Types for Correct Communication in Client-Server Systems

Simon Gay
Department of Computing Science
University of Glasgow
simon@dcs.gla.ac.uk

Malcolm Hole
Department of Computer Science
Royal Holloway, University of London
M.Hole@dcs.rhbnc.ac.uk

Technical Report

CSD-TR-00-07

December 18, 2000

Royal Holloway
University of London

Department of Computer Science
Egham, Surrey TW20 0EX, England

Abstract

We define an extension of the π-calculus which supports extended patterns of communication, such as client-server protocols. We then present a static type system for the language which allows concise high-level specifications of these protocols. The channels on which the constituent communications of these protocols take place are given linear types. This allows us to guarantee that a protocol will not fail to complete due to the pattern of communication between client and server being incompatible. We use the POP3 protocol as a concrete example of this technique. Finally, we discuss our plans to add subtyping to the system and to implement the type system as an addition to the Pict programming language.

Contents

List of Figures

1 Introduction

Following its early success as a framework for the investigation of the foundations of various concurrent programming styles, the π-calculus [7] has also become established as a vehicle for the exploration of type systems for concurrent programming languages [1, 4, 5, 6, 9, 13]. Inter-process communication in the π-calculus is based on point-to-point transmission of messages along named channels, and many proposed type systems have started from the assignment of types to channels, so that the type of a channel determines what kind of message it can carry. Because messages can themselves be channel names, this straightforward idea leads to detailed specifications of the intended uses of channels within a system. This line of research has not been purely theoretical: the Pict programming language [10] is directly based on the π-calculus and has a rich type system which incorporates subtyping on channel types and higher-order polymorphism.

Honda et al. [3, 12] have proposed π-calculus-like languages in which certain channels can be given *session types*. Such a channel, which we will call a *session channel*, is not restricted to carrying a single type of message for the whole of its lifetime. Instead, its type specifies a sequence of message types. Some of the messages might indicate choices between a range of possibilities, and different choices could lead to different specifications of the types of subsequent messages. Session types therefore have a branching structure.

While our work is based on session types, it differs from that of Honda et al. in that the majority of special treatment of session channels is handled by the typing rules and special syntax is restricted to a minimum. We anticipate that this will make it easier to achieve our goal of incorporating our type system into a modified version of the Pict language and compiler. Honda et al. use a special request and accept syntax for the purpose of establishing a channel. Their work also concentrates on modelling existing primitives such as remote procedure call and method invocation, and making implementaion of these in communication-based languages more elegant. Our aim is to look at the application of session types in the specification of complex protocols in client-server systems. In particular, we hope to extend the work presented here to include a system of subtyping to allow provision for extensible client-server systems which maintain backward compatibility, as detailed in Section 5.

The differences in syntax between our language and the π-calculus are small. A session channel is accessed via two ports which are distinct names in the environment and are distingushed by their polarity, either $+$ or $-$. The majority of special treatment of session channels is handled by the typing rules.

Consider a server for mathematical operations, which initially offers a choice between addition and negation. A client must choose an operation and then send the appropriate number of arguments to the server, which responds by sending back the result. All communications take place on a single session channel called x, whose session type is

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

More precisely, this is the type of the server side of the channel, x^+. The $\&\langle\ldots\rangle$ constructor specifies that a choice is offered between, in this case, two options, labelled plus and negate. Each label leads to a type which describes the subsequent communication on x. Note that the two branches have different types in which $?[\mathsf{int}]$ indicates receiving an integer, $![\mathsf{int}]$ indicates sending an integer, . is the sequencing constructor and end indicates the end of the interaction. The client side of the channel x^- has a dual or complementary type, written $\overline{S}$. Explicitly,

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

The $\oplus\langle\ldots\rangle$ constructor specifies that the client makes a choice between plus and negate. Again, each label is followed by a type which describes the subsequent interaction. The pattern of sending and receiving is the opposite of the pattern which appears on the server side.

An implementation of a maths server must use x^+ in accordance with the type S, and an implementation of a client must use x^- in accordance with the type $\overline{S}$. These requirements can be enforced by static typechecking, and it is then guaranteed that no communication errors will occur at runtime.

Generally, for any type correct client-server system, if either a client or server chooses a label, it is guaranteed to be one of the labels offered by the other. Similarly, when one sends a message, it is guaranteed to be of the type expected by the other.

The typing rules to be introduced in Section 3 will allow the derivation of

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

where

$$\begin{array}{rcl} \mathsf{server} & = & x^+ \triangleright \{\ \ \mathsf{plus}\colon x^+\ ?\ [a : \mathsf{int}]\,.\,x^+\ ?\ [b : \mathsf{int}]\,.\,x^+\ !\ [a+b]\,.\,\mathbf{0}, \\ & & \qquad\ \ \mathsf{negate}\colon x^+\ ?\ [a : \mathsf{int}]\,.\,x^+\ !\ [-a]\,.\,\mathbf{0}\} \end{array}$$

The operation $\triangleright$ allows a message on x^+ to be any of the listed alternatives, making a choice as to how the process continues. The labels are the same as those in S, and the pattern of inputs ($x^+\ ?\ [a : \mathsf{int}]\ldots.$) and outputs ($x^+\ !\ [a+b]\ldots.$) matches that in S.

One possible definition of a client is

$$\mathsf{client} = x^- \triangleleft \mathsf{negate} \,.\, x^- \,!\, [2] \,.\, x^- \,?\, [a : \mathsf{int}] \,.\, \mathbf{0}$$

and we can derive the typing judgement

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

Note the use of $\triangleleft$ to select from the available options, and that the subsequent pattern of inputs and outputs matches the specification in $\overline{S}$. This client does not do anything with the value received from the server. More realistically, $\mathbf{0}$ would be replaced by some continuation process which used a.

The client and the server can be put in parallel using the typing rule T-PAR for parallel composition.

$$\frac{x^+ : S \vdash \mathsf{server} \quad x^- : \overline{S} \vdash \mathsf{client}}{x^+ : S, x^- : \overline{S} \vdash \mathsf{server} \mid \mathsf{client}} \text{ T-PAR}$$

We will see later that putting another copy of client in parallel with this system is not a valid application of the typing rule T-PAR. The system server | client | client is erroneous because both clients are trying to use x^- to communicate with the same server. If this system is executed, one client could use x^- to choose either plus or negate. After that, the server expects to receive an integer on x^+, but the other client could again use x^- to choose between plus and negate. This is a runtime type error of the kind that the type system is designed to avoid.

To avoid this class of error we must ensure that each communication capability in a session type is used exactly once. Our typing rules use techniques similar to those Kobayashi et al. [5] use to enforce linearity [2]. 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, $y : \,\hat{}\,[\mathsf{int}]$ is a channel which can be used freely to send or receive integers.

How, then, can we implement a server which can be used by more than one client? One might imagine that replicating the server would help, as !(server) | client | client can evolve to !(server) | (server | client) | (server | client) and there is then a server for each client. But this does not solve the problem, because both servers and both clients are still using the same channel, x. There is, therefore, no guarantee that consecutive messages from one client will both go to the same server.

The solution is for each client to create a session channel which it will use for its own interaction with the server. The server consists of a replicated thread process. Each copy of thread receives a session channel along a channel called $\mathsf{trigger}$ and uses it to interact with a single client.

$$
\begin{array}{rcl}
\mathsf{thread} & = & \mathsf{trigger}\ ?\ [x^+ : S]\ .\ \mathsf{server} \\
\mathsf{server} & = & x^+ \triangleright \{\mathsf{plus}{:}\, x^+\ ?\ [a : \mathsf{int}]\ .\ x^+\ ?\ [b : \mathsf{int}]\ .\ x^+\ !\ [a+b]\ .\ \mathbf{0}, \\
& & \qquad \mathsf{negate}{:}\, x^+\ ?\ [a : \mathsf{int}]\ .\ x^+\ !\ [-a]\ .\ \mathbf{0}\} \\
\mathsf{newserver} & = & !\mathsf{thread} \\
\mathsf{client1body} & = & y^- \triangleleft \mathsf{negate}\ .\ y^-\ !\ [2]\ .\ y^-\ ?\ [a : \mathsf{int}]\ .\ \mathbf{0} \\
\mathsf{client1} & = & (\nu y^\pm : S)\mathsf{trigger}\ !\ [y^+]\ .\ \mathsf{client1body} \\
\mathsf{client2body} & = & z^- \triangleleft \mathsf{plus}\ .\ z^-\ !\ [1]\ .\ z^-\ !\ [2]\ .\ z^-\ ?\ [b : \mathsf{int}]\ .\ \mathbf{0} \\
\mathsf{client2} & = & (\nu z^\pm : S)\mathsf{trigger}\ !\ [z^+]\ .\ \mathsf{client2body}
\end{array}
$$

Now

$$\mathsf{trigger} : {}^\frown[S] \vdash \mathsf{newserver} \mid \mathsf{client1} \mid \mathsf{client2}$$

is a valid typing judgement. Or, to finish the derivation of a complete program

$$\vdash (\nu\mathsf{trigger} : {}^\frown[S])(\mathsf{newserver} \mid \mathsf{client1} \mid \mathsf{client2})$$

Because $\mathsf{trigger}$ does not have a session type, it can be used by all three processes. When this system is executed, $\mathsf{client1}$ sends the y^+ port of the local channel y to one copy of thread. The standard π-calculus scope extrusion allows the scope of y^+ to expand to include that copy of thread. $\mathsf{client1body}$ and server then have a private interaction on y. Similarly $\mathsf{client2}$ and another copy of thread use the private channel z for their communication. Notice that y is a session channel with two ports in $\mathsf{client1}$, and indeed both sides of y are used. y^+, whose type is S, is used by being sent away on $\mathsf{trigger}$, and y^-, whose type is $\overline{S}$, is used for communication by $\mathsf{client1body}$. Finally, note that x^+ is bound at the point of input in thread and has no complementary port, exchanging messages instead with either y^- or z^-, as appropriate. It is arbitrarily given the polarity $+$ to indicate it is a session channel.

The remainder of the paper is organised as follows. Section 2 defines the syntax of processes, types and environments, and some basic operations on them. The typing 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 the POP3 protocol, and discusses the role of subtyping. Finally we discuss related work, and outline our future plans, in Section 6.

2 Syntax and Notation

Our language is based on a polyadic π-calculus with output prefixing [7]. We omit the original π-calculus choice construct $P + Q$, partly in order to keep the language close to the core of Pict [10]. However, we have the constructs introduced in Section 1 for choosing between a collection of labelled processes, as proposed by Honda et al. [3, 12]. We also omit the matching construct, which allows channel names to be tested for equality, again because it is not present in core Pict. 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. It is convenient to add a conditional process expression, written $\mathsf{if}\ b\ \mathsf{then}\ P\ \mathsf{else}\ Q$, where b is a boolean value, and therefore we also have a ground type of booleans. Other ground types, such as int as used in the examples in Section 1, could be added along with appropriate primitive operations. As is standard, we use the replication operator ! instead of recursive process definitions.

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, and $\tilde{x} : \tilde{T}$ for a finite sequence $x_1^{p_1} : T_1, \ldots, x_n^{p_n} : T_n$ of typed names.

2.1 Processes

The syntax of processes is defined by the following grammar. Note that T , $\tilde{T}$ and S stand for types, lists of types and session types, which have not yet been defined.

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

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. 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.

As usual we define a *structural congruence* relation, written $\equiv$, which helps to define the operational semantics. It is the smallest congruence (on α-equivalence classes of processes) closed under the following rules.

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

In rule S-Offer, σ is a permutation on $\{1, \ldots, n\}$.

2.2 Types

The syntax of types is defined by the following grammar.

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

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, $\mathit{true} : \mathsf{bool}$ and $\mathit{false} : \mathsf{bool}$ appear in every environment.

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

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

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 The Type System

The typing rules are defined in Figure 1. T-Nil types the process **0**, providing 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 and T-In are standard as they deal with input and output on a non-session channel. T-OutSeq and T-InSeq are similar to the output and input rules for non-session channels, but the type of the session port in the environment has an output or input prefix added as appropriate. T-Offer types a process offering a choice between a number of processes on a channel providing the type associated with each process is appropriate. Conversely, T-Choose types a process that chooses between a number of processes providing the type associated with the chosen process is correct. Finally, T-Cond types a conditional expression providing the condition has type bool and that the processes representing the two possible outcomes are well typed, and T-Rep types the replication of a process providing the process is well typed in an environment containing no session channels.

