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha;\cdot}.$$

By $\dagger_1''$, and $\dagger_2''$, we have as assumptions

$$\dagger_1''' \, \forall m.(\mathcal{D}_1'; \mathcal{D}_2') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot} \quad \text{and} \quad \dagger_2''' \, \forall m.\,(\mathcal{D}_2'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{m+1}_{x_\alpha;\cdot}.$$

We consider cases based on the type of x_α. We provide the detailed proof for a few interesting cases. The proof of other cases is similar.

Case 1. **(**$\Delta = \Delta', x_\alpha{:}A \otimes B$**)** By $\dagger_1'''$ and $\dagger_2'''$we get for all $y_\beta \notin dom(\Delta, x_\alpha : A \otimes B, K)$.

$$\forall m.(\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{D}_1'; \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{D}_2') \in \mathcal{E}[\![\Delta, y_\beta : A, x_{\alpha+1} : B \Vdash K]\!]^m$$
$$\forall m.(\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{D}_2'; \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{D}_3') \in \mathcal{E}[\![\Delta, y_\beta : A, x_{\alpha+1} : B \Vdash K]\!]^m$$

We apply the induction hypothesis on a smaller observation index k' to get

$$\forall y_\beta \notin dom(\Delta, x_\alpha : A \otimes B, K).\, (\mathbf{msg}(\mathbf{send} y_\beta\, u_\alpha)\mathcal{D}_1'; \mathbf{msg}(\mathbf{send} y_\beta\, x_\alpha)\mathcal{D}_3') \in \mathcal{E}[\![\Delta, y_\beta : A, x_{\alpha+1} : B \Vdash K]\!]^{k'}$$

Which by **Row 9** of the logical relation is wnough to prove the goal of this subcase, i.e.,

$$(\mathcal{D}_1'; \mathcal{D}_3') \in \mathcal{V}[\![\Delta \Vdash K]\!]^{k'+1}_{x_\alpha;\cdot}$$

Case 2. **(**$K = x_\alpha{:}A \multimap B$**)**

□

Internal transition $\xrightarrow{\tau}$ *defined as:*
$\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ iff $\mathcal{D}_1 \mapsto \mathcal{D}_1'$

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

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

Fig. 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* $\Delta \Vdash \mathcal{D} :: \Delta'$*, we define* $\mathsf{fn}(\mathcal{D})$ *as* $dom(\Delta, \Delta')$*.*

Definition 23 (Weak transition relations)**.**

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

Definition 24 (Asynchronous bisimilarity)**.** *Asynchronous bisimilarity is the largest symmetric relation such that whenever* $\mathcal{D}_1 \approx_a \mathcal{D}_2$*, we have*
1) $(\tau - \mathtt{step})$ *if* $\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'. \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathcal{D}_2'$*,*
2) (output) *if* $\mathcal{D}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'. \mathcal{D}_2 \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathcal{D}_2'$*.*
3) (left input) *for all* $q \notin \mathsf{fn}(\mathcal{D}_1)$*., if* $\mathcal{D}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'. \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}_2'$*,*
4) (right input) *for all* $q \notin \mathsf{fn}(\mathcal{D}_1)$*., if* $\mathcal{D}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{D}_1'$ *then* $\exists \mathcal{D}_2'. \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}_2'$ *and* $\mathcal{D}_1' \approx_a \mathcal{D}_2'\mathbf{msg}(x_\alpha.q)$*.*
where $\mathbf{msg}(x_\alpha.q)$ *is defined as* $\mathbf{msg}(\mathbf{close}\, x_\alpha)$ *if* $q = \mathbf{close}$*,* $\mathbf{msg}(x_\alpha.k)$ *if* $q = k$*, and* $\mathbf{msg}(\mathbf{send}\, z_\delta\, x_\alpha)$ *if* $q = z_\delta$*.* ◇

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

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

◇

Definition 26. *For $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$, $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$ we define*

$$\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1] \approx^{\xi}_{a} \Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2] \text{ as}$$

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

Corollary 4. *For all $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash x_\alpha{:}A_1)$ and $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash y_\beta{:}A_2)$, we have $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^{\Psi_0}_{\xi} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ iff $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \approx^{\xi}_{a} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$.*

Proof. The proof is straightforward by considering Def. 17, Def. 26, and the following corollary (Corollary 5). □

Corollary 5. *for all $(\mathcal{D}_1, \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$, we have $\forall m.(\mathcal{D}_1, \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m.(\mathcal{D}_2, \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ iff $\mathcal{D}_1 \approx_a \mathcal{D}_2$.*

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

Lemma 24. *Consider a pair of session-typed forests $(\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{Forest}(\Delta \Vdash \Delta')$ consisting of multiple session-typed trees indexed in the set I, i.e., $\mathcal{C}_i = \{\mathcal{C}^j_i\}_{j \in I}$ for $i \in \{1, 2\}$, and $(\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$ with $\Delta = \{\Delta^j\}_{j\in I}$ and $\Delta' = \{K^j\}_{j \in I}$. We have:*

$$\forall j \in I. \forall m.\, (\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m \text{ and } (\mathcal{C}^j_2, \mathcal{C}^j_1) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m \text{ iff } \mathcal{C}_1 \approx_a \mathcal{C}_2.$$

Proof. The proof consists of two parts.

(1) Soundness. We want to prove for any arbitrary pair of forests $\mathcal{C}_1 = \{\mathcal{C}^j_1\}_{j\in I}$ and $\mathcal{C}_2 = \{\mathcal{C}^j_2\}_{j \in I}$ such that $\forall j \in I.\, (\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, if $\forall j \in I. \forall m.(\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ and $\forall j \in I. \forall m.(\mathcal{C}^j_2, \mathcal{C}^j_1) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ then $\{\mathcal{C}^j_1\}_{j \in I} \approx_a \{\mathcal{C}^j_2\}_{j\in I}$.

We proceed the proof by coninduction on the generating function of the bisimilarity. Consider an arbitrary $\mathcal{C}_1 = \{\mathcal{C}^j_1\}_{j\in I}$ and $\mathcal{C}_2 = \{\mathcal{C}^j_2\}_{j\in I}$ such that $\forall j \in I.\, (\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, and assume $\forall j \in I. \forall m.(\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$. Our goal is to show $\{\mathcal{C}^j_1\}_{j\in I} \approx_a \{\mathcal{C}^j_2\}_{j\in I}$. We prove it by showing the items (1)-(4) Def. 24 by assuming $\forall j \in I. \forall m.(\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ only. The symmetric relation can be established with the other assumption $\forall j \in I. \forall m.(\mathcal{C}^j_2, \mathcal{C}^j_1) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ We continue by establishing items (1)-(4) in Def. 24:

1) Assume $\mathcal{C}_1 \xrightarrow{\tau} \mathcal{C}'_1$, then by the definition of $\tau-$transition, we have $\mathcal{C}_1 \mapsto \mathcal{C}'_1$. In other words we have $\{\mathcal{C}^j_1\}_{j\in I} \mapsto \{\mathcal{C}^{j'}_1\}_{j \in I}$ for $\mathcal{C}'_1 = \{\mathcal{C}^{j'}_1\}_{j\in I}$. To establish item (1) in the definition, it is enough to prove that $\{\mathcal{C}^{j'}_1\}_{j\in I} \approx_a \{\mathcal{C}^j_2\}_{j \in I}$. Since we unfolded the conindcutive definition of the bisimilarity's generating function once, we can apply the coinductive case if the assumptions are satisified, i.e., we need to prove that each tree in the post-step forest is session-typed and the pairs are related by the logical relation.
By assumption $\forall j \in I. \forall m.(\mathcal{C}^j_1, \mathcal{C}^j_2) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ and by the forward closure lemma (Lem. 14), we get $\forall j \in I. \forall m.(\mathcal{C}^{j'}_1, \mathcal{C}^j_2) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$. This along with type-preservation of session-typed programs, is enough to get $\{\mathcal{C}^{j'}_1\}_{j\in I} \approx_a \{\mathcal{C}^j_2\}_{j\in I}$ by coinduction.
2) Assume $\mathcal{C}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{C}'_1$, then by Fig. 7 we have $\mathcal{C}_1 = \mathcal{C}^1_1 \mathbf{msg}(x_\alpha.q)\, \mathcal{C}^2_1$, with $\mathcal{C}'_1 = \mathcal{C}^1_1, \mathcal{C}^2_1$.
This means that for some $\kappa \in I$, we have either $\mathcal{C}^\kappa_1 = \mathcal{C}^{\kappa'}_1 \mathbf{msg}(x_\alpha.q)$ or $\mathcal{C}^\kappa_1 = \mathbf{msg}(x_\alpha.q)\mathcal{C}^{\kappa'}_1$, and $\mathcal{C}'_1 = \{\mathcal{C}^j_1\}_{j \in I - \{\kappa\}}\mathcal{C}^{\kappa'}_1$.
In particular, we know that $\forall m.(\mathcal{C}^\kappa_1, \mathcal{C}^\kappa_2) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m$, and $\mathcal{C}^\kappa_1 \mapsto^{0\Upsilon^\kappa_1;\Theta^\kappa_1} \mathcal{C}^\kappa_1$, such that $x_\alpha \in \Upsilon^\kappa_1$. By the definition of the term interpretation and Lem. 16, we get

$$\begin{aligned}&\forall\, \Upsilon^\kappa_1, \Theta^\kappa_1, \mathcal{C}^{\kappa'}_1 .\mathsf{if}\, \mathcal{C}^\kappa_1 \mapsto^{*\Upsilon^\kappa_1;\Theta^\kappa_1} \mathcal{C}^{\kappa'}_1, \mathsf{then}\, \exists \Upsilon^\kappa_2, \mathcal{C}^{\kappa'}_2 \,\mathsf{such\,that}\, \mathcal{C}^\kappa_2 \mapsto^{*\Upsilon^\kappa_2} \mathcal{C}^{\kappa'}_2 \text{ and } \Upsilon^\kappa_1 \subseteq \Upsilon^\kappa_2 \text{ and}\\ &\quad \forall\, x_\alpha \in \mathbf{Out}(\Delta^\kappa \Vdash K^\kappa).\, \mathbf{if}\, x_\alpha \in \Upsilon^\kappa_1.\, \mathbf{then}\, \forall\, k.\, (\mathcal{C}^{\kappa'}_1; \mathcal{C}^{\kappa'}_2) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{\cdot;x_\alpha} \text{ and}\\ &\quad \forall\, x_\alpha \in \mathbf{In}(\Delta^\kappa \Vdash K^\kappa).\mathbf{if}\, x_\alpha \in \Theta^\kappa_1.\, \mathbf{then}\, \forall k.\, (\mathcal{C}^{\kappa'}_1; \mathcal{C}^{\kappa'}_2) \in \mathcal{V}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{k+1}_{x_\alpha;\cdot}.\end{aligned}$$

We can instantiate the $\forall$ quantifiers and apply an $\mathbf{if}-$Elimination rule to get $\mathcal{C}^\kappa_2 \mapsto^{*\Upsilon^\kappa_2} \mathcal{C}^{\kappa'}_2$ for some $\Upsilon^\kappa_2 \supseteq \Upsilon^\kappa_1$ and some $\mathcal{C}^{\kappa'}_2$. Since $x_\alpha \in \Upsilon^\kappa_1$, we get by the definition of the logical relation $\forall m.(\mathcal{C}^\kappa_1, \mathcal{C}^{\kappa'}_2) \in \mathcal{V}^{\xi}_{\Psi_0}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m+1}_{\cdot;x_\alpha}$. This (depending on the type of the channel x_α) gives us $\mathcal{C}^{\kappa'}_2 = \mathcal{C}_{\kappa''_2}\mathbf{msg}(x_\alpha.q)$ or $\mathcal{C}^{\kappa'}_2 = \mathbf{msg}(x_\alpha.q)\mathcal{C}_{\kappa''_2}$. Note that if q is a channel name, $\mathcal{C}^{\kappa''}_2$ is a forest, rather than a tree, and if q is the label $\mathbf{close}$, the configuration $\mathcal{C}^{\kappa''}_2$ is empty.
Put $\mathcal{C}'_2 = \{\mathcal{C}^j_2\}_{j \in I-\{\kappa\}}, \mathcal{C}^{\kappa''}_2$. To establish item (2) in the definition, it is enough to prove that $\mathcal{C}'_1 \approx_a \mathcal{C}'_2$. Since we unfolded the conindcutive definition of the bisimilarity's generating function once, we can apply the coinductive case if the assumptions are satisified, i.e., we need to prove that each tree in the post-step forest is session-typed and the pairs are related by the logical relation.
The proof proceeds by cases on the type of x_α. Here we only provide one interesting case, the proof of other cases is similar.

Subcase 1. $K^\kappa = x_\alpha{:}A \otimes B \in \Delta'$**.** We have $\Delta^\kappa = \Delta_1^\kappa, \Delta_2^\kappa$ and $\mathcal{C}_1^\kappa = \mathcal{C}_1^{\kappa'}\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$ with $\mathcal{C}_1^{\kappa'} = \mathcal{C}_1^{\kappa''}\mathcal{A}_1$. Moreover, we have $\mathcal{C}_2^{\kappa'} = \mathcal{C}_2^{\kappa''}\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$ with $\mathcal{C}_2^{\kappa''} = \mathcal{C}_2^{\kappa'''}\mathcal{A}_2$. And $\forall m.(\mathcal{C}_1^{\kappa''};\mathcal{C}_2^{\kappa'''}) \in \mathcal{E}[\![\Delta_1^\kappa \Vdash x_{\alpha+1}{:}B]\!]^m$ and $\forall m.(\mathcal{A}_1;\mathcal{A}_2) \in \mathcal{E}[\![\Delta_2^\kappa \Vdash y_\delta{:}A]\!]^m$.
Recall that all other trees in the forest are still related, i.e. $\forall j \in I - \{\kappa\}$, we have $\forall m.(\mathcal{C}_1^j;\mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ This is enough to apply the coinductive argument and get $\mathcal{C}_1' \approx_a \mathcal{C}_2'$ where $\mathcal{C}_1' = \{\mathcal{C}_1^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_1^{\kappa'}$ and $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_2^{\kappa''}$.

3) Assume for an arbitrary q, we have $\mathcal{C}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{C}_1'$. By Fig. 7, there is a process $\mathbf{proc}(z_\beta, P_{x_\alpha}) \in \mathcal{C}_1$ which is waiting to receive a message along x_α. In particular, for some tree $\mathcal{C}_1^\kappa$ for $\kappa \in I$, we have $\mathbf{proc}(z_\beta, P_{x_\alpha}) \in \mathcal{C}_1^\kappa$.
We know that $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^\kappa) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m$, and $\mathcal{C}_1^\kappa \mapsto^{0\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^\kappa$, such that $x_\alpha \in \Theta_1^\kappa$. By the definition of the term interpretation and Lem. 16, we get

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

We can instantiate the $\forall$ quantifiers and apply an $\mathbf{if}-$Elimination rule to get $\mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'}$ for some $\Upsilon_2^\kappa \supseteq \Upsilon_1^\kappa$ and some $\mathcal{C}_2^{\kappa'}$. Since $x_\alpha \in \Theta_1^\kappa$, we get by the definition of the logical relation $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^{\kappa'}) \in \mathcal{V}^\xi_{\Psi_0}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m+1}_{x_\alpha;\cdot}$. We put $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_2^{\kappa'}$.
The proof proceeds by cases on the type of x_α. Here we only provide one interesting case, the proof of other cases is similar.

