Global Progress in Dynamically Interleaved Multiparty Sessions

Lorenzo Bettini, Mario Coppo, Mariangiola Dezani-Ciancaglini
Dipartimento di Informatica, Università di Torino
and
Nobuko Yoshida
Department of Computing, Imperial College London

A multiparty session forms a unit of structured interactions among many participants which follow a prescribed scenarios specified as a global type signature. When a distributed protocol is engaged in two or more specifications simultaneously, each session following a distinct global type can be dynamically merged and interfered by another at runtime (through the channel delegation operation). Previous work on multiparty session types has ignored this dynamic nature, providing a limited progress property ensured only within a single session, by assuming non-interference among different sessions and by forbidding delegation.

This paper develops, besides a more traditional *communication* type system, a novel static *interaction* type system for global progress in dynamically merged and interfered multiparty sessions. High-level session processes equipped with global signatures are translated into low-level processes, in which type-soundness is guaranteed against the local, compositional type system. This avoids a global linearity-check without scarifying the original expressivity. The interaction type system automatically infers causalities of channels for the low level processes, ensuring the entire protocol, starting from the high-level processes which consist of multiple sessions, does not get stuck at intermediate sessions also in presence of delegation.

Categories and Subject Descriptors: []:

General Terms:

Additional Key Words and Phrases:

December 20, 2009

1. INTRODUCTION

Widespread use of message-based communication for developing network applications to combine numerous distributed services has provoked urgent interests in structuring series of interactions to specify and program communication-safe software. The actual development of such applications still leaves to the programmer much of the responsibility in guaranteeing that communication will evolve as agreed by all the involved distributed

The work is partially supported by IST-3-016004-IP-09 SENSORIA, EPSRC GR/T03208, EPSRC EP/F003757 and IST2005-015905 MOBIUS.

peers. *Multiparty session type discipline* proposed in [Honda et al. 2008] offers a type-theoretic framework to validate a messages-exchange among concurrently running multiple peers in the distributed environment, generalising the existing two-party session types [Honda 1993; Honda et al. 1998]; interaction sequences are abstracted as a global type signature, which precisely declares how multiple participants communicate and synchronise with each other. Recently the two standardisation bodies for web-based business and finance protocols [Web Services Choreography Working Group 2002; UNIFI 2002] have investigated a design and implementation framework for standardising message exchange rules and validating business logic based on a notion of multiparty sessions, where a global type plays as a "shared agreement" between a team of programmers who are developing (possibly) a large size of distributed protocol or software by collaborations.

The initial multiparty session type discipline aims to retain the powerful dynamic features from the original binary sessions [Honda et al. 1998], incorporating features such as recursion and choice of interactions. Among features, *session delegation* is a key operation which permits to rely on other parties for completing specific tasks transparently in a type safe manner. A typical scenario is a web server delegating remaining interactions with a client to an application server to complete a transaction. The customer and the application server are initially unknown to each other but later communicate directly (transparently to the customer), through dynamic mobility of the session. When this mechanism is extended to multiparty interactions engaged in two or more specifications simultaneously, further complex interactions can be modelled: each multiparty session following a distinct global type can be dynamically merged and interfered by another at runtime via the channel delegation operation, grouping several structured conversations.

Previous work on multiparty session types [Honda et al. 2008] has ignored this dynamic nature, providing a limited progress property ensured only within a single session, by assuming non-interference among different sessions and by forbidding delegation. More precisely, although the previous system assures that the multiple participants respect the protocol, by checking the types of exchanged messages and the order of communications, it cannot guarantee a *global progress*, i.e, that a protocol which merges several global scenarios will not get stuck in the middle of a session. This limitation prohibits to ensure a successful termination of a transaction, making the framework practically inapplicable to a large size of dynamically reconfigured conversations.

