Types and Subtypes for Correct Communication in Client-Server Systems

Simon Gay and Malcolm Hole

UNIVERSITY
of
GLASGOW

Department of Computing Science
University of Glasgow
Glasgow G12 8QQ
Scotland

TR-2003-131
February 2003

Types and Subtypes for Correct Communication in Client-Server Systems

Simon Gay[1] and Malcolm Hole[2]

[1] Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK.
Email: <simon@dcs.gla.ac.uk>
[2] Department of Computer Science, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK. Email: <M.Hole@cs.rhul.ac.uk>

February 12, 2003

Abstract

We review an extension of the π-calculus with a static type system which supports high-level specifications of extended patterns of communication, such as client-server protocols. We then present a subtype relation that allows protocol specifications to be extended in order to describe richer behaviour; an implemented server can then be replaced by a refined implementation, without invalidating type-correctness of the overall system. We show how this subtyping relation can be integrated into the type system, and use the POP3 protocol as a concrete example of this technique. This report is a revised and extended version of our paper *Types and Subtypes for Client-Server Interaction*, presented at the ESOP'99 conference.

This research was funded by the EPSRC project "Novel Type Systems for Concurrent Programming Languages" (GR/L75177,GR/N39494).

Contents

1 Introduction

The π-calculus [15, 24] has been used as a vehicle for much work on types for concurrent programming languages [2, 10, 12, 13, 14, 18, 19, 26].

A system of *session types* for use in concurrent programming languages was first proposed by Honda et al. [6, 8, 25]. Such a type defines a sequence of message types that can be exchanged over a *session channel*. These messages can include both offering and making choices between certain possibilities and the types therefore have a branching structure. When implementing a client-server system it is useful to have such a type system as it enforces some structure on the ordering of messages, rather than just checking the types of individual messages that are sent on channels.

We have shown [4] how session types can be incorporated into the π-calculus with a minimum of additional syntax, the majority of the special treatment of session channels being handled by the typing rules. We have also advocated using a session type as part of the published specification of a server's protocol, so that static type-checking can be used to verify client implementations. Furthermore, we have proposed [3] a notion of subtyping on session types, which allows a protocol specification to be refined while maintaining compatibility with earlier users of the protocol. This report is a revised and extended version of our ESOP'99 paper [3], presenting that work in the same framework as our earlier report [4].

The remainder of this introduction presents an example of the use of session types, and subtyping on session types, in the π-calculus.

Consider a server for mathematical operations which initially offers the choice between addition and the sine function. The session type, S, of the server side of the channel is

$$S = \&\langle \mathsf{plus}\colon ?[\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, \mathsf{end}, \mathsf{sin}\colon ?[\mathsf{int}] \,.\, ![\mathsf{real}] \,.\, \mathsf{end} \rangle$$

The $\&\langle \ldots \rangle$ constructor specifies that a choice is offered between, in this case, two options, labelled plus and sin. Each label leads to a type describing the subsequent pattern of communication. Note that the pattern in the two branches are different. ? and ! indicate the receiving and sending of a name respectively. The . character is the sequencing constructor and end indicates the end of the interaction. Such a server would publish the dual or complementary type,

$$\overline{S} = \oplus\langle \mathsf{plus}\colon ![\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end}, \mathsf{sin}\colon ![\mathsf{int}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end} \rangle$$

as part of its specification. The $\oplus\langle \ldots \rangle$ constructor specifies that a choice is made. Again, each label is followed by a type which describes the subsequent interaction. Note that the pattern of sending and receiving is the opposite of the pattern which appears in the type of the server.

We would expect that a server that followed the pattern of communications in S and a client that followed the pattern of communications in $\overline{S}$ would interact correctly. Indeed, the typing rules we have presented [4] will allow the derivation of

$$x^+ : S, x^- : \overline{S} \vdash \mathsf{server} \mid \mathsf{client}$$

where

$$\begin{aligned} \mathsf{server} \;=\; & x^+ \triangleright \{\mathsf{plus}\colon x^+ \,?\, [a : \mathsf{real}] \,.\, x^+ \,?\, [b : \mathsf{real}] \,.\, x^+ \,!\, [a + b] \,.\, \mathbf{0}, \\ & \quad \mathsf{sin}\colon x^+ \,?\, [a : \mathsf{int}] \,.\, x^+ \,!\, [sin(a)] \,.\, \mathbf{0}\} \end{aligned}$$

and

$$\mathsf{client} = x^- \triangleleft \mathsf{sin} \,.\, x^- \,!\, [45] \,.\, x^- \,?\, [a : \mathsf{real}] \,.\, \mathbf{0}$$

In server, the offer operation, $\triangleright$, allows a message received on the session port x^+ to be any of the listed alternatives, offering a choice as to how the process continues. The labels and the pattern of inputs and outputs match those in S. Conversely, in client, the choose operation, $\triangleleft$, selects from the available options. The pattern of inputs and outputs then match those in $\overline{S}$ appropriate to the choice made. Note that this client does nothing with the value received from the server. More realistically, $\mathbf{0}$ would be replaced by some continuation process which used the name a.

One problem that occurs in client-server applications is the issue of backward compatibility when a server is upgraded. We might, for example, decide to introduce a replacement maths server with more functionality and which can handle real number input to trigonometrical operations. We could define this server as follows.

$$\begin{aligned}\mathsf{newserver} \;=\; x^+ \triangleright \{&\mathsf{plus} : x^+ \,?\, [a : \mathsf{real}] \,.\, x^+ \,?\, [b : \mathsf{real}] \,.\, x^+ \,!\, [a+b] \,.\, \mathbf{0},\\ &\mathsf{minus} : x^+ \,?\, [a : \mathsf{real}] \,.\, x^+ \,?\, [b : \mathsf{real}] \,.\, x^+ \,!\, [a-b] \,.\, \mathbf{0},\\ &\mathsf{sin} : x^+ \,?\, [a : \mathsf{real}] \,.\, x^+ \,!\, [sin(a)] \,.\, \mathbf{0}\},\\ &\mathsf{cos} : x^+ \,?\, [a : \mathsf{real}] \,.\, x^+ \,!\, [cos(a)] \,.\, \mathbf{0}\},\end{aligned}$$

This type of the server side of the channel is now

$$\begin{aligned}S' = \&\langle &\mathsf{plus} : ?[\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, \mathsf{end}, \mathsf{minus} : ?[\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, \mathsf{end},\\ &\mathsf{sin} : ?[\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, \mathsf{end}, \mathsf{cos} : ?[\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, \mathsf{end}\rangle\end{aligned}$$

This server would then publish

$$\begin{aligned}\overline{S'} = \oplus\langle &\mathsf{plus} : ![\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end}, \mathsf{minus} : ![\mathsf{real}] \,.\, ![\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end},\\ &\mathsf{sin} : ![\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end}, \mathsf{cos} : ![\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end}\rangle\end{aligned}$$

as the type of the client side of the channel.

We would like either

$$x^+ : S, x^- : \overline{S} \vdash \mathsf{newserver} \mid \mathsf{client}$$

or

$$x^+ : S', x^- : \overline{S'} \vdash \mathsf{newserver} \mid \mathsf{client}$$

to be correct. That is, we would like our old client to be able to have an interaction with the new server, whether it is on a channel with the old type, S, or the new type, S'. It seems reasonable that the interaction can happen as client uses one of the labels offered by the server, and the integer sent by the client can be used sensibly where it is expecting a real number. If we try to use a channel of type S, the client can make the choice as it is one of the available labels on the channel. The new server cannot offer the choice, however, as it is offering more labels than the type of the channel specifies, and typechecking therefore fails. i.e. the typing judgement

$$x^+ : S \vdash \mathsf{newserver}$$

is incorrect. If we try to use a channel of type S', the client can make the choice as, again, the chosen label is one of those available. This time, the new server can also offer the choice as the labels it is offering match those in the channel type. When the client tries to send

the integer 45, however, this does not match the type real specified in the channel type and again typechecking fails. This time

$$x^- : \overline{S'} \vdash \mathsf{client}$$

is incorrect, so the interaction is not allowed by our session type system. What is needed is a structured way to upgrade the functionality of a server, whilst maintaining type correctness for previous implementations of clients.

We present a type system that allows subtyping on session channels such that, for the above definitions, S is a subtype of S' and $\overline{S'}$ is a subtype of $\overline{S}$. We write this

$$S \leqslant S' \qquad \text{and} \qquad \overline{S'} \leqslant \overline{S}$$

When a process such as client immediately makes a choice, it is safe to use a channel with a type that contains a superset of the labels that the process is choosing between. In this situation, we can be sure that the chosen label will be one of the components of the type. A subtype of a session type that begins with a choice being made will, therefore, contain a greater number of labels. Conversely, when a process immediately offers a choice, the type of the channel used must contain a subset of the labels offered by the process. This ensures that whatever label is received on the channel by the process, it is one of those being offered. A subtype of a session type that begins with a choice being offered will, therefore, contain fewer labels. In either case, the type associated with each label in the subtype must be a subtype of the type associated with the corresponding label in the original type. A channel that is a subtype of the expected type can always be used in its place. This means that both of the above typing derivations are then correct: in the first derivation, newserver uses a channel of type S, which is a subtype of the type S' that it is expecting; in the second client uses a channel of type $\overline{S'}$, which is a subtype of the type $\overline{S}$ that it is expecting.

As with our type system without subtyping, each communication capability in a session type must be used exactly once, and to make sure of this we use techniques similar to those Kobayashi et al. [12, 13] use to enforce linearity [5]. The type system also allows non-session types to be specified, and there are no restrictions on how many processes may use them. For example, $y : \widehat{\ }[\mathsf{int}]$ is a channel which can be used freely to send or receive integers. To allow subtyping on non-session types, we draw on the work of Pierce and Sangiorgi [18, 19].

Returning to our mathematical server, we can now see how the revised version could work when placed in parallel with a pair of clients, one adhering to the new protocol and the other still using a channel of type S. As the server is to be used with multiple clients, we must also make the server multi-threaded. This is done using the standard π-calculus replication operator, !. This simply replicates a process the necessary number of times to fulfill any possible communications. Note that the channel trigger, although used to carry session channels, is not a session channel itself. It can therefore be used by the server and

both clients.

$$
\begin{array}{rcl}
\mathsf{multithreadedserver} & = & !\mathsf{thread} \\
\mathsf{thread} & = & \mathsf{trigger}\,?\,[x^+ : S'].\,\mathsf{newserver} \\
\mathsf{newserver} & = & x^+ \triangleright \{\mathsf{plus}\!:\!x^+\,?\,[a:\mathsf{real}].\,x^+\,?\,[b:\mathsf{real}].\,x^+\,!\,[a+b].\,\mathbf{0}, \\
 & & \quad \mathsf{minus}\!:\!x^+\,?\,[a:\mathsf{real}].\,x^+\,?\,[b:\mathsf{real}].\,x^+\,!\,[a-b].\,\mathbf{0}, \\
 & & \quad \mathsf{sin}\!:\!x^+\,?\,[a:\mathsf{real}].\,x^+\,!\,[sin(a)].\,\mathbf{0}\}, \\
 & & \quad \mathsf{cos}\!:\!x^+\,?\,[a:\mathsf{real}].\,x^+\,!\,[cos(a)].\,\mathbf{0}\}, \\
\mathsf{newclient} & = & (\nu y^{\pm} : S')\mathsf{trigger}\,!\,[y^+].\,\mathsf{newclientbody} \\
\mathsf{newclientbody} & = & y^- \triangleleft \mathsf{sin}.\,y^-\,!\,[90.0].\,y^-\,?\,[a:\mathsf{real}].\,\mathbf{0} \\
\mathsf{oldclient} & = & (\nu z^{\pm} : S)\mathsf{trigger}\,!\,[z^+].\,\mathsf{oldclientbody} \\
\mathsf{oldclientbody} & = & z^- \triangleleft \mathsf{sin}.\,z^-\,!\,[45].\,z^-\,?\,[b:\mathsf{real}].\,\mathbf{0}
\end{array}
$$

When the system is executed, newclient uses the ν construct to create a session channel of type S'. This has two ports: the server side, y^+, with type S', and the client side, y^-, of type $\overline{S'}$. It sends the y^+ port of the local channel y to one copy of the process thread and retains the y^- port for its own use. The type of y^+ is the same as the type which thread is expecting to receive and the standard π-calculus scope extrusion allows the scope of y^+ to expand to include that copy of thread. The processes newclient and newserver, therefore, both continue by having a private interaction on the channel y. The process oldclient creates a different session channel, z, of type S. Again this has two ports, and it sends the z^+ port to a separate copy of the process thread and retains the z^- port. Although the type of z^+, S, is not that which is expected by thread, it is a subtype of the expected type. It can, therefore, be used by newserver, and so oldclient and newserver both continue by having a private interaction on z. So,

$$\mathsf{trigger} : \widehat{\ }[S] \vdash \mathsf{newserver} \mid \mathsf{newclient} \mid \mathsf{oldclient}$$

is a valid typing judgement.

The remainder of this paper is organised as follows. Section 2 defines the syntax of processes, types and environments, and some basic operations on them. The typing and subtyping rules are presented in Section 3. Section 4 defines the operational semantics of the language and states the main technical results leading to type soundness. Section 5 uses our type system to specify a basic version of the POP3 protocol, and shows how subtyping can be used to add functionality whilst maintaining compatibility with the previous version. Finally, we discuss related work and outline our future plans in Section 6.

2 Syntax and Notation

Our language is the same as for our system of session types without subtyping [4] and is based on a polyadic π-calculus with output prefixing [15]. For simplicity we omit the original π-calculus choice construct $P+Q$ and the matching construct, which allows channel names to be tested for equality. However, we have the constructs introduced in Section 1 for choosing between a collection of labelled processes, as proposed by Honda et al. [6, 8, 25]. The inclusion of output prefixing is different from many recent presentations of the π-calculus, but it is essential because our type system must be able to impose an order on separate outputs on the same channel. Because it has an interesting effect on the typing rules, we add a conditional process expression, written $\mathsf{if}\ b\ \mathsf{then}\ P\ \mathsf{else}\ Q$, where b is a boolean value. As is standard, we use the replication operator ! instead of recursive process definitions.

As our language has conditional process expressions, we have a ground type of booleans. Other ground types such as int and real, as used in the examples in Section 1, could be added along with appropriate primitive operations. We also include Milner's standard π-calculus channel types [15], Pierce and Sangiorgi's input and output channel types [18, 19] and session types as proposed by Honda et al. [8, 25]. We include type variables, and allow all types to be recursive. An environment, which is particular to a process, describes the types of the names that are free in that process and our typing rules in Section 3 will allow us to derive processes that use names correctly with respect to their environment.

In general we use lower case letters for channel names, superscript p, q etc. for polarities, $l_1, \ldots, l_n$ for labels of choices, upper case P, Q, R for processes, upper case S, S_1, S_2 etc for session types and upper case T, U etc. for types. We write $\tilde{x}$ for a finite sequence $x_1^p, \ldots, x_n^q$ of names, $\tilde{x} : \tilde{T}$ for a finite sequence $x_1^{p_1} : T_1, \ldots, x_n^{p_n} : T_n$ of typed names and Γ, Γ_1, Γ_2 etc for environments.

2.1 Processes

The syntax of processes is defined by the grammar in Figure 1. Note that T , $\tilde{T}$ and S stand for types, lists of types and session types, which have not yet been defined. p is either $+$ or $-$ if the associated name is a session port, or is omitted in all other cases. $+$ and $-$ are dual (or complementary) polarities and we write $\overline{p}$ to denote the dual of p. Most other syntax is fairly standard. $\mathbf{0}$ is the inactive process, $|$ is parallel composition and $!P$ represents a potentially infinite supply of parallel copies of P. $(\nu x : T)P$ declares a local channel x of type T for use in P and $(\nu x^{\pm} : S)P$ declares a local session channel with two ports, x^+ and x^- of types S and $\bar{S}$ respectively, again for use in P. $x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,P$ receives the names $\tilde{y}$, which have types $\tilde{T}$, on the port x^p, and then executes P. $x^p\,!\,[\tilde{y}]\,.\,P$ outputs the names $\tilde{y}$ on the port x^p and then executes P. There should be no confusion between the use of ! for output and its use for replication, as the surrounding syntax is quite different in each case. $x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}$ offers a choice of subsequent behaviours on the port x^p – one of the P_i can be selected as the continuation process by sending the appropriate label l_i on the other port of the channel x, as explained in Section 1. $x^p \triangleleft l\ .\ P$ sends the label l on port x^p in order to make a selection from an choice offered in the other port of x, and then executes P. The conditional expression has already been mentioned.

We define free and bound names as usual: x is bound in $(\nu x : T)P$, x^+ and x^- are bound in $(\nu x^{\pm} : S)P$, the names in $\tilde{y}$ are bound in $x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,P$, and all other occurrences are free. We then define α-equivalence as usual, and identify processes which are α-equivalent.

$$
\begin{array}{rcl}
P & ::= & \mathbf{0} \\
& | & P \mid Q \\
& | & x^p \,?\, [\tilde{y} : \tilde{T}] \,.\, P \\
& | & x^p \,!\, [\tilde{y}] \,.\, P \\
& | & (\nu x : T)P \\
& | & (\nu x^{\pm} : S)P \\
& | & x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \\
& | & x^p \triangleleft l \,.\, P \\
& | & !P \\
& | & \mathsf{if}\ x\ \mathsf{then}\ P\ \mathsf{else}\ Q
\end{array}
$$

Figure 1: Syntax of Process

$$
\begin{array}{rcl}
\mathbf{0}\{\tilde{u}/\tilde{v}\} & = & \mathbf{0} \\
(P \mid Q)\{\tilde{u}/\tilde{v}\} & = & P\{\tilde{u}/\tilde{v}\} \mid Q\{\tilde{u}/\tilde{v}\} \\
(x^p \,?\, [\tilde{y} : \tilde{T}] \,.\, P)\{\tilde{u}/\tilde{v}\} & = & x^p\{\tilde{u}/\tilde{v}\} \,?\, [\tilde{y} : \tilde{T}] \,.\, P\{\tilde{u}/\tilde{v}\} \\
(x^p \,!\, [\tilde{y}] \,.\, P)\{\tilde{u}/\tilde{v}\} & = & x^p\{\tilde{u}/\tilde{v}\} \,!\, [\tilde{y}]\{\tilde{u}/\tilde{v}\} \,.\, P\{\tilde{u}/\tilde{v}\} \\
((\nu x : T)P)\{\tilde{u}/\tilde{v}\} & = & (\nu x : T)P\{\tilde{u}/\tilde{v}\} \\
((\nu x^{\pm} : S)P)\{\tilde{u}/\tilde{v}\} & = & (\nu x^{\pm} : S)P\{\tilde{u}/\tilde{v}\} \\
(x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\})\{\tilde{u}/\tilde{v}\} & = & x^p\{\tilde{u}/\tilde{v}\} \triangleright \{l_1 : P_1\{\tilde{u}/\tilde{v}\}, \ldots, l_n : P_n\{\tilde{u}/\tilde{v}\}\} \\
(x^p \triangleleft l \,.\, P)\{\tilde{u}/\tilde{v}\} & = & x^p\{\tilde{u}/\tilde{v}\} \triangleleft l \,.\, P\{\tilde{u}/\tilde{v}\} \\
(!P)\{\tilde{u}/\tilde{v}\} & = & !P\{\tilde{u}/\tilde{v}\} \\
(\ \mathsf{if}\ x\ \mathsf{then}\ P\ \mathsf{else}\ Q)\{\tilde{u}/\tilde{v}\} & = & \mathsf{if}\ x\{\tilde{u}/\tilde{v}\}\ \mathsf{then}\ P\{\tilde{u}/\tilde{v}\}\ \mathsf{else}\ Q\{\tilde{u}/\tilde{v}\}
\end{array}
$$

Figure 2: Substitution

We include the standard operation of substitution of names for names: $P\{\tilde{x}/\tilde{y}\}$ denotes P with the names $x_1^{p_1}, \ldots, x_n^{p_n}$ simultaneously substituted for $y_1^{q_1}, \ldots, y_n^{q_n}$, assuming that bound names are renamed if necessary to avoid capture of substituting names. Substitution is defined in Figure 2.

Figure 3 defines a standard *structural congruence* relation, written $\equiv$, which helps to define the operational semantics. In rule S-Offer, σ is a permutation on $\{1, \ldots, n\}$.

2.2 Types

The syntax of types is defined by the grammar in Figure 3. Ground types are self-explanatory. Here we only use bool, but others could easily be added, along with their respective operations. For channel types, $\hat{}[T_1, \ldots, T_n]$ represents a channel that has input and output capabilities and carries names of the types $T_1, \ldots, T_n$. $?[T_1, \ldots, T_n]$ and $![T_1, \ldots, T_n]$ also represent channels that carry names of these types, but with only input or output capability respectively.