Subcase 1. $x_\alpha{:}A \otimes B \in \Delta^\kappa \subseteq \Delta$ **(i.e.,** $\Delta^\kappa = \Delta^{\kappa'}, x_\alpha{:}A \otimes B$**)** In this case, we know that $\mathcal{C}_1^\kappa = \mathcal{C}_1^{\kappa_1}\,\mathbf{proc}(z_\delta, w \leftarrow \mathbf{recv}\, x_\alpha)\,\mathcal{C}_1^{\kappa_2}$ and q is a channel y_δ which is fresh in both $\mathcal{C}_1^\kappa$ and $\mathcal{C}_2^{\kappa'}$. Moreover, we have $\mathcal{C}_1^{\kappa'} = \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_1^\kappa$, and thus $\mathcal{C}_1' = \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_1$.
By the logical relation, we get $\forall m.(\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_1^\kappa; \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_2^{\kappa'}) \in \mathcal{E}[\![\Delta^{\kappa'}, y_\delta{:}A, x_{\alpha+1}{:}B \Vdash K^\kappa]\!]^m$.
Recall that all other trees in the forest are still related, i.e. $\forall j \in I - \{\kappa\}$, we have $\forall m.(\mathcal{C}_1^j;\mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ This is enough to apply the coinductive argument and get $\mathcal{C}_1' \approx_a \mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)\mathcal{C}_2'$ as required.

4) Assume for an arbitrary q, we have $\mathcal{C}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{C}_1'$. The proof is similar to the previous case:
By Fig. 7, there is a process $\mathbf{proc}(x_\alpha, P_{x_\alpha}) \in \mathcal{C}_1$ which is waiting to receive a message along x_α. In particular, for some tree $\mathcal{C}_1^\kappa$ for $\kappa \in I$, we have $\mathbf{proc}(x_\alpha, P) \in \mathcal{C}_1^\kappa$.
We know that $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^\kappa) \in \mathcal{E}[\![\Delta^\kappa \Vdash K^\kappa]\!]^m$, and $\mathcal{C}_1^\kappa \mapsto^{0\Upsilon_1^\kappa;\Theta_1^\kappa} \mathcal{C}_1^\kappa$, such that $x_\alpha \in \Theta_1^\kappa$. By the definition of the term interpretation and Lem. 16, we get

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

We can instantiate the $\forall$ quantifiers and apply an $\mathbf{if}-$Elimination rule to get $\mathcal{C}_2^\kappa \mapsto^{*\Upsilon_2^\kappa} \mathcal{C}_2^{\kappa'}$ for some $\Upsilon_2^\kappa \supseteq \Upsilon_1^\kappa$ and some $\mathcal{C}_2^{\kappa'}$. Since $x_\alpha \in \Theta_1^\kappa$, we get by the definition of the logical relation $\forall m.(\mathcal{C}_1^\kappa, \mathcal{C}_2^{\kappa'}) \in \mathcal{V}^\xi_{\Psi_0}[\![\Delta^\kappa \Vdash K^\kappa]\!]^{m+1}_{x_\alpha;\cdot}$. We put $\mathcal{C}_2' = \{\mathcal{C}_2^j\}_{j\in I-\{\kappa\}}, \mathcal{C}_2^{\kappa'}$.
The proof proceeds by cases on the type of x_α. Here we only provide one interesting case, the proof of other cases is similar.

Subcase 1. $K^\kappa = x_\alpha{:}A \multimap B \in \Delta'$**.** In this case, we know that $\mathcal{C}_1^\kappa = \mathcal{C}_1^{\kappa_1}\,\mathbf{proc}(x_\alpha, w \leftarrow \mathbf{recv}\, x_\alpha)\,\mathcal{C}_1^{\kappa_2}$ and q is a channel y_δ which is not free in $\mathcal{C}_1^\kappa$ (and by the typing not in $\mathcal{C}_2^{\kappa'}$ either). Moreover, we have $\mathcal{C}_1^{\kappa'} = \mathcal{C}_1^\kappa\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$, and thus $\mathcal{C}_1' = \mathcal{C}_1\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$.
By the logical relation, we get $\forall m.(\mathcal{C}_1^\kappa\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha); \mathcal{C}_2^{\kappa'}\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)) \in \mathcal{E}[\![\Delta^\kappa, y_\delta{:}A \Vdash x_{\alpha+1}{:}B]\!]^m$.
Recall that all other trees in the forest are still related, i.e. $\forall j \in I - \{\kappa\}$, we have $\forall m.(\mathcal{C}_1^j;\mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$ This is enough to apply the coinductive argument and get $\mathcal{C}_1' \approx_a \mathcal{C}_2'\mathbf{msg}(\mathbf{send}\, y_\delta\, x_\alpha)$ as required.

(2) Completeness. We want to prove for any arbitrary pair of forests $\mathcal{C}_1 = \{\mathcal{C}_1^j\}_{j\in I}$ and $\mathcal{C}_2 = \{\mathcal{C}_2^j\}_{j\in I}$ such that $\forall j \in I.\, (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, if $\{\mathcal{C}_1^j\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$ then $\forall j \in I.\forall m.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$.
We instead prove an equivalent statement that says

For any natural number m and any arbitrary pair of forests $\mathcal{C}_1 = \{\mathcal{C}_1^j\}_{j\in I}$ and $\mathcal{C}_2 = \{\mathcal{C}_2^j\}_{j\in I}$ such that $\forall j \in I.\, (\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathsf{Tree}(\Delta^j \Vdash K^j)$, if $\{\mathcal{C}_1^j\}_{j\in I} \approx_a \{\mathcal{C}_2^j\}_{j\in I}$ then $\forall j \in I.(\mathcal{C}_1^j, \mathcal{C}_2^j) \in \mathcal{E}[\![\Delta^j \Vdash K^j]\!]^m$.

We proceed by induction on m.

Base case. $(m = 0)$ The proof is straightforward by the definition of logical relation and the fact that the configurations are session-typed.

Inductive case. $(m = m' + 1)$ Consider an arbitrary $\kappa \in I$. By the definition of the term interpretation, we need to prove

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

Consider an arbitrary Υ_1^κ, Θ_1^κ, and $\mathcal{C}_1^{\kappa'}$, and assume $\mathcal{C}_1^{\kappa'}\mapsto^{*\Upsilon_1^\kappa;\Theta_1^\kappa}\mathcal{C}_1^{\kappa'}$, we need to prove

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

By local transition steps, we know that $\mathcal{C}_1\mapsto^*\mathcal{C}_1'$, where $\mathcal{C}_1'=\{\mathcal{C}_1^\kappa\}_{j\in I-\{\kappa\}}\mathcal{C}_1^{\kappa'}$. By $\mathcal{C}_1\approx_a\mathcal{C}_2$, we can apply the clause for τ-transitions in Def. 24 for zero or multiple times, to get $\mathcal{C}_2\mapsto^*\mathcal{C}_2'$ such that $\mathcal{C}_1'\approx_a\mathcal{C}_2'$. Consider the channels in the sets Υ_1^κ and Θ_1^k. There are three cases:

A. By item (2) of Def. 24 and $\mathcal{C}_1'\approx_a\mathcal{C}_2'$, we know that for all $x_\alpha\in\mathbf{Out}(\Delta\Vdash K)$ such that $x_\alpha\in\Upsilon_1^\kappa$, we have $\mathcal{C}_2'\mapsto^*\mathcal{C}_2^{x_s}$ such that $\mathcal{C}_2^{x_s}$ sends along the channel x_α and $\mathcal{C}_1'=\mathcal{C}_1^1\mathbf{msg}(x_\alpha.q)\mathcal{C}_1^2$ and $\mathcal{C}_2^{x_s}=\mathcal{C}_2^{x_s'}\mathbf{msg}(x_\alpha.q)\mathcal{C}_2^{x_s''}$ and we have $\mathcal{C}_1^1\mathcal{C}_1^2\approx_a\mathcal{C}_2^{x_s'}\mathcal{C}_2^{x_s''}$.