This paper develops, besides a more traditional *communication* type system (§ 4), a novel static *interaction* type system (§ 5) for global progress in dynamically merged and interfered multiparty, asynchronous sessions. High-level session processes equipped with global signatures are translated into low-level processes which have explicit senders and receivers (§ 2.5). Type-soundness of low-level processes is guaranteed against the local, compositional communication type system (§ 4.4).

The new calculus for multiparty sessions offers three technical merits without sacrificing the original simplicity and expressivity in [Honda et al. 2008]. First it avoids the overhead of global linearity-check in [Honda et al. 2008]; secondly it provides a more liberal policy in the use of variables, both in delegation and in recursive definitions; finally it implicitly provides each participant of a service with a runtime channel indexed by its role with which he can communicate with all other participants, permitting also broadcast in a natural way. The use of indexed channels, moreover, permits to define a light-weight interaction type system for global progress.

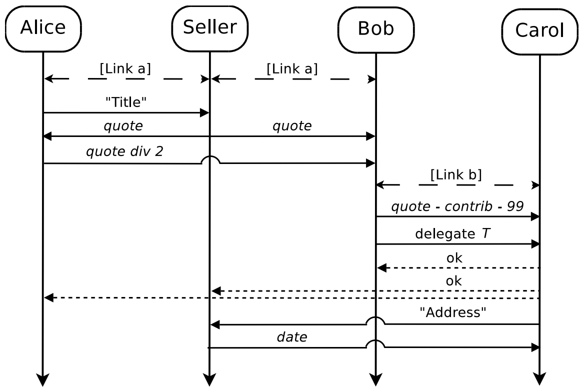

Fig. 1. The three buyer protocol interactions

The interaction type system automatically infers causalities of channels for the low level processes, ensuring the entire protocol, starting from the high-level processes which consist of multiple sessions, does not get stuck at intermediate sessions also in the presence of delegation.

2. SYNTAX

Before presenting the syntax of our calculus formally, we provide a small example, in order to give a basic idea of the functionalities and the linguistic features of our language.

2.1 Merging Two Conversations: Three Buyer Protocol

We introduce our calculus through an example, the three-buyer protocol, extending the two-buyer protocol from [Honda et al. 2008], which includes the new features, session-multicasting and dynamically merging two conversations. The overall scenario, involving a Seller (S), Alice (A), Bob (B) and Carol (C), proceeds as follows.

(1) Alice sends a book title to Seller, then Seller sends back a quote to Alice and Bob. Then Alice tells Bob how much she can contribute.
(2) If the price is within Bob's budget, Bob notifies both Seller and Alice he accepts, then sends his address, and Seller sends back the delivery date.
(3) If the price exceeds the budget, Bob asks Carol to collaborate together by establishing a new session. Then Bob sends how much Carol must pay, then *delegates* the remaining interactions with Alice and Seller to Carol.
(4) If the rest of the price is within Carol's budget, Carol accepts the quote and notifies Alice, Bob and Seller, and continues the rest of the protocol with Seller and Alice transparently, *as if she were Bob*. Otherwise she notifies Alice, Bob and Seller to quit the protocol.

Figure 1 depicts an execution of the above protocol where Bob asks Carol to collaborate (by delegating the remaining interactions with Alice and Seller) and the transaction terminates successfully.

Then multiparty session programming consists of two steps: specifying the intended communication protocols using global types, and implementing these protocols using processes. The specifications of the three-buyer protocol are given as two separated global types: one is G_a among Alice, Bob and Seller and the other is G_b between Bob and Carol. We write principals with legible symbols though they will actually be coded by numbers: in G_a we have $\mathtt{S} = 3$, $\mathtt{A} = 1$ and $\mathtt{B} = 2$, while in G_b we have $\mathtt{B} = 2$, $\mathtt{C} = 1$.