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

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

$$\frac{\Gamma \vdash P \quad \Gamma \vdash x : \hat{}[\tilde{T}]}{\Gamma + \tilde{y} : \tilde{T} \vdash x\,!\,[\tilde{y}]\,.\,P}\ \text{T-OUT} \qquad \frac{\Gamma, \tilde{y} : \tilde{T} \vdash P \quad \Gamma \vdash x : \hat{}[\tilde{T}]}{\Gamma \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,P}\ \text{T-IN}$$

$$\frac{\Gamma, x^p : S \vdash P}{(\Gamma, x^p : ![\tilde{T}]\,.\,S) + \tilde{y} : \tilde{T} \vdash x^p\,!\,[\tilde{y}]\,.\,P}\ \text{T-OUTSEQ} \qquad \frac{\Gamma, x^p : S, \tilde{y} : \tilde{T} \vdash P}{\Gamma, x^p : ?[\tilde{T}]\,.\,S \vdash x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,P}\ \text{T-INSEQ}$$

$$\frac{\Gamma, x^p : S_1 \vdash P_1; \ldots; \Gamma, x^p : S_n \vdash P_n}{\Gamma, x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}\ \text{T-OFFER}$$

$$\frac{\Gamma, x^p : S_i \vdash P \quad 1 \leq i \leq n}{\Gamma, x^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^p \triangleleft l_i\,.\,P}\ \text{T-CHOOSE}$$

$$\frac{\Gamma \vdash x : \mathsf{bool} \quad \Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash \ \mathsf{if}\ x\ \mathsf{then}\ P\ \mathsf{else}\ Q}\ \text{T-COND} \qquad \frac{\Gamma \vdash P \quad \Gamma \text{ unlimited}}{\Gamma \vdash !P}\ \text{T-REP}$$

Figure 1 Typing rules

In rules T-OUT and T-OUTSEQ, the names being output are added to the environment. This means that if a session channel is output, then the part of its usage which is given away 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 channel. Of course, the first process must not use the channel again. Such behaviour arises when a recursive process, which uses a session channel with a recursive type, is represented in terms of replication. When the next instance of the recursive process is invoked, the session channel must be passed on. Below is a recursive maths server similar to the non-recursive example in Section 1.

$$\begin{array}{rcll} S & = & \mu X.\&\langle & \mathsf{plus}:?[\mathsf{int}]\,.\,?[\mathsf{int}]\,.\,![\mathsf{int}]\,.\,X, \\ & & & \mathsf{negate}:?[\mathsf{int}]\,.\,![\mathsf{int}]\,.\,X, \\ & & & \mathsf{stop}:\mathsf{end} \quad \rangle \end{array}$$

$$
\begin{array}{rcl}
\mathsf{thread} & = & \mathsf{trigger}\ ?\ [x^+ : S]\ .\ \mathsf{server} \\
\mathsf{server} & = & x^+ \triangleright \{\mathsf{plus}\!:\! x^+\ ?\ [a : \mathsf{int}]\ .\ x^+\ ?\ [b : \mathsf{int}]\ .\ x^+\ !\ [a+b]\ .\ \mathsf{trigger}\ !\ [x^+]\ .\ \mathbf{0}, \\
 & & \qquad \mathsf{negate}\!:\! x^+\ ?\ [a : \mathsf{int}]\ .\ x^+\ !\ [-a]\ .\ \mathsf{trigger}\ !\ [x^+]\ .\ \mathbf{0}, \\
 & & \qquad \mathsf{stop}\!:\!\mathbf{0}\} \\
\mathsf{recserver} & = & !\mathsf{thread} \\
\mathsf{clientbody} & = & y^- \triangleleft \mathsf{plus}\ .\ y^-\ !\ [2]\ .\ y^-\ !\ [3]\ .\ y^-\ ?\ [a : \mathsf{int}]\ .\ y^- \triangleleft \mathsf{negate}\ .\ y^-\ !\ [a]\ .\ y^-\ ?\ [b : \mathsf{int}]\ . \\
 & & y^- \triangleleft \mathsf{stop}\ .\ \mathbf{0} \\
\mathsf{client} & = & (\nu y^\pm : S)\mathsf{trigger}\ !\ [y^+]\ .\ \mathsf{clientbody}
\end{array}
$$

The complete system can then be represented by the following typing judgement

$$\vdash (\nu \mathsf{trigger} : {}^\frown[S])(\mathsf{recserver} \mid \mathsf{client})$$

The recursive server is very similar to the non-recursive one. Notice, however, that before coming to the end of each recursive part, the server effectively restarts itself by sending the session channel, x^+, on trigger.

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. Consider again the applications of T-Par to the non-threaded maths server and client in Section 1. The first, server | client, is correct because the resulting environment, $x^+ : S, x^- : \overline{S}$, is defined. The second, server | client | client is incorrect, however. The rule cannot be applied as shown below, because the addition forming the resulting environment is not defined. This prevents the session channel x from being used simultaneously by both copies of client.

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

4 Operational Semantics

As usual for π-calculus-like languages, the operational semantics is defined by means of a reduction relation [6]. $P \xrightarrow{\alpha,l} Q$ means that the process P reduces to the process Q by executing a single communication step (or evaluating a conditional expression). The labels on the reduction are as follows. 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 2. 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.

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

In order to prove type soundness we will need to prove a subject reduction theorem showing that if $\Gamma \vdash P$ and $P \xrightarrow{\alpha,l} P'$ then there is an environment Γ' such that $\Gamma' \vdash P'$. We will then prove that if $\Gamma \vdash P$ then the immediately available communications in P do not cause type errors. Together these results imply that a well-typed process can be executed safely through any sequence of reduction steps.

Unlike other type systems for the π-calculus, both theorems include an additional condition that, for a reduction $P \xrightarrow{\alpha,l} Q$, the environment in which P is typed must be balanced with respect to α, or α must be τ. The reason for this is as follows: We are only interested in reduction derivations for processes obtainable by any number of reductions of a program. Before any reductions have occured, all programs are typeable in the empty environment. All session channels are balanced at the point of introduction to the environment (using the T-NEWSEQ rule) and all reductions on the channel will maintain the balance between the two ports of the channel (as a reduction simply removes the first item from the type

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

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

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

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

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

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

Figure 2 The reduction relation

of each port). Therefore, at all stages in the derivation of a reduction resulting from the execution of a program, we know that the environment will be balanced with respect to the channel on which the reduction takes place. So, although processes that do not meet this condition can be typeable, and indeed may have reductions, they are not of interest as they do not constitute a step in the reduction derivation of a program or its execution steps.

The following lemmas are steps towards establishing Theorem 1:

Lemma 1 (Substitution) *If* $\Gamma, x : T \vdash P$ *then* $\Gamma + y : T \vdash P\{y/x\}$

Lemma 2 (Structural Equivalence) *If* $\Gamma \vdash P$ *and* $P \equiv Q$ *then* $\Gamma \vdash Q$

The proofs of theses lemmas and the following theorem can be found in Appendix A.

Theorem 1 (Subject Reduction) *If $\Gamma \vdash P$ and $P \xrightarrow{\alpha,l} P'$ where Γ is balanced with respect to α (if $\alpha \neq \tau$), then $\Gamma' \vdash P'$ where*

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

The proof of the following theorem can be found in Appendix B.

Theorem 2 (Run-Time Safety) *If $\Gamma \vdash P$, $P \equiv (\nu\tilde{w} : \tilde{W})(Q \mid R)$, $Q \xrightarrow{\alpha,l} Q'$ and Γ is balanced with respect to α (if $\alpha \neq \tau$), then*

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

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

and

$\Gamma \vdash x : \hat{}[\tilde{T}]$ *or* $(\nu x : \hat{}[\tilde{T}]) \in (\nu\tilde{w} : \tilde{W})$

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

$p = \overline{q}$

and

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

and

$\Gamma \vdash x^p : ?[\tilde{T}] \,.\, S, x^q : ![\tilde{T}] \,.\, \overline{S}$ *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

$\Gamma \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^q : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle$

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

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

5 The POP3 Protocol

As a more substantial example, we will now use our type system to specify the POP3 protocol [8]. 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. 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. Note that the POP3 protocol does not deal with sending messages or routing messages through a network.

5.1 Types for the Protocol

To specify the behaviour of a channel which can be used for a POP3 session, we use the following type definitions:

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

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. In addition to interacting with clients, the POP3 server must also retrieve messages from the message database. We can specify a separate session type, M, for this.

$$\begin{array}{rcll} M & = & ?[\mathsf{str}] \,.\, \mu X.\&\langle & \mathsf{getstats} : ![\mathsf{int},\mathsf{int}] \,.\, X, \\ & & & \mathsf{getmessagestats} : ?[\mathsf{int}] \,.\, ![\mathsf{int},\mathsf{int}] \,.\, X, \\ & & & \mathsf{getnoofmessages} : ![\mathsf{int}] \,.\, \mu Y.\oplus\langle \quad \mathsf{message} : ![\mathsf{int},\mathsf{int}] \,.\, Y, \\ & & & \qquad\qquad\qquad\qquad\qquad\qquad\quad \mathsf{end} : X \quad \rangle, \\ & & & \mathsf{getamessage} : ?[\mathsf{int}] \,.\, ![\mathsf{str},\mathsf{str}] \,.\, X, \\ & & & \mathsf{deletemessage} : ?[\mathsf{int}] \,.\, ![\mathsf{bool}] \,.\, X, \\ & & & \mathsf{reset} : X, \\ & & & \mathsf{exit} : \mathbf{0} \quad \rangle \end{array}$$

As in this case the message database is the server and the POP3 server the client, M is the type used by the message database and $\overline{M}$ is that used by the POP3 server.

5.2 POP3 with Session Channels

We can use the above type definitions in the implementation of a POP3 server as follows:

$$\begin{array}{rcl} \mathsf{POP3server} & = & !\mathsf{thread} \\ \mathsf{thread} & = & \mathsf{trigger}\,?\,[x^+ : A] \,.\, \mathsf{auth} \\ \mathsf{auth} & = & x^+ \triangleright \{\, \mathsf{quit} : x^+ \triangleleft \mathsf{ok} \,.\, x^+\,!\,[\text{“+OK Server Signing Off”}] \,.\, \mathbf{0}, \\ & & \quad \mathsf{user} : x^+\,?\,[\mathsf{name} : \mathsf{str}] \,. \\ & & \qquad \mathsf{if}\ \mathsf{checkname}![\mathsf{name}]\ \mathsf{then} \\ & & \qquad\quad x^+ \triangleleft \mathsf{ok} \,.\, x^+![\text{“+OK Valid Mailbox”}]. \\ & & \qquad\quad x^+ \triangleright \{\, \mathsf{quit} :\ x^+ \triangleleft \mathsf{ok}. \\ & & \qquad\qquad\qquad\quad x^+![\text{“+OK Server Signing Off”}] \,.\, \mathbf{0}, \\ & & \qquad\qquad\quad \mathsf{pass} : x^+?[\mathsf{pwd} : \mathsf{str}]. \\ & & \qquad\qquad\qquad\quad \mathsf{if}\ \mathsf{checkpwd}![\mathsf{name},\mathsf{pwd}]\ \mathsf{then} \\ & & \qquad\qquad\qquad\qquad x^+ \triangleleft \mathsf{ok}. \\ & & \qquad\qquad\qquad\qquad x^+![\text{“+OK Mailbox Locked and Ready”}]. \\ & & \qquad\qquad\qquad\qquad \mathsf{transtrig} \\ & & \qquad\qquad\qquad\quad \mathsf{else} \\ & & \qquad\qquad\qquad\qquad x^+ \triangleleft \mathsf{error}. \\ & & \qquad\qquad\qquad\qquad x^+![\text{“-ERR Invalid Password”}] \,.\, \mathsf{auth}\,\} \\ & & \qquad \mathsf{else} \\ & & \qquad\quad x^+ \triangleleft \mathsf{error} \,.\, x^+![\text{“-ERR Unknown Mailbox”}] \,.\, \mathsf{auth}\,\} \end{array}$$