B. By item (3) of Def. 24 and $\mathcal{C}_1'\approx_a\mathcal{C}_2'$, we know that for all $x_\alpha\in\mathbf{In}(\Delta\Vdash\cdot)$ such that $x_\alpha\in\Theta_1^\kappa$, we have $\mathcal{C}_2'\mapsto^*\mathcal{C}_2^{x_r}$ such that $\mathbf{msg}(x_\alpha.q)\mathcal{C}_1'\approx_a\mathbf{msg}(x_\alpha.q)\mathcal{C}_2^{x_r}$.

C. By item (4) of Def. 24 and $\mathcal{C}_1'\approx_a\mathcal{C}_2'$, we know that for all $x_\alpha\in\mathbf{In}(\cdot\Vdash K)$ such that $x_\alpha\in\Theta_1^\kappa$, we have $\mathcal{C}_2'\mapsto^*\mathcal{C}_2^{x_r}$ such that $\mathcal{C}_1'\mathbf{msg}(x_\alpha.q)\approx_a\mathcal{C}_2^{x_r}\mathbf{msg}(x_\alpha.q)$.

Apply the confluence lemma (Lem. 11) on (i) $\mathcal{C}_2'\mapsto^*\mathcal{C}_2^{x_s}$ for all $x_\alpha\in\mathbf{Out}(\Delta\Vdash K)$ such that $x_\alpha\in\Upsilon_1^\kappa$ and (ii) $\mathcal{C}_2'\mapsto^*\mathcal{C}_2^{x_r}$ for all $x_\alpha\in\mathbf{In}(\Delta\Vdash K)$ such that $x_\alpha\in\Theta_1^\kappa$ to build $\mathcal{C}_2''$. In particular, by the confluence lemma, we get $\mathcal{C}_2'\mapsto^{*\Upsilon_2}\mathcal{C}_2''$ where $\Upsilon_1^\kappa\subseteq\Upsilon_2$, and (i') $\mathcal{C}_2^{x_s}\mapsto^*\mathcal{C}_2''$ for all $x_\alpha\in\mathbf{Out}(\Delta\Vdash K)$ such that $x_\alpha\in\Upsilon_1^\kappa$ and (ii) $\mathcal{C}_2^{x_r}\mapsto^*\mathcal{C}_2''$ for all $x_\alpha\in\mathbf{In}(\Delta\Vdash K)$ such that $x_\alpha\in\Theta_1^\kappa$.

More precisely, by the forest structure, we have $\mathcal{C}_2''=\{\mathcal{C}_2^{\kappa''}\}_{j\in I-\{\kappa\}}\mathcal{C}_2^{\kappa''}$, with $\mathcal{C}_2^\kappa\mapsto^{*\Upsilon_2^\kappa}\mathcal{C}_2^{\kappa''}$ with $\Upsilon_1^\kappa\subseteq\Upsilon_2^\kappa$. We use Υ_2^κ and $\mathcal{C}_2^{\kappa''}$ to instantiate the existential quantifier in the goal. We need to prove that

$$\begin{array}{l}\forall\,x_\alpha\in\mathbf{Out}(\Delta^\kappa\Vdash K^\kappa).\,\mathbf{if}\,x_\alpha\in\Upsilon_1^\kappa.\,\mathbf{then}\,(\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''})\in\mathcal{V}[\![\Delta^\kappa\Vdash K^\kappa]\!]^{m'+1}_{\cdot;x_\alpha}\text{ and}\\ \forall\,x_\alpha\in\mathbf{In}(\Delta^\kappa\Vdash K^\kappa).\mathbf{if}\,x_\alpha\in\Theta_1^\kappa.\,\mathbf{then}\,(\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''})\in\mathcal{V}[\![\Delta^\kappa\Vdash K^\kappa]\!]^{m'+1}_{x_\alpha;\cdot}.\end{array}$$

By the forward closure lemma for the bisimulation (Lem. 25), we get

A'. By item $[A.]$ we get, for all $x_\alpha\in\mathbf{Out}(\Delta\Vdash K)$ such that $x_\alpha\in\Upsilon_1^\kappa$, we have $\mathcal{C}_2'\mapsto^*\mathcal{C}_2''$ such that $\mathcal{C}_2''$ sends along the channel x_α and we have $\mathcal{C}_1'=\mathcal{C}_1^1\mathbf{msg}(x_\alpha.q)\mathcal{C}_1^2$ and $\mathcal{C}_2''=\mathcal{C}^{x_s'}\mathbf{msg}(x_\alpha.q)\mathcal{C}^{x_s''}$ and $\mathcal{C}_1^1\mathcal{C}_1^2\approx_a\mathcal{C}^{x_s'}\mathcal{C}^{x_s''}$.

B'. By item $B.$, the confluence lemma, and the forward closure lemma for the bisimulation (Lem. 25), we get: for all $x_\alpha\in\mathbf{In}(\Delta\Vdash\cdot)$ such that $x_\alpha\in\Theta_1^\kappa$, we have $\mathcal{C}_2'\mapsto^*\mathcal{C}_2''$ such that $\mathbf{msg}(x_\alpha.q)\mathcal{C}_1'\approx_a\mathbf{msg}(x_\alpha.q)\mathcal{C}_2''$.

C'. By item $C.$, the confluence lemma, and the forward closure lemma for the bisimulation (Lem. 25), we get: for all $x_\alpha\in\mathbf{In}(\cdot\Vdash K)$ such that $x_\alpha\in\Theta_1^\kappa$, we have $\mathcal{C}_2'\mapsto^*\mathcal{C}_2''$ such that $\mathcal{C}_1'\mathbf{msg}(x_\alpha.q)\approx_a\mathcal{C}_2''\mathbf{msg}(x_\alpha.q)$.

There are two parts we need to prove:

Part 1. $\forall\,x_\alpha\in\mathbf{Out}(\Delta^\kappa\Vdash K^\kappa).\,\mathbf{if}\,x_\alpha\in\Upsilon_1^\kappa.\,\mathbf{then}\,(\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''})\in\mathcal{V}[\![\Delta^\kappa\Vdash K^\kappa]\!]^{m'+1}_{\cdot;x_\alpha}$

Assume an arbitrary $x_\alpha\in\mathbf{Out}(\Delta^\kappa\Vdash K^\kappa)$ with $x_\alpha\in\Upsilon_1$. we consider cases based on the type of x_α. Here we provide the detailed proof for one case, the proof of other cases is similar.

Subcase 1. $K^\kappa=x_\alpha{:}A\otimes B$. We need to prove $(\mathcal{C}_1^{\kappa'},\mathcal{C}_2^{\kappa''})\in\mathcal{V}[\![\Delta^\kappa\Vdash x_\alpha{:}A\otimes B]\!]^{m'+1}_{\cdot;x_\alpha}$. We first establish $\mathcal{C}_1^{\kappa'}=\mathcal{C}_1^{\kappa''}\mathcal{A}_1\mathbf{msg}(\mathbf{send}y_\beta\,x_\alpha)$ and $\mathcal{C}_2^{\kappa''}=\mathcal{C}_2^{\kappa'''}\mathcal{A}_2\mathbf{msg}(\mathbf{send}y_\beta\,x_\alpha)$, using item [A'] above and well-typedness of programs. Next, we apply the inductive hypothesis to show $(\mathcal{A}_1;\mathcal{A}_2)\in\mathcal{E}[\![\Delta_1^\kappa\Vdash y_\beta{:}A]\!]^{m'}$ and also $(\mathcal{C}_1^{\kappa''};\mathcal{C}_2^{\kappa'''})\in\mathcal{E}[\![\Delta_2^\kappa\Vdash x_{\alpha+1}{:}B]\!]^{m'}$ with $\Delta^\kappa=\Delta_1^\kappa,\Delta_2^\kappa$. We get this by configuration typing, $\mathcal{C}_1^1\mathcal{C}_1^2\approx_a\mathcal{C}^{x_s'}\mathcal{C}^{x_s''}$ given in item $[A']$ above and the fact that both $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{C}_1^{\kappa''}$ and $\mathcal{C}_2^{\kappa'''}$ are separate trees in the forests $\mathcal{C}_1^1\mathcal{C}_1^2$ and $\mathcal{C}^{x_s'}\mathcal{C}^{x_s''}$.