For session types, end, represents a port of a session channel that has completed its sequence of communications. $?[T_1, \ldots, T_n].S$ and $![T_1, \ldots, T_n].S$ represent ports that will either

$$
\begin{array}{rcll}
P \mid \mathbf{0} & \equiv & P & \text{S-Unit} \\
P \mid Q & \equiv & Q \mid P & \text{S-Comm} \\
P \mid (Q \mid R) & \equiv & (P \mid Q) \mid R & \text{S-Assoc} \\
(\nu x^p : T)P \mid Q & \equiv & (\nu x^p : T)(P \mid Q) \text{ if } x \text{ is not free in } Q & \text{S-Extr} \\
(\nu x : T)\mathbf{0} & \equiv & \mathbf{0} \text{ if } x \text{ is not a session channel} & \text{S-Nil} \\
(\nu x^p : T)(\nu y^q : U)P & \equiv & (\nu y^q : U)(\nu x^p : T)P & \text{S-Switch} \\
!P & \equiv & P|!P & \text{S-Rep} \\
x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} & \equiv & x^p \triangleright \{l_{\sigma(1)} : P_{\sigma(1)}, \ldots, l_{\sigma(n)} : P_{\sigma(n)}\} & \text{S-Offer}
\end{array}
$$

Figure 3: Stuctural Congruence Rules

$$
\begin{array}{lrcl}
\text{Types} & T & ::= & G \\
 & & | & C \\
 & & | & S \\
 & & | & \mu X.T \text{(recursive type)} \\
 & & | & X \text{(type variable)} \\
\text{Ground types} & G & ::= & \mathsf{bool} \\
\text{Channel types} & C & ::= & \hat{}[T_1, \ldots, T_n] \\
 & & | & ?[T_1, \ldots, T_n] \\
 & & | & ![T_1, \ldots, T_n] \\
\text{Session types} & S & ::= & \mathsf{end} \\
 & & | & ?[T_1, \ldots, T_n] \,.\, S \\
 & & | & ![T_1, \ldots, T_n] \,.\, S \\
 & & | & \&\langle l_1 : S_1, \ldots, l_n : S_n \rangle \\
 & & | & \oplus\langle l_1 : S_1, \ldots, l_n : S_n \rangle \\
 & & | & \mu X.S \text{(recursive session type)} \\
 & & | & X \text{(type variable)}
\end{array}
$$

Figure 4: Syntax of Types

input or output names of the types $T_1, \ldots, T_n$ and then have the type S. $\&\langle l_1 : S_1, \ldots, l_n : S_n \rangle$ represents a port that will receive one of n labels and then continue with the corresponding continuation type. $\oplus\langle l_1 : S_1, \ldots, l_n : S_n \rangle$ represents a port that will send one of n labels and then continue with the corresponding continuation type. Finally, $\mu X.S$ and X allow the definition of recursive types and type variables.

If S is a session type then $\overline{S}$, the dual (or complementary) type of S, is defined inductively as follows.

$$\begin{array}{rcl}
\overline{?[\tilde{T}].S} & = & ![\tilde{T}].\overline{S} \\
\overline{![\tilde{T}].S} & = & ?[\tilde{T}].\overline{S} \\
\overline{\&\langle l_1 : S_1, \ldots, l_n : S_n \rangle} & = & \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n} \rangle \\
\overline{\oplus\langle l_1 : S_1, \ldots, l_n : S_n \rangle} & = & \&\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n} \rangle \\
\overline{\mathsf{end}} & = & \mathsf{end} \\
\overline{\mu X.T} & = & \mu X.\overline{T} \\
\overline{X} & = & X
\end{array}$$

2.3 Environments

An environment is a set of typed names, written $x_1^{p_1} : T_1, \ldots, x_n^{p_n} : T_n$. We use Γ, Γ_1, Γ_2 etc. and Γ', Γ'' etc. to stand for environments. We assume that all the names in an environment are distinct. We write $x^p \in \Gamma$ to indicate that x^p is one of the names appearing in Γ, and write $\Gamma \vdash x^p : T$ or $\Gamma(x^p) = T$ to indicate x^p has the type T in Γ. When $x^p \notin \Gamma$ we write $\Gamma, x^p : T$ for the environment formed by adding $x^p : T$ to the set of typed names in Γ. When Γ_1 and Γ_2 have disjoint sets of names, we write Γ_1, Γ_2 for their union. Implicitly, *true* : bool and *false* : bool appear in every environment.

The partial operation $+$, combining a typed name with an environment, is defined as follows:

$$\begin{array}{rcll}
\Gamma + x^p : T & = & \Gamma, x^p : T & \text{if } x^p \notin \Gamma \\
(\Gamma, x : T) + x : U & = & \Gamma, x : T & \text{if } x \text{ is not a session port and } T \leqslant U \\
(\Gamma, x : T) + x : U & = & \Gamma, x : U & \text{if } x \text{ is not a session port and } U \leqslant T
\end{array}$$

and is undefined in all other cases.

We extend $+$ to a partial operation on environments by defining

$$\Gamma + (x_1^{p_1} : T_1, \ldots, x_n^{p_n} : T_n) \quad = \quad (\cdots(\Gamma + x_1^{p_1} : T_1) + \cdots +) + x_n^{p_n} : T_n$$

We say that an environment is *completed* if any session ports it contains have type end, and *unlimited* if it contains no session types at all. An environment, Γ, is *balanced* with respect to a channel x if

$$\Gamma \vdash x^p : S \implies \Gamma \vdash x^{\overline{p}} : \overline{S}$$
or
$$\Gamma \vdash x : T \text{ where } T \text{ is not a session type}$$

A process that is typable in the empty environment (i.e. a process with no free variables) is called a *program*.

3 Type System

The typing rules allow us to derive processes that are correctly typed with respect to their environment. Derivations start with the nil process, $\mathbf{0}$, typed in some environment that is completed and apply the rules to add operations as prefixes, making the appropriate modifications to the environment. For example, the process oldclient from Section 1 is typable in the environment $x^- : \overline{S'}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x^- : \mathsf{end}, a : \mathsf{real}\ \text{completed}}{x^- : \mathsf{end}, a : \mathsf{real} \vdash \mathbf{0}}\ \text{T-Nil} \quad \mathsf{real} \leqslant \mathsf{real}}{x^- : ?[\mathsf{real}] \,.\, \mathsf{end} \vdash x^- \,?\, [a : \mathsf{real}] \,.\, \mathbf{0}}\ \text{T-InSeq} \quad \mathsf{int} \leqslant \mathsf{real}}{x^- : \overline{S'}_{sin}, 45 : \mathsf{int} \vdash x^- \,!\, [45] \,.\, x^- \,?\, [a : \mathsf{real}] \,.\, \mathbf{0}}\ \text{T-OutSeq} \quad 1 \leq 3 \leq 4 \quad \overline{S'}_{sin} \leqslant \overline{S'}_{sin}}{x^- : \overline{S'} \vdash x^- \triangleleft sin \,.\, x^- \,!\, [45] \,.\, x^- \,?\, [a : \mathsf{real}] \,.\, \mathbf{0}}\ \text{T-Offer}$$

The type $\overline{S'}_{sin}$ is used above to make the typing derivation more compact and represents $![\mathsf{real}] \,.\, ?[\mathsf{real}] \,.\, \mathsf{end}$, the type associated with the label sin in the type $\overline{S'}$. The values in the inequality $1 \leq 3 \leq 4$ come simply from enumeration of the labels in $\overline{S'}$.

The typing rules can also be used to typecheck a process in an environment. This is done by reading the rules upwards instead of down. Depending on the actions that the process performs, the appropriate typing rules are applied in reverse until the process $\mathbf{0}$ is reached, typed in a completed environment. If no such path through the rules exists, the process does not typecheck in that environment. If we wanted to typecheck the process client in the environment $x^- : \overline{S'}$, for example, the derivation above read backwards shows a valid path and thus demonstates type correctness.

3.1 Typing Rules

The typing rules are defined in Figure 5. T-Nil types the process $\mathbf{0}$, provided that there are no session channels in the environment that have not completed their communication session. T-Par adds together the environments that type two processes individually to obtain a new environment to type them in parallel. T-New removes a channel from the environment in the normal way when it is to be bound by a ν. T-NewSeq does the same for both ports of a session channel. T-Out deals with output on a non-session channel, making sure that the channel in the environment has a type that is a subtype of a channel type that has the output capability for the type being sent. T-In does the same for input on non-session channels, making sure that the channel in the environment has a type that is a subtype of a channel type that has the input capability for the type being received. T-OutSeq and T-InSeq are similar to the output and input rules for non-session channels. The type of the session port in the environment has an output or input prefix added as appropriate, though, to indicate the additional capability being used by the port. T-Offer types a process offering a choice between a number of labels on a session port provided that the type associated with each label in the environment is a subtype of the type being used in the process by the port. The number of labels in the type of the port in the environment must also be no more than those offered in the process. T-Choose types a process that chooses between a number of labels provided that the type associated with the chosen label in the environment is a subtype of the type used in the process. This time the number of labels in the type of the port in the

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash \mathbf{0}}\ \text{T-Nil} \qquad \frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1 + \Gamma_2 \vdash P \mid Q}\ \text{T-Par}$$

$$\frac{\Gamma, x : T \vdash P \quad T \text{ is not a session}}{\Gamma \vdash (\nu x : T)P}\ \text{T-New} \qquad \frac{\Gamma, x^+ : S, x^- : \bar{S} \vdash P}{\Gamma \vdash (\nu x^{\pm} : S)P}\ \text{T-NewSeq}$$

$$\frac{\Gamma \vdash P \quad \Gamma \vdash x \leqslant ![\tilde{T}]}{\Gamma + \tilde{y} : \tilde{T} \vdash x\,![\tilde{y}]\,.\,P}\ \text{T-Out} \qquad \frac{\Gamma, x^p : S \vdash P \quad \tilde{U} \leqslant \tilde{T}}{(\Gamma, x^p : ![\tilde{T}]\,.\,S) + \tilde{y} : \tilde{U} \vdash x^p\,![\tilde{y}]\,.\,P}\ \text{T-OutSeq}$$

$$\frac{\Gamma, \tilde{y} : \tilde{T} \vdash P \quad \Gamma \vdash x \leqslant ?[\tilde{T}]}{\Gamma \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,P}\ \text{T-In} \qquad \frac{\Gamma, x^p : S, \tilde{y} : \tilde{U} \vdash P \quad \tilde{T} \leqslant \tilde{U}}{\Gamma, x^p : ?[\tilde{T}]\,.\,S \vdash x^p\,?\,[\tilde{y} : \tilde{U}]\,.\,P}\ \text{T-InSeq}$$

$$\frac{\Gamma, x^p : S_1 \vdash P_1; \ldots; \Gamma, x^p : S_n \vdash P_n \quad m \leq n \quad \forall i \in \{1, \ldots, m\}.T_i \leqslant S_i}{\Gamma, x^p : \&\langle l_1 : T_1, \ldots, l_n : T_m\rangle \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}\ \text{T-Offer}$$

$$\frac{\Gamma, x^p : S_i \vdash P \quad 1 \leq i \leq n \quad T_i \leqslant S_i}{\Gamma, x^p : \oplus\langle l_1 : T_1, \ldots, l_n : T_n\rangle \vdash x^p \triangleleft l_i\,.\,P}\ \text{T-Choose}$$

$$\frac{\Gamma \vdash x \leqslant \mathsf{bool} \quad \Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash \ \mathsf{if}\ x\ \mathsf{then}\ P\ \mathsf{else}\ Q}\ \text{T-Cond} \qquad \frac{\Gamma \vdash P \quad \Gamma \text{ unlimited}}{\Gamma \vdash !P}\ \text{T-Rep}$$

Figure 5: Typing Rules

environment must be no less than those the process is choosing between. Finally, T-Cond types a conditional expression provided that the condition is a subtype of bool and that the processes representing the two possible outcomes are well typed, and T-Rep types the replication of a process provided that the process is well typed in an environment containing no session channels.

Each typing rule is only applicable when any instances of $+$ which it contains are actually defined. This ensures that the environment correctly records the use being made of session channels. In rules T-Out and T-OutSeq, for example, the names being output are added to the environment. This means that if a session port is output, then it cannot be used again by the remainder of the process. This allows a process to begin a communication on a session channel, then delegate the rest of the session to another process by sending it the port that it was using. Of course, the first process must not use the channel again.

3.2 Subtyping

Figure 6 defines the subtype relation by means of a collection of inference rules for judgements of the form $\Sigma \vdash T \leqslant U$, where Σ ranges over finite sets of instances of $\leqslant$. When $\emptyset \vdash T \leqslant U$ is derivable we simply write $T \leqslant U$. The inference rules can be interpreted as an algorithm for checking whether $T \leqslant U$ for given T and U, as follows. Beginning with the goal $\emptyset \vdash T \leqslant U$, apply the rules upwards to generate subgoals. Pattern matching on the structure of T and

$$\frac{T \leqslant U \in \Sigma}{\Sigma \vdash T \leqslant U}\ \text{AS-Assump}$$

$$\frac{}{\Sigma \vdash \mathsf{bool} \leqslant \mathsf{bool}}\ \text{AS-Bool} \qquad \frac{}{\Sigma \vdash \mathsf{end} \leqslant \mathsf{end}}\ \text{AS-End}$$

$$\frac{\forall i \in \{1,\ldots,n\}.\Sigma \vdash T_i \leqslant U_i}{\Sigma \vdash ?[\tilde{T}] \leqslant ?[\tilde{U}] \quad \Sigma \vdash \hat{}[\tilde{T}] \leqslant ?[\tilde{U}]}\ \text{AS-In}$$

$$\frac{\forall i \in \{1,\ldots,n\}.\Sigma \vdash U_i \leqslant T_i}{\Sigma \vdash ![\tilde{T}] \leqslant ![\tilde{U}] \quad \Sigma \vdash \hat{}[\tilde{T}] \leqslant ![\tilde{U}]}\ \text{AS-Out}$$

$$\frac{\forall i \in \{1,\ldots,n\}.(\Sigma \vdash T_i \leqslant U_i \text{ and } \Sigma \vdash U_i \leqslant T_i)}{\Sigma \vdash \hat{}[\tilde{T}] \leqslant \hat{}[\tilde{U}]}\ \text{AS-InOut}$$

$$\frac{\Sigma \vdash V \leqslant W \quad \forall i \in \{1,\ldots,n\}.\Sigma \vdash T_i \leqslant U_i}{\Sigma \vdash ?[\tilde{T}]\,.\,V \leqslant ?[\tilde{U}]\,.\,W}\ \text{AS-InSeq}$$

$$\frac{\Sigma \vdash V \leqslant W \quad \forall i \in \{1,\ldots,n\}.\Sigma \vdash U_i \leqslant T_i}{\Sigma \vdash ![\tilde{T}]\,.\,V \leqslant ![\tilde{U}]\,.\,W}\ \text{AS-OutSeq}$$

$$\frac{m \leqslant n \quad \forall i \in \{1,\ldots,m\}.\Sigma \vdash S_i \leqslant T_i}{\Sigma \vdash \&\langle l_1 : S_1, \ldots, l_m : S_m\rangle \leqslant \&\langle l_1 : T_1, \ldots, l_n : T_n\rangle}\ \text{AS-Branch}$$

$$\frac{m \leqslant n \quad \forall i \in \{1,\ldots,m\}.\Sigma \vdash S_i \leqslant T_i}{\Sigma \vdash \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle \leqslant \oplus\langle l_1 : T_1, \ldots, l_m : T_m\rangle}\ \text{AS-Choice}$$

$$\frac{\Sigma, \mu X.S \leqslant T \vdash \text{unwind}(\mu X.S) \leqslant T}{\Sigma \vdash \mu X.S \leqslant T}\ \text{AS-Rec-L}$$

$$\frac{\Sigma, T \leqslant \mu X.S \vdash T \leqslant \text{unwind}(\mu X.S)}{\Sigma \vdash T \leqslant \mu X.S}\ \text{AS-Rec-R}$$

Figure 6: Subtyping Rules

U determines which rule to use, except that the rule AS-Assump should always be used, causing the current subgoal to succeed, if it is applicable. If both AS-Rec-L and AS-Rec-R are applicable then they should be used in either order. If a subgoal is generated which does not match any of the rules, the algorithm returns false.

The rules AS-Bool and AS-End say that we can always assume that bool is a subtype of bool and end is a subtype of end.

3.3 Addition on Types and Environments

The + operation has two uses in the typing rules. The first use is seen in rules T-Out and T-OutSeq, where the names being output are added to the environment. This means that if a port of a session channel is output, then that port is given away and cannot be used again by the remainder of the process. It is possible, therefore, to begin an interaction on a session channel, then safely delegate the rest of the interaction to another process by sending it the port. If the first process attempted to use the port again, this would be rejected by the type system.

Delegation arises in recursive processes, where a session channel with a recursive type is used and the process is replicated. In this situation, a new instance of the recursive process is invoked to which the session channel must be passed on: the process actually delegates to a separate copy of itself. Below is a recursive example of an addition server – a cut-down version of the maths server in Section 1.

$$\begin{array}{rcl}
\mathsf{recserver} & = & !\mathsf{thread} \\
\mathsf{thread} & = & \mathsf{trigger}\,?\,[x^+ : \mu X.?[\mathsf{int}]\,.\,?[\mathsf{int}]\,.\,![\mathsf{int}]\,.\,X]\,.\,\mathsf{server} \\
\mathsf{server} & = & x^+\,?\,[a : \mathsf{int}]\,.\,x^+\,?\,[b : \mathsf{int}]\,.\,x^+\,!\,[a+b]\,.\,\mathsf{trigger}\,!\,[x^+]\,.\,\mathbf{0} \\
\mathsf{client} & = & (\nu y^{\pm} : \mu X.?[\mathsf{int}]\,.\,?[\mathsf{int}]\,.\,![\mathsf{int}]\,.\,X)\mathsf{trigger}\,!\,[y^+]\,.\,\mathsf{clientbody} \\
\mathsf{clientbody} & = & y^-\,!\,[2]\,.\,y^-\,!\,[3]\,.\,y^-\,?\,[a : \mathsf{int}]\,.\,y^-\,!\,[a]\,.\,y^-\,!\,[4]\,.\,y^-\,?\,[b : \mathsf{int}]\,.\,P
\end{array}$$

The client creates the recursive session channel, y, and passes the y^+ port to a copy of thread on the trigger channel. server then uses it to receive two integers and sends back the result. At this point, server sends the port on the trigger channel. This is received by a separate copy of thread, and the new copy can perform the second addition. In this situation, there is no option to end the recursion. More realistically, the recursion would be embedded in an offer construct that included some termination option.

The other occurence of the + operation is in the T-Par rule. Here, the operation is used to prevent multiple processes using a single session port being put in parallel. Two clients which both interact with a server using the same port should not be allowed, for example, and indeed,

$$\frac{x^+ : S, x^- : \overline{S} \vdash \mathsf{server} \mid \mathsf{client1} \qquad x^- : \overline{S} \vdash \mathsf{client2}}{(x^+ : S, x^- : \overline{S}) + x^- : \overline{S} \vdash \mathsf{server} \mid \mathsf{client} \mid \mathsf{client}}\ \text{T-Par}$$

is not a valid application of the rule, as the environment resulting from the addition is not defined.

4 Operational Semantics

The operational semantics is defined by means of a reduction relation, where $P \xrightarrow{\alpha,l} Q$ means that the process P reduces to the process Q by executing a single communication step or evaluating a conditional expression. The reductions are labelled in the following way: If the channel on which the communication takes place is free in P then α is the name of that channel. If it is a bound channel, or if the reduction corresponds to the evaluation of a conditional expression, $\alpha = \tau$. l is the label passed on the channel during the reduction if the reduction in question is a selection. $l = \_$ in all other cases.

The reduction relation is the smallest relation closed under the rules in Figure 7. The rules R-Comm and R-Select introduce communication steps. R-Comm is standard apart from a pre-condition that the polarities must be opposite if the reduction is on a session channel. R-Select deals with reductions where one process selects a labelled choice from a list offered by the other. Note that R-Comm applies to both session and non-session channels, as there is no indication of the type of x in either process.

R-True and R-False describe the two possible reductions of a conditional expression, R-Par allows reductions under parallel composition and R-Cong allows any process that is structurally congruent to a process with a reduction, to reduce to a process structurally congruent to the process resulting from that reduction. R-New and R-NewSeq describe reductions under a new binding, where the channel on which the reduction takes place is not the one being bound. R-NewX and R-NewXSeq again describe reductions under a new binding, but where the reductions is on the channel being bound. The tail function used in R-NewXSeq takes a session type and returns a new session type representing the original after a single reduction step.