$$
\begin{array}{rcl}
\mathsf{transtrig} & = & (\nu y^{\pm} : M)(\mathsf{dbtrigger}\,!\,[y^{+}]\,.\,y^{-}\,!\,[\mathsf{name}]\,.\,\mathsf{trans}) \\
\mathsf{trans} & = & x^{+} \rhd \{\, \mathsf{stat} : x^{+} \lhd \mathsf{ok}\,.\,y^{-} \lhd \mathsf{getstats}\,.\,y^{-}?[\mathsf{num} : \mathsf{int}, \mathsf{size} : \mathsf{int}]\,.\,x^{+}![\mathsf{num}, \mathsf{size}]. \\
& & \qquad\qquad \mathsf{trans}, \\
& & \quad \mathsf{list} : x^{+} \rhd \{\, \mathsf{one} : x^{+}?[\mathsf{num} : \mathsf{int}]\,.\,y^{-} \lhd \mathsf{getmessagestats}. \\
& & \qquad\qquad\qquad y^{-}![\mathsf{num}]\,.\,y^{-}?[\mathsf{m} : \mathsf{int}, \mathsf{size} : \mathsf{int}]. \\
& & \qquad\qquad\qquad \mathsf{if}\ \mathsf{m} = 0\ \mathsf{then} \\
& & \qquad\qquad\qquad\quad x^{+} \lhd \mathsf{error}\,.\,x^{+}![\text{“-ERR No Such Message”}]\,.\,\mathsf{trans} \\
& & \qquad\qquad\qquad \mathsf{else} \\
& & \qquad\qquad\qquad\quad x^{+} \lhd \mathsf{ok}\,.\,x^{+}![\mathsf{m}, \mathsf{size}]\,.\,\mathsf{trans}, \\
& & \qquad\qquad\quad \mathsf{all} : y^{-} \lhd \mathsf{getnoofmessages}\,.\,y^{-}?[\mathsf{n} : \mathsf{int}]\,.\,x^{+}![\mathsf{n}]\,.\,\mathsf{list}\,\}, \\
& & \quad \mathsf{retr} : x^{+}?[\mathsf{num} : \mathsf{int}]\,.\,y^{-} \lhd \mathsf{getamessage}\,.\,y^{-}![\mathsf{num}]\,.\,y^{-}?[\mathsf{m} : \mathsf{str}, \mathsf{size} : \mathsf{str}]. \\
& & \qquad\quad \mathsf{if}\ \mathsf{m} = \emptyset\ \mathsf{then} \\
& & \qquad\qquad x^{+} \lhd \mathsf{error}\,.\,x^{+}![\text{“-ERR No Such Message”}]\,.\,\mathsf{trans} \\
& & \qquad\quad \mathsf{else} \\
& & \qquad\qquad x^{+} \lhd \mathsf{ok}\,.\,x^{+}![\mathsf{m}]\,.\,x^{+}![\mathsf{size}]\,.\,\mathsf{trans}, \\
& & \quad \mathsf{dele} : x^{+}?[\mathsf{num} : \mathsf{int}]\,.\,y^{-} \lhd \mathsf{deletemessage}\,.\,y^{-}![\mathsf{num}]\,.\,y^{-}?[\mathsf{b} : \mathsf{bool}]. \\
& & \qquad\quad \mathsf{if}\ \mathsf{b} = \mathit{false}\ \mathsf{then} \\
& & \qquad\qquad x^{+} \lhd \mathsf{ok}\,.\,x^{+}![\text{“+OK Message Deleted”}]\,.\,\mathsf{trans} \\
& & \qquad\quad \mathsf{else} \\
& & \qquad\qquad x^{+} \lhd \mathsf{error}\,.\,x^{+}![\text{“-ERR No Such Message”}]\,.\,\mathsf{trans}, \\
& & \quad \mathsf{noop} : x^{+} \lhd \mathsf{ok}\,.\,x^{+}![\text{“+OK”}]\,.\,\mathsf{trans}, \\
& & \quad \mathsf{rset} : y^{-} \lhd \mathsf{reset}\,.\,x^{+} \lhd \mathsf{ok}\,.\,x^{+}![\text{“+OK”}]\,.\,\mathsf{trans}, \\
& & \quad \mathsf{quit} : x^{+} \lhd \mathsf{ok}\,.\,x^{+}![\text{“+OK Server Signing Off”}]\,.\,\mathsf{update}\,\} \\
\mathsf{list} & = & y^{-} \rhd \{\mathsf{message} : y^{-}?[\mathsf{m} : \mathsf{int}, \mathsf{size} : \mathsf{int}]\,.\,x^{+} \lhd \mathsf{message}\,.\,x![\mathsf{m}, \mathsf{size}]\,.\,\mathsf{list}, \\
& & \quad \mathsf{end} : x^{+} \lhd \mathsf{end}\,.\,\mathsf{trans}\,\} \\
\mathsf{update} & = & y^{-} \lhd \mathsf{exit}\,.\,\mathbf{0}
\end{array}
$$

To simplify the above example, we allow equality testing in conditional statements. We also use responsive output channels *à la* Pict [10]. Where the type of a channel includes a responsive output channel, sending a message on the channel evaluates to an expression of the type carried on the responsive output channel. For example, the channel checkname has type $\hat{}$[str, /bool], where / indicates a responsive output channel. Therefore, sending a string on the channel evaluates to a boolean expression.

These definitions illustrate the complex structure possible for session types, and show the use of recursive session channels to describe repetitive session behaviour. The server both offers and makes choices, in contrast to the example in Section 1. For a client to be type correct, it must use its end of the channel in accordance with the type $\overline{A}$. This ensures that all messages sent by the client will be of the type expected by the server and that the appropriate labels are offered whenever the server makes a choice. 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, however, because the different continuation types after ok and error are essential for an accurate description of the protocol's behaviour. Aside from the type of the channel it uses, we do not concern ourselves with the operation of the message database as, after receiving the name of the mailbox, it simply services requests from the POP3 server.

In similar fashion to the example in Section 1, we say

$$\mathsf{trigger} : {}^\frown[A] \vdash \mathsf{POP3}$$

which can be published as the specification of the server and its protocol.

5.3 POP3 without Session Channels

We can implement a POP3 server without session channels as follows:

$$
\begin{array}{rcl}
\mathsf{POP3server} & = & !\mathsf{thread} \\
\mathsf{thread} & = & \mathsf{trigger}?[\,\mathsf{command} : {}^\frown[\mathsf{str}], \mathsf{ok} : {}^\frown[\mathsf{str}], \mathsf{user} : {}^\frown[\mathsf{str}], \mathsf{pass} : {}^\frown[\mathsf{str}], \\
& & \qquad\quad \mathsf{continue} : {}^\frown[{}^\frown[\mathsf{int},\mathsf{int}], {}^\frown[\mathsf{int}], {}^\frown[\mathsf{int},\mathsf{int}], {}^\frown[\mathsf{str}]\,]\,]\;.\;\mathsf{auth} \\
\mathsf{auth} & = & \mathsf{command}?[\mathsf{c1} : \mathsf{str}]. \\
& & \mathsf{if}\ \mathsf{c1} = \text{“quit”}\ \mathsf{then} \\
& & \quad \mathsf{ok}![\text{“+OK Server Signing Off”}]\;.\;\mathbf{0} \\
& & \mathsf{else\ if}\ \mathsf{c1} = \text{“user”}\ \mathsf{then} \\
& & \quad \mathsf{user}?[\mathsf{name} : \mathsf{str}].\,\mathsf{if}\ \mathsf{checkname}![\mathsf{name}]\ \mathsf{then} \\
& & \qquad\qquad\qquad\qquad \mathsf{ok}![\text{“+OK Valid Mailbox”}]\;.\;\mathsf{command}?[\mathsf{c2} : \mathsf{str}]. \\
& & \qquad\qquad\qquad\qquad \mathsf{if}\ \mathsf{c2} = \text{“quit”}\ \mathsf{then} \\
& & \qquad\qquad\qquad\qquad\quad \mathsf{ok}![\text{“+OK Server Signing Off”}]\;.\;\mathbf{0} \\
& & \qquad\qquad\qquad\qquad \mathsf{else\ if}\ \mathsf{c2} = \text{“pass”}\ \mathsf{then} \\
& & \qquad\qquad\qquad\qquad\quad \mathsf{pass}?[\mathsf{pwd} : \mathsf{str}]. \\
& & \qquad\qquad\qquad\qquad\quad \mathsf{if}\ \mathsf{checkpass}![\mathsf{name},\mathsf{pwd}]\ \mathsf{then} \\
& & \qquad\qquad\qquad\qquad\qquad \mathsf{ok}![\text{“+OK Mailbox Locked and Ready”}]. \\
& & \qquad\qquad\qquad\qquad\qquad \mathsf{transtrig} \\
& & \qquad\qquad\qquad\qquad\quad \mathsf{else} \\
& & \qquad\qquad\qquad\qquad\qquad \mathsf{error}![\text{“-ERR Invalid Password”}]\;.\;\mathsf{auth}
\end{array}
$$

```
                              else
                                error![“-ERR Invalid Command”] . auth
                            else
                              error![“-ERR Unknown Mailbox”] . auth
               else
                 error![“-ERR Invalid Command”] . auth

transtrig  =   continue?[ stats : ^[int, int], message : ^[int], list : ^[int, int], retr : ^[str] ] . trans
    trans  =   command?[c3 : str].
               if c3 = “stat” then
                 (νgetnum : ^[int])(νgetsize : ^[int])
                 ( getstats![name, getnum, getsize] . getnum?[num : int] . getsize?[size : int].
                   stats![num, size] . trans )
               else if c3 = “list” then
                 command?[c4 : str].
                 if c4 = “one” then
                   message?[num : int].(νreadmessage : ^[int, int])
                                      ( getmessagestats![num, readmessage].
                                        readmessage?[m : int, size : int].
                                        if m = 0 then
                                          error![“-ERR No Such Message”] . trans
                                        else
                                          list![m, size] . trans )
                 else if c4 = “all” then
                   (νhowmany : ^[int])
                   ( getnoofmessages![howmany] . howmany?[n : int] . message![n] . list )
                 else
                   error![“-ERR Invalid Command”]
               else if c3 = “retr” then
                 message?[num : int].(νgetmessage : ^[str, str])
                                    ( getamessage![num, getmessage].
                                      getmessage?[m : str, size : str].
                                      if m = ∅ then
                                        error![“-ERR No Such Message”] . trans
                                      else
                                        retr![m] . retr![size] . trans
```

```
else if c3 = "dele" then
  message?[num : int].if deletemessage![num] then
                          ok!["+OK Message Deleted"] . trans
                        else
                          error!["-ERR No Such Message"]
else if c3 = "noop" then
  ok!["+OK"] . trans
else if c3 = "rset" then
  rset![] . ok!["+OK"] . trans
else if c3 = "quit" then
  ok!["+OK Server Signing Off"] . update
else
  error!["-ERR Invalid Command"] . trans
```

```
  list   =  (νnextstats : ^[int, int])
            ( getnextmessagestats![nextstats] . nextstats?[m : int, size : int].
              if m = 0 then
                 trans
              else
                 list![m, size]  )
update   =  exit![] . 0
```

The server uses many channels to communicate with the message database - one for each function offered by the server. We assume these are private channels known only to the server and database, and to aid comparison with the server using session channels, these are given names corresponding to the labels in the session example.

The code using session channels is shorter than that without sessions. We would also argue that it is more readable, as it is more similar to a case expression than the nested ifs used in the non-session example. (It would, however, be possible to add a case expression to the π-calculus without adding session types.) Using a single channel for all communications between a client and server also improves the readability: it is immediately clear that any message sent by the server on x is for the POP3 client and any message sent on y is for the message database.

As we said in Section 1, we hope to incorporate session types into a modified version of the Pict compiler. It is therefore worth noting that session channels reduce the overheads associated with initialising client-server interactions. In the non-session example, the client must create nine channels and send them to the server. The server must create another five channels and send these to the message database. This contrasts with the session example, where the client only needs to create a single channel for communication with the server, and the server creates just one for communication with the message database.

The advantages of most type systems lie in the enforcement of certain properties, removing the responsibility to enforce these from the programmer. Because the type of a session channel imposes an order on the messages sent on the channel, a type correct process will be guaranteed to send messages on the channel in the order that the recipient expects - it will not be up to the programmer to ensure this. Similarly, where a type correct process offers a choice between labels on a channel, it is guaranteed to offer all choices that the chooser could make. Because this is not true without session channels, whenever we read a command from the client in the non-session POP3 example, we must include a default else statement at the end to deal with cases where the client's chosen command is not provided by the server.

5.4 Subtyping

Protocols in computing are frequently subject to ongoing revision. This can create situations where different servers and clients implement a different range of services depending on the version of the protocol that they comply with. Ideally we would like all clients to be compatible with all new versions of the server. The type system for session types we have presented in this report does not currently support this, but it is our intention that this feature should be added by means of a subtype relation, $\leqslant$. This will include subtyping on ordinary channels as defined by Pierce and Sangiorgi [9], but will be extended to deal with session channels. Figure 3 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 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.

As an example of how subtyping can be used for this purpose, the POP3 protocol permits an alternative authentication scheme to that specified in the previous example. It is accessed by the apop command and allows the client to send a mailbox name 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 3 Inference rules for subtyping 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

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

A server, newPOP3 which implements the apop command can be typed as follows:

$$\mathsf{port} : {}^\frown[B] \vdash \mathsf{newPOP3}$$

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

$$\mathsf{port} : {}^\frown[A], x^{-} : \overline{A} \vdash \mathsf{clientbody}$$

This gives

$$\mathsf{port} : {}^\frown[A] \vdash \mathsf{client}$$

The following derivation shows that client can also be typed in the environment $\mathsf{port} : {}^\frown[B]$, and can therefore be put in parallel with newPOP3. We can assume

$$\mathsf{port} : {}^\frown[B], x^{-} : \overline{A} \vdash \mathsf{clientbody}$$

as port is not a session channel and is not used in clientbody.

$$\dfrac{\dfrac{\mathsf{port} : {}^\frown[B], x^{-} : \overline{A} \vdash \mathsf{clientbody} \qquad \mathsf{port} : {}^\frown[B], x^{-} : \overline{A}(\mathsf{port}) \leqslant ![A]}{\mathsf{port} : {}^\frown[B], x^{+} : A, x^{-} : \overline{A} \vdash \mathsf{port}\,!\,[x^{+}]\,.\,\mathsf{clientbody}}\ \text{T-OUT}}{\mathsf{port} : {}^\frown[B] \vdash (\nu x^{\pm} : A)\mathsf{port}\,!\,[x^{+}]\,.\,\mathsf{clientbody}}\ \text{T-NEW}$$

The point to note is that the T-OUT rule has been modified to allow the type of port, ${}^\frown[B]$, to be a subtype of $![A]$, the type of a channel that can output a message of type A. This is true because $A \leqslant B$ and, if we look at the AS-OUT rule, output behaves contravariantly, as in Pierce and Sangiorgi's type system for the π-calculus [9].