$$
\begin{array}{ll}
G_a = & \\
\mathtt{1.}\ \mathtt{A} \longrightarrow \mathtt{S}: & \langle \mathsf{string}\rangle. \\
\mathtt{2.}\ \mathtt{S} \longrightarrow \{\mathtt{A},\mathtt{B}\}: & \langle \mathsf{int}\rangle. \\
\mathtt{3.}\ \mathtt{A} \longrightarrow \mathtt{B}: & \langle \mathsf{int}\rangle. \\
\mathtt{4.}\ \mathtt{B} \longrightarrow \{\mathtt{S},\mathtt{A}\}: & \{\mathsf{ok}: \mathtt{B} \longrightarrow \mathtt{S}: \langle \mathsf{string}\rangle. \\
\mathtt{5.} & \qquad \mathtt{S} \longrightarrow \mathtt{B}: \langle \mathsf{date}\rangle; \mathsf{end} \\
\mathtt{6.} & \ \mathsf{quit}: \mathsf{end}\}
\end{array}
\qquad
\begin{array}{l}
G_b = \\
\mathtt{1.}\ \mathtt{B} \longrightarrow \mathtt{C}: \langle \mathsf{int}\rangle. \\
\mathtt{2.}\ \mathtt{B} \longrightarrow \mathtt{C}: \langle T\rangle. \\
\mathtt{3.}\ \mathtt{C} \longrightarrow \mathtt{B}: \{\mathsf{ok}: \mathsf{end},\quad \mathsf{quit}: \mathsf{end}\}. \\
\ T = \\
\ \oplus(\{\mathtt{S},\mathtt{A}\}, \\
\quad \{\mathsf{ok}: !\langle \mathtt{S}, \mathsf{string}\rangle; ?\langle \mathtt{S}, \mathsf{date}\rangle; \mathsf{end}, \\
\quad \mathsf{quit}: \mathsf{end}\})
\end{array}
$$

The types give a global view of the two conversations, directly abstracting the scenario given by the diagram. In G_a, line 1 denotes A sends a string value to S. Line 2 says S multicasts the same integer value to A and B and line 3 says that A sends an integer to B. In lines 4-6 B sends either ok or quit to S and A. In the first case B sends a string to S and receives a date from S, in the second case there are no further communications.

Line 2 in G_b represents the delegation of the capability specified by the action type T of channels (formally defined later) from B to C (note that S and A in T concern the session on a).

We now give the code, associated to G_a and G_b, for S, A, B and C in a "user" syntax formally defined in the following section[1]:

$$
\begin{array}{lcl}
\mathtt{S} & = & \overline{a}[3](y_3).y_3?(\mathit{title}); y_3!\langle \mathsf{quote}\rangle; y_3\&\{\mathsf{ok}: y_3?(\mathit{address}); y_3!\langle \mathsf{date}\rangle; \mathbf{0},\ \mathsf{quit}: \mathbf{0}\} \\
\mathtt{A} & = & a[1](y_1).y_1!\langle \texttt{"Title"}\rangle; y_1?(\mathit{quote}); y_1!\langle \mathit{quote}\ \mathsf{div}\ 2\rangle; y_1\&\{\mathsf{ok}: \mathbf{0},\ \mathsf{quit}: \mathbf{0}\} \\
\mathtt{B} & = & a[2](y_2).y_2?(\mathit{quote}); y_2?(\mathit{contrib}); \\
& & \quad \mathsf{if}\ (\mathit{quote}\ \text{-}\ \mathit{contrib} < 100)\ \mathsf{then}\ \ y_2 \oplus \mathsf{ok}; y_2!\langle \texttt{"Address"}\rangle; y_2?(\mathit{date}); \mathbf{0} \\
& & \quad \mathsf{else}\ \overline{b}[2](z_2).z_2!\langle \mathit{quote}\ \text{-}\ \mathit{contrib}\ \text{-}\ 99\rangle; z_2!\langle\langle y_2\rangle\rangle; z_2\&\{\mathsf{ok}: \mathbf{0},\ \mathsf{quit}: \mathbf{0}\} \\
\mathtt{C} & = & b[1](z_1).z_1?(x); z_1?((t)); \\
& & \qquad \mathsf{if}\ (x < 100)\ \mathsf{then}\ z_1 \oplus \mathsf{ok}; t \oplus \mathsf{ok}; t!\langle \texttt{"Address"}\rangle; t?(\mathit{date}); \mathbf{0} \\
& & \qquad \mathsf{else}\ z_1 \oplus \mathsf{quit}; t \oplus \mathsf{quit}; \mathbf{0}
\end{array}
$$