As for our system without subtypes, we must prove that a well typed process can never reduce to a process with communication errors. For that system, we proved two theorems: A subject reduction theorem, showing that a well typed process always reduces to another well typed processs, and secondly a run-time safety theorem, showing that a well typed process has no communication errors. The type system with subtyping is more complicated, however, as a name can be promoted in a process to a supertype of its actual type in the environment. The following typing judgement is an example of this.

$$\Gamma, x^+ : ?[\hat{}[\mathsf{int}]] \,.\, \mathsf{end}, x^- : ![\hat{}[\mathsf{int}]] \,.\, \mathsf{end}, z : \hat{}[\mathsf{int}] \vdash x^+ \,?\, [y : ![\mathsf{int}]] \,.\, y \,!\, [4] \,.\, \mathbf{0} \mid x^- \,!\, [z] \,.\, \mathbf{0} \mid z \,!\, [3] \,.\, \mathbf{0}$$

The process is typable – a valid path exists through the typing rules. The capabilities in the environment, however, are the sum of the three environments that type the three parallel processes. Because of this, it is not obvious that the output capability for the channel z (used by the first process after it binds z to y) comes from the environment that types the second process. It could come only from the environment that types the third process, which clearly must have output capability on z, with the second process only having input capability. We would hope that if this were true, the process would not be well typed, but to prove this for all processes we clearly need to know the capabilities that are available on a channel at the time they are sent across a parallel composition. Pierce and Sangiorgi [18, 19] use a system of tagging to keep track of promoted types in their system of input and output types, and we use a similar approach, tagging every name that exists without a typing constraint.

First we define tagged processes, which are the same as untagged processes, but with a typing constraint on every occurence of a name. The stuctural congruence rules must also be modified to include the tags. Next, we define a tagging function from untagged typable

$$\frac{\text{if } x \text{ is a session channel then } p = \bar{q}}{x^p\,?\,[\tilde{y}:\tilde{T}]\,.\,P \mid x^q\,!\,[\tilde{z}]\,.\,Q \xrightarrow{x,\text{-}} P\{\tilde{z}/\tilde{y}\} \mid Q}\ \text{R-Comm}$$

$$\frac{l \in \{l_1,\ldots,l_n\} \quad p = \bar{q}}{x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i\,.\,Q \xrightarrow{x,l_i} P_i \mid Q}\ \text{R-Select}$$

$$\frac{}{\mathsf{if}\ \mathit{true}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau,\text{-}} P}\ \text{R-True} \qquad \frac{}{\mathsf{if}\ \mathit{false}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau,\text{-}} Q}\ \text{R-False}$$

$$\frac{P \xrightarrow{\alpha,l} P'}{P \mid Q \xrightarrow{\alpha,l} P' \mid Q}\ \text{R-Par} \qquad \frac{P' \equiv P \quad P \xrightarrow{\alpha,l} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha,l} Q'}\ \text{R-Cong}$$

$$\frac{P \xrightarrow{\alpha,l} P' \quad \alpha \neq x}{(\nu x : T)P \xrightarrow{\alpha,l} (\nu x : T)P'}\ \text{R-New} \qquad \frac{P \xrightarrow{\alpha,l} P' \quad \alpha \neq x}{(\nu x^{\pm} : S)P \xrightarrow{\alpha,l} (\nu x^{\pm} : S)P'}\ \text{R-NewSeq}$$

$$\frac{P \xrightarrow{x,l} P'}{(\nu x : T)P \xrightarrow{\tau,\text{-}} (\nu x : T)P'}\ \text{R-NewX} \qquad \frac{P \xrightarrow{x,l} P'}{(\nu x^{\pm} : S)P \xrightarrow{\tau,\text{-}} (\nu x^{\pm} : \mathrm{tail}(S,l))P'}\ \text{R-NewXSeq}$$

Figure 7: The reduction relation

$$\begin{aligned}
\mathrm{tail}(?[\tilde{T}].S, _) &= S \\
\mathrm{tail}(![\tilde{T}].S, _) &= S \\
\mathrm{tail}(\&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, l_i) &= S_i \\
\mathrm{tail}(\oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle, l_i) &= S_i \\
\mathrm{tail}(\mu X.S) &= \mathrm{tail}(S\{\mu X.S/X\})
\end{aligned}$$

Figure 8: The tail function

processes to tagged processes, and an erase function from tagged to untagged processes. Finally, the typing rules and reduction rules must also be modified to include the tags. The tagged typing rules are shown in Figure 14 and the tagged reduction rules which are different from the untagged ones are shown in Figure 13.

We can prove type soundness for this tagged system in the usual way: we prove a subject reduction theorem and a run-time safety theorem. Together these imply that a well-typed process can be executed safely through any sequence of reduction steps. As with our standard session type system, we are only interested in processes typed in a balanced environment as these are the only processes that can result from the execution of a program. The following lemmas are steps towards establishing Theorem 1. Where the + operation is used, we can assume it is a valid application because all occurences will occur as a result of a substitution when a communication takes place in a well-typed process. The only invalid use of + is to add a session port to an environment that already contains it and this cannot happen in a well typed process as the typing rules remove a session port from the environment whenever

$$
\begin{array}{rcl}
P & ::= & \mathbf{0} \\
& | & P \mid Q \\
& | & (x^p : T)\,?\,[\tilde{y} : \tilde{U}]\,.\,P \\
& | & (x^p : T)\,!\,[y : \tilde{U}]\,.\,P \\
& | & (\nu x : T)P \\
& | & (\nu x^{\pm} : S)P \\
& | & (x^p : S) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \\
& | & (x^p : S) \triangleleft l\,.\,P \\
& | & !P \\
& | & \mathsf{if}\ (x : T)\ \mathsf{then}\ P\ \mathsf{else}\ Q
\end{array}
$$

Figure 9: Syntax of Tagged Processes

$$
\begin{array}{rcll}
P \mid \mathbf{0} & \equiv & P & \text{TS-Unit} \\
P \mid Q & \equiv & Q \mid P & \text{TS-Comm} \\
P \mid (Q \mid R) & \equiv & (P \mid Q) \mid R & \text{TS-Assoc} \\
(\nu x^p : T)P \mid Q & \equiv & (\nu x^p : T)(P \mid Q) \text{ if } x \text{ is not free in } Q & \text{TS-Extr} \\
(\nu x : T)\mathbf{0} & \equiv & \mathbf{0} \text{ if } x \text{ is not a session channel} & \text{TS-Nil} \\
(\nu x^p : T)(\nu y^q : U)P & \equiv & (\nu y^q : U)(\nu x^p : T)P & \text{TS-Switch} \\
!P & \equiv & P|!P & \text{TS-Rep} \\
(x^p : S) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} & \equiv & (x^p : S) \triangleright \{l_{\sigma(1)} : P_{\sigma(1)}, \ldots, l_{\sigma(n)} : P_{\sigma(n)}\} & \text{TS-Offer}
\end{array}
$$

Figure 10: Tagged Structural Congruence Rules

$$
\begin{array}{rcl}
\mathrm{tag}_\Gamma(\mathbf{0}) & = & \mathbf{0} \\
\mathrm{tag}_\Gamma(P \mid Q) & = & \mathrm{tag}_{\Gamma_1}(P) \mid \mathrm{tag}_{\Gamma_2} Q \ {}^{(1)} \\
\mathrm{tag}_\Gamma((\nu x : T)P) & = & (\nu x : T)(\mathrm{tag}_{\Gamma, x:T}(P)) \\
\mathrm{tag}_\Gamma((\nu x^{\pm} : S)P) & = & (\nu x : S)(\mathrm{tag}_{\Gamma, x^+:S, x^-:\overline{S}}(P)) \\
\mathrm{tag}_{(\Gamma, x:C)+y:T}(x\,!\,[\tilde{y}]\,.\,P) & = & (x : C)\,!\,[\tilde{y} : \tilde{T}]\,.\,\mathrm{tag}_{\Gamma, x:C}(P) \\
\mathrm{tag}_{\Gamma, x:C}(x\,?\,[\tilde{y} : \tilde{T}]\,.\,P) & = & (x : C)\,?\,[\tilde{y} : \tilde{T}]\,.\,\mathrm{tag}_{(\Gamma, x:C)+\tilde{y}:\tilde{T}}(P) \\
\mathrm{tag}_{(\Gamma, x^p:S)+y:T}(x^p\,!\,[\tilde{y}]\,.\,P) & = & (x^p : S)\,!\,[\tilde{y} : \tilde{T}]\,.\,\mathrm{tag}_{\Gamma, x^p:\mathrm{tail}(S)}(P) \\
\mathrm{tag}_{\Gamma, x^p:S}(x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,P) & = & (x^p : S)\,?\,[\tilde{y} : \tilde{T}]\,.\,\mathrm{tag}_{\Gamma, x^p:\mathrm{tail}(S)+\tilde{y}:\tilde{T}}(P) \\
\mathrm{tag}_{\Gamma, x:\&\langle l_i T_i\rangle_{1\leqslant i\leqslant m}}(x^p \triangleright \{l_i : P_i\}_{1\leqslant i\leqslant n}) & = & (x^p : \&\langle l_i : T_i\rangle_{1\leqslant i\leqslant m}) \triangleright \{l_i : \mathrm{tag}_{\Gamma, x:S_i}(P_i)\}_{1\leqslant i\leqslant n} \ {}^{(2)} \\
\mathrm{tag}_{\Gamma, x:\oplus\langle l_i S_i\rangle_{1\leqslant i\leqslant n}}(x \triangleleft l\,.\,P) & = & (x : \oplus\langle l_i : S_i\rangle_{1\leqslant i\leqslant n}) \triangleleft l\,.\,\mathrm{tag}_{\Gamma, x:S_i}(P) \\
\mathrm{tag}_{\Gamma, x:T}(\ \mathsf{if}\ x\ \mathsf{then}\ P\ \mathsf{else}\ Q) & = & \ \mathsf{if}\ x : T\ \mathsf{then}\ \mathrm{tag}_{\Gamma, x:T}(P)\ \mathsf{else}\ \mathrm{tag}_{\Gamma, x:T}(Q) \\
\mathrm{tag}_\Gamma(!P) & = & !\mathrm{tag}_\Gamma(P)
\end{array}
$$

(1) where the last step of the typing derivation is

$$\frac{\Gamma_1 \vdash P \qquad \Gamma_2 \vdash Q}{\Gamma \vdash P \mid Q}\ \text{T-Par}$$

(2) where the last step of the typing derivation is

$$\frac{\Gamma, x^p : S_1 \vdash P_1; \ldots; \Gamma, x^p : S_n \vdash P_n \quad m \leq n \quad \forall i \in \{1, \ldots, m\}.T_i \leqslant S_i}{\Gamma, x^p : \&\langle l_1 : T_1, \ldots, l_n : T_m\rangle \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}\ \text{T-Offer}$$

Figure 11: Tag function

it is occurs in an output message.

Lemma 1 (Non-Session Subtypes) *If T is not a session type and $U \leqslant T$ then U is not a session type.*

Proof Straightforward from the definition of subtyping - no subtyping rule allows a session type to be a subtype of non-session type.

□

Lemma 2 (Completion) *If* $\Gamma, x^p : T$ completed *and* $U \leqslant T$ *then* $\Gamma + y^q : U$ completed

Proof This splits into two cases: one where T is a session type and another where it is not.

case T is a session type
As $\Gamma, x^p : T$ is completed, Γ must be completed and T must be end. From the definition of subtyping it can be seen that the only subtype of end is end, so U must also be end. From the definition of $+$, $\Gamma + y^q : \mathsf{end} = \Gamma, y^q : \mathsf{end}$ which, as Γ is completed, must itself be completed also.

case T is not a session type
As $\Gamma, x^p : T$ is completed, Γ must be completed. From Lemma 1, U cannot be a session type. As Γ is completed and U is not a session type, $\Gamma + y^q : U$ must also be completed.

□

$$
\begin{array}{rcl}
\mathrm{erase}(\mathbf{0}) & = & \mathbf{0} \\
\mathrm{erase}(P \mid Q) & = & \mathrm{erase}(P) \mid \mathrm{erase}(Q) \\
\mathrm{erase}((\nu x^p : T)P) & = & (\nu x^p : T)(\mathrm{erase}(P)) \\
\mathrm{erase}((x : T)\,!\,[\tilde{y} : \tilde{U}]\,.\,P) & = & x\,!\,[\tilde{y}]\,.\,\mathrm{erase}(P) \\
\mathrm{erase}((x : T)\,?\,[\tilde{y} : \tilde{U}]\,.\,P) & = & x\,?\,[\tilde{y} : \tilde{U}]\,.\,\mathrm{erase}(P) \\
\mathrm{erase}((x^p : S) \triangleright \{l_1 : (P_1), \ldots, l_n : (P_n)\}) & = & x^p \triangleright \{l_1 : (\mathrm{erase}(P_1)), \ldots, l_n : \mathrm{erase}((P_n))\} \\
\mathrm{erase}((x^p : S) \triangleleft l\,.\,P) & = & x^p \triangleleft l\,.\,\mathrm{erase}(P) \\
\mathrm{erase}(\ \mathsf{if}\ x : T\ \mathsf{then}\ P\ \mathsf{else}\ Q) & = & \ \mathsf{if}\ x\ \mathsf{then}\ \mathrm{erase}(P)\ \mathsf{else}\ \mathrm{erase}(Q) \\
\mathrm{erase}(!P) & = & !\mathrm{erase}(P)
\end{array}
$$

Figure 12: Erase Function

$$\frac{\text{if } x \text{ is a session channel then } p = \bar{q}}{(x^p : U)\,?\,[\tilde{y} : \tilde{T}]\,.\,P \mid (x^q : V)\,!\,[\tilde{z} : \tilde{W}]\,.\,Q \xrightarrow{x,-}_{T} P\{\tilde{z}/\tilde{y}\} \mid Q}\ \text{TR-Comm}$$

$$\frac{l \in \{l_1, \ldots, l_n\} \quad p = \bar{q}}{(x^p : U) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid (x^q : V) \triangleleft l_i\,.\,Q \xrightarrow{x,l_i}_{T} P_i \mid Q}\ \text{TR-Select}$$

$$\frac{}{\mathsf{if}\ \mathit{true} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau,-}_{T} P}\ \text{TR-True}$$

$$\frac{}{\mathsf{if}\ \mathit{false} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau,-}_{T} Q}\ \text{TR-False}$$

Figure 13: The tagged reduction relation

Lemma 3 (Unlimitedness) *If* $\Gamma, x : T$ unlimited *and* $U \leqslant T$ *then* $\Gamma + y : U$ unlimited

Proof If $\Gamma, x : T$ unlimited, Γ must be unlimited and T cannot be a session type. From Lemma 1, U cannot be a session type. If Γ is unlimited and U is not a session type, $\Gamma + y : U$ is unlimited.

□

Lemma 4 (Type Addition) *If* $y \notin \tilde{x}$ *then* $(\Gamma, \tilde{x} : \tilde{T}) + y : U = (\Gamma + y : U), \tilde{x} : \tilde{T}$

Proof As $y \notin \tilde{x}$, the addition can only apply to types in Γ and, therefore, $(\Gamma, \tilde{x} : \tilde{T}) + y : U = (\Gamma + y : U), \tilde{x} : \tilde{T}$.

□

Lemma 5 (Subtype Transitivity) *If* $W \leqslant U$ *and* $U \leqslant T$ *then* $W \leqslant T$

This lemma is proved by structural induction on the derivations of $W \leqslant U$ and $U \leqslant T$. Due to the large number of rules the proof is rather long. Two example cases are given in Appendix A.

Lemma 6 (Substitution Typing) *If* $\Gamma, x : T \vDash_{\mathcal{T}} z \leqslant V$ *and* $U \leqslant T$ *then* $\Gamma + y : U \vDash_{\mathcal{T}} z\{y/x\} \leqslant V$

$$\frac{\Gamma \text{ completed}}{\Gamma \vDash_{\mp} \mathbf{0}}\ \text{TT-Nil} \qquad \frac{\Gamma_1 \vDash_{\mp} P \quad \Gamma_2 \vDash_{\mp} Q}{\Gamma_1 + \Gamma_2 \vDash_{\mp} P \mid Q}\ \text{TT-Par}$$

$$\frac{\Gamma, x : T \vDash_{\mp} P \quad T \text{ is not a session}}{\Gamma \vDash_{\mp} (\nu x : T)P}\ \text{TT-New} \qquad \frac{\Gamma, x^{+} : S, x^{-} : \bar{S} \vDash_{\mp} P}{\Gamma \vDash_{\mp} (\nu x^{\pm} : S)P}\ \text{TT-NewSeq}$$

$$\frac{\Gamma \vDash_{\mp} P \quad \Gamma \vDash_{\mp} x \leqslant V \leqslant ![\tilde{U}] \quad \tilde{T} \leqslant \tilde{U}}{\Gamma + \tilde{y} : \tilde{T} \vDash_{\mp} (x : V)\,![\tilde{y} : \tilde{U}]\,.\,P}\ \text{TT-Out} \qquad \frac{\Gamma, \tilde{y} : \tilde{T} \vDash_{\mp} P \quad \Gamma \vDash_{\mp} x \leqslant V \leqslant ?[\tilde{T}]}{\Gamma \vDash_{\mp} (x : V)\,?\,[\tilde{y} : \tilde{T}]\,.\,P}\ \text{TT-In}$$

$$\frac{\Gamma, x^{p} : S \vDash_{\mp} P \qquad S \leqslant B \qquad \tilde{U} \leqslant \tilde{W} \leqslant \tilde{A} \leqslant \tilde{T}}{(\Gamma, x^{p} : ![\tilde{T}]\,.\,S) + \tilde{y} : \tilde{U} \vDash_{\mp} (x^{p} : ![\tilde{A}]\,.\,B)\,![\tilde{y} : \tilde{W}]\,.\,P}\ \text{TT-OutSeq}$$

$$\frac{\Gamma, x^{p} : S, \tilde{y} : \tilde{U} \vDash_{\mp} P \qquad S \leqslant B \qquad \tilde{T} \leqslant \tilde{A} \leqslant \tilde{U}}{\Gamma, x^{p} : ?[\tilde{T}]\,.\,S \vDash_{\mp} (x^{p} : ?[\tilde{A}]\,.\,B)\,?\,[\tilde{y} : \tilde{U}]\,.\,P}\ \text{TT-InSeq}$$

$$\frac{\begin{array}{c}\Gamma, x^{p} : S_1 \vDash_{\mp} P_1; \ldots; \Gamma, x^{p} : S_n \vDash_{\mp} P_n \quad m \leq r \leq n \\ \forall i \in \{1, \ldots, m\}.T_i \leqslant U_i \quad \forall i \in \{1, \ldots, r\}.U_i \leqslant S_i\end{array}}{\Gamma, x^{p} : \&\langle l_1 : T_1, \ldots, l_n : T_m\rangle \vDash_{\mp} x^{p} \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}\ \text{TT-Offer}$$

$$\frac{\Gamma, x^{p} : S_i \vDash_{\mp} P \quad U_i \leqslant S_i \quad 1 \leq i \leq m \leq n \quad \forall i \in \{1, \ldots, m\}.T_i \leqslant U_i}{\Gamma, x^{p} : \oplus\langle l_1 : T_1, \ldots, l_n : T_n\rangle \vDash_{\mp} (x^{p} : \oplus\langle l_1 : U_1, \ldots, l_m : U_m\rangle) \triangleleft l_i\,.\,P}\ \text{TT-Choose}$$

$$\frac{\Gamma \vDash_{\mp} x \leqslant \mathsf{bool} \quad \Gamma \vDash_{\mp} P \quad \Gamma \vDash_{\mp} Q}{\Gamma \vDash_{\mp} \mathsf{if}\ x : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q}\ \text{TT-Cond} \qquad \frac{\Gamma \vDash_{\mp} P \quad \Gamma \text{ unlimited}}{\Gamma \vDash_{\mp} !P}\ \text{TT-Rep}$$

Figure 14: Tagged Typing Rules

Proof This splits into two cases: one where $z = x$ and another where $z \neq x$.

case $z = x$
As $\Gamma, x : T \vDash_{\mp} x \leqslant V$, $T \leqslant V$. From Lemma 5, $U \leqslant V$. As $U \leqslant V$, $\Gamma + y : U \vDash_{\mp} x\{y/x\} \leqslant V$.

case $z \neq x$
As $\Gamma, x : T \vDash_{\mp} z \leqslant V$ and $z \neq x$, $\Gamma \vDash_{\mp} z \leqslant V$. As $\Gamma \vDash_{\mp} z \leqslant V$ and $z \neq x$, $\Gamma + y : U \vDash_{\mp} z\{y/x\} \leqslant V$.

□

Lemma 7 (No Substitution) *If* $\Gamma \vDash_{\mp} P$ *and* $x \notin \Gamma$ *then* $\Gamma \vDash_{\mp} P\{y/x\}$

Proof As $\Gamma \vDash_{\mp} P$ and $x \notin \Gamma$, $x \notin fn(P)$. As $x \notin fn(P)$, $P\{y/x\} = P$ and, therefore, $\Gamma \vDash_{\mp} P\{y/x\}$.

□

Lemma 8 (Substitution) *If* $\Gamma, x : T \vDash_{\mp} P$ *and* $U \leqslant T$ *then* $\Gamma + y : U \vDash_{\mp} P\{y/x\}$

The proof of this lemma is in Appendix A.