So, in addition to the benefits of using session channels described previously, session types with subtyping allow us to add extra features to a new version of a server in the form of extra labels in a choice. Providing all previous labels are maintained, the type of the new server will be a supertype of the old server. This means that a client expecting an interaction with the old server will provide the new server with a channel which is a subtype of what is expected, and therefore the system will still be type correct.

6 Conclusions and Future Work

We have defined a language whose type system incorporates session types, as suggested by Honda et al. [3, 12]. 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. [5] 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. A session type for the POP3 protocol and the definition of the POP3 server were given as a concrete example of this. Finally, we have defined a subtype relation and shown how subtyping increases the utility of these ideas: 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.

The present paper is the first report of our work on a longer term project to investigate advanced type systems in the context of the Pict programming language. Our next goal is to extend the typing rules to incorporate the ideas for subtyping shown in chapter 5. We then hope to implement a type checker for the type system including subtyping. We have already produced a version of the Pict compiler without polymorphism, as we hope this will simplify the integration of our type system with it, which is one of our principal goals. Beyond this, there are many possible areas to look at. We hope to study the interaction of session types and polymorphism with a view to reintroducing polymorphism to the modified compiler. The Pict compiler uses a powerful partial type inference technique [11], and it will be interesting to see how it can be extended to handle session types. Because of the value of explicit session types as specifications, we might not want to allow the programmer to omit them completely. However, automatic inference of, for example, polarities will probably be very useful. The modified compiler will allow us to gain experience in programming with sessions and should suggest other useful typing features that could be added.

Acknowledgements

Simon Gay and Malcolm Hole are funded by the EPSRC project "Novel Type Systems for Concurrent Programming Languages" (GR/N39494).

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] B. 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.

[10] 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. To appear in *Proof, Language and Interaction: Essays in Honour of Robin Milner*, Gordon Plotkin, Colin Stirling, and Mads Tofte, editors, MIT Press, 1998.

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

[12] 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.

[13] 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 using a certain set of rules, perform some operation (such as applying the induction hypothesis), and reconstruct the process 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 1 [Substitution]

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

Proof By structural induction on the derivation of $\Gamma \vdash P$

T-Nil

The substitution is :

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

We have :

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

We need :

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

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

If $\Gamma, x : T$ is completed then $\Gamma + y : T$ (which must be either Γ or $\Gamma, y : T$) is also completed.

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

So, we have :

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

which is what we need.

T-Par

The substitution is :

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

We have :

$$\Gamma, x : T \vdash P \mid Q$$

We need :

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

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

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

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

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

1.1 : $x \notin \Gamma_2$

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

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 :

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

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 $\Gamma_3 \vdash Q$.

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

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 :

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

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

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

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

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

and, $\Gamma_2 + y : T \vdash Q\{y/x\}$

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

Given that $\Gamma_1, x : T + \Gamma_2, x : T = \Gamma, x : T$, we know $\Gamma_1 + \Gamma_2 = \Gamma$.

Also, given that x is free in in both P and Q, T cannot be a session type.

So, $y : T + y : T = y : T$.

So, we have :

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

which is what we need.

T-NEW

The substitution is :

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

We have :

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

We need :

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

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

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

As z is bound in P by the new declaration, we assume $z \neq y$, as it can be renamed.

So, $(\Gamma, z : Z) + y : T = (\Gamma + y : T), z : Z$.

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

So, we have :

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

which is what we need.

T-NEWSEQ

The substitution is :

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

We have :

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

We need :

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

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

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

As z^{+} and z^{-} are bound in P by the new declaration,

we know $z^{+} \neq y$ and $z^{-} \neq y$.

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

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

So, we have :

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

which is what we need.

T-REP

The substitution is :

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

We have :

$$\Gamma, x : T \vdash !P$$

We need :

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

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

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

If $\Gamma, x : T$ is unlimited, then $\Gamma + y : T$ (which must be either Γ or $\Gamma + y : T$) is also unlimited.

$$\frac{\Gamma + y : T \vdash P\{y/x\} \qquad \Gamma + y : T \text{ unlimited}}{\Gamma + y : T \vdash !P\{y/x\}} \text{ T-REP}$$

So, we have :

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

which is what we need.

T-COND

The substitution is :

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

We have :

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

We need :

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

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

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

If $\Gamma, x : T \vdash z : \mathsf{bool}$ then $\Gamma + y : T \vdash z\{y/x\} : \mathsf{bool}$.

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

So, we have :

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

which is what we need.

T-OUT

The substitution is :

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

We have :

$$\Gamma, x : T \vdash u\,!\,[\tilde{v}]\,.\,P$$

We need :

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

Case 1 : $x \notin \tilde{v}$

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

$$\frac{(\Gamma', x : T) + \tilde{v} : \tilde{V} \vdash u\,!\,[\tilde{v}]\,.\,P}{\Gamma', x : T \vdash P \qquad \Gamma', x : T \vdash u : \hat{}[\tilde{V}]}\ \textrm{T-OUT}$$

By the induction hypothesis, $\Gamma' + y : T \vdash P\{y/x\}$.

If $\Gamma', x : T \vdash u : \hat{}[\tilde{V}]$ then $\Gamma' + y : T \vdash u\{y/x\} : \hat{}[\tilde{V}]$.

$$\frac{\Gamma' + y : T \vdash P\{y/x\} \qquad \Gamma' + y : T \vdash u\{y/x\} : \hat{}[\tilde{V}]}{\Gamma' + y : T + \tilde{v} : \tilde{V} \vdash (u\,!\,[\tilde{v}]\,.\,P)\{y/x\}}\ \textrm{T-OUT}$$

Given that $\Gamma, x : T = \Gamma', x : T + \tilde{v} : \tilde{V}$ and $x \notin \tilde{v}$, we know $\Gamma = \Gamma' + \tilde{v} : \tilde{V}$.

So, $\Gamma' + y : T + \tilde{v} : \tilde{V} = \Gamma + y : T$.

So, we have :

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

which is what we need.

Case 2 : $x \in \tilde{v}$, *say* $x = v_1$

Let $\Gamma, x : T = \Gamma' + (x : T, v_2 : V_2, \ldots, v_n : V_n)$.

<u>*2.1*</u> *:* $x \in \Gamma'$

Let $\Gamma' = \Gamma'', x : T$.

$$\frac{(\Gamma'', x : T) + (x : T, v_2 : V_2, \ldots, v_n : V_n) \vdash u\,!\,[\tilde{v}]\,.\,P}{\Gamma'', x : T \vdash P \qquad \Gamma'', x : T \vdash u : \hat{}[\tilde{V}]}\ \text{T-OUT}$$

By the induction hypothesis, $\Gamma'' + y : T \vdash P\{y/x\}$.

If $\Gamma'', x : T \vdash u : \hat{}[\tilde{V}]$ then $\Gamma'' + y : T \vdash u\{y/x\} : \hat{}[\tilde{V}]$.

$$\frac{\Gamma'' + y : T \vdash P\{y/x\} \qquad \Gamma'' + y : T \vdash u\{y/x\} : \hat{}[\tilde{V}]}{\Gamma'' + y : T + (y : T, v_2 : V_2, \ldots, v_n : V_n) \vdash (u\,!\,[\tilde{v}]\,.\,P)\{y/x\}}\ \text{T-OUT}$$

Given that $(\Gamma'', x : T) + (x : T, v_2 : V_2, \ldots, v_n : V_n) = \Gamma, x : T$,

we know $\Gamma'' + (v_2 : V_2, \ldots, v_n : V_n) = \Gamma$.

So, $\Gamma'' + y : T + (y : T, v_2 : V_2, \ldots, v_n : V_n) = \Gamma + y : T$.

So, we have :

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

which is what we need.

<u>*2.2*</u> *:* $x \notin \Gamma'$

$$\frac{\Gamma' + (x : T, v_2 : V_2, \ldots, v_n : V_n) \vdash u\,!\,[\tilde{v}]\,.\,P}{\Gamma' \vdash P \qquad \Gamma' \vdash u : \hat{}[\tilde{V}]}\ \text{T-OUT}$$

As $x \notin \Gamma'$, we know $\Gamma' \vdash P\{y/x\}$ and $\Gamma' \vdash u\{y/x\} : \hat{}[\tilde{V}]$.

$$\frac{\Gamma' \vdash P\{y/x\} \qquad \Gamma' \vdash u\{y/x\} : \hat{}[\tilde{V}]}{\Gamma' + (y : T, v_2 : V_2, \ldots, v_n : V_n) \vdash (u\,!\,[\tilde{v}]\,.\,P)\{y/x\}}\ \text{T-OUT}$$

Given that $\Gamma, x : T = \Gamma' + (x : T, v_2 : V_2, \ldots, v_n : V_n)$ and $x \notin \Gamma'$,

we know $\Gamma' + (v_2 : V_2, \ldots, v_n : V_n) = \Gamma$.

So, $\Gamma' + (y : T, v_2 : V_2, \ldots, v_n : V_n) = \Gamma + y : T$.

So, we have :

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

which is what we need.

T-In

The substitution is :

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

We have :

$$\Gamma, x : T \vdash u\ ?\ [\tilde{v} : \tilde{V}]\ .\ P$$

We need :

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

$$\dfrac{\Gamma, x : T \vdash u\ ?\ [\tilde{v} : \tilde{V}]\ .\ P}{\Gamma, x : T, \tilde{v} : \tilde{V} \vdash P \qquad \Gamma, x : T \vdash u : \hat{}[\tilde{V}]}\ \textbf{T-In}$$

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

As $\tilde{v}$ are bound variables, we know $y \notin \tilde{v}$.

So, $(\Gamma, \tilde{v} : \tilde{V}) + y : T = (\Gamma + y : T), \tilde{v} : \tilde{V}$.

If $\Gamma, x : T \vdash u : \hat{}[\tilde{V}]$ then $\Gamma + y : T \vdash u\{y/x\} : \hat{}[\tilde{V}]$.

$$\dfrac{(\Gamma + y : T), \tilde{v} : \tilde{V} \vdash P\{y/x\} \qquad \Gamma + y : T \vdash u\{y/x\} : \hat{}[\tilde{V}]}{\Gamma + y : T \vdash (u\ ?\ [\tilde{v}]\ .\ P)\{y/x\}}\ \textbf{T-In}$$

So, we have :

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

which is what we need.

T-OutSeq

Note that for the T-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 : $x \notin \tilde{v}$, $x \neq u$

The substitution is :

$$(u^p\ !\ [\tilde{v}]\ .\ P)\{y/x\}$$

We have :

$$((\Gamma, u^p : ![\tilde{V}]\ .\ S) + \tilde{v} : \tilde{V}), x : T \vdash u^p\ !\ [\tilde{v}]\ .\ P$$

We need :

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

As $x \notin \tilde{v}$, we know $((\Gamma, u^p : ![\tilde{V}] . S) + \tilde{v} : \tilde{V}), x : T = ((\Gamma, u^p : ![\tilde{V}] . S), x : T) + \tilde{v} : \tilde{V}$.

$$\frac{((\Gamma, u^p : ![\tilde{V}] . S), x : T) + \tilde{v} : \tilde{V} \vdash u^p ! [\tilde{v}] . P}{\Gamma, u^p : S, x : T \vdash P} \text{ T-OutSeq}$$

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

We know $y \neq u^p$, as $+$ is not defined on session ports.

So, $\Gamma, u^p : S + y : T = (\Gamma + y : T), u^p : S$.

$$\frac{(\Gamma + y : T), u^p : S \vdash P\{y/x\}}{((\Gamma + y : T), u^p : ![\tilde{V}] . S) + \tilde{v} : \tilde{V} \vdash (u^p ! [\tilde{v}] . P)\{y/x\}} \text{ T-OutSeq}$$

As $y \neq u^p$, $((\Gamma + y : T), u^p : ![\tilde{V}] . S) + \tilde{v} : \tilde{V} = ((\Gamma, u^p : ![\tilde{V}] . S) + \tilde{v} : \tilde{V}) + y : T$.

So, we have :

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

which is what we need.