Session name a establishes the session corresponding to G_a. S initiates a session involving three bodies as third participant by $\overline{a}[3](y_3)$: A and B participate as first and second participants by $a[1](y_1)$ and $a[2](y_2)$, respectively. Then S, A and B communicate using the channels y_3, y_1 and y_2, respectively. Each channel $y_{\mathtt{p}}$ can be seen as a port connecting participant p with all other ones; the receivers of the data sent on $y_{\mathtt{p}}$ are specified by the global type (this information will be included in the runtime code). The first line of G_a is implemented by the input and output actions $y_3?(\mathit{title})$ and $y_1!\langle \texttt{"Title"}\rangle$. The last line of G_b is implemented by the branching and selection actions $z_2\&\{\mathsf{ok}: \mathbf{0},\ \mathsf{quit}: \mathbf{0}\}$ and $z_1 \oplus \mathsf{ok}$, $z_1 \oplus \mathsf{quit}$.

In B, if the quote minus A's contribution exceeds 100€ (i.e., quote - $\mathit{contrib} \geq 100$), another session between B and C is established dynamically through shared name b. The del-

[1]In the examples we will use the following font conventions: variables (bound by an input action) are in *italics* and constants (when their value is not relevant) are in sans serif; string literals are in `monospace` font and double quoted.

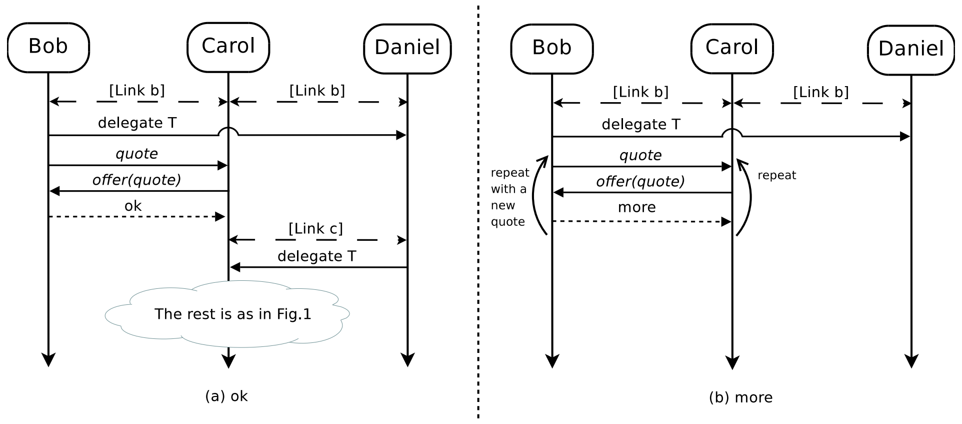

Fig. 2. The three buyer protocol (with recursion): additional interactions

egation is performed by passing the channel y_2 from B to C (actions $z_2!\langle\langle y_2\rangle\rangle$ and $z_1?((t))$), and so the rest of the session is carried out by C with S and A. We can further enrich this protocol with recursive-branching behaviours in interleaved sessions (for example, C can repeatedly negotiate the quote with S as if she were B). What we want to guarantee by static type-checking is that the whole conversation between the four parties preserves progress as if it were a single conversation.