Lemma 9 (Structural Equivalence) *If* $\Gamma \models_{\mathcal{T}} P$ *and* $P \equiv Q$ *then* $\Gamma \models_{\mathcal{T}} Q$

This lemma is proved by structural induction on the derivation of $P \equiv Q$. It is essentially the same as the proof of the structural equivalence lemma for our system without subtyping [4], but with the additional type annotations introduced by the tag function.

Theorem 1 (Subject Reduction) *If* $\Gamma \models_{\mathcal{T}} P$ *and* $P \xrightarrow[\mathcal{T}]{\alpha,l} P'$ *where* Γ *is balanced with respect to* α *(if* $\alpha \neq \tau$*), then* $\Gamma' \models_{\mathcal{T}} P'$ *where*

$$\Gamma' = \Gamma \text{ if } \alpha = \tau \text{ or if } \alpha \text{ is not a session channel}$$
$$\Gamma' = \Gamma'', \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l)$$
$$\text{where } \Gamma'', \alpha^+ : S, \alpha^- : \overline{S} = \Gamma \text{ if } \alpha \text{ is a session channel.}$$

This theorem is proved by structural induction on the derivation of $P \xrightarrow[\mathcal{T}]{\alpha,l} P'$. It is similar to the proof of the same theorem for our system without subtyping. The proof of this theorem is given in Appendix A.

Lemma 10 *If* $\Gamma \vdash P$ *is a derivable untagged typing judgement, then* $\Gamma \models_{\mathcal{T}} \mathrm{tag}_\Gamma P$ *is a derivable tagged typing judgement.*

Proof A straightforward induction on the derivation of $\Gamma \vdash P$. □

Theorem 2 *If* $\Gamma \vdash P$ *is derivable and* $\Gamma \models_{\mathcal{T}} E$ *is derivable and* $P = \mathrm{Erase}(E)$ *and* $P \longrightarrow^* Q$ *then there exists* $\Delta \models_{\mathcal{T}} F$ *such that* $E \longrightarrow^* F$ *and* $Q = \mathrm{Erase}(F)$.

Proof By breaking the sequence of reductions into individual steps, and showing that the result holds for each step; the latter fact can be proved by induction on the derivation of the reduction step. □

The Tagged Subject Reduction Theorem, Lemma 10 and Theorem 2 imply that any sequence of reductions from a well-typed untagged process can be mirrored by a sequence of reductions from a well-typed tagged process. The final theorem establishes that well-typed tagged processes do not contain any immediate possibilities for incorrect communication. It follows easily from the tagged typing rules; most of the work in proving type soundness is concentrated into the proof of the Tagged Subject Reduction Theorem. Each case of the conclusion shows that whenever a tagged process appears to contain a potential reduction, the preconditions for the relevant tagged reduction rule are satisfied and the reduction can safely be carried out.

Theorem 3 (Run-Time Safety) *If* $\Gamma \models_{\mathcal{T}} P$, $P \equiv (\nu \tilde{w} : \tilde{W})(Q \mid R)$, $Q \xrightarrow[\mathcal{T}]{\alpha,l} Q'$ *and* Γ *is*

balanced with respect to α *(if* $\alpha \neq \tau$*), then*

i)If $Q \equiv x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[z]\,.\,Q_2$ *where* x *is not a session channel, then*

For all $z_i \in \tilde{z}$, $\Gamma \vDash_{\mp} z_i : T_i$ *or* $(\nu z_i : T_i) \in (\nu \tilde{w} : \tilde{W})$

and

$\Gamma \vDash_{\mp} x : \widehat{\ }[\tilde{T}]$ *or* $(\nu x : \widehat{\ }[\tilde{T}]) \in (\nu \tilde{w} : \tilde{W})$

ii)If $Q \equiv x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2$ *where* x *is a session channel, then*

$p = \overline{q}$

and

For all $z_i \in \tilde{z}$, $\Gamma \vDash_{\mp} z_i : T_i$ *or* $(\nu z_i : T_i) \in (\nu \tilde{w} : \tilde{W})$

and

$\Gamma \vDash_{\mp} x^p : ?[\tilde{T}]\,.\,S, x^q : ![\tilde{T}]\,.\,\overline{S}$ *or* $(\nu x^{\pm} : ?[\tilde{T}]\,.\,S) \in (\nu \tilde{w} : \tilde{W})$

or $(\nu x^{\pm} : ![\tilde{T}]\,.\,\overline{S}) \in (\nu \tilde{w} : \tilde{W})$

iii)If $Q \equiv x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i\,.\,Q_2$*, then*

$p = \overline{q}$

and

$l_i \in \{l_1, \ldots, l_n\}$

and

$\Gamma \vDash_{\mp} x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^q : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle$

or $(\nu x^{\pm} : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle) \in (\nu \tilde{w} : \tilde{W})$

or $(\nu x^{\pm} : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle) \in (\nu \tilde{w} : \tilde{W})$

iv)If $Q \equiv$ $\mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2$*, then*

$\Gamma \vDash_{\mp} x : \mathsf{bool}$ *or* $(\nu x : \mathsf{bool}) \in (\nu \tilde{w} : \tilde{W})$

If we take a well-typed untagged process and convert it into a tagged process, no reduction sequence can lead to a type error. Because every reduction sequence from a well-typed untagged process can be matched by a reduction sequence from a well-typed tagged process, we conclude that no type errors can result from executing a well-typed untagged process.

5 The POP3 Protocol

As a more substantial example, we will now use our type system to specify the POP3 protocol [16]. This protocol is typically used by electronic mail software to download new messages from a remote mailbox, so that they can be read and processed locally; it does not deal with sending messages or routing messages through a network. A POP3 server requires a client to authenticate itself by means of the user and pass commands. A client may then use commands such as stat to obtain the status of the mailbox, retr to retrieve a particular message, and quit to terminate the session. Some of these commands require additional information to be sent, for example the number of a message. We have omitted several POP3 commands from our description, but it is straightforward to fill in the missing ones.

To specify the behaviour of a channel which can be used for a POP3 session, we use the type definitions in Figure 15: A describes interactions with the authentication state, and T describes interactions with the transaction state. These definitions are for the server side of the channel, and we assume that there is a ground type str of strings. These definitions illustrate the complex structure possible for session types, and show the use of recursive types to describe repetitive session behaviour. The server both offers and makes choices, in contrast to the example in Section 1. After receiving a command (a label) from the client, the server can respond with either ok or error (except for the quit command, which always succeeds and does not allow an error response). The client implements an interaction of type $\overline{A}$, and therefore must offer a choice between ok and error when waiting for a response to a command. In the published description of the protocol, ok and error responses are simply strings prefixed with +OK or -ERR. This does not allow us to replace the corresponding $\oplus$ by ![str] in the above definitions because the different continuation types after ok and error are essential for an accurate description of the protocol's behaviour. We specify a string message as well, because a POP3 server is allowed to provide extra information such as an error code.

As in Section 1 we could implement a process POP3body such that $x^+ : A \vdash \mathsf{POP3body}$. Defining $\mathsf{POP3} = \mathsf{port}?[x^+ : A].\,\mathsf{POP3body}$ gives $\mathsf{port} : \widehat{\ }[A] \vdash \mathsf{POP3}$, which can be published as the specification of the server and its protocol.

The POP3 protocol permits an alternative authentication scheme, accessed by the apop command, in which the client sends a mailbox name and an authenticating string simultaneously. This option does not have to be provided, but a server which does implement it requires a channel of type B, where B is obtained from A by adding a third option to the

$$
\begin{array}{rcl}
A & = & \mu X.\&\langle\ \mathsf{quit}:\oplus\langle\mathsf{ok}:![\mathsf{str}]\,.\,\mathsf{end}\rangle, \\
& & \quad \mathsf{user}:?[\mathsf{str}]\,.\,\oplus\langle\ \mathsf{error}:![\mathsf{str}]\,.\,X, \\
& & \qquad\qquad \mathsf{ok}:![\mathsf{str}]\,.\,\&\langle\ \mathsf{quit}:\oplus\langle\mathsf{ok}:![\mathsf{str}]\,.\,\mathsf{end}\rangle, \\
& & \qquad\qquad\qquad\qquad \mathsf{pass}:?[\mathsf{str}]\,.\,\oplus\langle\ \mathsf{error}:![\mathsf{str}]\,.\,X, \\
& & \qquad\qquad\qquad\qquad\qquad\qquad \mathsf{ok}:![\mathsf{str}]\,.\,T\rangle\rangle\rangle\rangle \\
T & = & \mu X.\&\langle\ \mathsf{stat}:\oplus\langle\mathsf{ok}:![\mathsf{int},\mathsf{int}]\,.\,X\rangle, \\
& & \quad \mathsf{retr}:?[\mathsf{int}]\,.\,\oplus\langle\ \mathsf{ok}:![\mathsf{str}]\,.\,![\mathsf{str}]\,.\,X, \\
& & \qquad\qquad \mathsf{error}:![\mathsf{str}]\,.\,X\rangle, \\
& & \quad \mathsf{quit}:\oplus\langle\mathsf{ok}:![\mathsf{str}]\,.\,\mathsf{end}\rangle\rangle
\end{array}
$$

Figure 15: Types for the POP3 protocol

first choice:

$$\mathsf{apop} : ?[\mathsf{str}, \mathsf{str}] \,.\, \oplus\langle \mathsf{error} : ![\mathsf{str}] \,.\, X, \mathsf{ok} : ![\mathsf{str}] \,.\, T\rangle$$

A server which implements the apop command can be typed as follows: $\mathsf{port} : \hat{}[B] \vdash \mathsf{newPOP3}$. Now suppose that client is a POP3 client which does not know about the apop command. As before, we have $\mathsf{client} = (\nu x^{\pm} : A)\mathsf{port}\,![x^{+}] \,.\, \mathsf{clientbody}$ where $x^{-} : \overline{A} \vdash \mathsf{clientbody}$. This gives $\mathsf{port} : \hat{}[A] \vdash \mathsf{client}$. The following derivation shows that client can also be typed in the environment $\mathsf{port} : \hat{}[B]$, and can therefore be put in parallel with newPOP3. The key fact is that $A \leqslant B$, because the top-level options provided by A are a subset of those provided by B.

$$\dfrac{\dfrac{x^{-} : \overline{A} \vdash \mathsf{clientbody} \quad \hat{}[B] \leqslant ![A]}{\mathsf{port} : \hat{}[B], x^{+} : A, x^{-} : \overline{A} \vdash \mathsf{port}\,![x^{+}] \,.\, \mathsf{clientbody}}\ \text{T-Out}}{\mathsf{port} : \hat{}[B^{1}] \vdash (\nu x^{\pm} : A)\mathsf{port}\,![x^{+}] \,.\, \mathsf{clientbody}}\ \text{T-New}$$

6 Conclusions

We have defined a language whose type system incorporates session types, as suggested by Honda et al. [8, 25], and subtyping, based on Pierce and Sangiorgi's work [18, 19] and extended to session types. Session channels must be controlled linearly in order to guarantee that messages go to the correct destinations, and we have adapted the work of Kobayashi et al. [12, 13] for this purpose. Our language differs minimally from the π-calculus, the only additions being primitives for offering and making labelled choices. Unlike Honda et al. we do not introduce special syntax for establishing and manipulating session channels; everything is taken care of by the typing rules. We have advocated using a session type as part of the published specification of a server's protocol, so that static type-checking can be used to verify that client implementations behave correctly. Using the POP3 protocol as an example, we have shown that subtyping increases the utility of this idea: if upgrading a server causes its protocol to have a session type which is a supertype of its original session type, then existing client implementations are still type-correct with respect to the new server.

Session types have some similarities to the types for active objects studied by Nierstrasz [17] and Puntigam [22, 23]. Both incorporate the idea of a type which specifies a sequence of interactions. The main difference seems to be that in the case of active objects, types can specify interdependencies between interactions on different channels. However, the underlying framework (concurrent objects with method invocation, instead of channel-based communication between processes) is rather different, and we have not yet made a detailed comparison of the two systems.

Acknowledgements

This research was funded by the EPSRC project "Novel Type Systems for Concurrent Programming Languages" (GR/L75177,GR/N39494). Paul Taylor's proof tree macros were used in the production of this report.

References

[1] G. Boudol. Asynchrony and the π-calculus (note). Rapporte de Recherche 1702, INRIA Sofia-Antipolis, May 1992.

[2] S. J. Gay. A sort inference algorithm for the polyadic π-calculus. In *Proceedings, 20th ACM Symposium on Principles of Programming Languages*. ACM Press, 1993.

[3] S. J. Gay and M. J. Hole. Types and subtypes for client-server interactions. In S. D. Swierstra, editor, *ESOP'99: Proceedings of the European Symposium on Programming Languages and Systems*, number 1576 in Lecture Notes in Computer Science, pages 74–90. Springer-Verlag, 1999.

[4] S. J. Gay and M. J. Hole. Types for correct communication in client-server systems. Technical Report CSD-TR-00-07, Department of Computer Science, Royal Holloway, University of London, 2000.

[5] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[6] K. Honda. Types for dyadic interaction. In *CONCUR 93*, Lecture Notes in Computer Science. Springer-Verlag, 1993.

[7] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *Proceedings of the European Conference on Object-Oriented Programming*, LNCS. Springer-Verlag, 1994.

[8] K. Honda, V. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *Proceedings of the European Symposium on Programming*, Lecture Notes in Computer Science. Springer-Verlag, 1998.

[9] K. Honda and N. Yoshida. Combinatory representation of mobile processes. In *Proceedings, 21st ACM Symposium on Principles of Programming Languages*, 1994.

[10] N. Kobayashi. A partially deadlock-free typed process calculus. In *Proceedings, Twelfth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1997.

[11] N. Kobayashi. A partially deadlock-free typed process calculus. *ACM Transactions on Programming Languages and Systems*, 20:436–482, 1998.

[12] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the pi-calculus. In *Proceedings, 23rd ACM Symposium on Principles of Programming Languages*, 1996.

[13] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the Pi-Calculus. *ACM Transactions on Programming Languages and Systems*, 21(5):914–947, September 1999.

[14] R. Milner. The polyadic π-calculus: A tutorial. Technical Report 91-180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1991.

[15] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–77, September 1992.

[16] J. Myers and M. Rose. Post office protocol version 3, May 1996. Internet Standards RFC1939.

[17] O. Nierstrasz. Regular types for active objects. *ACM Sigplan Notices*, 28(10):1–15, October 1993.

[18] B. C. Pierce and D. Sangiorgi. Types and subtypes for mobile processes. In *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1993.

[19] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5), 1996.

[20] B. C. Pierce and D. N. Turner. Pict: A programming language based on the pi-calculus. Technical Report CSCI 476, Computer Science Department, Indiana University, 1997. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, MIT Press, 2000.

[21] B. C. Pierce and D. N. Turner. Local type inference. In *Proceedings, 25th ACM Symposium on Principles of Programming Languages*, 1998.

[22] F. Puntigam. Synchronization expressed in types of communication channels. In *Proceedings of the European Conference on Parallel Processing (Euro-Par'96)*, volume 1123 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[23] F. Puntigam. Coordination requirements expressed in types for active objects. In M. Aksit and S. Matsuoka, editors, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'97)*, volume 1241 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[24] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[25] K. Takeuchi, K. Honda, and M. Kubo. An interaction-based language and its typing system. In *Proceedings of the 6th European Conference on Parallel Languages and Architectures*, number 817 in Lecture Notes in Computer Science. Springer-Verlag, 1994.

[26] D. N. Turner. *The Polymorphic Pi-Calculus: Theory and Implementation*. PhD thesis, University of Edinburgh, 1996.

A Subject Reduction Theorem

It is common in the following proofs that it is necessary to deconstruct a process or typing derivation using a certain set of rules, perform some operation (such as applying the induction hypothesis), and reconstruct what we need using the same set of rules. As the rules are used in reverse during the deconstruction stage, we write the rule applications upside down in order to improve the clarity of the proofs.

A.1 Proof of Lemma 5 [Subtype Transitivity]

$$\text{If } W \leqslant U \text{ and } U \leqslant T \text{ then } W \leqslant T$$

Proof By structural induction on the derivations of $W \leqslant U$ and $U \leqslant T$. Two cases are given here as an outline of how the proof works.

AS-InOut/AS-Out

The subtypes are:

$$\hat{}[\tilde{T}] \leqslant \hat{}[\tilde{U}] \text{ and } \hat{}[\tilde{U}] \leqslant ![\tilde{V}]$$

We have :

$$\Sigma \vdash \hat{}[\tilde{T}] \leqslant \hat{}[\tilde{U}] \qquad \Sigma \vdash \hat{}[\tilde{U}] \leqslant ![\tilde{V}]$$

We need :

$$\Sigma \vdash \hat{}[\tilde{T}] \leqslant ![\tilde{V}]$$

$$\frac{\Sigma \vdash \hat{}[\tilde{T}] \leqslant \hat{}[\tilde{U}]}{\forall i \in \{1,\ldots,n\}.(\Sigma \vdash T_i \leqslant U_i \text{ and } \Sigma \vdash U_i \leqslant T_i)}\ \text{AS-InOut}$$

$$\frac{\Sigma \vdash \hat{}[\tilde{U}] \leqslant ![\tilde{V}]}{\forall i \in \{1,\ldots,n\}.\Sigma \vdash V_i \leqslant U_i}\ \text{AS-Out}$$

By the induction hypothesis, $\forall i \in \{1,\ldots,n\}.\Sigma \vdash V_i \leqslant T_i$

$$\frac{\forall i \in \{1,\ldots,n\}.\Sigma \vdash V_i \leqslant T_i}{\Sigma \vdash \hat{}[\tilde{T}] \leqslant ![\tilde{V}]}\ \text{AS-Out}$$

So, we have :

$$\Sigma \vdash \hat{}[\tilde{T}] \leqslant ![\tilde{V}]$$

which is what we need.

AS-Branch

The subtypes are:

$$\&\langle l_1 : S_1, \ldots, l_l : S_l\rangle \leqslant \&\langle l_1 : S_1', \ldots, l_m : S_m'\rangle$$
$$\text{and}$$
$$\&\langle l_1 : S_1', \ldots, l_m : S_m'\rangle \leqslant \&\langle l_1 : S_1'', \ldots, l_n : S_n''\rangle$$

We have :

$$\Sigma \vdash \&\langle l_1 : S_1, \ldots, l_l : S_l \rangle \leqslant \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}' \rangle$$
$$\Sigma \vdash \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}' \rangle \leqslant \&\langle l_1 : {S_1}'', \ldots, l_n : {S_n}'' \rangle$$

We need :

$$\Sigma \vdash \&\langle l_1 : S_1, \ldots, l_l : S_l \rangle \leqslant \&\langle l_1 : {S_1}'', \ldots, l_n : {S_n}'' \rangle$$

$$\frac{\Sigma \vdash \&\langle l_1 : S_1, \ldots, l_l : S_l \rangle \leqslant \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}' \rangle}{l \leq m \qquad \forall i \in \{1, \ldots, l\}.\Sigma \vdash S_i \leqslant {S_i}'} \text{AS-Branch}$$

$$\frac{\Sigma \vdash \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}' \rangle \leqslant \&\langle l_1 : {S_1}'', \ldots, l_n : {S_n}'' \rangle}{m \leq n \qquad \forall i \in \{1, \ldots, m\}.\Sigma \vdash {S_i}' \leqslant {S_i}''} \text{AS-Branch}$$

By the induction hypothesis, $\forall i \in \{1, \ldots, l\}.\Sigma \vdash S_i \leqslant {S_i}''$

$$\frac{l \leq n \qquad \forall i \in \{1, \ldots, l\}.\Sigma \vdash S_i \leqslant {S_i}''}{\Sigma \vdash \&\langle l_1 : S_1, \ldots, l_l : S_l \rangle \leqslant \&\langle l_1 : {S_1}'', \ldots, l_n : {S_n}'' \rangle} \text{AS-Branch}$$

So, we have :

$$\Sigma \vdash \&\langle l_1 : S_1, \ldots, l_l : S_l \rangle \leqslant \&\langle l_1 : {S_1}'', \ldots, l_n : {S_n}'' \rangle$$

which is what we need.

□

A.2 Proof of Lemma 8 [Substitution]

$$\text{If } \Gamma, x : T \vdash P \text{ and } U \leqslant T \text{ then } \Gamma + y : U \vdash P\{y/x\}$$

Proof By structural induction on the derivation of $\Gamma, x : T \vdash P$

TT-Nil

The substitution is :

$$\mathbf{0}\{y/x\}$$

We have :

$$\Gamma, x : T \vdash \mathbf{0} \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash \mathbf{0}$$

$$\frac{\Gamma, x : T \vdash \mathbf{0}}{\Gamma, x : T \text{ completed}} \text{TT-Nil}$$

We know (Lemma 2 [Completion]) : $\Gamma + y : U$ completed

$$\frac{\Gamma + y : U \text{ completed}}{\Gamma + y : U \vdash \mathbf{0}} \text{TT-Nil}$$

So, we have :

$$\Gamma + y : U \vdash \mathbf{0}$$

which is what we need.