Part 2. $\forall\,x_\alpha\in\mathbf{In}(\Delta^\kappa\Vdash K^\kappa).\mathbf{if}\,x_\alpha\in\Theta_1^\kappa.\,\mathbf{then}\,(\mathcal{C}_1^{\kappa'};\mathcal{C}_2^{\kappa''})\in\mathcal{V}[\![\Delta^\kappa\Vdash K^\kappa]\!]^{m'+1}_{x_\alpha;\cdot}$. Assume an arbitrary $x_\alpha\in\mathbf{In}(\Delta^\kappa\Vdash K^\kappa)$ with $x_\alpha\in\Theta_1$. we consider cases based on the type of x_α. Here we provide the detailed proof for one case, the proof of other cases is similar.

Subcase 1. $\Delta^\kappa=\Delta_1^\kappa x_\alpha{:}A\otimes B$. We need to prove $(\mathcal{C}_1^{\kappa'},\mathcal{C}_2^{\kappa''})\in\mathcal{V}[\![\Delta_1^\kappa,x_\alpha{:}A\otimes B\Vdash K^\kappa]\!]^{m'+1}_{x_\alpha;\cdot}$. We apply the inductive hypothesis on $\mathbf{msg}(\mathbf{send}\,y_\beta\,x_\alpha)\mathcal{C}_1'\approx_a\mathbf{msg}(\mathbf{send}y_\beta\,x_\alpha)\mathcal{C}_2''$ which is given by item [B'] above to get $(\mathbf{msg}(\mathbf{send}y_\beta\,x_\alpha)\mathcal{C}_1^{\kappa'},\mathbf{msg}(\mathbf{send}y_\beta\,x_\alpha)\mathcal{C}_2^{\kappa''})\in\mathcal{E}[\![\Delta_1^\kappa,y_\beta{:}A,x_{\alpha+1}{:}B\Vdash K^\kappa]\!]^{m'}$ which completes the proof of this case.

□

Lemma 25 (Forward closure)**.** *For all $\mathcal{D}_1,\mathcal{D}_2$ if $\mathcal{D}_1\approx_a\mathcal{D}_2$ and $\mathcal{D}_2\Rightarrow\mathcal{D}_2'$, then $\mathcal{D}_1\approx_a\mathcal{D}_2'$.*

Proof. The proof is by coinduction on the generative function of the bisimulation ($\mathcal{D}_1 \approx_a \mathcal{D}'_2$). We consider four cases required to establish $\mathcal{D}_1 \approx_a \mathcal{D}'_2$.

1) ($\tau - \texttt{step}$) if $\mathcal{D}_1 \xrightarrow{\tau} \mathcal{D}'_1$ then by the assumption $\mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}''_2$ and $\mathcal{D}'_1 \approx_a \mathcal{D}''_2$. This gives us $\mathcal{D}_2 \mapsto^* \mathcal{D}''_2$. We also know by assumption that $\mathcal{D}_2 \mapsto^* \mathcal{D}'_2$. By the confluence lemma(Lem. 11), we can build a $\mathcal{D}$ such that $\mathcal{D}''_2 \mapsto^* \mathcal{D}$ and $\mathcal{D}'_2 \mapsto^* \mathcal{D}$. By coinduction on $\mathcal{D}'_1 \approx_a \mathcal{D}''_2$ having the assumption $\mathcal{D}''_2 \mapsto^* \mathcal{D}$, we get $\mathcal{D}'_1 \approx_a \mathcal{D}$, which along with $\mathcal{D}'_2 \xRightarrow{\tau} \mathcal{D}$ is enough to get the result.
2) (output) if $\mathcal{D}_1 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}'_1$ then by the assumption $\mathcal{D}_2 \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}^4_2$ and $\mathcal{D}'_1 \approx_a \mathcal{D}^4_2$. This gives us $\mathcal{D}_2 \mapsto^* \mathcal{D}''_2$, and $\mathcal{D}''_2 \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}'''_2$, and $\mathcal{D}'''_2 \mapsto^* \mathcal{D}^4_2$. We also know by assumption that $\mathcal{D}_2 \mapsto^* \mathcal{D}'_2$. By the confluence lemma(Lem. 11), we can build a $\mathcal{D}$ such that $\mathcal{D}''_2 \mapsto^* \mathcal{D}$ and $\mathcal{D}'_2 \mapsto^* \mathcal{D}$. Moreover, for some $\mathcal{D}'$ we have $\mathcal{D} \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}'$, such that $\mathcal{D}'''_2 \mapsto^* \mathcal{D}'$. Recall that we also have $\mathcal{D}'''_2 \mapsto^* \mathcal{D}^4_2$. Again, we apply the confluence lemma(Lem. 11) to get a $\mathcal{D}''$ such that $\mathcal{D}^4_2 \mapsto^* \mathcal{D}''$ and $\mathcal{D}' \mapsto^* \mathcal{D}''$. By coinduction on $\mathcal{D}'_1 \approx_a \mathcal{D}^4_2$ having the assumption $\mathcal{D}^4_2 \mapsto^* \mathcal{D}''$, we get $\mathcal{D}'_1 \approx_a \mathcal{D}''$. Moreover, we have $\mathcal{D}'_2 \xRightarrow{\overline{x_\alpha}\, q} \mathcal{D}''$, i.e., $\mathcal{D}'_2 \mapsto^* \mathcal{D}$, and $\mathcal{D} \xrightarrow{\overline{x_\alpha}\, q} \mathcal{D}'$, and $\mathcal{D}' \mapsto^* \mathcal{D}''$. This is enough to get the result.
3) ($\mathsf{left\,input}$) Consider an arbitrary q which is not free in $\mathcal{D}_1$, and assume $\mathcal{D}_1 \xrightarrow{\mathbf{L}\, x_\alpha\, q} \mathcal{D}'_1$, then $\exists \mathcal{D}''_2 . \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}''_2$ and $\mathcal{D}'_1 \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}''_2$,. This gives us $\mathcal{D}_2 \mapsto^* \mathcal{D}''_2$. We also know by assumption that $\mathcal{D}_2 \mapsto^* \mathcal{D}'_2$. By the confluence lemma(Lem. 11), we can build a $\mathcal{D}$ such that $\mathcal{D}''_2 \mapsto^* \mathcal{D}$ and $\mathcal{D}'_2 \mapsto^* \mathcal{D}$. This also gives us $\mathbf{msg}(x_\alpha.q)\mathcal{D}''_2 \mapsto^* \mathbf{msg}(x_\alpha.q)\mathcal{D}$. By coinduction on $\mathcal{D}'_1 \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}''_2$ having the assumption $\mathbf{msg}(x_\alpha.q)\mathcal{D}''_2 \mapsto^* \mathbf{msg}(x_\alpha.q)\mathcal{D}$, we get $\mathcal{D}'_1 \approx_a \mathbf{msg}(x_\alpha.q)\mathcal{D}$. Moreover, we have $\mathcal{D}'_2 \Rightarrow \mathcal{D}$, i.e., $\mathcal{D}'_2 \mapsto^* \mathcal{D}$. This is enough to complete the proof of this case.
4) ($\mathsf{right\,input}$) for all $q \notin \mathsf{fn}(\mathcal{D}_1)$, if $\mathcal{D}_1 \xrightarrow{\mathbf{R}\, x_\alpha\, q} \mathcal{D}'_1$ then $\exists \mathcal{D}'_2 . \mathcal{D}_2 \xRightarrow{\tau} \mathcal{D}'_2$ and $\mathcal{D}'_1 \approx_a \mathcal{D}'_2\mathbf{msg}(x_\alpha.q)$. The proof is similar to the previous case.