2.2 Three Buyer Protocol (with recursion)

We now describe a variant of the example of Section 2.1 that uses recursion; in particular, the scenario is basically the same, the only part that changes is that, if the price exceeds the budget, Bob initiates a negotiate with Carol to collaborate together by establishing a new session: Bob starts asking a first proposal of contribution to Carol. At each step Carol answers with a new offer. Bob can accept the offer, try with a new proposal or give up. When Bob decides to end the negotiate (accepting the offer or giving up) he communicated the exit to Carol and, as before, Carol concludes the protocol with Seller. The auxiliary process Daniel (D) (as well as the session c) is used as a kind of temporary store for the delegation channel y_2 during the negotiation between Alice and Bob. Figure 2 depicts the part of the protocol involving possible recursion (case (b), right).

The communication protocols are described by the following global types; these are similar to the ones of Section 2.1. In particular G_a is exactly the same (since the server does not notice the further interactions among the buyers). Instead, G_b is now more involved since we have a recursive part which represents the (possibly) recursive negotiation between Bob and Carol; finally G_c simply register the transfer the delegation channel to Carol.

$$
\begin{array}{lll}
\mathtt{S} & = & \overline{a}[3](y_3).y_3?(\mathit{title});y_3!\langle \mathsf{quote}\rangle;y_3\&\{\mathsf{ok}: y_3?(\mathit{address});y_3!\langle \mathsf{date}\rangle;\mathbf{0},\ \mathsf{quit}:\mathbf{0}\} \\
\mathtt{A} & = & a[1](y_1).y_1!\langle \texttt{"Title"}\rangle;y_1?(\mathit{quote});y_1!\langle \mathit{quote}\ \mathsf{div}\ 2\rangle;y_1\&\{\mathsf{ok}:\mathbf{0},\ \mathsf{quit}:\mathbf{0}\} \\
\mathtt{B} & = & a[2](y_2).y_2?(\mathit{quote});y_2?(\mathit{contrib}); \\
 & & \mathsf{if}\ (\mathit{quote}\ \text{-}\ \mathit{contrib} < 100)\ \mathsf{then}\ \ y_2 \oplus \mathsf{ok};y_2!\langle \texttt{"Address"}\rangle;y_2?(\mathit{date});\mathbf{0} \\
 & & \mathsf{else}\ \overline{b}[2](z_2). \\
 & & \quad \mathsf{def}\ X(x,z,y) = \\
 & & \qquad z!\langle x\rangle;z?(x_1); \\
 & & \qquad \mathsf{if}\ \mathit{good}(x_1)\ \mathsf{then}\ z\oplus \mathsf{ok};z!\langle\langle y\rangle\rangle;\mathbf{0} \\
 & & \qquad \mathsf{else\ if}\ \mathit{negotiable}(x_1)\ \mathsf{then}\ y\oplus \mathsf{more};X(\mathit{newproposal}(x_1),z,y) \\
 & & \qquad\quad \mathsf{else}\ y\oplus \mathsf{quit};z!\langle\langle y\rangle\rangle;\mathbf{0} \\
 & & \quad \mathsf{in}\ X(\mathit{firstproposal}(\mathit{quote}),z_2,y_2) \\
\mathtt{C} & = & b[1](z_1).\ \mathsf{def}\ Y(u) = u?(x');u!\langle \mathit{offer}(x')\rangle; \\
 & & \quad y\&\{\mathsf{ok}: u?((t));t\oplus \mathsf{ok};t!\langle \texttt{"Address"}\rangle;t?(\mathit{date});\mathbf{0} \\
 & & \qquad \mathsf{more}: Y(u), \\
 & & \qquad \mathsf{quit}: u?((t));t\oplus \mathsf{quit};\mathbf{0}\} \\
 & & \quad \mathsf{in}\ Y(z_1)
\end{array}
$$

Fig. 3. The three buyer example with recursion.