TT-Par

The substitution is :

$$(P \mid Q)\{y/x\}$$

We have :

$$\Gamma, x : T \vdash P \mid Q \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (P \mid Q)\{y/x\}$$

Case 1 : x is free in P, x is not free in Q

$$\frac{\Gamma, x : T \vdash P \mid Q}{\Gamma_1, x : T \vdash P \qquad \Gamma_2 \vdash Q}\ \text{TT-Par}$$

where $\Gamma_1, x : T + \Gamma_2 = \Gamma, x : T$

$\Gamma_1, x : T \vdash P$ and $U \leqslant T$, so, by the induction hypothesis, $\Gamma_1 + y : U \vdash P\{y/x\}$.

1.1 : $x \notin \Gamma_2$

$$\frac{\Gamma_1 + y : U \vdash P\{y/x\} \qquad \Gamma_2 \vdash Q}{\Gamma_1 + y : U + \Gamma_2 \vdash P\{y/x\} \mid Q}\ \text{TT-Par}$$

As $x \notin \Gamma_2$, we know $\Gamma_1 + \Gamma_2 = \Gamma$.

x is not free in Q, so $P\{y/x\} \mid Q \equiv (P \mid Q)\{y/x\}$.

So, we have :

$$\Gamma + y : U \vdash (P \mid Q)\{y/x\}$$

which is what we need.

1.2 : $x \in \Gamma_2$

Let $\Gamma_2 = \Gamma_3, x : T$. We know T is not a session type and that $\Gamma_1 + \Gamma_3 = \Gamma$.

As $\Gamma_3, x : T \vdash Q$ and x is not free in Q, we know $\Gamma_3 \vdash Q$.

$$\frac{\Gamma_1 + y : U \vdash P\{y/x\} \qquad \Gamma_3 \vdash Q}{\Gamma_1 + y : U + \Gamma_3 \vdash P\{y/x\} \mid Q}$$

We know $\Gamma_1 + \Gamma_3 = \Gamma$, and x is not free in Q, so $P\{y/x\} \mid Q \equiv (P \mid Q)\{y/x\}$.

So, we have :

$$\Gamma + y : U \vdash (P \mid Q)\{y/x\}$$

which is what we need.

Case 2 : x is not free in P, x is free in Q

This case is the opposite of case one and is proved in the same way. *Case 3 : x is free in P, x is free in Q*

$$\frac{\Gamma, x : T \vdash P \mid Q}{\Gamma_1, x : T \vdash P \qquad \Gamma_2, x : T \vdash Q}\ \text{TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$ and T is not a session type

By the induction hypothesis, $\Gamma_1 + y : U \vdash P\{y/x\}$

By the induction hypothesis, $\Gamma_2 + y : U \vdash Q\{y/x\}$

$$\frac{\Gamma_1 + y : U \vdash P\{y/x\} \qquad \Gamma_2 + y : U \vdash Q\{y/x\}}{\Gamma_1 + y : U + \Gamma_2 + y : U \vdash (P \mid Q)\{y/x\}}\ \text{TT-Par}$$

We know (Lemma 1 [Non-Session Subtypes]) : U is not a session type.

From the definition of $+$:$\Gamma_1 + y : U + \Gamma_2 + y : U = \Gamma + y : U$

So, we have :

$$\Gamma + y : U \vdash (P \mid Q)\{y/x\}$$

which is what we need.

TT-New

The substitution is :

$$((\nu z : Z)P)\{y/x\}$$

We have :

$$\Gamma, x : T \vdash (\nu z : Z)P \qquad U \leqslant T$$

We need :

$$\Gamma + y : T \vdash ((\nu z : Z)P)\{y/x\}$$

$$\frac{\Gamma, x : T \vdash (\nu z : Z)P}{\Gamma, x : T, z : Z \vdash P}\ \text{TT-New}$$

By the induction hypothesis, $(\Gamma, z : Z) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) : $(\Gamma, z : Z) + y : U = (\Gamma + y : U), z : Z$

$$\frac{(\Gamma + y : U), z : Z \vdash P\{y/x\}}{\Gamma + y : U \vdash (\nu z : Z)(P\{y/x\})}\ \text{TT-New}$$

So, we have :

$$\Gamma + y : U \vdash ((\nu z : Z)P)\{y/x\}$$

which is what we need.

TT-NewSeq

The substitution is :

$$((\nu z^{\pm} : Z)P)\{y/x\}$$

We have :

$$\Gamma, x : T \vdash (\nu z^{\pm} : Z)P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash ((\nu z^{\pm} : Z)P)\{y/x\}$$

$$\frac{\Gamma, x : T \vdash (\nu z^{\pm} : Z)P}{\Gamma, x : T, z^{+} : Z, z^{-} : \overline{Z} \vdash P}\ \text{TT-NewSeq}$$

By the induction hypothesis, $(\Gamma, z^{+} : Z, z^{-} : \overline{Z}) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) :

$$(\Gamma, z^{+} : Z, z^{-} : \overline{Z}) + y : U = (\Gamma + y : U), z^{+} : Z, z^{-} : \overline{Z}$$

$$\frac{(\Gamma + y : U), z^{+} : Z, z^{-} : \overline{Z} \vdash P\{y/x\}}{\Gamma + y : U \vdash (\nu z^{\pm} : Z)(P\{y/x\})}$$

So, we have :

$$\Gamma + y : U \vdash ((\nu z^{\pm} : Z)P)\{y/x\}$$

which is what we need.

TT-Rep

The substitution is :

$$!P\{y/x\}$$

We have :

$$\Gamma, x : T \vdash !P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash !P\{y/x\}$$

$$\frac{\Gamma, x : T \vdash !P}{\Gamma, x : T \vdash P \qquad \Gamma, x : T \text{ unlimited}} \text{ TT-Rep}$$

By the induction hypothesis, $\Gamma + y : U \vdash P\{y/x\}$.

We know (Lemma 3 [Unlimitedness]) : $\Gamma + y : U$ unlimited

$$\frac{\Gamma + y : U \vdash P\{y/x\} \qquad \Gamma + y : U \text{ unlimited}}{\Gamma + y : U \vdash !P\{y/x\}} \text{ TT-Rep}$$

So, we have :

$$\Gamma + y : U \vdash !P\{y/x\}$$

which is what we need.

TT-Cond

The substitution is :

$$(\mathsf{if}\ z : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q)\{y/x\}$$

We have :

$$\Gamma, x : T \vdash \mathsf{if}\ z : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash \mathsf{if}\ z\{y/x\} : \mathsf{bool}\ \mathsf{then}\ P\{y/x\}\ \mathsf{else}\ Q\{y/x\}$$

$$\frac{\Gamma, x : T \vdash \mathsf{if}\ z : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q}{\Gamma, x : T \vdash z \leqslant \mathsf{bool} \qquad \Gamma, x : T \vdash P \qquad \Gamma, x : T \vdash Q} \text{ TT-Cond}$$

By the induction hypothesis, $\Gamma + y : U \vdash P\{y/x\}$ and $\Gamma + y : U \vdash Q\{y/x\}$.

We know (Lemma 6 [Substitution Typing]) : $\Gamma + y : U \vdash z\{y/x\} \leqslant \mathsf{bool}$.

$$\frac{\Gamma + y : U \vdash z\{y/x\} : \mathsf{bool} \qquad \Gamma + y : U \vdash P\{y/x\} \qquad \Gamma + y : U \vdash Q\{y/x\}}{\Gamma + y : U \vdash \mathsf{if}\ z\{y/x\} : \mathsf{bool}\ \mathsf{then}\ P\{y/x\}\ \mathsf{else}\ Q\{y/x\}} \text{ TT-Cond}$$

So, we have :

$$\Gamma + y : U \vdash \mathsf{if}\ z\{y/x\}\ \mathsf{then}\ P\{y/x\}\ \mathsf{else}\ Q\{y/x\}$$

which is what we need.

TT-Out

The substitution is :

$$((u : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P)\{y/x\}$$

Case 1 : $x \notin \tilde{v}$, $x \neq u$

We have :

$$\Gamma, x : T \vdash (u : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (u : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

Let $\Gamma, x : T = (\Gamma', x : T) + \tilde{v} : \tilde{X}$ where $\Gamma' + \tilde{v} : \tilde{X} = \Gamma$

$$\frac{(\Gamma', x : T) + \tilde{v} : \tilde{X} \vdash (u : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P}{\Gamma', x : T \vdash P \qquad \Gamma', x : T \vdash u \leqslant V \leqslant\, !\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}\ \text{TT-Out}$$

By the induction hypothesis, $\Gamma' + y : U \vdash P\{y/x\}$.

We know (Lemma 6 [Substitution Typing]) : $\Gamma' + y : U \vdash u \leqslant V \leqslant\, !\,[\tilde{W}]$.

$$\frac{\Gamma' + y : U \vdash P\{y/x\} \qquad \Gamma' + y : U \vdash u \leqslant V \leqslant\, !\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}{\Gamma' + y : U + \tilde{v} : \tilde{X} \vdash (u : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}}\ \text{TT-Out}$$

But, $\Gamma' + \tilde{v} : \tilde{X} = \Gamma$.

So, we have :

$$\Gamma + y : U \vdash (u : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

which is what we need.

Case 2 : $x \notin \tilde{v}$, $x = u$

We have :

$$\Gamma, x : T \vdash (x : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (y : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

Let $\Gamma, x : T = (\Gamma', x : T) + \tilde{v} : \tilde{X}$ where $\Gamma' + \tilde{v} : \tilde{X} = \Gamma$

$$\frac{(\Gamma', x : T) + \tilde{v} : \tilde{X} \vdash (x : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P}{\Gamma', x : T \vdash P \qquad \Gamma', x : T \vdash x \leqslant V \leqslant\, !\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}\ \text{TT-Out}$$

By the induction hypothesis, $\Gamma' + y : U \vdash P\{y/x\}$.

We know (Lemma 6 [Substitution Typing]) : $\Gamma' + y : U \vdash y \leqslant V \leqslant\, !\,[\tilde{W}]$.

$$\frac{\Gamma' + y : U \vdash P\{y/x\} \qquad \Gamma' + y : U \vdash y \leqslant V \leqslant\, !\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}{\Gamma' + y : U + \tilde{v} : \tilde{X} \vdash (y : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}}\ \text{TT-Out}$$

But, $\Gamma' + \tilde{v} : \tilde{X} = \Gamma$.

So, we have :

$$\Gamma + y : U \vdash (y : V)\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

which is what we need.

Case 3 : $x \in \tilde{v}$, say $x = v_1$, $x \neq u$

We have :

$$\Gamma, x : T \vdash (u : V)\,!\,[x : W_1, v_2 : W_2, \ldots, v_n : W_n]\,.\,P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (u : V) \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

Let $\Gamma, x : T = \Gamma' + (x : X_1, v_2 : X_2, \ldots, v_n : X_n)$ where $T \leqslant X_1$.

Case 3.1 : $x \in \Gamma'$

Let $\Gamma' = \Gamma'', x : T$ where $\Gamma'' + (v_2 : X_2, \ldots, v_n : X_n) = \Gamma$.

$$\frac{\begin{array}{c}(\Gamma'', x : T) + (x : X_1, v_2 : X_2, \ldots, v_n : X_n) \vdash \\ (u : V) \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\end{array}}{\Gamma'', x : T \vdash P \qquad \Gamma'', x : T \vdash u \leqslant V \leqslant \,!\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}\ \text{TT-Out}$$

By the induction hypothesis, $\Gamma'' + y : U \vdash P\{y/x\}$.

We know (Lemma 6 [Substitution Typing]) : $\Gamma'' + y : U \vdash u \leqslant V \leqslant \,!\tilde{W}$.

$$\frac{\Gamma'' + y : U \vdash P\{y/x\} \qquad \Gamma'' + y : U \vdash u \leqslant V \leqslant \,!\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}{\begin{array}{c}\Gamma'' + y : U + (y : X_1, v_2 : X_2, \ldots, v_n : X_n) \vdash \\ (u : V) \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}\end{array}}\ \text{TT-Out}$$

But, $\Gamma'' + (v_2 : X_2, \ldots, v_n : X_n) = \Gamma$ and $U \leqslant X_1$.

So, we have :

$$\Gamma + y : U \vdash (u : V) \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

which is what we need.

Case 3.2 : $x \notin \Gamma'$

$$\frac{\begin{array}{c}\Gamma' + (x : X_1, v_2 : X_2, \ldots, v_n : X_n) \vdash \\ (u : V) \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\end{array}}{\Gamma' \vdash P \qquad \Gamma' \vdash u \leqslant V \leqslant \,!\,[\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}\ \text{TT-Out}$$

We know (Lemma 7 [No Substitution]) : $\Gamma' \vdash P\{y/x\}$

As $x \notin \Gamma'$, we know $\Gamma' + (v_2 : X_2, \ldots, v_n : X_n) = \Gamma$ and $T = X_1$.

We know (Lemma 5 [Subtype Transitivity]) : $U, X_2, \ldots, X_n \leqslant \tilde{W}$.

$$\frac{\Gamma' \vdash P\{y/x\} \qquad \Gamma' \vdash u \leqslant V \leqslant \,!\,[\tilde{W}] \qquad [U, X_2, \ldots, X_n] \leqslant \tilde{W}}{\begin{array}{c}\Gamma' + (y : U, v_2 : X_2, \ldots, v_n : X_n) \vdash \\ (u : V) \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}\end{array}}\ \text{TT-Out}$$

But, $\Gamma' + (v_2 : X_2, \ldots, v_n : X_n) = \Gamma$.

So, we have :

$$\Gamma + y : U \vdash (u : V) \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

which is what we need.

Case 4 : $x \in \tilde{v}$, say $x = v_1$, $x = u$
We have :

$$\Gamma, x : T \vdash (x : V)\,![x : W_1, v_2 : W_2, \ldots, v_n : W_n]\,.\,P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (y : V)\,![y : W_1, v_2 : W_2, \ldots, v_n : W_n]\,.\,P\{y/x\}$$

Let $\Gamma, x : T = (\Gamma', x : T) + (x : X_1, v_2 : X_2, \ldots, v_n : X_n)$
where $\Gamma = \Gamma' + (v_2 : X_2, \ldots, v_n : X_n)$
and $T \leqslant X_1$.

$$\frac{\begin{array}{c}(\Gamma', x : T) + (x : X_1, v_2 : X_2, \ldots, v_n : X_n) \vdash \\ (x : V)\,![x : W_1, v_2 : W_2, \ldots, v_n : W_n]\,.\,P\end{array}}{\Gamma', x : T \vdash P \qquad \Gamma', x : T \vdash x \leqslant V \leqslant\, ![\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}\ \text{TT-Out}$$

By the induction hypothesis, $\Gamma' + y : U \vdash P\{y/x\}$.
We know (Lemma 6 [Substitution Typing]) : $\Gamma' + y : U \vdash y \leqslant V \leqslant\, ![\tilde{W}]$

$$\frac{\Gamma' + y : U \vdash P\{y/x\} \qquad \Gamma' + y : U \vdash y \leqslant V \leqslant\, ![\tilde{W}] \qquad \tilde{X} \leqslant \tilde{W}}{\begin{array}{c}(\Gamma' + y : U) + (y : X_1, v_2 : X_2, \ldots, v_n : X_n) \vdash \\ (y : V)\,![y : W_1, v_2 : W_2, \ldots, v_n : W_n]\,.\,P\{y/x\}\end{array}}\ \text{TT-Out}$$

But, $\Gamma = \Gamma' + (v_2 : X_2, \ldots, v_n : X_n)$ and $U \leqslant T \leqslant X_1$.

So, we have :

$$\Gamma + y : U \vdash (y : V)\,![y : W_1, v_2 : W_2, \ldots, v_n : W_n]\,.\,P\{y/x\}$$

which is what we need.

TT-In
The substitution is :

$$((u : V)\,?\,[\tilde{v} : \tilde{V}]\,.\,P)\{y/x\}$$

Case 1 : $x \neq u$
We have :

$$\Gamma, x : T \vdash (u : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (u : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

$$\frac{\Gamma, x : T \vdash (u : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P}{\Gamma, x : T, \tilde{v} : \tilde{W} \vdash P \qquad \Gamma, x : T \vdash u \leqslant V \leqslant\, ?[\tilde{W}]}\ \text{TT-In}$$

By the induction hypothesis, $(\Gamma, \tilde{v} : \tilde{W}) + y : U \vdash P\{y/x\}$.
We know (Lemma 4 [Type Addition]) : $(\Gamma + y : U), \tilde{v} : \tilde{W} \vdash P\{y/x\}$.
We know (Lemma 6 [Substitution Typing]) : $\Gamma + y : U \vdash u \leqslant V \leqslant\, ?[\tilde{W}]$.

$$\frac{(\Gamma + y : U), \tilde{v} : \tilde{W} \vdash P\{y/x\} \qquad \Gamma + y : U \vdash u \leqslant V \leqslant\, ?[\tilde{W}]}{\Gamma + y : U \vdash (u : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}}\ \text{TT-In}$$

So, we have :

$$\Gamma + y : U \vdash (u : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

which is what we need.

Case 2 : $x = u$
We have :

$$\Gamma, x : T \vdash (x : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P \qquad U \leqslant T$$

We need :

$$\Gamma + y : U \vdash (y : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

$$\frac{\Gamma, x : T \vdash (x : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P}{\Gamma, x : T, \tilde{v} : \tilde{W} \vdash P \qquad \Gamma, x : T \vdash x \leqslant V \leqslant ?[\tilde{W}]}\ \text{TT-In}$$

By the induction hypothesis, $(\Gamma, \tilde{v} : \tilde{W}) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) : $(\Gamma + y : U), \tilde{v} : \tilde{W} \vdash P\{y/x\}$.

We know (Lemma 6 [Substitution Typing]) : $\Gamma + y : U \vdash y \leqslant V \leqslant ?[\tilde{W}]$.

$$\frac{(\Gamma + y : U), \tilde{v} : \tilde{W} \vdash P\{y/x\} \qquad \Gamma + y : U \vdash y \leqslant V \leqslant ?[\tilde{W}]}{\Gamma + y : U \vdash (y : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}}\ \text{TT-In}$$

So, we have :

$$\Gamma + y : U \vdash (y : V)\,?\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

which is what we need.

TT-OutSeq

The substitution is :

$$((u^p : ![\tilde{Z}]\,.\,S')\,!\,[\tilde{v} : \tilde{W}]\,.\,P)\{y/x\}$$

Note that for the TT-OutSeq case, $u \notin \tilde{v}$. This is because the typing rules do not allow a session channel to be sent on itself.
Case 1 : $x \notin \tilde{v}$, $x \neq u$
We have :

$$((\Gamma, u^p : ![\tilde{Y}]\,.\,S) + \tilde{v} : \tilde{X}), x : T \vdash (u^p : ![\tilde{Z}]\,.\,S')\,!\,[\tilde{v} : \tilde{W}]\,.\,P \qquad U \leqslant T$$

We need :

$$((\Gamma, u^p : ![\tilde{Y}]\,.\,S) + \tilde{v} : \tilde{X}) + y : U \vdash (u^p : ![\tilde{Z}]\,.\,S')\,!\,[\tilde{v} : \tilde{W}]\,.\,P\{y/x\}$$

We know (Lemma 4 [Type Addition]):

$((\Gamma, u^p : ![\tilde{Y}] \,.\, S), x : T) + \tilde{v} : \tilde{X} \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P$

$$\frac{((\Gamma, u^p : ![\tilde{Y}] \,.\, S), x : T) + \tilde{v} : \tilde{X} \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P}{\Gamma, u^p : S, x : T \vdash P \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}} \text{ TT-OutSeq}$$

By the induction hypothesis, $(\Gamma, u^p : S) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) :

$(\Gamma + y : U), u^p : S \vdash P\{y/x\}$.

$$\frac{(\Gamma + y : U), u^p : S \vdash P\{y/x\} \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}}{((\Gamma + y : U), u^p : ![\tilde{Y}] \,.\, S) + \tilde{v} : \tilde{X} \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P\{y/x\}} \text{ TT-OutSeq}$$

We know (Lemma 4 [Type Addition]) :

$((\Gamma + y : U), u^p : ![\tilde{Y}] \,.\, S) + \tilde{v} : \tilde{X} = ((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + \tilde{v} : \tilde{X}), y : U.$

So, we have :

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + \tilde{v} : \tilde{X}) + y : U \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P\{y/x\}$$

which is what we need. *Case 2 : $x \notin \tilde{v}$, $x^p = u$*
We have :

$$(\Gamma + \tilde{v} : \tilde{X}), x^p : ![\tilde{Y}] \,.\, S \vdash (x^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P \qquad ![\tilde{U}] \,.\, S'' \leqslant ![\tilde{Y}] \,.\, S$$

We need :

$$\Gamma + \tilde{v} : \tilde{X} + y^q : ![\tilde{U}] \,.\, S'' \vdash (y^q : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P\{y^q/x^p\}$$

We know (Lemma 4 [Type Addition]):

$(\Gamma, x^p : ![\tilde{Y}] \,.\, S) + \tilde{v} : \tilde{X} \vdash (x^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P$

$$\frac{(\Gamma, x^p : ![\tilde{Y}] \,.\, S) + \tilde{v} : \tilde{X} \vdash (x^p : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P}{\Gamma, x^p : S \vdash P \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}} \text{ TT-OutSeq}$$

$$\frac{![\tilde{U}] \,.\, S'' \leqslant ![\tilde{Y}] \,.\, S}{S'' \leqslant S \qquad \tilde{Y} \leqslant \tilde{U}} \text{ AS-OutSeq}$$

By the induction hypothesis, $\Gamma + y^q : S'' \vdash P\{y^q/x^p\}$.