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

The substitution is :

$$(x^p ! [\tilde{v}] . P)\{y^q/x^p\}$$

We have :

$$(\Gamma + \tilde{v} : \tilde{V}), x^p : ![\tilde{V}] . S \vdash x^p ! [\tilde{v}] . P$$

We need :

$$\Gamma + \tilde{v} : \tilde{V} + y^q : ![\tilde{V}] . S \vdash y^q ! [\tilde{v}] . P\{y^q/x^p\}$$

As $x \notin \tilde{v}$, we know $(\Gamma + \tilde{v} : \tilde{V}), x^p : ![\tilde{V}] . S = (\Gamma, x^p : ![\tilde{V}] . S) + \tilde{v} : \tilde{V}$.

$$\frac{(\Gamma, x^p : ![\tilde{V}] . S) + \tilde{v} : \tilde{V} \vdash x^p ! [\tilde{v}] . P}{\Gamma, x^p : S \vdash P} \text{ T-OutSeq}$$

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

As the addition is defined, we know $y^q \notin \Gamma$.

So, $\Gamma + y^q : S = \Gamma, y^q : S$.

$$\frac{\Gamma, y^q : S \vdash P\{y^q/x^p\}}{(\Gamma, y^q : ![\tilde{V}] . S) + \tilde{v} : \tilde{V} \vdash y^q ! [\tilde{v}] . P\{y^q/x^p\}} \text{ T-OutSeq}$$

We know $y^q \notin \Gamma$. So, $\Gamma, y^q : ![\tilde{V}] . S = \Gamma + y^q : ![\tilde{V}] . S$.

So, we have :

$$\Gamma + \tilde{v} : \tilde{V} + y^q : ![\tilde{V}] . S \vdash y^q ! [\tilde{v}] . P\{y^q/x^p\}$$

which is what we need.

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

The substitution is :

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

We have :

$$((\Gamma, u^p : ![\tilde{V}] \,.\, S) + (v_2 : V_2, \ldots, v_n : V_n)), x : T + V_1 \vdash u^p \,!\, [\tilde{v}] \,.\, P$$

We need :

$$(\Gamma, u^p : ![\tilde{V}] \,.\, S) + (v_2 : V_2, \ldots, v_n : V_n) + y : T + V_1 \vdash (u^p \,!\, [\tilde{v}] \,.\, P)\{y/x\}$$

As $x = v_1$, we know $((\Gamma, u^p : ![\tilde{V}] \,.\, S) + (v_2 : V_2, \ldots, v_n : V_n)), x : T + V_1 =$
$((\Gamma, u^p : ![\tilde{V}] \,.\, S), x : T) + (v_1 : V_1, \ldots, v_n : V_n)$.

$$\frac{((\Gamma, u^p : ![\tilde{V}] \,.\, S), x : T) + (v_1 : V_1, \ldots, v_n : V_n) \vdash u^p \,!\, [\tilde{v}] \,.\, P}{\Gamma, u^p : S, x : T \vdash P}\ \textsc{T-OutSeq}$$

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

As the addition is defined, we know $u \neq y$.

So, $(\Gamma, u^p : S) + y : T = (\Gamma + y : T), u^p : S$.

$$\frac{(\Gamma + y : T), u^p : S \vdash P\{y/x\}}{((\Gamma + y : T), u^p : ![\tilde{V}] \,.\, S) + \tilde{v} : \tilde{V}\{y/x\} \vdash (u^p \,!\, [\tilde{v}] \,.\, P)\{y/x\}}\ \textsc{T-OutSeq}$$

We know $u \neq y$. So, $((\Gamma + y : T), u^p : ![\tilde{V}] \,.\, S) + \tilde{v} : \tilde{V}\{y/x\} =$
$(\Gamma, u^p : ![\tilde{V}] \,.\, S) + (v_2 : V_2, \ldots, v_n : V_n) + y : T + V_1$.

So, we have :

$$((\Gamma, u^p : ![\tilde{V}] \,.\, S) + (v_2 : V_2, \ldots, v_n : V_n)), x : T + V_1 \vdash u^p \,!\, [\tilde{v}] \,.\, P$$

which is what we need.

T-InSeq

Case 1: $x \neq u$

The substitution is :

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

We have :

$$\Gamma, u^p : ?[\tilde{V}] \,.\, S, x : T \vdash u^p \mathbin{?} [\tilde{v} : \tilde{V}] \,.\, P$$

We need :

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

$$\frac{\Gamma, u^p : ?[\tilde{V}] \,.\, S, x : T \vdash u^p \mathbin{?} [\tilde{v} : \tilde{V}] \,.\, P}{\Gamma, u^p : S, \tilde{v} : \tilde{V}, x : T \vdash P}\ \text{T-InSeq}$$

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

As $\tilde{v}$ are bound, we know $y \notin \tilde{v}$.

We know also, as the addition is defined, $y \neq u$.

So, $(\Gamma, u^p : S, \tilde{v} : \tilde{V}) + y : T = (\Gamma + y : T), u^p : S, \tilde{v} : \tilde{V}$.

$$\frac{(\Gamma + y : T), u^p : S, \tilde{v} : \tilde{V} \vdash P\{y/x\}}{(\Gamma + y : T), u^p : ?[\tilde{V}] \,.\, S \vdash (u^p \mathbin{?} [\tilde{v} : \tilde{V}] \,.\, P)\{y/x\}}\ \text{T-InSeq}$$

We know $y \neq u$. So, $(\Gamma + y : T), u^p : ?[\tilde{V}] \,.\, S = (\Gamma, u^p : ?[\tilde{V}] \,.\, S) + y : T$.

So, we have :

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

which is what we need.

Case 2 : $x = u$

The substitution is :

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

We have :

$$\Gamma, x^p : ?[\tilde{V}] \,.\, S \vdash x^p \mathbin{?} [\tilde{v} : \tilde{V}] \,.\, P$$

We need :

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

$$\frac{\Gamma, x^p : ?[\tilde{V}] \,.\, S \vdash x^p \mathbin{?} [\tilde{v} : \tilde{V}] \,.\, P}{\Gamma, \tilde{v} : \tilde{V}, x^p : S \vdash P}\ \text{T-InSeq}$$

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

As $\tilde{v}$ are bound names, we know $y \notin \tilde{v}$.

We know also, as the addition is defined, $y \notin \Gamma$.

So, $(\Gamma, \tilde{v} : \tilde{V}) + y^q \,.\, S = \Gamma, y^q : S, \tilde{v} : \tilde{V}$.

$$\frac{\Gamma, y^q : S, \tilde{v} : \tilde{V} \vdash P\{y^q/x^p\}}{\Gamma, y^q : ?[\tilde{V}] \,.\, S \vdash y^q \mathbin{?} [\tilde{v} : \tilde{V}] \,.\, P\{y^q/x^p\}}\ \text{T-InSeq}$$

We know $y \notin \Gamma$. So, $\Gamma, y^q : ?[\tilde{V}] \,.\, S = \Gamma + y^q : ?[\tilde{V}] \,.\, S$.

So, we have :

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

which is what we need.

T-Offer

Case 1 : $x \neq u$

The substitution is :

$$(u^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\})\{y/x\}$$

We have :

$$\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x : T \vdash u^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}$$

We need :

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

$$\frac{\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x : T \vdash u^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}{\Gamma, u^p : S_i, x : T \vdash P_1, \ldots, \Gamma, u^p : S_n, x : T \vdash P_n} \text{T-OFFER}$$

By the induction hypothesis, $\forall i_{1,\ldots,n}\ (\Gamma, u^p : S_i) + y : T \vdash P_i\{y/x\}$.

We know, as the addition is defined, $y \neq u^p$.

So, $\forall i_{1,\ldots,n}\ (\Gamma, u^p : S_i) + y : T = (\Gamma + y : T), u^p : S_i$.

$$\frac{(\Gamma + y : T), u^p : S_1 \vdash P_1, \ldots, (\Gamma + y : T), u^p : S_n \vdash P_n}{(\Gamma + y : T), u^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash u^p \triangleright \{l_1 : P_1\{y/x\}, \ldots, l_n : P_n\{y/x\}\}} \text{T-OFFER}$$

We know $y \neq u^p$. So, $(\Gamma + y : T), u^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle =$
$(\Gamma, u^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle) + y : T$.

So, we have :

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

which is what we need.

Case 2 : $x = u$

The substitution is :

$$(x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\})\{y^q/x^p\}$$

We have :

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

We need :

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

$$\frac{\Gamma, x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}{\Gamma, x^p : S_i \vdash P_i, \ldots, \Gamma, x^p : S_n \vdash P_n} \text{T-OFFER}$$

By the induction hypothesis, $\forall i_{1,\ldots,n}\ \Gamma + y^q : S_i \vdash P_i\{y^q/x^p\}$.

We know, as the addition is defined, $y^q \notin \Gamma$.

So, $\forall i_{1,\ldots,n}\ \Gamma + y^q : S_i = \Gamma, y^q : S_i$.

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

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

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.

T-Choose

Case 1 : $x \neq u$

The substitution is :

$$u^p \triangleleft l_i \,.\, P\{y/x\}$$

We have :

$$\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x : T \vdash u^p \triangleleft l_i \,.\, P$$

We need :

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

$$\frac{\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x : T \vdash u^p \triangleleft l_i \,.\, P}{\Gamma, u^p : S_i, x : T \vdash P}\ \textsc{T-Choose}$$

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

We know, as the addition is defined, $u^p \neq y$.

So, $(\Gamma, u^p : S) + y : T = (\Gamma + y : T), u^p : S$.

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

We know, $u^p \neq y$. So, $(\Gamma + y : T), u^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle =$ $(\Gamma, u^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle) + y : T$.

So, we have :

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

which is what we need.

Case 2 : $x = u$

The substitution is :

$$(x^p \triangleleft l_i \,.\, P)\{y^q/x^p\}$$

We have :

$$\Gamma, x^p : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^p \triangleleft l_i \,.\, P$$

We need :

$$\Gamma + y^q : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash y^q \triangleleft l_i \,.\, P\{y^q/x^p\}$$

$$\frac{\Gamma, x^p : \oplus\langle l_1 \!:\! S_1, \ldots, l_n \!:\! S_n\rangle \vdash x^p \triangleleft l_i \;.\; P}{\Gamma, x^p : S_i \vdash P} \text{ T-Choose}$$

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

We know, as the addition is defined, $y^q \notin \Gamma$.

So, $\Gamma + y^q : S_i = \Gamma, y^q : S_i$.

$$\frac{\Gamma, y^q : S_i \vdash P\{y^q/x^i\}}{\Gamma, y^q : \oplus\langle l_1 \!:\! S_1, \ldots, l_n \!:\! S_n\rangle \vdash y^q \triangleleft l_i \;.\; P\{y^q/x^p\}} \text{ T-Choose}$$

We know $y^q \notin \Gamma$. So, $\Gamma, y^q : \oplus\langle l_1 \!:\! S_1, \ldots, l_n \!:\! S_n\rangle = \Gamma + y^q : \oplus\langle l_1 \!:\! S_1, \ldots, l_n \!:\! S_n\rangle$.

So, we have :

$$\Gamma + y^q : \oplus\langle l_1 \!:\! S_1, \ldots, l_n \!:\! S_n\rangle \vdash y^q \triangleleft l_i \;.\; P\{y^q/x^p\}$$

which is what we need.

□

A.2 Proof of Lemma 2 [Structural Equivalence]

If $\Gamma \vdash P$ and $P \equiv Q$ then $\Gamma \vdash Q$

Proof There is one case for each rule in the definition of structural congruence. Each case divides into two further cases: one for the application of the rule in each direction.

S-Unit

The rule is :

$$P \mid \mathbf{0} \equiv P$$

Case 1 :

We have :

$$\Gamma \vdash P \mid \mathbf{0}$$

We need :

$$\Gamma \vdash P$$

$$\frac{\Gamma \vdash P \mid \mathbf{0}}{\Gamma \vdash P \qquad \vdash \mathbf{0}} \text{ T-Par}$$

So, we have :

$$\Gamma \vdash P$$

which is what we need.

Case 2 :

We have :

$$\Gamma \vdash P$$

We need :

$$\Gamma \vdash P \mid \mathbf{0}$$

$$\dfrac{\Gamma \vdash P \qquad \dfrac{\text{completed}}{\vdash \mathbf{0}}\ \text{T-NIL}}{\Gamma \vdash P \mid \mathbf{0}}\ \text{T-PAR}$$

So, we have :

$$\Gamma \vdash P \mid \mathbf{0}$$

which is what we need.

S-COMM

The rule is :

$$P \mid Q \equiv Q \mid P$$

Case 1 :

We have :

$$\Gamma \vdash P \mid Q$$

We need :

$$\Gamma \vdash Q \mid P$$

$$\dfrac{\Gamma \vdash P \mid Q}{\Gamma_1 \vdash P \qquad \Gamma_2 \vdash Q}\ \text{T-PAR}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\dfrac{\Gamma_2 \vdash Q \qquad \Gamma_1 \vdash P}{\Gamma_2 + \Gamma_1 \vdash Q \mid P}$$

We know $\Gamma_2 + \Gamma_1 = \Gamma$.

So, we have :

$$\Gamma \vdash Q \mid P$$

which is what we need.

Case 2 :

This case is the opposite of case one and is proved in the same way.