□

XIII. Biorthogonality

A. Equivalence

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

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

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

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

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

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

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

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

Theorem 6 ($\top\top$-closure). *Consider $(\mathcal{D}_1, \mathcal{D}_2) \in \mathsf{Tree}(\Delta \Vdash K)$, we have*

$$(\Delta \Vdash \mathcal{D}_1 :: K) \equiv (\Delta \Vdash \mathcal{D}_2 :: K)$$

$$\textbf{iff}$$

$$\forall \mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2.\, \text{s.t.}\, (\Delta \dashv \mathcal{C}_1[\ ]\mathcal{F}_1 \dashv K) \equiv (\Delta \dashv \mathcal{C}_2[\ ]\mathcal{F}_2 \dashv K) \text{ we have}$$
$$\forall m.\, \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \approx_a \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2$$

Proof. There are two directions to consider:

Left to Right: The result is straightforward by the compositionality result(Corollary 2) and the soundness result (Lem. 24) and the definition of equivalence $\equiv$ for programs and environments (Def. 30).

Right to Left: Consider $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{F}_1 = \mathcal{F}_2 = \cdot$. By Lem. 21 and Def. 30, we know that $(\Delta \dashv \cdot [\ ] \cdot \dashv K) \equiv (\Delta \dashv \cdot [\ ] \cdot \dashv K)$. From this we get $\forall m.\, \mathcal{D}_1 \approx_a \mathcal{D}_2$, and by the completeness lemma (Lem. 24) we get $\forall m.\, (\mathcal{D}_1, \mathcal{D}_2) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$ and $\forall m.\, (\mathcal{D}_2, \mathcal{D}_1) \in \mathcal{E}[\![\Delta \Vdash K]\!]^m$. By Def. 30, we have $\Delta \Vdash \mathcal{D}_1 :: K \equiv \Delta \Vdash \mathcal{D}_2 :: K$, which completes the proof.

This result is enough to prove that our equivalence relation is $\top\top$-closed, i.e., following the results presented in the results presented in [1], it shows $r = s^{\top}$, 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 $\mathcal{C}[\ ]\mathcal{F}$, such that for some Δ' and K', we have $\Delta' \Vdash \mathcal{C} :: \Delta$ and $K \Vdash \mathcal{F} :: K'$.*

Definition 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 $\Delta \Vdash \mathcal{D} :: K$. Given a security level $\xi \in \Psi_0$, and two interfaces $\Gamma_1 \Vdash x_\alpha{:}A_1[c_1]$ and $\Gamma_2 \Vdash y_\beta{:}A_2[c_2]$ we write $PRel^{\xi}(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$ for the set of all security program relations that relate programs $|\Gamma_1| \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1$ and $|\Gamma_2| \Vdash \mathcal{D}_2 :: y_\beta{:}A_2$.*

A **low-security** *session environment-relation is a binary relation between low security session-typed environments. Given two interfaces $\Gamma_1 \Vdash x_\alpha{:}A_1[c_1]$ and $\Gamma_2 \Vdash y_\beta{:}A_2[c_2]$, we write $ERel^{\xi}(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$ for the set of all environment relations that relate low security session-typed environments $\mathcal{C}_1[\ ]\mathcal{F}_1$ with the interface $|\Gamma_1 \Downarrow \xi| \Vdash |x_\alpha{:}A_1[c_1] \Downarrow \xi|$ and $\mathcal{C}_2[\ ]\mathcal{F}_2$ with the interface $|\Gamma_2 \Downarrow \xi| \Vdash |y_\beta{:}A_2[c_2] \Downarrow \xi|$.*

Definition 33 (The $(\_)^\top$ operation on relations)**.** *Given two interfaces $\Gamma_1 \Vdash x_\alpha{:}A_1[c_1]$ and $\Gamma_2 \Vdash y_\beta{:}A_2[c_2]$ and the observer security level $\xi \in \Psi_0$ and $r \in PRel^\xi(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$, we define $r^\top \in ERel^\xi(\Gamma_1 \Vdash x_\alpha{:}A_1[c_1], \Gamma_2 \Vdash y_\beta{:}A_2[c_2])$ by*

$$(\mathcal{C}_1[\;]\mathcal{F}_1, \mathcal{C}_2[\;]\mathcal{F}_2) \in r^\top \text{ iff } \forall (\mathcal{D}_1, \mathcal{D}_2) \in r.$$
$$\forall \mathcal{B}_1 \in \text{H-Provider}^\xi(\Gamma_1), \mathcal{B}_2 \in \text{H-Provider}^\xi(\Gamma_2), \mathcal{T}_1 \in \text{H-CLient}^\xi(x_\alpha{:}A_1[c_1]),\ \mathcal{T}_2 \in \text{H-Client}^\xi(y_\beta{:}A_2[c_2]).$$
$$\mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1 \approx_a \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2$$

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

$$(\mathcal{D}_1, \mathcal{D}_2) \in s^\top \text{ iff } \forall (\mathcal{C}_1[\;]\mathcal{F}_1, \mathcal{C}_2[\;]\mathcal{F}_2) \in s.$$
$$\forall \mathcal{B}_1 \in \text{H-Provider}^\xi(\Gamma_1), \mathcal{B}_2 \in \text{H-Provider}^\xi(\Gamma_2), \mathcal{T}_1 \in \text{H-CLient}^\xi(x_\alpha{:}A_1[c_1]),\ \mathcal{T}_2 \in \text{H-Client}^\xi(y_\beta{:}A_2[c_2]).$$
$$\mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1 \approx_a \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2$$

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 34 (Equivalence of forests by the logical relation upto the observer level)**.** *We write $\Gamma_1' \Vdash \mathcal{C}_1 :: \Gamma_1 \equiv^\xi_{\Psi_0} \Gamma_2' \Vdash \mathcal{C}_2 :: \Gamma_2$ iff*

(i) **(Both forests are well-typed.)** $\mathcal{C}_1 \in \text{Forest}(|\Gamma_1'| \Vdash |\Gamma_1|)$ *and* $\mathcal{C}_2 \in \text{Forest}(|\Gamma_2'| \Vdash |\Gamma_2|)$

(ii) **(Their observable interface is the same.)** $\Gamma_1' \Downarrow \xi = \Gamma_2' \Downarrow \xi$ *and* $\Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi$*, and*

(iii) **(Each tree in the 1st forest with an observable interface has a counterpart in the 2nd forest which is equivalent to it.)** *for every $\mathcal{T}_1 \in \mathcal{C}_1$ such that $\mathcal{T}_1 \in \text{Tree}(|\Gamma_1''| \Vdash |x_\alpha{:}A_1[c_1]|)$, and $\Gamma_1'' \subseteq \Gamma_1'$ and $x_\alpha{:}A_1[c_1] \in \Gamma_1$ and $(\Gamma_1'', x_\alpha{:}A_1[c_1]) \Downarrow \xi \neq \cdot$ there exists a $\mathcal{T}_2 \in \mathcal{C}_2$ such that $\mathcal{T}_2 \in \text{Tree}(|\Gamma_2''| \Vdash |y_\beta{:}A_2[c_2]|)$, and $\Gamma_2'' \subseteq \Gamma_2'$ and $y_\beta{:}A_2[c_2] \in \Gamma_2$ for $\Gamma_1'' \Downarrow \xi = \Gamma_2'' \Downarrow \xi$ and $y_\beta{:}A_2[c_2] \Downarrow \xi = x_\alpha{:}A_1[c_1] \Downarrow \xi$, and*