We know from the definition of $+$:

$\Gamma, y^q : S'' \vdash P\{y^q/x^p\}$.

$$\frac{\Gamma, y^q : S'' \vdash P\{y^q/x^p\} \qquad S'' \leqslant S' \qquad \tilde{X} \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{U}}{(\Gamma, y^q : ![\tilde{U}] \,.\, S'') + \tilde{v} : \tilde{X} \vdash (y^q : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P\{y^q/x^p\}} \text{ TT-OutSeq}$$

We know from the definition of $+$: $(\Gamma, y^q : ![\tilde{U}] \,.\, S'') + \tilde{v} : \tilde{X} = \Gamma + \tilde{v} : \tilde{X} + y^q : ![\tilde{U}] \,.\, S''$.

So, we have :

$$\Gamma + \tilde{v} : \tilde{X} + y^q : ![\tilde{U}] \,.\, S'' \vdash (y^q : ![\tilde{Z}] \,.\, S') \,!\, [\tilde{v} : \tilde{W}] \,.\, P\{y^q/x^p\}$$

which is what we need. *Case 3 : $x \in \tilde{v}$ (say $x = v_1$), $x \neq u$*
We have :

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)), x : T \vdash \\ (u^p : ![\tilde{Z}] \,.\, S') \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P$$

and $U \leqslant T$.

We need :

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U \\ \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

Case 3.1 : $x \notin \Gamma$

We know from the definition of $+$:

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + x : T \vdash \\ (u^p : ![\tilde{Z}] \,.\, S') \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P.$$

$$\frac{((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + x : T \vdash \\ (u^p : ![\tilde{Z}] \,.\, S') \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P}{\Gamma, u^p : S \vdash P \qquad S \leqslant S' \qquad [T, X_1, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}} \text{ TT-OutSeq}$$

We know (Lemma 7 [No Substitution]) : $\Gamma, u^p : S \vdash P\{y/x\}$.

We know (Lemma 5 [Subtype Transitivity]) : $[U, X_2, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}$.

$$\frac{\Gamma, u^p : S \vdash P\{y/x\} \qquad S \leqslant S' \qquad [U, X_2, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}}{((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U \vdash \\ (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}} \text{ TT-OutSeq}$$

So, we have :

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U \vdash \\ (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

which is what we need.

Case 3.2 : $x \in \Gamma$

Let $((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)), x : T =$

$(\Gamma, x : T', u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) + x : X_1$ where $T' + X_1 = T$.

Case 3.2.1 : $T' \leqslant X_1$, $T = T'$

$$\frac{(\Gamma, x : T, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) + x : X_1 \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P}{\Gamma, x : T, u^p : S \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}} \text{ TT-OutSeq}$$

By the induction hypothesis, $(\Gamma, u^p : S) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) : $(\Gamma + y : U), u^p : S \vdash P\{y/x\}$.

We know (Lemma 5 [Subtype Transitivity]) : $[U, X_2, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}$.

$$\frac{(\Gamma + y : U), u^p : S \vdash P\{y/x\} \qquad S \leqslant S' \qquad [U, X_2, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}}{((\Gamma + y : U), u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}} \text{ TT-OutSeq}$$

We know (Lemma 4 [Type Addition])

$$((\Gamma + y : U), u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) = ((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U.$$

So, we have :

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

which is what we need.

Case 3.2.2 : $X_1 \leqslant T'$, $T = X_1$

$$\frac{(\Gamma, x : T', u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) + x : T \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [x : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P}{\Gamma, x : T', u^p : S \qquad S \leqslant S' \qquad [T, X_1, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}} \text{ TT-OutSeq}$$

By the induction hypothesis, $(\Gamma, u^p : S) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) : $(\Gamma + y : U), u^p : S \vdash P\{y/x\}$.

We know (Lemma 5 [Subtype Transitivity]) : $[U, X_2, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}$.

$$\frac{(\Gamma + y : U), u^p : S \vdash P\{y/x\} \qquad S \leqslant S' \qquad [U, X_2, \ldots, X_n] \leqslant \tilde{W} \leqslant \tilde{Z} \leqslant \tilde{Y}}{((\Gamma + y : U), u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) \vdash (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}} \text{ TT-OutSeq}$$

We know (Lemma 4 [Type Addition]):

$$((\Gamma + y : U), u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n) = ((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U.$$

So, we have :

$$((\Gamma, u^p : ![\tilde{Y}] \,.\, S) + (v_2 : X_2, \ldots, v_n : X_n)) + y : U \vdash \\ (u^p : ![\tilde{Z}] \,.\, S') \,!\, [y : W_1, v_2 : W_2, \ldots, v_n : W_n] \,.\, P\{y/x\}$$

which is what we need.

TT-InSeq

The substitution is :

$$((u^p : ?[\tilde{V}] \,.\, S) \,?\, [\tilde{v} : \tilde{W}] \,.\, P)\{y/x\}$$

Case 1: $x \neq u$

We have :

$$\Gamma, u^p : ?[\tilde{X}] \,.\, S, x : T \vdash (u^p : ?[\tilde{V}] \,.\, S) \,?\, [\tilde{v} : \tilde{W}] \,.\, P \qquad U \leqslant T$$

We need :

$$(\Gamma, u^p : ?[\tilde{X}] \,.\, S) + y : U \vdash (u^p : ?[\tilde{V}] \,.\, S) \,?\, [\tilde{v} : \tilde{W}] \,.\, P\{y/x\}$$

$$\frac{\Gamma, u^p : ?[\tilde{X}] \,.\, S, x : T \vdash (u^p : ?[\tilde{V}] \,.\, S) \,?\, [\tilde{v} : \tilde{W}] \,.\, P}{\Gamma, u^p : S, \tilde{v} : \tilde{W}, x : T \vdash P \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{V} \leqslant \tilde{W}} \text{ TT-InSeq}$$

By the induction hypothesis, $(\Gamma, u^p : S, \tilde{v} : \tilde{W}) + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]) : $(\Gamma + y : U), u^p : S, \tilde{v} : \tilde{W} \vdash P\{y/x\}$.

$$\frac{(\Gamma + y : U), u^p : S, \tilde{v} : \tilde{W} \vdash P\{y/x\} \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{V} \leqslant \tilde{W}}{(\Gamma + y : U), u^p : ?[\tilde{X}] \,.\, S \vdash (u^p : ?[\tilde{V}] \,.\, S) \,?\, [\tilde{v} : \tilde{W}] \,.\, P\{y/x\}} \text{ TT-InSeq}$$

We know (Lemma 4 [Type Addition]):

$(\Gamma + y : U), u^p : ?[\tilde{X}] \,.\, S = (\Gamma, u^p : ?[\tilde{X}] \,.\, S) + y : U.$

So, we have :

$$(\Gamma, u^p : ?[\tilde{X}] \,.\, S) + y : U \vdash (u^p : ?[\tilde{V}] \,.\, S) \,?\, [\tilde{v} : \tilde{W}] \,.\, P\{y/x\}$$

which is what we need.

Case 2 : $x = u$

We have :

$$\Gamma, x^p : ?[\tilde{X}] \,.\, S \vdash (x^p : ?[\tilde{V}] \,.\, S') \,?\, [\tilde{v} : \tilde{W}] \,.\, P \qquad ?[\tilde{Y}] \,.\, S'' \leqslant ?[\tilde{X}] \,.\, S$$

We need :

$$\Gamma + y^q : ?[\tilde{Y}] \,.\, S'' \vdash (y^q : ?[\tilde{V}] \,.\, S') \,?\, [\tilde{v} : \tilde{W}] \,.\, P\{y^q/x^p\}$$

$$\frac{\Gamma, x^p : ?[\tilde{X}] \,.\, S \vdash (x^p : ?[\tilde{V}] \,.\, S') \,?\, [\tilde{v} : \tilde{W}] \,.\, P}{\Gamma, \tilde{v} : \tilde{W}, x^p : S \vdash P \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{V} \leqslant \tilde{W}} \text{ TT-InSeq}$$

$$\frac{?[\tilde{Y}] \,.\, S'' \leqslant ?[\tilde{X}] \,.\, S}{S'' \leqslant S \qquad \tilde{Y} \leqslant \tilde{X}} \text{ AS-InSeq}$$

By the induction hypothesis, $(\Gamma, \tilde{v} : \tilde{W}) + y^q : S'' \vdash P\{y^q/x^p\}$.

We know from the definition of $+ : \Gamma, \tilde{v} : \tilde{W}, y^q : S'' \vdash P\{y^q/x^p\}$.

We know (Lemma 5 [Subtype Transitivity]) : $\tilde{Y} \leqslant \tilde{V} \leqslant \tilde{W}$.

$$\frac{\Gamma, y^q : S'', \tilde{v} : \tilde{W} \vdash P\{y^q/x^p\} \qquad S'' \leqslant S \qquad \tilde{Y} \leqslant \tilde{V} \leqslant \tilde{W}}{\Gamma, y^q : ?[\tilde{Y}] \,.\, S'' \vdash (y^q : ?[\tilde{V}] \,.\, S') \,?\, [\tilde{v} : \tilde{W}] \,.\, P\{y^q/x^p\}} \text{ TT-InSeq}$$

We know from the definition of $+ : \Gamma, y^q : ?[\tilde{Y}] \,.\, S'' = \Gamma + y^q : ?[\tilde{Y}] \,.\, S''$.

So, we have :

$$\Gamma + y^q : ?[\tilde{Y}] \,.\, S'' \vdash (y^q : ?[\tilde{V}] \,.\, S') \,?\, [\tilde{v} : \tilde{W}] \,.\, P\{y^q/x^p\}$$

which is what we need.

TT-Offer

The substitution is :

$$((u^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_m\})\{y/x\}$$

Case 1 : $x \neq u$

We have :

$$\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle, x : T \vdash (u^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_m\}$$

and $U \leqslant T$. We need :

$$(\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle) + y : U \vdash (u^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1\{y/x\}, \ldots, l_n : P_m\{y/x\}\}$$

$$\frac{\begin{array}{c}\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle, x : T \vdash \\ (u^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_m\}\end{array}}{\forall i_{1,\ldots,m}(\Gamma, u^p : S''_i, x : T \vdash P_i) \quad l \leq n \leq m \quad \forall i_{1,\ldots,l} S_i \leqslant S'_i \quad \forall i_{1,\ldots,n} S'_i \leqslant S''_i}\ \text{TT-Offer}$$

By the induction hypothesis, $\forall i_{1,\ldots,m}(\Gamma, u^p : S''_i) + y : U \vdash P_i\{y/x\}$.

We know (Lemma 4 [Type Addition]) : $\forall i_{1,\ldots,m}(\Gamma + y : U), u^p : S''_i \vdash P_i\{y/x\}$.

$$\frac{\begin{array}{l}\forall i_{1,\ldots,m}((\Gamma + y : U), u^p : S''_i \vdash \\ \quad P_i\{y/x\})\end{array} \qquad l \leq n \leq m \quad \forall i_{1,\ldots,l} S_i \leqslant S'_i \quad \forall i_{1,\ldots,n} S'_i \leqslant S''_i}{\begin{array}{c}(\Gamma + y : U), u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle \vdash \\ (u^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1\{y/x\}, \ldots, l_n : P_m\{y/x\}\}\end{array}}\ \text{TT-Offer}$$

We know (Lemma 4 [Type Addition]):

$(\Gamma + y : U), u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle = (\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle) + y : U$.

So, we have :

$$\begin{array}{c}(\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle) + y : U \vdash \\ (u^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1\{y/x\}, \ldots, l_n : P_m\{y/x\}\}\end{array}$$

which is what we need.

Case 2 : $x = u$

We have :

$$\begin{array}{l}\Gamma, x^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle \vdash (x^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_m\} \\ \&\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle \leqslant \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle\end{array}$$

We need :

$$\frac{\begin{array}{l}\Gamma + y^q : \&\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle \vdash \\ \quad (y^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1\{y/x\}, \ldots, l_n : P_m\{y/x\}\} \\ \Gamma, x^p : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle \vdash \\ \quad (x^p : \&\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleright \{l_1 : P_1, \ldots, l_m : P_m\}\end{array}}{\begin{array}{l}\forall i_{1,\ldots,m}(\Gamma, x^p : S''_i \vdash P_i) \qquad l \leq n \leq m \qquad \forall i_{1,\ldots,l} S_i \leqslant S'_i \qquad \forall i_{1,\ldots,n} S'_i \leqslant S''_i \\ \&\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle \leqslant \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle\end{array}}\ \text{TT-Offer}$$
$$\frac{}{k \leq l \qquad \forall i \in \{1, \ldots, k\}. S'''_i \leqslant S_i}\ \text{AS-Branch}$$

By the induction hypothesis, $\Gamma + y^q : S'''_1 \vdash P_1\{y^q/x^p\}, \ldots, \Gamma + y^q : S'''_m \vdash P_m\{y^q/x^p\}$.

We know from the definition of $+$: $\Gamma, y^q : S'''_1 \vdash P_1\{y^q/x^p\}, \ldots, \Gamma, y^q : S'''_m \vdash P_m\{y^q/x^p\}$.

$$\frac{\forall i_{1,\ldots,m}(\Gamma, y^q : S'''_i \vdash P_i) \qquad k \leq n \leq m \qquad \forall i_{1,\ldots,k} S'''_i \leqslant S'_i \qquad \forall i_{1,\ldots,n} S'_i \leqslant S''_i}{\begin{array}{c}\Gamma, y^q : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash \\ y^q \triangleright \{l_1 : P_1\{y^q/x^p\}, \ldots, l_n : P_n\{y^q/x^p\}\}\end{array}}\ \text{TT-Offer}$$

We know from the definition of $+$:

$\Gamma, y^q : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle = \Gamma + y^q : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle$.

So, we have :

$$\Gamma + y^q : \&\langle l_1 : S_1, \ldots, l_n : S_n \rangle \vdash y^q \triangleright \{l_1 : P_1\{y^q/x^p\}, \ldots, l_n : P_n\{y^q/x^p\}\}$$

which is what we need.

TT-Choose

The substitution is :

$$(u^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n \rangle) \triangleleft l_i \,.\, P\{y/x\}$$

Case 1 : $x \neq u$

We have :

$$\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle, x : T \vdash (u^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P \qquad U \leqslant T$$

We need :

$$(\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle) + y : U \vdash (u^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P\{y/x\}$$

$$\frac{\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle, x : T \vdash (u^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P}{\Gamma, u^p : S'', x : T \vdash P \qquad S' \leqslant S''_i \qquad 1 \leq i \leq n \leq l \qquad \forall j_{1,\ldots,n} S_j \leqslant S'_j} \text{ TT-Choose}$$

By the induction hypothesis, $(\Gamma, u^p : S'') + y : U \vdash P\{y/x\}$.

We know (Lemma 4 [Type Addition]): $(\Gamma + y : U), u^p : S'' \vdash P\{y/x\}$.

$$\frac{(\Gamma + y : U), u^p : S'' \vdash P\{y/x\} \quad S' \leqslant S''_i \quad 1 \leq i \leq n \leq l \quad \forall j_{1,\ldots,n} S_j \leqslant S'_j}{(\Gamma + y : U), u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle \vdash (u^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P\{y/x\}} \text{ TT-Choose}$$

We know (Lemma 4 [Type Addition]):

$(\Gamma + y : U), u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle = (\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle) + y : U.$

So, we have :

$$(\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle) + y : U \vdash (u^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P\{y/x\}$$

which is what we need.

Case 2 : $x = u$

We have :

$$\Gamma, x^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle \vdash (x^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P$$
$$\oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k \rangle \leqslant \oplus\langle l_1 : S_1, \ldots, l_l : S_l \rangle$$

We need :

$$\Gamma + y^q : \oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k \rangle \vdash (y^q : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n \rangle) \triangleleft l_i \,.\, P\{y^q/x^p\}$$

$$\dfrac{\dfrac{\Gamma, x^p : \oplus\langle l_1 : S_1, \ldots, l_l : S_l\rangle \vdash (x^p : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleleft l_i \;.\; P}{\Gamma, x^p : S'' \vdash P \qquad S'_i \leqslant S'' \qquad 1 \leq i \leq n \leq l \qquad \forall j_{1,\ldots,n} S_j \leqslant S'_j}\ \text{TT-Choose} \qquad \dfrac{\oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle \leqslant \oplus\langle l_1 : S_1, \ldots, l_l : S_l\rangle}{l \leq k \qquad \forall i_{1,\ldots,l} S'''_l \leqslant S_l}\ \text{AS-Choice}}{\Gamma, y^q : \oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle \vdash (y^q : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleleft l_i \;.\; P\{y^q/x^p\}}\ \text{TT-Choose}$$

By the induction hypothesis, $\Gamma + y^q : S'' \vdash P\{y^q/x^p\}$.
We know from the definition of $+$: $\Gamma, y^q : S'' \vdash P\{y^q/x^p\}$.
We know from the definition of $+$:
$\Gamma, y^q : \oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle = \Gamma + y^q : \oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle$.

So, we have :

$$\Gamma + y^q : \oplus\langle l_1 : S'''_1, \ldots, l_k : S'''_k\rangle \vdash (y^q : \oplus\langle l_1 : S'_1, \ldots, l_n : S'_n\rangle) \triangleleft l_i \;.\; P\{y^q/x^p\}$$

which is what we need.

□

A.3 Proof of Theorem 1 [Subject Reduction Theorem]

If $\Gamma \Vdash_{\mathcal{T}} P$ and $P \xrightarrow{\alpha, l}_{\mathcal{T}} P'$ where Γ is balanced with respect to α (if $\alpha \neq \tau$), then $\Gamma' \Vdash_{\mathcal{T}} P'$ where

$\Gamma' = \Gamma$ if $\alpha = \tau$ or if α is not a session channel
$\Gamma' = \Gamma'', \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l)$
where $\Gamma'', \alpha^+ : S, \alpha^- : \overline{S} = \Gamma$ if α is a session channel.

Proof By structural induction on the derivation of $P \xrightarrow{\alpha, l}_{\mathcal{T}} P'$

TR-Comm

The reduction is :

$$(x^p : T') \,?\, [\tilde{y} : \tilde{V}] \,.\, P \mid (x^q : T'') \,!\, [\tilde{z} : \tilde{W}] \,.\, Q \xrightarrow{x,\tau} P\{\tilde{z}/\tilde{y}\} \mid Q$$

Case 1 : x is not a session channel

We have :

$$\Gamma, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} (x : T') \,?\, [\tilde{y} : \tilde{V}] \,.\, P \mid (x : T'') \,!\, [\tilde{z} : \tilde{W}] \,.\, Q$$

We need :

$$\Gamma, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \mid Q$$

$$\frac{\Gamma, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} (x : T') \,?\, [\tilde{y} : \tilde{V}] \,.\, P \mid (x : T'') \,!\, [\tilde{z} : \tilde{W}] \,.\, Q}{\Gamma_1, x : \widehat{\ }[\tilde{T}] \vDash_{\mp} (x : T') \,?\, [\tilde{y} : \tilde{V}] \,.\, P \qquad \Gamma_2, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} (x : T'') \,!\, [\tilde{z} : \tilde{W}] \,.\, Q} \text{ TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_1, x : \widehat{\ }[\tilde{T}] \vDash_{\mp} (x : T') \,?\, [\tilde{y} : \tilde{V}] \,.\, P}{\Gamma_1, x : \widehat{\ }[\tilde{T}], \tilde{y} : \tilde{V} \vDash_{\mp} P \qquad \widehat{\ }[T] \leqslant T' \leqslant ?[V]} \text{ TT-In}$$

$$\frac{\Gamma_2, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} (x : T'') \,!\, [\tilde{z} : \tilde{W}] \,.\, Q}{\Gamma_2, x : \widehat{\ }[\tilde{T}] \vDash_{\mp} Q \qquad \widehat{\ }[\tilde{T}] \leqslant T'' \leqslant ![\tilde{W}] \qquad \tilde{U} \leqslant \tilde{W}} \text{ TT-Out}$$

$$\frac{\widehat{\ }[T] \leqslant ![W]}{\tilde{W} \leqslant \tilde{T}} \text{ AS-Out}$$

$$\frac{\widehat{\ }[T] \leqslant ?[V]}{\tilde{T} \leqslant \tilde{V}} \text{ AS-In}$$

We know (Lemma 5 [Subtype Transitivity]) : $\tilde{U} \leqslant \tilde{V}$

We know (Lemma 8 [Substitution]) : $\Gamma_1, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\}$

$$\frac{\Gamma_1, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \qquad \Gamma_2, x : \widehat{\ }[\tilde{T}] \vDash_{\mp} Q}{(\Gamma_1, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U}) + (\Gamma_2, x : \widehat{\ }[\tilde{T}]) \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \mid Q} \text{ TT-Par}$$

We know from the definition of $+$:

$(\Gamma_1, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U}) + (\Gamma_2, x : \widehat{\ }[\tilde{T}]) = \Gamma, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U}.$

So, we have :

$$\Gamma, x : \widehat{\ }[\tilde{T}] + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \mid Q$$

which is what we need.