S-ASSOC

The rule is :

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

Case 1 :

We have :

$$\Gamma \vdash P \mid (Q \mid R)$$

We need :

$$\Gamma \vdash (P \mid Q) \mid R$$

$$\frac{\Gamma \vdash P \mid (Q \mid R)}{\Gamma_1 \vdash P \qquad \Gamma_2 \vdash Q \mid R}\text{ T-PAR}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_2 \vdash Q \mid R}{\Gamma_3 \vdash Q \qquad \Gamma_4 \vdash R}\text{ T-PAR}$$

where $\Gamma_3 + \Gamma_4 = \Gamma_2$

$$\frac{\Gamma_1 \vdash P \qquad \Gamma_3 \vdash Q}{\Gamma_1 + \Gamma_3 \vdash P \mid Q}\text{ T-PAR}$$

$$\frac{\Gamma_1 + \Gamma_3 \vdash P \mid Q \qquad \Gamma_4 \vdash R}{\Gamma_1 + \Gamma_3 + \Gamma_4 \vdash (P \mid Q) \mid R}\text{ T-PAR}$$

We know $\Gamma_3 + \Gamma_4 = \Gamma_2$ and $\Gamma_1 + \Gamma_2 = \Gamma$.

So, $\Gamma_1 + \Gamma_3 + \Gamma_4 = \Gamma$.

So, we have :

$$\Gamma \vdash (P \mid Q) \mid R$$

which is what we need.

Case 2 :

This case is the opposite of case one and is proved in the same way.

S-Extr

The rule is :

$$(\nu x^p : T)P \mid Q \equiv (\nu x^p : T)(P \mid Q) \text{ if } x^p \text{ is not free in } Q$$

Case 1 :

1.1 : x is a session channel

We have :

$$\Gamma \vdash (\nu x^{\pm} : S)P \mid Q$$

We need :

$$\Gamma \vdash (\nu x^{\pm} : S)(P \mid Q)$$

$$\frac{\Gamma \vdash (\nu x^{\pm} : S)P \mid Q}{\Gamma_1 \vdash (\nu x^{\pm} : S)P \qquad \Gamma_2 \vdash Q} \text{ T-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_1 \vdash (\nu x^{\pm} : S)P}{\Gamma_1, x^+ : S, x^- : \overline{S} \vdash P} \text{ T-NewSeq}$$

$$\frac{\Gamma_1, x^+ : S, x^- : \overline{S} \vdash P \qquad \Gamma_2 \vdash Q}{(\Gamma_1, x^+ : S, x^- : \overline{S}) + \Gamma_2 \vdash P \mid Q} \text{ T-Par}$$

As x was bound in P, we know $x \notin \Gamma_2$.

So, $(\Gamma_1, x^+ : S, x^- : \overline{S}) + \Gamma_2 = (\Gamma_1 + \Gamma_2), x^+ : S, x^- : \overline{S}$.

We know $\Gamma_1 + \Gamma_2 = \Gamma$. So, $(\Gamma_1 + \Gamma_2), x^+ : S, x^- : \overline{S} = \Gamma, x^+ : S, x^- : \overline{S}$.

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P \mid Q}{\Gamma \vdash (\nu x^{\pm} : S)(P \mid Q)} \text{ T-NewSeq}$$

So, we have :

$$\Gamma \vdash (\nu x^{\pm} : S)(P \mid Q)$$

which is what we need.

<u>1.2</u> : x is not a session channel
We have :

$$\Gamma \vdash (\nu x : T)P \mid Q$$

We need :

$$\Gamma \vdash (\nu x : T)(P \mid Q)$$

$$\frac{\Gamma \vdash (\nu x : T)P \mid Q}{\Gamma_1 \vdash (\nu x : T)P \qquad \Gamma_2 \vdash Q} \text{T-PAR}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_1 \vdash (\nu x : T)P}{\Gamma_1, x : T \vdash P} \text{T-NEW}$$

$$\frac{\Gamma_1, x : T \vdash P \qquad \Gamma_2 \vdash Q}{(\Gamma_1, x : T) + \Gamma_2 \vdash P \mid Q} \text{T-PAR}$$

As x was bound in P, we know $x \notin \Gamma_2$.

So, $(\Gamma_1, x : T) + \Gamma_2 = (\Gamma_1 + \Gamma_2), x : T$.

We know $\Gamma_1 + \Gamma_2 = \Gamma$. So, $(\Gamma_1 + \Gamma_2), x : T = \Gamma, x : T$.

$$\frac{\Gamma, x : T \vdash P \mid Q}{\Gamma \vdash (\nu x : T)(P \mid Q)} \text{T-NEW}$$

So, we have :

$$\Gamma \vdash (\nu x : T)(P \mid Q)$$

which is what we need.

Case 2 :

2.1 : x is a session channel

We have :

$$\Gamma \vdash (\nu x^{\pm} : S)(P \mid Q)$$

We need :

$$\Gamma \vdash (\nu x^{\pm} : S)P \mid Q$$

$$\dfrac{\Gamma \vdash (\nu x^{\pm} : S)(P \mid Q)}{\Gamma, x^{+} : S, x^{-} : \overline{S} \vdash P \mid Q}\ \textsc{T-NewSeq}$$

$$\dfrac{\Gamma, x^{+} : S, x^{-} : \overline{S} \vdash P \mid Q}{\Gamma_1 \vdash P \qquad \Gamma_2 \vdash Q}\ \textsc{T-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, x^{+} : S, x^{-} : \overline{S}$ and $x \in \Gamma_1$ but $x \notin \Gamma_2$

Let $\Gamma_1 = \Gamma_1', x^{+} : S, x^{-} : \overline{S}$.

$$\dfrac{\Gamma_1', x^{+} : S, x^{-} : \overline{S} \vdash P}{\Gamma_1' \vdash (\nu x^{\pm} : S)P}\ \textsc{T-NewSeq}$$

$$\dfrac{\Gamma_1' \vdash (\nu x^{\pm} : S)P \qquad \Gamma_2 \vdash Q}{\Gamma_1' + \Gamma_2 \vdash (\nu x^{\pm} : S)P \mid Q}\ \textsc{T-Par}$$

Given $\Gamma_1 + \Gamma_2 = \Gamma, x^{+} : S, x^{-} : \overline{S}$ and $\Gamma_1 = \Gamma_1', x^{+} : S, x^{-} : \overline{S}$ and $x \notin \Gamma_2$, we know $\Gamma_1' + \Gamma_2 = \Gamma$.

So, we have :

$$\Gamma \vdash (\nu x^{\pm} : S)P \mid Q$$

which is what we need.

2.2 : x is not a session channel

We have :

$$\Gamma \vdash (\nu x : T)(P \mid Q)$$

We need :

$$\Gamma \vdash (\nu x : T)P \mid Q$$

$$\frac{\Gamma \vdash (\nu x : T)(P \mid Q)}{\Gamma, x : T \vdash P \mid Q}\ \text{T-New}$$

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

where $\Gamma_1 + \Gamma_2 = \Gamma, x : T$ and $x \in \Gamma_1$ but $x \notin \Gamma_2$

Let $\Gamma_1 = \Gamma'_1, x : T$.

$$\frac{\Gamma'_1, x : T \vdash P}{\Gamma'_1 \vdash (\nu x : T)P}\ \text{T-New}$$

$$\frac{\Gamma'_1 \vdash (\nu x : T)P \qquad \Gamma_2 \vdash Q}{\Gamma'_1 + \Gamma_2 \vdash (\nu x : T)P \mid Q}\ \text{T-Par}$$

Given that $\Gamma_1 + \Gamma_2 = \Gamma, x : T$ and $\Gamma_1 = \Gamma'_1, x : T$ and $x \notin \Gamma_2$, we know $\Gamma'_1 + \Gamma_2 = \Gamma$.

So, we have :

$$\Gamma \vdash (\nu x : T)P \mid Q$$

which is what we need.

S-Nil

The rule is :

$$(\nu x : T)\mathbf{0} \equiv \mathbf{0} \text{ if } x \text{ is not a session channel}$$

Case 1 :

We have :

$$\Gamma \vdash (\nu x : T)\mathbf{0}$$

We need :

$$\Gamma \vdash \mathbf{0}$$

$$\frac{\Gamma \vdash (\nu x : T)\mathbf{0}}{\Gamma, x : T \vdash \mathbf{0}}\ \text{T-New}$$

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

If $\Gamma, x : T$ is completed then we know Γ is completed.

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash \mathbf{0}}\ \text{T-Nil}$$

So, we have :

$$\Gamma \vdash \mathbf{0}$$

which is what we need.

Case 2 :

We have :

$$\Gamma \vdash \mathbf{0}$$

We need :

$$\Gamma \vdash (\nu x : T)\mathbf{0}$$

$$\frac{\Gamma \vdash \mathbf{0}}{\Gamma \text{ completed}}\text{T-NIL}$$

We know x is not a session channel. So, if Γ is completed then $\Gamma, x : T$ is completed.

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

$$\frac{\Gamma, x : T \vdash \mathbf{0}}{\Gamma \vdash (\nu x : T)\mathbf{0}}\text{T-NEW}$$

So, we have :

$$\Gamma \vdash (\nu x : T)\mathbf{0}$$

which is what we need.

S-SWITCH

The rule is :

$$(\nu x^p : T)(\nu y^q : U)P \equiv (\nu y^q : U)(\nu x^p : T)P$$

Case 1 :

1.1 : x is a session channel

We have :

$$\Gamma \vdash (\nu x^{\pm} : T)(\nu y^{\pm} : U)P$$

We need :

$$\Gamma \vdash (\nu y^{\pm} : U)(\nu x^{\pm} : T)P$$

$$\dfrac{\Gamma \vdash (\nu x^{\pm} : T)(\nu y^{\pm} : U)P}{\Gamma, x^{+} : T, x^{-} : \overline{T} \vdash (\nu y^{\pm} : U)P} \text{ T-NewSeq}$$
$$\dfrac{\Gamma, x^{+} : T, x^{-} : \overline{T} \vdash (\nu y^{\pm} : U)P}{\Gamma, x^{+} : T, x^{-} : \overline{T}, y^{+} : U, y^{-} : \overline{U} \vdash P} \text{ T-NewSeq}$$
$$\dfrac{\Gamma, x^{+} : T, x^{-} : \overline{T}, y^{+} : U, y^{-} : \overline{U} \vdash P}{\Gamma, y^{+} : U, y^{-} : \overline{U} \vdash (\nu x^{\pm} : T)P} \text{ T-NewSeq}$$
$$\dfrac{\Gamma, y^{+} : U, y^{-} : \overline{U} \vdash (\nu x^{\pm} : T)P}{\Gamma \vdash (\nu y^{\pm} : U)(\nu x^{\pm} : T)P} \text{ T-NewSeq}$$

So, we have :

$$\Gamma \vdash (\nu y^{\pm} : U)(\nu x^{\pm} : T)P$$

which is what we need.

1.2 : x is not a session channel

We have :

$$\Gamma \vdash (\nu x : T)(\nu y : U)P$$

We need :

$$\Gamma \vdash (\nu y : U)(\nu x : T)P$$

$$\dfrac{\Gamma \vdash (\nu x : T)(\nu y : U)P}{\Gamma, x : T \vdash (\nu y : U)P} \text{ T-New}$$
$$\dfrac{\Gamma, x : T \vdash (\nu y : U)P}{\Gamma, x : T, y : U \vdash P} \text{ T-New}$$
$$\dfrac{\Gamma, x : T, y : U \vdash P}{\Gamma, y : U \vdash (\nu x : T)P} \text{ T-New}$$
$$\dfrac{\Gamma, y : U \vdash (\nu x : T)P}{\Gamma \vdash (\nu y : U)(\nu x : T)P} \text{ T-New}$$

So, we have :

$$\Gamma \vdash (\nu y : U)(\nu x : T)P$$

which is what we need.

Case 2 :

This case is the opposite of case one and is proved in the same way.

S-Rep

The rule is :

$$!P \equiv P|!P$$

Case 1 :

We have :

$$\Gamma \vdash !P$$

We need :

$$\Gamma \vdash P|!P$$

$$\dfrac{\dfrac{\Gamma \vdash !P}{\Gamma \text{ unlimited} \qquad \Gamma \vdash P}\,\text{T-Rep}}{}$$

$$\dfrac{\Gamma \vdash P \qquad \Gamma \vdash !P}{\Gamma + \Gamma \vdash P|!P}\,\text{T-Par}$$

We know there are no session channels in Γ.

So, $\Gamma + \Gamma = \Gamma$.

So, we have :

$$\Gamma \vdash P|!P$$

which is what we need.

Case 2 :

We have :

$$\Gamma \vdash P|!P$$

We need :

$$\Gamma \vdash !P$$

$$\dfrac{\Gamma \vdash P|!P}{\Gamma_1 \vdash P \qquad \Gamma_2 \vdash !P}\,\text{T-Par}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

We know there are no session channels in Γ.

So, $\Gamma_1 = \Gamma_2 = \Gamma$.

So, we have :

$$\Gamma \vdash !P$$

which is what we need.