(iv) **(Each tree in the 2nd forest with an observable interface has a counterpart in the 1st forest which is equivalent to it.)** *vice versa.*

Definition 35. *Define the relation*

$$\Gamma_1 \dashv \mathcal{C}_1[\;]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1] \equiv^\xi_{\Psi_0} \Gamma_2 \dashv \mathcal{C}_2[\;]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2] \text{ as}$$

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

(ii) **(The configurations $\mathcal{C}_1$ and $\mathcal{C}_2$ offer channels annotated as observable and use the same channels and are equivalent for any observable (low-confidentiality) annotation of their resource channel.)** *for some Δ, we have $(\mathcal{C}_1, \mathcal{C}_2) \in \text{Forest}(\Delta \Vdash |\Gamma|)$. In particular, if $\Gamma = \cdot$, we have $\mathcal{C}_1 = \mathcal{C}_2 = \cdot$.*
Moreover, for every Γ_1' and Γ_2' such that $|\Gamma_1'| = |\Gamma_2'| = \Delta$, and $\forall w_\eta{:}A[c] \in \Gamma_1'.\ c \sqsubseteq \xi$ and $\forall w_\eta{:}A[c] \in \Gamma_2'.\ c \sqsubseteq \xi$ we have $\Gamma_1' \Vdash \mathcal{C}_1 :: \Gamma \equiv^\xi_{\Psi_0} \Gamma_2' \Vdash \mathcal{C}_2 :: \Gamma$.

(iii) **(The configurations $\mathcal{F}_1$ and $\mathcal{F}_2$ use channels annotated as observable, offer the same channels, and are equivalent for any observable (low-confidentiality) annotation of their offering channels.)** *If $K^s = \_{:}1[\top]$, then $\mathcal{F}_1 = \mathcal{F}_2 = \cdot$. Otherwise, for some $w_\eta{:}A$, we have $(\mathcal{F}_1, \mathcal{F}_2) \in \text{Tree}(K^s \Vdash w_\eta{:}A)$. Moreover, $\forall c, c' \sqsubseteq \xi$, we have $K^s \Vdash \mathcal{F}_1 :: w_\eta{:}A[c] \equiv^\xi_{\Psi_0} K^s \Vdash \mathcal{F}_2 :: w_\eta{:}A[c']$ and.*

◇

Lemma 26 (Compositionality of $\equiv^\xi_{\Psi_0}$)**.** *If*

(i) $\quad(\Gamma_1' \Vdash \mathcal{C}_1 :: \Gamma) \equiv^\xi_{\Psi_0} (\Gamma_2' \Vdash \mathcal{C}_2 :: \Gamma)$ with $\Gamma = \Gamma_1 \Downarrow_\xi = \Gamma_2 \Downarrow_\xi$ and

(ii) $\quad(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^\xi_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ and

(iii) $\quad(K^s \Vdash \mathcal{F}_1 :: K_1^s) \equiv^\xi_{\Psi_0} (K^s \Vdash \mathcal{F}_2 :: K_2^s)$ with $K^s = x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi$ then

$$(\Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: K_1^s) \equiv^\xi_{\Psi_0} (\Gamma_2', \Gamma_2^h \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: K_2^s),$$

where Γ_1^h is the set of all channels $w_\eta{:}C[d] \in \Gamma_1$ with $d \not\sqsubseteq \xi$ and Γ_2^h is the set of all channels $w_\eta{:}C[d] \in \Gamma_2$ with $d \not\sqsubseteq \xi$.

Proof. By the configuration typing and definition of equivalence (Def. 17) and (i), (ii), (iii), we know that $\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \in \text{Tree}(|\Gamma_1'| \Vdash |K_1^s|)$ and $\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 \in \text{Tree}(|\Gamma_2'| \Vdash |K_2^s|)$. Moreover, by definition of equivalence (Def. 17) and (i), (iii) we get $\Gamma' = (\Gamma_1', \Gamma_1^h) \Downarrow \xi = (\Gamma_2', \Gamma_1^h) \Downarrow \xi$ and $K^{s'} = K_1^s \Downarrow \xi = K_2^s \Downarrow \xi$.

Assume arbitrary configurations $\mathcal{B}_1 \in$ **H-Provider**$^\xi(\Gamma_1')$, $\mathcal{B}_1^h \in$ **H-Provider**$^\xi(\Gamma_1^h)$, $\mathcal{B}_2 \in$ **H-Provider**$^\xi(\Gamma_2')$, $\mathcal{B}_2^h \in$ **H-Provider**$^\xi(\Gamma_2^h)$, $\mathcal{T}_1 \in$ **H-Client**$^\xi(K_1^s)$, and $\mathcal{T}_2 \in$ **H-Client**$^\xi(K_2^s)$. Our goal is to prove

$$\forall m.\ (\mathcal{B}_1\mathcal{B}_1^h\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1, \mathcal{B}_2\mathcal{B}_2^h\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2) \in \mathcal{E}[\![|\Gamma'| \Vdash |K^{s'}|]\!]^m, \text{ and}$$
$$\forall m.\ (\mathcal{B}_2\mathcal{B}_2^h\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2, \mathcal{B}_1\mathcal{B}_1^h\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1) \in \mathcal{E}[\![|\Gamma'| \Vdash |K^{s'}|]\!]^m.$$

By assumptions (i)-(iii) we have:

(i') Note that for any trees $\mathcal{A}_1'' \in \mathcal{C}_1$ and $\mathcal{A}_2'' \in \mathcal{C}_2$ they have an observable offering channel occuring in Γ. Assume that $\mathcal{C}_1$ as a forest includes n separate trees. By the previous observation, we know that $\mathcal{C}_2$ also consists of n separate trees. For any $j \leq n$ consider the tree $\mathcal{A}_1^j \in \mathcal{C}_1$ consider the corresponding tree $\mathcal{A}_2^j \in \mathcal{C}_2$ that exists by Def. 34 and for which we have $(\dagger_j)$ $\Gamma_1^{j'} \Vdash \mathcal{A}_1^j :: w_\eta^j{:}A^j[c^j] \equiv^\xi_{\Psi_0} \Gamma_2^{j'} \Vdash \mathcal{A}_2^j :: w_\eta^j{:}A^j[c^j]$.

From $\dagger_j$, we get $\Gamma^{j'} = \Gamma_1^{j'} = \Gamma_2^{j'}$ and for arbitrary configurations $\mathcal{T}_1^j \in \textbf{H-Client}^\xi(\Gamma_1^{j'})$, and $\mathcal{T}_2^j \in \textbf{H-Client}^\xi(\Gamma_2^{j'})$, we have

$$\forall\, m.\, (\mathcal{B}_1^j\mathcal{C}_1, \mathcal{B}_2^j\mathcal{C}_2) \in \mathcal{E}[\![|\Gamma^{j'} \Downarrow \xi| \Vdash w_\eta^j{:}A^j]\!]^m, \text{ and}$$
$$\forall\, m.\, (\mathcal{B}_2^j\mathcal{C}_2, \mathcal{B}_1^j\mathcal{C}_1) \in \mathcal{E}[\![|\Gamma^{j'} \Downarrow \xi| \Vdash w_\eta^j{:}A^j]\!]^m.$$

Observe that $\Gamma_1' = \{\Gamma_1^{j'}\}_{j\leq n}$ and $\Gamma_2' = \{\Gamma_2^{j'}\}_{j\leq n}$, and $\Gamma = \{w_\eta^j{:}A^j[c^j]\}_{j\leq n}$.