Case 2 : x is a session channel
We have :

$$\Gamma, x^+ : ?[\tilde{T}].S, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{U} \vDash_{\mp} (x^+ : ?[\tilde{T'}] \,.\, S')?[\tilde{y} : \tilde{V}].P \mid (x^- : ![\tilde{T''}] \,.\, S'')![\tilde{z} : \tilde{W}].Q$$

We need :

$$\Gamma, x^+ : S, x^- : \overline{S} + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \mid Q$$

$$\frac{\Gamma, x^+ : ?[\tilde{T}].S, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{U} \vDash_{\mp} (x^+ : ?[\tilde{T'}] \,.\, S')?[\tilde{y} : \tilde{V}].P \mid (x^- : ![\tilde{T''}] \,.\, S'')![\tilde{z} : \tilde{W}].Q}{\Gamma_1, x^+ : ?[\tilde{T}].S \vDash_{\mp} (x^+ : ?[\tilde{T'}] \,.\, S')?[\tilde{y} : \tilde{V}].P \qquad \Gamma_2, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{U} \vDash_{\mp} (x^- : ![\tilde{T''}] \,.\, S'')![\tilde{z} : \tilde{W}].Q} \text{ TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_1, x^+ : ?[\tilde{T}].S \vDash_{\mp} (x^+ : ?[\tilde{T'}] \,.\, S')?[\tilde{y} : \tilde{V}].P}{\Gamma_1, x^+ : S, \tilde{y} : \tilde{V} \vDash_{\mp} P \qquad S \leqslant S' \qquad \tilde{T} \leqslant \tilde{T'} \leqslant \tilde{V}} \text{ TT-InSeq}$$

$$\frac{\Gamma_2, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{U} \vDash_{\mp} (x^- : ![\tilde{T''}] \,.\, S'')![\tilde{z} : \tilde{W}].Q}{\Gamma_2, x^- : \overline{S} \vDash_{\mp} Q \qquad \overline{S} \leqslant S'' \qquad \tilde{U} \leqslant \tilde{W} \leqslant \tilde{T''} \leqslant \tilde{T}} \text{ TT-OutSeq}$$

We know (Lemma 5 [Subtype Transitivity]) : $\tilde{U} \leqslant \tilde{V}$

We know (Lemma 8 [Substitution]) : $\Gamma_1, x^+ : S + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\}$

$$\frac{\Gamma_1, x^+ : S + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \qquad \Gamma_2, x^- : \overline{S} \vDash_{\mp} Q}{(\Gamma_1, x^+ : S + \tilde{z} : \tilde{U}) + (\Gamma_2, x^- : \overline{S}) \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \mid Q} \text{ TT-Par}$$

We know from the definition of $+$:
$(\Gamma_1, x^+ : S + \tilde{z} : \tilde{U}) + (\Gamma_2, x^- : \overline{S}) = \Gamma, x^+ : S, x^- : \overline{S} + \tilde{z} : \tilde{U}.$

So, we have :

$$\Gamma, x^+ : S, x^- : \overline{S} + \tilde{z} : \tilde{U} \vDash_{\mp} P\{\tilde{z}/\tilde{y}\} \mid Q$$

which is what we need.

TR-Select
The reduction is :

$$(x^+ : \&\langle l_1 : S_1', \ldots, l_m : S_m' \rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid (x^- : \oplus\langle l_1 : S_1'', \ldots, l_k : S_k'' \rangle) \triangleleft l_i \,.\, Q \xrightarrow{x, l_i}_{T} P_i \mid Q$$

We have :

$$\Gamma, x^+ : \&\langle l_1 : S_1, \ldots, l_l : S_l \rangle, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_l : \overline{S_l} \rangle \vDash_{\mp} (x^+ : \&\langle l_1 : S_1', \ldots, l_m : S_m' \rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid (x^- : \oplus\langle l_1 : S_1'', \ldots, l_k : S_k'' \rangle) \triangleleft l_i \,.\, Q$$

We need :

$$\Gamma, x^+ : S_i, x^- : \overline{S_i} \vDash_{\mp} P_i \mid Q$$

$$\frac{\begin{array}{l}\Gamma, x^+ : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_l : \overline{S_l}\rangle \vDash_\mp \\ \quad (x^+ : \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}'\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} | \\ \quad (x^- : \oplus\langle l_1 : {S_1}'', \ldots, l_k : {S_k}''\rangle) \triangleleft l_i \ . \ Q\end{array}}{\begin{array}{l}\Gamma_1, x^+ : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle \vDash_\mp \\ \quad (x^+ : \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}'\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \\ \quad \Gamma_2, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_l : \overline{S_l}\rangle \vDash_\mp (x^- : \oplus\langle l_1 : {S_1}'', \ldots, l_k : {S_k}''\rangle) \triangleleft l_i \ . \ Q\end{array}} \text{TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\begin{array}{l}\Gamma_1, x^+ : \&\langle l_1 : S_1, \ldots, l_l : S_l\rangle \vDash_\mp \\ \quad (x^+ : \&\langle l_1 : {S_1}', \ldots, l_m : {S_m}'\rangle) \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}\end{array}}{\forall i \in \{1, \ldots, n\}.(\Gamma_1, x^+ : S_i''' \vDash_\mp P_i) \quad l \leq m \leq n \quad \begin{array}{l}\forall i \in \{1, \ldots, l\}.S_i \leqslant S_i' \\ \forall i \in \{1, \ldots, m\}.S_i' \leqslant S_i'''\end{array}} \text{TT-Offer}$$

We know (Lemma 5 [Subtype Transitivity]) : $S_i \leqslant S_i'''$

By the induction hypothesis : $\Gamma_1 + x^+ : S_i \vDash_\mp P_i$

From the definition of + : $\Gamma_1, x^+ : S_i \vDash_\mp P_i$

$$\frac{\Gamma_2, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_l : \overline{S_l}\rangle \vDash_\mp (x^- : \oplus\langle l_1 : {S_1}'', \ldots, l_k : {S_k}''\rangle) \triangleleft l_i \ . \ Q}{\Gamma_2, x^- : S_i'''' \vDash_\mp Q \qquad S_i'' \leqslant S_i'''' \qquad 1 \leq i \leq k \leq l \qquad \forall i \in \{1, \ldots, q\}.\overline{S_i} \leqslant S_i''} \text{TT-Choose}$$

We know (Lemma 5 [Subtype Transitivity]) : $\overline{S_i} \leqslant S_i''''$

By the induction hypothesis : $\Gamma_2 + x^- : \overline{S_i} \vDash_\mp Q$

From the definition of + : $\Gamma_2, x^- : \overline{S_i} \vDash_\mp Q$

$$\frac{\Gamma_1, x^+ : S_i \vDash_\mp P_i \qquad \Gamma_2, x^- : \overline{S_i} \vDash_\mp Q}{(\Gamma_1, x^+ : S_i) + (\Gamma_2, x^- : \overline{S_i}) \vDash_\mp P_i \mid Q} \text{TT-Par}$$

But, $(\Gamma_1, x^+ : S_i) + (\Gamma_2, x^- : \overline{S_i}) = \Gamma, x^+ : S_i, x^- : \overline{S_i}$.

So, we have :

$$\Gamma, x^+ : S_i, x^- : \overline{S_i} \vDash_\mp P_i \mid Q$$

which is what we need.

TR-True

The reduction is :

$$\mathsf{if}\ \mathit{true} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow{\tau, -}_{T} P$$

We have :

$$\Gamma \vDash_\mp \mathsf{if}\ \mathit{true} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q$$

We need :

$$\Gamma \vDash_\mp P$$

$$\frac{\Gamma \vDash_\mp \mathsf{if}\ \mathit{true} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q}{\Gamma \vDash_\mp \mathit{true} \leqslant \mathsf{bool} \qquad \Gamma \vDash_\mp P \qquad \Gamma \vDash_\mp Q} \text{TT-Cond}$$

So, we have :

$$\Gamma \vDash_T P$$

which is what we need.

TR-False

The reduction is :

$$\mathsf{if}\ \mathit{false} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q \xrightarrow[T]{\tau,-} Q$$

We have :

$$\Gamma \vDash_T \mathsf{if}\ \mathit{false} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q$$

We need :

$$\Gamma \vDash_T Q$$

$$\frac{\Gamma \vDash_T \mathsf{if}\ \mathit{false} : \mathsf{bool}\ \mathsf{then}\ P\ \mathsf{else}\ Q}{\Gamma \vDash_T \mathit{false} \leqslant \mathsf{bool} \qquad \Gamma \vDash_T P \qquad \Gamma \vDash_T Q}\ \text{TT-Cond}$$

So, we have :

$$\Gamma \vDash_T Q$$

which is what we need.

TR-Par

The derivation ends with :

$$\frac{P \xrightarrow[T]{\alpha,l} P'}{P \mid Q \xrightarrow[T]{\alpha,l} P' \mid Q'}$$

Case 1 : $\alpha = \tau$

We have :

$$\Gamma \vDash_T P \mid Q$$

We need :

$$\Gamma \vDash_T P' \mid Q$$

$$\frac{\Gamma \vDash_T P \mid Q}{\Gamma_1 \vDash_T P \qquad \Gamma_2 \vDash_T Q}\ \text{TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$\Gamma_1 \vDash_T P$ and $P \xrightarrow[T]{\tau,-} P'$, so, by the induction hypothesis, $\Gamma_1 \vDash_T P'$.

$$\frac{\Gamma_1 \vDash_T P' \qquad \Gamma_2 \vDash_T Q}{\Gamma_1 + \Gamma_2 \vDash_T P' \mid Q}\ \text{TT-Par}$$

But, $\Gamma_1 + \Gamma_2 = \Gamma$.

So, we have :

$$\Gamma \vDash_T P' \mid Q$$

which is what we need.

Case 2 : $\alpha \neq \tau$, α is not a session channel
We have :

$$\Gamma \vDash_{\mathcal{T}} P \mid Q$$

We need :

$$\Gamma \vDash_{\mathcal{T}} P' \mid Q$$

$$\frac{\Gamma \vDash_{\mathcal{T}} P \mid Q}{\Gamma_1 \vDash_{\mathcal{T}} P \qquad \Gamma_2 \vDash_{\mathcal{T}} Q}\ \text{TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$\Gamma_1 \vDash_{\mathcal{T}} P$ and $P \xrightarrow{\alpha, l}_{\mathcal{T}} P'$, so, by the induction hypothesis, $\Gamma_1 \vDash_{\mathcal{T}} P'$.

$$\frac{\Gamma_1 \vDash_{\mathcal{T}} P' \qquad \Gamma_2 \vDash_{\mathcal{T}} Q}{\Gamma_1 + \Gamma_2 \vDash_{\mathcal{T}} P' \mid Q}\ \text{TT-Par}$$

But, $\Gamma_1 + \Gamma_2 = \Gamma$.

So, we have :

$$\Gamma \vDash_{\mathcal{T}} P' \mid Q$$

which is what we need.

Case 3 : $\alpha \neq \tau$, α is a session channel
We have :

$$\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mathcal{T}} P \mid Q$$

We need :

$$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mathcal{T}} P' \mid Q$$

$$\frac{\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mathcal{T}} P \mid Q}{\Gamma_1, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mathcal{T}} P \qquad \Gamma_2 \vDash_{\mathcal{T}} Q}\ \text{TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$ and $\alpha \notin \Gamma_2$

$\Gamma_1, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mathcal{T}} P$ and $P \xrightarrow{\alpha, l}_{\mathcal{T}} P'$, so, by the induction hypothesis, as
α is a session channel, $\Gamma_1, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mathcal{T}} P'$.

$$\frac{\Gamma_1, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mathcal{T}} P' \qquad \Gamma_2 \vDash_{\mathcal{T}} Q}{(\Gamma_1, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l)) + \Gamma_2 \vDash_{\mathcal{T}} P' \mid Q}\ \text{TT-Par}$$

We know (Lemma 4 [Type Addition]):
$(\Gamma_1, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l)) + \Gamma_2 = (\Gamma_1 + \Gamma_2), \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l)$
But, $\Gamma_1 + \Gamma_2 = \Gamma$ and $\alpha \notin \Gamma_2$.

So, we have :

$$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mathcal{T}} P' \mid Q$$

which is what we need.

TR-Cong

The derivation ends with :

$$\frac{P' \equiv P \quad P \xrightarrow[T]{\alpha,l} Q \quad Q \equiv Q'}{P' \xrightarrow[T]{\alpha,l} Q'}$$

Case 1 : $\alpha = \tau$ or α is not a session channel

We have :

$$\Gamma \vDash_{\mp} P'$$

We need :

$$\Gamma \vDash_{\mp} Q'$$

We know (Lemma 9 [Structural Equivalence]) : $\Gamma \vDash_{\mp} P$

$\Gamma \vDash_{\mp} P$ and $P \xrightarrow[T]{\alpha,l} Q$, so, by the induction hypothesis, $\Gamma \vDash_{\mp} Q$.

We know (Lemma 9 [Structural Equivalence]) : $\Gamma \vDash_{\mp} Q'$

So, we have :

$$\Gamma \vDash_{\mp} Q'$$

which is what we need.

Case 2 : α is a session channel

We have :

$$\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} P'$$

We need :

$$\Gamma, \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l) \vDash_{\mp} Q'$$

We know (Lemma 9 [Structural Equivalence]) : $\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} P$

$\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} P$ and $P \xrightarrow[T]{\alpha,l} Q$, so, by the induction hypothesis,
$\Gamma, \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l) \vDash_{\mp} Q$.

We know (Lemma 9 [Structural Equivalence]) :

$$\Gamma, \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l) \vDash_{\mp} Q'$$

So, we have :

$$\Gamma\alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l) \vDash_{\mp} Q'$$

which is what we need.

TR-New

The derivation ends with :

$$\frac{P \xrightarrow[T]{\alpha,l} P' \quad \alpha \neq x}{(\nu x : T)P \xrightarrow[T]{\alpha,l} (\nu x : T)P'}$$

Case 1 : $\alpha = \tau$ or is not a session channel

We have :

$$\Gamma \vDash_{\mp} (\nu x : T)P$$

We need :

$$\Gamma \vDash_{\mathcal{T}} (\nu x : T)P'$$

$$\frac{\Gamma \vDash_{\mathcal{T}} (\nu x : T)P}{\Gamma, x : T \vDash_{\mathcal{T}} P}\ \text{TT-New}$$

$\Gamma, x : T \vDash_{\mathcal{T}} P$ and $P \xrightarrow[T]{\alpha, l} P'$ where $\alpha = \tau$ or is not a session channel, so, by the induction hypothesis, $\Gamma, x : T \vDash_{\mathcal{T}} P'$.

$$\frac{\Gamma, x : T \vDash_{\mathcal{T}} P'}{\Gamma \vDash_{\mathcal{T}} (\nu x : T)P'}\ \text{TT-New}$$

So, we have :

$$\Gamma \vDash_{\mathcal{T}} (\nu x : T)P'$$

which is what we need.

Case 2 : α is a session channel
We have :

$$\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} (\nu x : T)P$$

We need :

$$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mp} (\nu x : T)P'$$

$$\frac{\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} (\nu x : T)P}{\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x : T \vDash_{\mp} P} \text{ TT-New}$$

$\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x : T \vDash_{\mp} P$ and $P \xrightarrow{\alpha, l}_{\mp} P'$, so, by the induction hypothesis,
$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l), x : T \vDash_{\mp} P'$.

$$\frac{\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l), x : T \vDash_{\mp} P'}{\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mp} (\nu x : T)P'} \text{ TT-New}$$

So, we have :

$$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mp} (\nu x : T)P'$$

which is what we need.

TR-NewSeq
The derivation ends with :

$$\frac{P \xrightarrow{\alpha, l}_{\mp} P' \quad \alpha \neq x}{(\nu x^{\pm} : S)P \xrightarrow{\alpha, l}_{\mp} (\nu x^{\pm} : S)P'}$$

Case 1 : $\alpha = \tau$ or is not a session channel
We have :

$$\Gamma \vDash_{\mp} (\nu x^{\pm} : S)P$$

We need :

$$\Gamma \vDash_{\mp} (\nu x^{\pm} : S)P'$$

$$\frac{\Gamma \vDash_{\mp} (\nu x^{\pm} : S)P}{\Gamma, x^+ : S, x^- : \overline{S} \vDash_{\mp} P} \text{ TT-NewSeq}$$

$\Gamma, x^+ : S, x^- : \overline{S} \vDash_{\mp} P$ and $P \xrightarrow{\alpha, l}_{\mp} P'$, so, by the induction hypothesis,
$\Gamma, x^+ : S, x^- : \overline{S} \vDash_{\mp} P'$.

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vDash_{\mp} P'}{\Gamma, \vDash_{\mp} (\nu x^{\pm} : S)P'} \text{ TT-NewSeq}$$

So, we have :

$$\Gamma \vDash_{\mp} (\nu x^{\pm} : S)P'$$

which is what we need.

Case 2 : α is a session channel

We have :

$$\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} (\nu x^{\pm} : S_2)P$$

We need :

$$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mp} (\nu x^{\pm} : S_2)P'$$

$$\frac{\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vDash_{\mp} (\nu x^{\pm} : S)P}{\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x^+ : S_2, x^- : \overline{S_2} \vDash_{\mp} P} \text{ TT-NewSeq}$$

$\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x^+ : S_2, x^- : \overline{S_2} \vDash_{\mp} P$ and $P \xrightarrow[T]{\alpha, l} P'$, so, by the induction hypothesis, $\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l), x^+ : S_2, x^- : \overline{S_2} \vDash_{\mp} P'$.

$$\frac{\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l), x^+ : S_2, x^- : \overline{S_2} \vDash_{\mp} P'}{\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mp} (\nu x^{\pm} : S)P'} \text{ TT-NewSeq}$$

So, we have :

$$\Gamma, \alpha^+ : \text{tail}(S, l), \alpha^- : \text{tail}(\overline{S}, l) \vDash_{\mp} (\nu x^{\pm} : S_2)P'$$

which is what we need.

TR-NewX

The derivation ends with :

$$\frac{P \xrightarrow[T]{x, l} P'}{(\nu x : T)P \xrightarrow[T]{\tau, \_} (\nu x : T)P'}$$

We have :

$$\Gamma \vDash_{\mp} (\nu x : T)P$$

We need :

$$\Gamma \vDash_{\mp} (\nu x : T)P'$$

$$\frac{\Gamma \vDash_{\mp} (\nu x : T)P}{\Gamma, x : T \vDash_{\mp} P} \text{ TT-New}$$

$\Gamma, x : T \vDash_{\mp} P$ and $P \xrightarrow[T]{x, l} P'$, so, by the induction hypothesis, $\Gamma, x : T \vDash_{\mp} P'$.

$$\frac{\Gamma, x : T \vDash_{\mp} P'}{\Gamma \vDash_{\mp} (\nu x : T)P'} \text{ TT-New}$$

So, we have :

$$\Gamma \vDash_{\mp} (\nu x : T)P'$$

which is what we need.

TR-NewXSeq

The derivation ends with :

$$\frac{P \xrightarrow[T]{x, l} P'}{(\nu x^{\pm} : T)P \xrightarrow[T]{\tau, \_} (\nu x^{\pm} : \text{tail}(T, l))P'}$$

We have :

$$\Gamma \vDash_{\mp} (\nu x^{\pm} : T)P$$

We need :

$$\Gamma \vDash_{\mp} (\nu x^{\pm} : tail(T, l))P'$$

$$\frac{\Gamma \vDash_{\mp} (\nu x^{\pm} : T)P}{\Gamma, x^{+} : T, x^{-} : \overline{T} \vDash_{\mp} P}\ \text{TT-NewSeq}$$

$\Gamma, x^{+} : T, x^{-} : \overline{T} \vDash_{\mp} P$ and $P \xrightarrow[T]{x,l} P'$, so, by the induction hypothesis,

$\Gamma, x^{+} : \text{tail}(T, l), x^{-} : \text{tail}(\overline{T}, l) \vDash_{\mp} P'$.

$$\frac{\Gamma, x^{+} : \text{tail}(T, l), x^{-} : \text{tail}(\overline{T}, l) \vDash_{\mp} P'}{\Gamma \vDash_{\mp} (\nu x^{\pm} : T)P'}\ \text{TT-NewSeq}$$

So, we have :

$$\Gamma \vDash_{\mp} (\nu x^{\pm} : tail(T, l))P'$$

which is what we need.