S-OFFER

The rule is :

$$x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \equiv x^p \triangleright \{l_{\sigma(1)} : P_{\sigma(1)}, \ldots, l_{\sigma(n)} : P_{\sigma(n)}\}$$

where σ is a permutation on $\{1, \ldots, n\}$

e.g. $\sigma(1) = n$, $\sigma(2) = n - 1$ etc.

Case 1 :

We have :

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

We need :

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

$$\cfrac{\cfrac{\Gamma, x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^p \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}{\Gamma_1, x^p : S_1 \vdash P_1; \ldots; \Gamma_n, x^p : S_n \vdash P_n} \text{ T-OFFER}}{\cfrac{\Gamma_n, x^p : S_n \vdash P_n; \ldots; \Gamma_1, x^p : S_1 \vdash P_1}{\Gamma, x^p : \&\langle l_n : S_n, \ldots, l_1 : S_1\rangle \vdash x^p \triangleright \{l_n : P_n, \ldots, l_1 : P_1\}} \text{ T-OFFER}}$$

So, we have :

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

which is what we need.

Case 2 :

This case is the opposite of case one and is proved in the same way.

$\square$

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

If $\Gamma \vdash P$ and $P \xrightarrow{\alpha, l} P'$ where Γ is balanced with respect to α (if $\alpha \neq \tau$), then $\Gamma' \vdash P'$ where

$$\Gamma' = \Gamma \text{ if } \alpha = \tau \text{ or if } \alpha \text{ is not a session channel}$$

$$\Gamma' = \Gamma'', \alpha^+ : \mathsf{tail}(S, l), \alpha^- : \mathsf{tail}(\overline{S}, l)$$

$$\text{where } \Gamma'', \alpha^+ : S, \alpha^- : \overline{S} = \Gamma \text{ if } \alpha \text{ is a session channel.}$$

Proof By structural induction on the derivation of $P \xrightarrow{\alpha, l} P'$

R-COMM

The reduction is :

$$x^p \,?\, [\tilde{y} : \tilde{T}] \,.\, P \mid x^q \,!\, [\tilde{z}] \,.\, Q \xrightarrow{x,-} P\{\tilde{z}/\tilde{y}\} \mid Q$$

Case 1 : x is not a session channel

We have :

$$\Gamma, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash x \,?\, [\tilde{y} : \tilde{T}] \,.\, P \mid x \,!\, [\tilde{z}] \,.\, Q$$

We need :

$$\Gamma, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash P\{\tilde{z}/\tilde{y}\} \mid Q$$

$$\frac{\Gamma, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash x \,?\, [\tilde{y} : \tilde{T}] \,.\, P \mid x \,!\, [\tilde{z}] \,.\, Q}{\Gamma_1, x : \hat{}[\tilde{T}] \vdash x \,?\, [\tilde{y} : \tilde{T}] \,.\, P \qquad \Gamma_2, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash x \,!\, [\tilde{z}] \,.\, Q} \text{ T-PAR}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_1, x : \hat{}[\tilde{T}] \vdash x \,?\, [\tilde{y} : \tilde{T}] \,.\, P}{\Gamma_1, x : \hat{}[\tilde{T}], \tilde{y} : \tilde{T} \vdash P} \text{ T-IN}$$

We know (Lemma 1 [Substitution]) : $\quad \Gamma_1, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash P\{\tilde{z}/\tilde{y}\}$

$$\frac{\Gamma_2, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash x \,!\, [\tilde{z}] \,.\, Q}{\Gamma_2, x : \hat{}\tilde{T} \vdash Q} \text{ T-OUT}$$

$$\frac{\Gamma_1, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash P\{\tilde{z}/\tilde{y}\} \qquad \Gamma_2, x : \hat{}[\tilde{T}] \vdash Q}{(\Gamma_1, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T}) + (\Gamma_2, x : \hat{}\tilde{T}) \vdash P\{\tilde{z}/\tilde{y}\} \mid Q} \text{ T-PAR}$$

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

So, we have :

$$\Gamma, x : \hat{}[\tilde{T}] + \tilde{z} : \tilde{T} \vdash P\{\tilde{z}/\tilde{y}\} \mid Q$$

which is what we need.

Case 2 : x is a session channel

We have :

$$\Gamma, x^+ : ?[\tilde{T}].S, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{T} \vdash x?[\tilde{y} : \tilde{T}].P \mid x![\tilde{z}].Q$$

We need :

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

$$\frac{\Gamma, x^+ : ?[\tilde{T}].S, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{T} \vdash x?[\tilde{y} : \tilde{T}].P \mid x![\tilde{z}].Q}{\Gamma_1, x^+ : ?[\tilde{T}].S \vdash x?[\tilde{y} : \tilde{T}].P \qquad \Gamma_2, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{T} \vdash x![\tilde{z}].Q} \text{ T-PAR}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma_1, x^+ : ?[\tilde{T}].S \vdash x?[\tilde{y} : \tilde{T}].P}{\Gamma_1, x^+ : S, \tilde{y} : \tilde{T} \vdash P} \text{ T-INSEQ}$$

We know (Lemma 1 [Substitution]) : $\quad \Gamma_1, x^+ : S + \tilde{z} : \tilde{T} \vdash P\{\tilde{z}/\tilde{y}\}$

$$\frac{\Gamma_2, x^- : ![\tilde{T}].\overline{S} + \tilde{z} : \tilde{T} \vdash x![\tilde{z}].Q}{\Gamma_2, x^- : \overline{S} \vdash Q} \text{ T-OUTSEQ}$$

$$\frac{\Gamma_1, x^+ : S + \tilde{z} : \tilde{T} \vdash P\{\tilde{z}/\tilde{y}\} \qquad \Gamma_2, x^- : \overline{S} \vdash Q}{(\Gamma_1, x^+ : S + \tilde{z}) + (\Gamma_2, x^- : \overline{S}) \vdash P\{\tilde{z}/\tilde{y}\} \mid Q} \text{ T-PAR}$$

But, $(\Gamma_1, x^+ : S + \tilde{z}) + (\Gamma_2, x^- : \overline{S}) = \Gamma, x^+ : S, x^- : \overline{S} + \tilde{z} : \tilde{T}$.

So, we have :

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

which is what we need.

R-SELECT

The reduction is :

$$x^+ \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^- \triangleleft l_i \,.\, Q \xrightarrow{x, l_i} P_i \mid Q$$

We have :

$$\Gamma, x^+ : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle \vdash \\ x^+ \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^- \triangleleft l_i \,.\, Q$$

We need :

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

$$\frac{\Gamma, x^+ : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle \vdash x^+ \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \mid x^- \triangleleft l_i\,.\,Q}{\Gamma_1, x^+ : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^+ \triangleright \{l_1 : P_1, \ldots, l_n : P_n\} \quad \Gamma_2, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle \vdash x^- \triangleleft l_i\,.\,Q}\ \text{T-PAR}$$

where $\Gamma_1 + \Gamma_2 = \Gamma$

$$\frac{\Gamma, x^+ : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle \vdash x^+ \triangleright \{l_1 : P_1, \ldots, l_n : P_n\}}{\Gamma_1, x^+ : S_1 \vdash P_1, \ldots, \Gamma_1, x^+ : S_n \vdash P_n}\ \text{T-OFFER}$$

$$\frac{\Gamma_2, x^- : \oplus\langle l_1 : \overline{S_1}, \ldots, l_n : \overline{S_n}\rangle \vdash x^- \triangleleft l_i\,.\,Q}{\Gamma_2, x^- : \overline{S_i} \vdash Q}\ \text{T-CHOOSE}$$

$$\frac{\Gamma_1, x^+ : S_i \vdash P_i \qquad \Gamma_2, x^- : \overline{S_i} \vdash Q}{(\Gamma_1, x^+ : S_i) + (\Gamma_2, x^- : \overline{S_i}) \vdash P_i \mid Q}\ \text{T-PAR}$$

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

So, we have :

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

which is what we need.

R-TRUE

The reduction is :

$$\text{if } \mathit{true} \text{ then } P \text{ else } Q \xrightarrow{\tau,-} P$$

We have :

$$\Gamma \vdash \text{if } \mathit{true} \text{ then } P \text{ else } Q$$

We need :

$$\Gamma \vdash P$$

$$\frac{\Gamma \vdash \text{if } \mathit{true} \text{ then } P \text{ else } Q}{\Gamma \vdash \mathit{true} : \text{bool} \qquad \Gamma \vdash P \qquad \Gamma \vdash Q}\ \text{T-COND}$$

So, we have :

$$\Gamma \vdash P$$

which is what we need.

R-False

The reduction is :

$$\text{if } \mathit{false} \text{ then } P \text{ else } Q \xrightarrow{\tau,-} Q$$

We have :

$$\Gamma \vdash \text{if } \mathit{false} \text{ then } P \text{ else } Q$$

We need :

$$\Gamma \vdash Q$$

$$\dfrac{\Gamma \vdash \text{if } \mathit{false} \text{ then } P \text{ else } Q}{\Gamma \vdash \mathit{false} : \text{bool} \qquad \Gamma \vdash P \qquad \Gamma \vdash Q}\ \text{T-Cond}$$

So, we have :

$$\Gamma \vdash Q$$

which is what we need.

R-Par

The derivation ends with :

$$\dfrac{P \xrightarrow{\alpha,l} P'}{P \mid Q \xrightarrow{\alpha,l} P' \mid Q'}$$

Case 1 : $\alpha = \tau$

We have :

$$\Gamma \vdash P \mid Q$$

We need :

$$\Gamma \vdash P' \mid Q$$

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

where $\Gamma_1 + \Gamma_2 = \Gamma$

$\Gamma_1 \vdash P$ and $P \xrightarrow{\tau,-} P'$, so, by the induction hypothesis, $\Gamma_1 \vdash P'$.

$$\dfrac{\Gamma_1 \vdash P' \qquad \Gamma_2 \vdash Q}{\Gamma_1 + \Gamma_2 \vdash P' \mid Q}\ \text{T-Par}$$

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

So, we have :

$$\Gamma \vdash P' \mid Q$$

which is what we need.

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

We have :

$$\Gamma \vdash P \mid Q$$

We need :

$$\Gamma \vdash P' \mid Q$$

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

where $\Gamma_1 + \Gamma_2 = \Gamma$

$\Gamma_1 \vdash P$ and $P \xrightarrow{\alpha,l} P'$, so, by the induction hypothesis, $\Gamma_1 \vdash P'$.

$$\frac{\Gamma_1 \vdash P' \qquad \Gamma_2 \vdash Q}{\Gamma_1 + \Gamma_2 \vdash P' \mid Q}\ \textsc{T-Par}$$

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

So, we have :

$$\Gamma \vdash P' \mid Q$$

which is what we need.

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

We have :

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

We need :

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

$$\frac{\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vdash P \mid Q}{\Gamma_1, \alpha^+ : S, \alpha^- : \overline{S} \vdash P \qquad \Gamma_2 \vdash Q}\ \textsc{T-Par}$$

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

$\Gamma_1, \alpha^+ : S, \alpha^- : \overline{S} \vdash P$ and $P \xrightarrow{\alpha,l} P'$, so, by the induction hypothesis, as α is a session channel, $\Gamma_1, \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l) \vdash P'$.

$$\frac{\Gamma_1, \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l) \vdash P' \qquad \Gamma_2 \vdash Q}{(\Gamma_1, \alpha^+ : \mathrm{tail}(S, l), \alpha^- : \mathrm{tail}(\overline{S}, l)) + \Gamma_2 \vdash P' \mid Q}\ \textsc{T-Par}$$

But, $\Gamma_1 + \Gamma_2 = \Gamma$ and $\alpha \notin \Gamma_2$.

So, we have :

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

which is what we need.

R-CONG

The derivation ends with :

$$\frac{P' \equiv P \quad P \xrightarrow{\alpha,l} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha,l} Q'}$$

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

$$\Gamma \vdash P'$$

We need :

$$\Gamma \vdash Q'$$

We know (Lemma 2 [Structural Equivalence]) : $\Gamma \vdash P$

$\Gamma \vdash P$ and $P \xrightarrow{\alpha,l} Q$, so, by the induction hypothesis, $\Gamma \vdash Q$.

We know (Lemma 2 [Structural Equivalence]) : $\Gamma \vdash Q'$

So, we have :

$$\Gamma \vdash Q'$$

which is what we need.

Case 2 : α is a session channel
We have :

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

We need :

$$\Gamma, \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l) \vdash Q'$$

We know (Lemma 2 [Structural Equivalence]) : $\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vdash P$

$\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vdash P$ and $P \xrightarrow{\alpha,l} Q$, so, by the induction hypothesis,
$\Gamma, \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l) \vdash Q$.

We know (Lemma 2 [Structural Equivalence]) :

$$\Gamma, \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l) \vdash Q'$$

So, we have :

$$\Gamma \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l) \vdash Q'$$

which is what we need.