(ii') By Def. 17, we know that for arbitrary configurations $\mathcal{B}_1^h \in \textbf{H-Provider}^\xi(\Gamma_1)$, and $\mathcal{B}_2^d \in \textbf{H-Provider}^\xi(\Gamma_2)$, and $\mathcal{T}_1^d \in \textbf{H-Client}^\xi(x_\alpha{:}A_1[c_1])$, and $\mathcal{T}_2^d \in \textbf{H-Client}^\xi(y_\beta{:}A_2[c_2])$.

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

(iii') By Def. 17, if $c_1 \sqsubseteq \xi$ and $c_2 \sqsubseteq \xi$, we have for any $\mathcal{T}_1 \in \textbf{H-Client}^\xi(K_1^s)$, and $\mathcal{T}_2 \in \textbf{H-Client}^\xi(K_2^s)$

$$\forall\, m.\, (\mathcal{F}_1\mathcal{T}_1^h, \mathcal{F}_2\mathcal{T}_2^h) \in \mathcal{E}[\![|K^s| \Vdash |K^{s'}|]\!]^m, \text{ and}$$
$$\forall\, m.\, (\mathcal{F}_2\mathcal{T}_2^h, \mathcal{F}_1\mathcal{T}_1^h) \in \mathcal{E}[\![|K^s| \Vdash |K^{s'}|]\!]^m,$$

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

Theorem 7 ($\top\top$-closure)**.** *Consider $\mathcal{D}_1 \in \mathsf{Tree}(|\Gamma_1| \Vdash |x_\alpha{:}A_1[c_1]|)$ and $\mathcal{D}_2 \in \mathsf{Tree}(|\Gamma_2| \Vdash |y_\beta{:}A_2[c_2]|)$ and a given observer level $\xi \in \Psi_0$. We have*

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

iff

$$\forall \mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2.\, \text{s.t.}\, (\Gamma_1 \dashv \mathcal{C}_1[\;]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1]) \equiv^\xi_{\Psi_0} (\Gamma_2 \dashv \mathcal{C}_2[\;]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2]) \text{ we have}$$
$$\forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^\xi(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^\xi(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^\xi(x_\alpha{:}A_1[c_1]),\, \mathcal{T}_2 \in \mathsf{H\text{-}Client}^\xi(y_\beta{:}A_2[c_2]).$$
$$\mathcal{B}_1\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1\mathcal{T}_1 \;\approx_a\; \mathcal{B}_2\mathcal{C}_2\mathcal{D}_2\mathcal{F}_2\mathcal{T}_2$$

Proof. There are two directions to consider:

Left to Right: Consider $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^\xi_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$ and arbitrary $\mathcal{C}_1, \mathcal{C}_2, \mathcal{F}_1, \mathcal{F}_2$ such that we have

$$(\Gamma_1 \dashv \mathcal{C}_1[\;]\mathcal{F}_1 \dashv x_\alpha{:}A_1[c_1]) \equiv^\xi_{\Psi_0} (\Gamma_2 \dashv \mathcal{C}_2[\;]\mathcal{F}_2 \dashv y_\beta{:}A_2[c_2]).$$

By Def. 35, we get that $\Gamma = \Gamma_1 \Downarrow \xi = \Gamma_2 \Downarrow \xi$ and $K^s = x_\alpha{:}A_1[c_1] \Downarrow \xi = y_\beta{:}A_2[c_2] \Downarrow \xi$. Moreover for some Δ, we have $(\mathcal{C}_1, \mathcal{C}_2) \in \mathsf{Forest}(\Delta \Vdash |\Gamma|)$ and for any low secrecy annotations of Δ as Γ_1' and Γ_2' we get $\Gamma_1' \Vdash \mathcal{C}_1 :: \Gamma \equiv^\xi_{\Psi_0} \Gamma_2' \Vdash \mathcal{C}_2 :: \Gamma$. Also if $K^s = \_{:}1[\top]$, we have $\mathcal{F}_1 = \mathcal{F}_2 = \cdot$, and otherwise $(\mathcal{F}_1, \mathcal{F}_2) \in \mathsf{Tree}(K^s \Vdash w_\eta{:}A)$ for some $w_\eta{:}A$, for any low secrecy annotations (with $c_3, c_4 \sqsubseteq \xi$) of $w_\eta{:}A$ as $w_\eta{:}A[c_3]$ and $w_\eta{:}A[c_4]$ we get $K^s \Vdash \mathcal{C}_1 :: w_\eta{:}A[c_3] \equiv^\xi_{\Psi_0} K^s \Vdash \mathcal{C}_2 :: w_\eta{:}A[c_4]$.
Our goal is to prove:

$$\forall \mathcal{B}_1 \in \mathsf{H\text{-}Provider}^\xi(\Gamma_1), \mathcal{B}_2 \in \mathsf{H\text{-}Provider}^\xi(\Gamma_2), \mathcal{T}_1 \in \mathsf{H\text{-}CLient}^\xi(x_\alpha{:}A_1[c_1]),\, \mathcal{T}_2 \in \mathsf{H\text{-}Client}^\xi(y_\beta{:}A_2[c_2]).$$
$$\mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 \;\approx_a\; \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2.$$

By compositionality of equivalence upto relation (Lem. 26), we get $\Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: x_\alpha{:}A_1[c_1] \equiv^\xi_{\Psi_0} \Gamma_2', \Gamma_2^h \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: y_\beta{:}A_2[c_2]$ in the case where $K^s := \_{:}1[\top]$, i.e., $c_1, c_2 \not\sqsubseteq \xi$, and otherwise we get $\Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: w_\eta{:}A[c_3] \equiv^\xi_{\Psi_0} \Gamma_2', \Gamma_2^h \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: w_\eta{:}A[c_4]$ where Γ_1^h is the set of all channels $z_\gamma{:}C[d] \in \Gamma_1$ with $d \not\sqsubseteq \xi$ and Γ_2^h is the set of all channels $z_\gamma{:}C[d] \in \Gamma_2$ with $d \not\sqsubseteq \xi$.
By the soundness theorem for equivalence relation (Corollary 4), we get $\Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: x_\alpha{:}A_1[c_1] \approx^\xi_a \Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: y_\beta{:}A_2[c_2]$ in the first case where $K^s = \_{:}1[\top]$ and otherwise $\Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_1\mathcal{D}_1\mathcal{F}_1 :: w_\eta{:}A[c_3] \approx^\xi_a \Gamma_1', \Gamma_1^h \Vdash \mathcal{C}_2\mathcal{D}_2\mathcal{F}_2 :: w_\eta{:}A[c_4]$. In the first case, the proof is complete by Def. 26 and the observation that all channels in Γ_1' and Γ_2' are annotated with channels of secrecy $d' \sqsubseteq \xi$. In the second case, the proof is complete by Def. 26 and observing that all channels in Γ_1' and Γ_2' are annotated with channels of secrecy $d' \sqsubseteq \xi$ and that $c_1, c_2 \sqsubseteq \xi$ and $c_3, c_4 \sqsubseteq \xi$.

Right to Left: Assume $\mathcal{C}_1 = \mathcal{C}_2 = \mathcal{F}_1 = \mathcal{F}_2 = \cdot$. By Lem. 21 and Def. 30, we know that $(\Gamma_1 \dashv \cdot [\,] \cdot \dashv x_\alpha{:}A_1[c_1]) \equiv (\Gamma_2 \dashv \cdot [\,] \cdot \dashv y_\beta{:}A_1[c_2])$. From this we get $\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1] \approx^\xi_a \Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2]$. Now, we can apply the completeness lemma (Corollary 4) to get $(\Gamma_1 \Vdash \mathcal{D}_1 :: x_\alpha{:}A_1[c_1]) \equiv^\xi_{\Psi_0} (\Gamma_2 \Vdash \mathcal{D}_2 :: y_\beta{:}A_2[c_2])$.

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

$\square$

References

[1] Andrew M. Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in Computer Science*, 10(3):321–359, 2000.