□

B Run-Time Safety Theorem

In the following proof, wherever it is necessary to remove multiple new bindings, we write this as a single application of the T-New rule for simplicity. In fact it could represent multiple applications of both the T-New and T-NewSeq rules.

B.1 Proof of Theorem 3 [Run-Time Safety]

If $\Gamma \vDash_{\mp} P$, $P \equiv (\nu \tilde{w} : \tilde{W})(Q \mid R)$, $Q \xrightarrow{\alpha,l}_{\tau} Q'$ and Γ is balanced with respect to α (if $\alpha \neq \tau$), then

i)If $Q \equiv (x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2$
where x is not a session channel, then
$\forall i \in \{1, \ldots, n\}.\Gamma \vDash_{\mp} z_i : V_i$ or $(\nu \tilde{z_i} : V_i) \in (\nu \tilde{w} : \tilde{W})$ where $V_i \leqslant T_i$
and
$\Gamma \vDash_{\mp} x : \frown[\tilde{X}]$ or $(\nu x : \frown[\tilde{X}]) \in (\nu \tilde{w} : \tilde{W})$
for some $\tilde{X}$ such that $\frown[\tilde{X}] \leqslant ?[\tilde{T}]$ and $\frown[\tilde{X}] \leqslant ![\tilde{U}]$

ii)If $Q \equiv (x^p : ?[\tilde{A}]\,.\,S')\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x^q : ![\tilde{B}]\,.\,\overline{S'})\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2$
where x is a session channel, then
$p = \overline{q}$
and
$\forall i \in \{1, \ldots, n\}.\Gamma \vDash_{\mp} z_i : V_i$ or $(\nu \tilde{z_i} : V_i) \in (\nu \tilde{w} : \tilde{W})$ where $V_i \leqslant T_i$
and
$\Gamma \vDash_{\mp} x^p : ?[\tilde{X}]\,.\,S, x^q : ![\tilde{X}]\,.\,\overline{S}$ or $(\nu x^{\pm} : ?[\tilde{X}]\,.\,S) \in (\nu \tilde{w} : \tilde{W})$
or $(\nu x^{\pm} : ![\tilde{X}]\,.\,\overline{S}) \in (\nu \tilde{w} : \tilde{W})$ for some $\tilde{X}$ and S such that
$?[\tilde{X}]\,.\,S \leqslant ?[\tilde{T}]\,.\,S'$ and $![\tilde{X}]\,.\,\overline{S} \leqslant ![\tilde{U}]\,.\,\overline{S'}$

iii)If $Q \equiv x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i\,.\,Q_2$, then
$p = \overline{q}$
and
$l_i \in \{l_1, \ldots, l_n\}$
and
$\Gamma \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^q : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle$
or $(\nu x^{\pm} : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle) \in (\nu \tilde{w} : \tilde{W})$
or $(\nu x^{\pm} : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle) \in (\nu \tilde{w} : \tilde{W})$

iv)If $Q \equiv$ if x then Q_1 else Q_2, then
$\Gamma \vdash x :$ bool or $(\nu x :$ bool$) \in (\nu \tilde{w} : \tilde{W})$

Proof There are four cases - one for each type of reduction.

Case 1 : $Q \equiv (x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2$ *where* x *is not a session channel*

We have :

$$\Gamma \vDash_{\mp} (\nu\tilde{w} : \tilde{W})(((x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2) \mid R)$$

We need :

$$\forall i \in \{1, \ldots, n\}.\Gamma \vDash_{\mp} z_i : V_i \text{ or } (\nu\tilde{z_i} : V_i) \in (\nu\tilde{w} : \tilde{W}) \text{ where } V_i \leqslant T_i$$
and
$$\Gamma \vDash_{\mp} x : \hat{}[\tilde{X}] \text{ or } (\nu x : \hat{}[\tilde{X}]) \in (\nu\tilde{w} : \tilde{W})$$
for some $\tilde{X}$ such that $\hat{}[\tilde{X}] \leqslant ?[\tilde{T}]$ and $\hat{}[\tilde{X}] \leqslant ![\tilde{U}]$

$$\frac{\Gamma \vDash_{\mp} (\nu\tilde{w} : \tilde{W})(((x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2) \mid R)}{\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} ((x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2) \mid R}\ \text{TT-New}$$

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} ((x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2) \mid R}{\Gamma_1 \vDash_{\mp} (x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2 \qquad \Gamma_2 \vDash_{\mp} R}\ \text{TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$

$$\frac{\Gamma_1 \vDash_{\mp} (x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2 \qquad \Gamma_2 \vDash_{\mp} R}{\Gamma_3 \vDash_{\mp} (x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \qquad \Gamma_4 \vDash_{\mp} (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2}\ \text{TT-Par}$$

where $\Gamma_3 + \Gamma_4 = \Gamma_1$

$$\frac{\Gamma_3 \vDash_{\mp} (x : A)\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1}{\Gamma_3, \tilde{y} : \tilde{T} \vDash_{\mp} Q_1 \qquad \Gamma_3 \vDash_{\mp} x \leqslant A \leqslant ?[\tilde{T}]}\ \text{TT-In}$$

We know (Lemma 5 [Subtype Transitivity]) : $\Gamma_3 \vDash_{\mp} x \leqslant ?[\tilde{T}]$

Let $\Gamma_4 = \Gamma_5 + z : \tilde{V}$ where $\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} \tilde{z} : \tilde{V}$

$$\frac{\Gamma_5 + z : \tilde{V} \vDash_{\mp} (x : B)\,!\,[\tilde{z} : \tilde{U}]\,.\,Q_2}{\Gamma_5 \vDash_{\mp} Q_2 \qquad \Gamma_5 \vDash_{\mp} x \leqslant B \leqslant ![\tilde{U}] \qquad \tilde{V} \leqslant \tilde{U}}\ \text{TT-Out}$$

We know (Lemma 5 [Subtype Transitivity]) : $\Gamma_5 \vDash_{\mp} x \leqslant ![\tilde{U}]$

$\Gamma_2 + \Gamma_3 + \Gamma_5 + \tilde{z} : \tilde{V} = \Gamma, \tilde{w} : \tilde{W}$

From the definition of $+$: $\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} x : \hat{}[\tilde{X}]$ for some $\tilde{X}$
such that $\hat{}[\tilde{X}] \leqslant ?[\tilde{T}]$ and $\hat{}[\tilde{X}] \leqslant ![\tilde{U}]$

So, $\Gamma \vDash_{\mp} x : \hat{}[\tilde{X}]$ or $(\nu x : \hat{}[\tilde{X}]) \in (\nu\tilde{w} : \tilde{W})$ for some $\tilde{X}$ such that $\hat{}[\tilde{X}] \leqslant ?[\tilde{T}]$
and $\hat{}[\tilde{X}] \leqslant ![\tilde{U}]$

$$\frac{\hat{}[\tilde{X}] \leqslant ?[\tilde{T}]}{\forall i \in \{1, \ldots, n\}.X_i \leqslant T_i}\ \text{AS-In}$$

$$\frac{\hat{}[\tilde{X}] \leqslant ![\tilde{U}]}{\forall i \in \{1, \ldots, n\}.U_i \leqslant X_i}\ \text{AS-Out}$$

We know (Lemma 5 [Subtype Transitivity]) : $\forall i \in \{1, \ldots, n\}.V_i \leqslant T_i$

From the definition of $+$: $\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} \tilde{z} : \tilde{V}$

So, $\forall i \in \{1, \ldots, n\}.\Gamma \vDash_{\mp} z_i : V_i$ or $(\nu\tilde{z_i} : V_i) \in (\nu\tilde{w} : \tilde{W})$

So, we have :

$$\begin{array}{l}\forall i \in \{1,\ldots,n\}.\Gamma \vDash_{\overline{\tau}} z_i : V_i \text{ or } (\nu \tilde{z_i} : V_i) \in (\nu \tilde{w} : \tilde{W}) \text{ where } V_i \leqslant T_i \\ \text{and} \\ (\Gamma \vDash_{\overline{\tau}} x : \hat{}[\tilde{X}] \text{ or } (\nu x : \hat{}[\tilde{X}]) \in (\nu \tilde{w} : \tilde{W}) \\ \quad \text{for some } \tilde{X} \text{ such that } \hat{}[\tilde{X}] \leqslant ?[\tilde{T}] \text{ and } \hat{}[\tilde{X}] \leqslant ![\tilde{U}]\end{array}$$

which is what we need.

Case 2 : $Q \equiv (x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \,|\, (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2$ *where x is a session channel*
We have :

$$\Gamma \vdash (\nu \tilde{w} : \tilde{W})(((x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \,|\, (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2) \,|\, R)$$

We need :

$$\begin{array}{l}p = \overline{q} \\ \text{and} \\ \forall i \in \{1,\ldots,n\}.\Gamma \vDash_{\overline{\tau}} z_i : V_i \text{ or } (\nu \tilde{z_i} : V_i) \in (\nu \tilde{w} : \tilde{W}) \text{ where } V_i \leqslant T_i \\ \text{and} \\ \Gamma \vDash_{\overline{\tau}} x^p : ?[\tilde{X}] \,.\, S, x^q : ![\tilde{X}] \,.\, \overline{S} \text{ or } (\nu x^{\pm} : ?[\tilde{X}] \,.\, S) \in (\nu \tilde{w} : \tilde{W}) \\ \quad \text{or } (\nu x^{\pm} : ![\tilde{X}] \,.\, \overline{S}) \in (\nu \tilde{w} : \tilde{W}) \text{ for some } \tilde{X} \text{ and } S \text{ such that} \\ \quad ?[\tilde{X}] \,.\, S \leqslant ?[\tilde{T}] \,.\, S' \text{ and } ![\tilde{X}] \,.\, \overline{S} \leqslant ![\tilde{U}] \,.\, \overline{S'}\end{array}$$

$$\frac{\Gamma \vDash_{\mp} (\nu\tilde{w} : \tilde{W})(((x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \mid (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2) \mid R)}{\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} ((x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \mid (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2) \mid R} \text{ TT-New}$$

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} ((x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \mid (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2) \mid R}{\Gamma_1 \vDash_{\mp} (x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \mid (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2 \qquad \Gamma_2 \vDash_{\mp} R} \text{ TT-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$

$$\frac{\Gamma_1 \vDash_{\mp} (x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \mid (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2}{\Gamma_3 \vDash_{\mp} (x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1 \qquad \Gamma_4 \vDash_{\mp} (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2} \text{ TT-Par}$$

where $\Gamma_3 + \Gamma_4 = \Gamma_1$

As the above addition is defined, $p \neq q$

Let $\Gamma_3 = \Gamma_5, x^p : ?[\tilde{X}] \,.\, S$ for some $\tilde{X}$ and S

$$\frac{\Gamma_5, x^p : ?[\tilde{X}] \,.\, S \vDash_{\mp} (x^p : ?[\tilde{A}] \,.\, S') \,?\, [\tilde{y} : \tilde{T}] \,.\, Q_1}{\Gamma_5, x^p : S, y : T \vDash_{\mp} Q_1 \qquad S \leqslant S' \qquad \tilde{X} \leqslant \tilde{A} \leqslant \tilde{T}} \text{ TT-InSeq}$$

We know (Lemma 5 [Subtype Transitivity]) : $\tilde{X} \leqslant \tilde{T}$

$$\frac{S \leqslant S' \qquad \forall i \in \{1, \ldots, n\}. X_i \leqslant T_i}{?[\tilde{X}] \,.\, S \leqslant ?[\tilde{T}] \,.\, S'} \text{ AS-InSeq}$$

Let $\Gamma_4 = (\Gamma_6, x^q : ![\tilde{X}] \,.\, \overline{S}) + \tilde{z} : \tilde{V}$ where $\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} \tilde{z} : \tilde{V}$

$$\frac{(\Gamma_6, x^q : ![\tilde{X}] \,.\, \overline{S}) + \tilde{z} : \tilde{V} \vDash_{\mp} (x^q : ![\tilde{B}] \,.\, \overline{S'}) \,!\, [\tilde{z} : \tilde{U}] \,.\, Q_2}{\Gamma_6, x_q : \overline{S} \vdash Q_2 \qquad \overline{S} \leqslant \overline{S'} \qquad \tilde{V} \leqslant \tilde{U} \leqslant \tilde{A} \leqslant \tilde{X}} \text{ TT-OutSeq}$$

We know (Lemma 5 [Subtype Transitivity]) : $\tilde{U} \leqslant \tilde{X}$

$$\frac{\overline{S} \leqslant \overline{S'} \qquad \forall i \in \{1, \ldots, n\}. U_i \leqslant X_i}{![\tilde{X}] \,.\, \overline{S} \leqslant ![\tilde{U}] \,.\, \overline{S'}} \text{ AS-OutSeq}$$

$\Gamma_2 + (\Gamma_5, x^p : ?[\tilde{X}] \,.\, S) + ((\Gamma_6, x^q : ![\tilde{X}] \,.\, \overline{S}) + \tilde{z} : \tilde{V}) = \Gamma, \tilde{w} : \tilde{W}$

From the definition of $+$: $\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} x^p : ?[\tilde{X}] \,.\, S, x^q : ![\tilde{X}] \,.\, \overline{S}$

So, $\Gamma \vdash x^p : ?[\tilde{X}] \,.\, S, x^q : ![\tilde{X}] \,.\, \overline{S}$ or $(\nu x^{\pm} : ?[\tilde{X}] \,.\, S) \in (\nu\tilde{w} : \tilde{W})$
or $(\nu x^{\pm} : ![\tilde{X}] \,.\, \overline{S}) \in (\nu\tilde{w} : \tilde{W})$ for some $\tilde{X}$ and S such that
$?[\tilde{X}] \,.\, S \leqslant ?[\tilde{T}] \,.\, S'$ and $![\tilde{X}] \,.\, \overline{S} \leqslant ![\tilde{U}] \,.\, \overline{S'}$

We know (Lemma 5 [Subtype Transitivity]) : $\tilde{V} \leqslant \tilde{T}$

From the definition of $+$: $\Gamma, \tilde{w} : \tilde{W} \vDash_{\mp} \tilde{z} : \tilde{V}$

So, $\forall i \in \{1, \ldots, n\}. \Gamma \vDash_{\mp} z_i : V_i$ or $(\nu\tilde{z_i} : V_i) \in (\nu\tilde{w} : \tilde{W})$ where $V_i \leqslant T_i$

So, we have :

$$
\begin{array}{l}
p = \overline{q} \\
\text{and} \\
\forall i \in \{1, \ldots, n\}. \Gamma \vDash_{\mp} z_i : V_i \text{ or } (\nu \tilde{z_i} : V_i) \in (\nu \tilde{w} : \tilde{W}) \text{ where } V_i \leqslant T_i \\
\text{and} \\
\Gamma \vdash x^p : ?[\tilde{X}] \,.\, S, x^q : ![\tilde{X}] \,.\, \overline{S} \text{ or } (\nu x^{\pm} : ?[\tilde{X}] \,.\, S) \in (\nu \tilde{w} : \tilde{W}) \\
\quad \text{or } (\nu x^{\pm} : ![\tilde{X}] \,.\, \overline{S}) \in (\nu \tilde{w} : \tilde{W}) \text{ for some } \tilde{X} \text{ and } S \text{ such that} \\
\quad ?[\tilde{X}] \,.\, S \leqslant ?[\tilde{T}] \,.\, S' \text{ and } ![\tilde{X}] \,.\, \overline{S} \leqslant ![\tilde{U}] \,.\, \overline{S'}
\end{array}
$$

which is what we need.

Case 3 : $Q \equiv x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2$
We have :

$$\Gamma \vdash (\nu \tilde{w} : \tilde{W})((x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2) \mid R)$$

We need :

$$
\begin{array}{l}
p = \overline{q} \\
\text{and} \\
l_i \in \{l_1, \ldots, l_n\} \\
\text{and} \\
\Gamma \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n \rangle, x^q : \oplus \langle l_1 : S_1, \ldots, l_n : S_n \rangle \\
\qquad \text{or } (\nu x^{\pm} : \&\langle l_1 : S_1, \ldots, l_n : S_n \rangle) \in (\nu \tilde{w} : \tilde{W}) \\
\qquad \text{or } (\nu x^{\pm} : \oplus \langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n} \rangle) \in (\nu \tilde{w} : \tilde{W})
\end{array}
$$

$$\frac{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2) \mid R)}{\Gamma, \tilde{w} : \tilde{W} \vdash (x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2) \mid R}\ \text{T-New}$$

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vdash (x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2) \mid R}{\Gamma_1 \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2 \qquad \Gamma_2 \vdash R}\ \text{T-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$

$$\frac{\Gamma_1 \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2}{\Gamma_3 \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \qquad \Gamma_4 \vdash x^q \triangleleft l_i \,.\, Q_2}\ \text{T-Par}$$

As the above addition is defined, we know $p = \overline{q}$.

If $\Gamma_3 \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}$ then $\Gamma_3 \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle$.

If $\Gamma_4 \vdash x^q \triangleleft l_i \,.\, Q_2$ then $\Gamma_4 \vdash x^q : \oplus\langle l'_1 : S'_1, \ldots, l'_m : S'_m\rangle$ where $l_i \in \{l'_1, \ldots, l'_m\}$.

We know $\Gamma_3 + \Gamma_4 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$.

So, $\Gamma, \tilde{w} : \tilde{W} \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle$ and $\Gamma, \tilde{w} : \tilde{W} \vdash x^q : \oplus\langle l'_1 : S'_1, \ldots, l'_m : S'_m\rangle$.

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vdash (x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2) \mid R}{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^q \triangleleft l_i \,.\, Q_2) \mid R)}\ \text{T-New}$$

We know x^+ and x^- are either in Γ or are bound in $(\nu\tilde{w} : \tilde{W})$.

If x^+ and x^- are in Γ, then $x = \alpha$ and Γ is balanced with respect to x.

So, $m = n$ and for all $j \in \{1, \ldots, n\}$ $S'_j = \overline{S_j}$ and $l'_j = l_j$, and therefore

$l_i \in \{l_1, \ldots, l_n\}$, $\Gamma \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle$ and $\Gamma \vdash x^q : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_m}\rangle$.

If x^+ and x^- are bound in $(\nu\tilde{w} : \tilde{W})$ then $\Gamma, \tilde{w} : \tilde{W}(x^+) = \overline{\Gamma, \tilde{w} : \tilde{W}(x^-)}$.

So, $m = n$ and for all $j \in \{1, \ldots, n\}$ $S'_j = \overline{S_j}$ and $l'_j = l_j$, and therefore

$l_i \in \{l_1, \ldots, l_n\}$ and, $(\nu x^{\pm} : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle) \in (\nu\tilde{w} : \tilde{W})$ or

$(\nu x^{\pm} : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle) \in (\nu\tilde{w} : \tilde{W})$.

So, we have :

$$\begin{aligned}
&p = \overline{q}\\
&\text{and}\\
&l_i \in \{l_1, \ldots, l_n\}\\
&\text{and}\\
&\Gamma \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^q : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle\\
&\qquad \text{or } (\nu x^{\pm} : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle) \in (\nu\tilde{w} : \tilde{W})\\
&\qquad \text{or } (\nu x^{\pm} : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle) \in (\nu\tilde{w} : \tilde{W})
\end{aligned}$$

which is what we need.

Case 4 : $Q \equiv$ $\mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2$

We have :

$$\Gamma \vdash (\nu\tilde{w} : \tilde{W})((\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2) \mid R)$$

We need :

$$\Gamma \vdash x : \mathsf{bool} \text{ or } (\nu x : \mathsf{bool}) \in (\nu\tilde{w} : \tilde{W})$$

$$\frac{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2) \mid R)}{\Gamma, \tilde{w} : \tilde{W} \vdash (\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2) \mid R}\ \text{T-New}$$

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vdash (\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2) \mid R}{\Gamma_1 \vdash\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2 \qquad \Gamma_2 \vdash R}\ \text{T-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$

$$\frac{\Gamma_1 \vdash\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2}{\Gamma_1 \vdash x : \mathsf{bool} \qquad \Gamma_1 \vdash Q_1 \qquad \Gamma_1 \vdash Q_2}\ \text{T-Cond}$$

We know $\Gamma_1 \vdash x : \mathsf{bool}$ and $\Gamma_1 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$.

So, $\Gamma, \tilde{w} : \tilde{W} \vdash x : \mathsf{bool}$.

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vdash (\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2) \mid R}{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((\ \mathsf{if}\ x\ \mathsf{then}\ Q_1\ \mathsf{else}\ Q_2) \mid R)}\ \text{T-New}$$

We know x is either in Γ or is bound in $(\nu\tilde{w} : \tilde{W})$.

So, we have :

$$\Gamma \vdash x : \mathsf{bool} \text{ or } (\nu x : \mathsf{bool}) \in (\nu\tilde{w} : \tilde{W})$$

which is what we need.

□