R-NEW

The derivation ends with :

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

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

We have :

$$\Gamma \vdash (\nu x : T)P$$

We need :

$$\Gamma \vdash (\nu x : T)P'$$

$$\frac{\Gamma \vdash (\nu x : T)P}{\Gamma, x : T \vdash P}\ \text{T-NEW}$$

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

$$\frac{\Gamma, x : T \vdash P'}{\Gamma \vdash (\nu x : T)P'}\ \text{T-NEW}$$

So, we have :

$$\Gamma \vdash (\nu x : T)P'$$

which is what we need.

Case 2 : α is a session channel

We have :

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

We need :

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

$$\frac{\Gamma, \alpha^+ : S, \alpha^- : \overline{S} \vdash (\nu x : T)P}{\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x : T \vdash P}\ \textsc{T-New}$$

$\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x : T \vdash P$ and $P \xrightarrow{\alpha,l} P'$, so, by the induction hypothesis,

$\Gamma, \alpha^+ : \mathsf{tail}(S, l), \alpha^- : \mathsf{tail}(\overline{S}, l), x : T \vdash P'$.

$$\frac{\Gamma, \alpha^+ : \mathsf{tail}(S, l), \alpha^- : \mathsf{tail}(\overline{S}, l), x : T \vdash P'}{\Gamma, \alpha^+ : \mathsf{tail}(S, l), \alpha^- : \mathsf{tail}(\overline{S}, l) \vdash (\nu x : T)P'}\ \textsc{T-New}$$

So, we have :

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

which is what we need.

R-NewSeq

The derivation ends with :

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

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

We have :

$$\Gamma \vdash (\nu x^\pm : S)P$$

We need :

$$\Gamma \vdash (\nu x^\pm : S)P'$$

$$\frac{\Gamma \vdash (\nu x^\pm : S)P}{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}\ \textsc{T-NewSeq}$$

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

$\Gamma, x^+ : S, x^- : \overline{S} \vdash P'$.

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P'}{\Gamma, \vdash (\nu x^\pm : S)P'}\ \textsc{T-NewSeq}$$

So, we have :

$$\Gamma \vdash (\nu x^\pm : S)P'$$

which is what we need.

Case 2 : α is a session channel
We have :

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

We need :

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

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

$\Gamma, \alpha^+ : S, \alpha^- : \overline{S}, x^+ : S_2, x^- : \overline{S_2} \vdash P$ and $P \xrightarrow{\alpha,l} P'$, so, by the induction hypothesis, $\Gamma, \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l), x^+ : S_2, x^- : \overline{S_2} \vdash P'$.

$$\frac{\Gamma, \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l), x^+ : S_2, x^- : \overline{S_2} \vdash P'}{\Gamma, \alpha^+ : \mathsf{tail}(S,l), \alpha^- : \mathsf{tail}(\overline{S},l) \vdash (\nu x^\pm : S)P'}\ \text{T-NewSeq}$$

So, we have :

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

which is what we need.

R-NewX

The derivation ends with :

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

We have :

$$\Gamma \vdash (\nu x : T)P$$

We need :

$$\Gamma \vdash (\nu x : T)P'$$

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

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

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

So, we have :

$$\Gamma \vdash (\nu x : T)P'$$

which is what we need.

R-NewXSeq

The derivation ends with :

$$\frac{P \xrightarrow{x,l} P'}{(\nu x^{\pm} : T)P \xrightarrow{\tau,-} (\nu x^{\pm} : \mathrm{tail}(T,l))P'}$$

We have :

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

We need :

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

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

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

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

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

So, we have :

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

which is what we need.

$\square$

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 2 [Run-Time Safety]

If $\Gamma \vdash P$, $P \equiv (\nu\tilde{w} : \tilde{W})(Q \mid R)$, $Q \xrightarrow{\alpha,l} Q'$ 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 $z_i \in \tilde{z}$, $\Gamma \vdash z_i : T_i$ or $(\nu z_i : T_i) \in (\nu\tilde{w} : \tilde{W})$

and

$\Gamma \vdash x : \hat{}[\tilde{T}]$ or $(\nu x : \hat{}[\tilde{T}]) \in (\nu\tilde{w} : \tilde{W})$

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 $z_i \in \tilde{z}$, $\Gamma \vdash z_i : T_i$ or $(\nu z_i : T_i) \in (\nu\tilde{w} : \tilde{W})$

and

$\Gamma \vdash x^p : ?[\tilde{T}]\ .\ S, x^q : ![\tilde{T}]\ .\ \overline{S}$ 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

$\Gamma \vdash x^p : \&\langle l_1 : S_1, \ldots, l_n : S_n\rangle, x^q : \oplus\langle l_1 : S_1, \ldots, l_n : S_n\rangle$
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

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

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

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

We have :

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

We need :

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

and

$\Gamma \vdash x : \,\hat{}\,[\tilde{T}]$ or $(\nu x : \,\hat{}\,[\tilde{T}]) \in (\nu\tilde{w} : \tilde{W})$

$$\frac{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2) \mid R)}{\Gamma, \tilde{w} : \tilde{W} \vdash (x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2) \mid R}\ \textsc{T-New}$$

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vdash (x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2) \mid R}{\Gamma_1 \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2 \qquad \Gamma_2 \vdash R}\ \textsc{T-Par}$$

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

$$\frac{\Gamma_1 \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2}{\Gamma_3 \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \qquad \Gamma_4 \vdash x\,!\,[\tilde{z}]\,.\,Q_2}\ \textsc{T-Par}$$

where $\Gamma_3 + \Gamma_4 = \Gamma_1$

$$\frac{\Gamma_3 \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1}{\Gamma_3, \tilde{y} : \tilde{T} \vdash Q_1 \qquad \Gamma_3 \vdash x : \,\hat{}\,[\tilde{T}]}\ \textsc{T-In}$$

Let $\Gamma_4 = \Gamma_5 + \tilde{z} : \tilde{U}$ for some $\tilde{U}$.

$$\frac{\Gamma_5 + \tilde{z} : \tilde{U} \vdash x\,!\,[\tilde{z}]\,.\,Q_2}{\Gamma_5 \vdash Q_2 \qquad \Gamma_5 \vdash x : \,\hat{}\,[\tilde{U}]}\ \textsc{T-Out}$$

$\Gamma_3 + \Gamma_4 = \Gamma_1$. So, $\Gamma_3 + \Gamma_5 + \tilde{z} : \tilde{U} = \Gamma_1$.

But, $\Gamma_3 \vdash x : \,\hat{}\,[\tilde{T}]$ and $\Gamma_5 \vdash x : \,\hat{}\,[\tilde{U}]$. So, $\tilde{U} = \tilde{T}$.

$\Gamma_3 \vdash x : \,\hat{}\,[\tilde{T}]$, $\Gamma_4 \vdash \tilde{z} : \tilde{T}$ and $\Gamma_3 + \Gamma_4 + \Gamma_2 = \Gamma, \tilde{w} : \tilde{W}$.

So, $\Gamma, \tilde{w} : \tilde{W} \vdash x : \,\hat{}\,[\tilde{T}]$ and $\Gamma, \tilde{w} : \tilde{W} \vdash \tilde{z} : \tilde{T}$.

$$\frac{\Gamma, \tilde{w} : \tilde{W} \vdash x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2 \qquad \Gamma_2 \vdash R}{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((x\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x\,!\,[\tilde{z}]\,.\,Q_2) \mid R)}\ \textsc{T-New}$$

We know, for both x and for all $z_i \in \tilde{z}$, that they are either in Γ or are bound in $(\nu\tilde{w} : \tilde{W})$.

So, we have :

$$\text{For all } z_i \in \tilde{z},\ \Gamma \vdash z_i : T_i \text{ or } (\nu z_i : T_i) \in (\nu \tilde{w} : \tilde{W})$$
$$\text{and}$$
$$\Gamma \vdash x : \hat{}[\tilde{T}] \text{ or } (\nu x : \hat{}[\tilde{T}]) \in (\nu \tilde{w} : \tilde{W})$$

which is what we need.

Case 2 : $Q \equiv x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2$ *where* x *is a session channel*

We have :

$$\Gamma \vdash (\nu \tilde{w} : \tilde{W})((x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2) \mid R)$$

We need :

$$p = \overline{q}$$
$$\text{and}$$
$$\text{For all } z_i \in \tilde{z},\ \Gamma \vdash z_i : T_i \text{ or } (\nu z_i : T_i) \in (\nu \tilde{w} : \tilde{W})$$
$$\text{and}$$
$$\Gamma \vdash x^p : ?[\tilde{T}]\,.\,S, x^q : ![\tilde{T}]\,.\,\overline{S} \text{ or } (\nu x^{\pm} : ?[\tilde{T}]\,.\,S) \in (\nu \tilde{w} : \tilde{W})$$
$$\text{or } (\nu x^{\pm} : ![\tilde{T}]\,.\,\overline{S}) \in (\nu \tilde{w} : \tilde{W})$$

$$\dfrac{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2) \mid R)}{\Gamma, \tilde{w} : \tilde{W} \vdash (x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2) \mid R}\;\textsc{T-New}$$

$$\dfrac{\Gamma, \tilde{w} : \tilde{W} \vdash (x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2) \mid R}{\Gamma_1 \vdash x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2 \qquad \Gamma_2 \vdash R}\;\textsc{T-Par}$$

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

$$\dfrac{\Gamma_1 \vdash x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2}{\Gamma_3 \vdash x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \qquad \Gamma_4 \vdash x^q\,!\,[\tilde{z}]\,.\,Q_2}\;\textsc{T-Par}$$

where $\Gamma_3 + \Gamma_4 = \Gamma_1$

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

Let $\Gamma_3 = \Gamma_5, x^p : ?[\tilde{T}]\,.\,S$.

$$\dfrac{\Gamma_5, x^p : ?[\tilde{T}]\,.\,S \vdash x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1}{\Gamma_5, x^p : S, \tilde{y} : \tilde{T} \vdash Q_1}\;\textsc{T-OutSeq}$$

Let $\Gamma_4 = \Gamma_6, x^q : ![\tilde{U}]\,.\,S' + \tilde{z} : \tilde{U}$ for some $\tilde{U}$.

$$\dfrac{\Gamma_6, x^q : ![\tilde{U}]\,.\,S' + \tilde{z} : \tilde{U} \vdash x^q\,!\,[\tilde{z}]\,.\,Q_2}{\Gamma_6, x^q : S' \vdash Q}\;\textsc{T-OutSeq}$$

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

So, $\Gamma, \tilde{w} : \tilde{W} \vdash x^p : ?[\tilde{T}]\,.\,S$ and $\Gamma, \tilde{w} : \tilde{W} \vdash x^q : ![\tilde{U}]\,.\,S'$.

$$\dfrac{\Gamma, \tilde{w} : \tilde{W} \vdash (x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2) \mid R}{\Gamma \vdash (\nu\tilde{w} : \tilde{W})((x^p\,?\,[\tilde{y} : \tilde{T}]\,.\,Q_1 \mid x^q\,!\,[\tilde{z}]\,.\,Q_2) \mid R)}\;\textsc{T-New}$$

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, $\tilde{U} = \tilde{T}$ and $S' = \overline{S}$ and therefore, $\Gamma \vdash x^p : ?[\tilde{T}]\,.\,S$ and $\Gamma \vdash x^q : ![\tilde{T}]\,.\,\overline{S}$.

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, $\tilde{U} = \tilde{T}$ and $S' = \overline{S}$ and therefore, $(\nu x^\pm : ?[\tilde{T}]\,.\,S) \in (\nu\tilde{w} : \tilde{W})$ or $(\nu x^\pm : ![\tilde{T}]\,.\,\overline{S}) \in (\nu\tilde{w} : \tilde{W})$.

In either case $\tilde{U} = \tilde{T}$. So, $\Gamma_4 \vdash \tilde{z} : \tilde{T}$.

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

Therefore, for all $z_i \in \tilde{z}$, we know either $z_i \in \Gamma$ or z_i is bound in $(\nu\tilde{w} : \tilde{W})$.

So, we have :

$$
\begin{aligned}
& p = \overline{q} \\
& \text{and} \\
& \text{For all } z_i \in \tilde{z},\ \Gamma \vdash z_i : T_i \text{ or } (\nu z_i : T_i) \in (\nu \tilde{w} : \tilde{W}) \\
& \text{and} \\
& \Gamma \vdash x^p : ?[\tilde{T}] \,.\, S, x^q : ![\tilde{T}] \,.\, \overline{S} \text{ or } (\nu x^{\pm} : ?[\tilde{T}] \,.\, S) \in (\nu \tilde{w} : \tilde{W}) \\
& \qquad \text{or } (\nu x^{\pm} : ![\tilde{T}] \,.\, \overline{S}) \in (\nu \tilde{w} : \tilde{W})
\end{aligned}
$$

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 :

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

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

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

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 : $Q \equiv \textsf{ if } x \textsf{ then } Q_1 \textsf{ else } Q_2$

We have :

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

We need :

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

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

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

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

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

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

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

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

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

So, we have :

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

which is what we need.

$\square$