$$
G_a =
\begin{array}{llllll}
1. & \mathtt{A} & \longrightarrow & \mathtt{S}: & \langle \mathsf{string}\rangle. & \\
2. & \mathtt{S} & \longrightarrow & \{\mathtt{A},\mathtt{B}\}: & \langle \mathsf{int}\rangle. & \\
3. & \mathtt{A} & \longrightarrow & \mathtt{B}: & \langle \mathsf{int}\rangle. & \\
4. & \mathtt{B} & \longrightarrow & \{\mathtt{S},\mathtt{A}\}: & \{\mathsf{ok}: \mathtt{B} \longrightarrow \mathtt{S}: \langle \mathsf{string}\rangle. & \\
5. & & & & \quad \mathtt{S} \longrightarrow \mathtt{B}: \langle \mathsf{date}\rangle;\mathsf{end} & \\
6. & & & & \mathsf{quit}: \mathsf{end}\} &
\end{array}
$$

$$
G_b =
\begin{array}{lllll}
1. & \mu\mathbf{t}.\mathtt{B} & \longrightarrow & \mathtt{C}: & \langle \mathsf{int}\rangle. \\
2. & \mathtt{C} & \longrightarrow & \mathtt{B}: & \langle \mathsf{int}\rangle. \\
3. & \mathtt{B} & \longrightarrow & \mathtt{C}: & \{\mathsf{ok}: \mathtt{B} \longrightarrow \mathtt{C}: \langle T\rangle;\mathsf{end}, \\
 & & & & \ \mathsf{more}: \mathbf{t}, \\
 & & & & \ \mathsf{quit}: \mathtt{B} \longrightarrow \mathtt{C}: \langle T\rangle;\mathsf{end}\}
\end{array}
$$

$$
\begin{array}{l}
T = \\
\oplus(\{\mathtt{S},\mathtt{A}\}, \\
\quad \{\mathsf{ok}: !\langle \mathtt{S},\mathsf{string}\rangle;?\langle \mathtt{S},\mathsf{date}\rangle;\mathsf{end}, \\
\quad \mathsf{quit}: \mathsf{end}\})
\end{array}
$$

The code of the example is in Figure 3. Again, it is similar to the previous one, but for the recursive negotiation between Bob and Carol, and for the presence of the auxiliary process and session.

2.3 User syntax

The syntax for processes initially written by the user, called *user-defined processes*, is based on [Honda et al. 2008]. We start from the following sets: *service names*, ranged over by $a, b, \ldots$ (representing public names of endpoints), *value variables*, ranged over by $x, x', \ldots$, *identifiers* , i.e., service names and variables, ranged over by $u, w, \ldots$, *channel variables*, ranged over by $y, z, t \ldots$, *labels*, ranged over by $l, l', \ldots$ (functioning like method names or labels in labelled records); *process variables*, ranged over by $X, Y, \ldots$ (used for representing recursive behaviour). Then *processes*, ranged over by $P, Q \ldots$, and *expressions*, ranged over by $e, e', \ldots$, are given by the grammar in Table I.

For the primitives for session initiation, $\overline{u}[n](y).P$ initiates a new session through an identifier u (which represents a shared interaction point) with the other multiple participants, each of shape $u[\mathtt{p}](y).Q_{\mathtt{p}}$ where $1 \leq \mathtt{p} \leq n-1$. The (bound) variable y is the channel used to do the communications. We call $\mathtt{p}$, $\mathtt{q}$,... (ranging over natural numbers) the *par-*

P	$::=$	$\overline{u}[n](y).P$	Multicast Request		$\mid$	if e then P else Q	Conditional
	$\mid$	$u[\mathtt{p}](y).P$	Accept		$\mid$	$P \mid Q$	Parallel
	$\mid$	$y!\langle e\rangle;P$	Value sending		$\mid$	$\mathbf{0}$	Inaction
	$\mid$	$y?(x);P$	Value reception		$\mid$	$(\nu a)P$	Hiding
	$\mid$	$y!\langle\langle z\rangle\rangle;P$	Session delegation		$\mid$	def D in P	Recursion
	$\mid$	$y?((z));P$	Session reception		$\mid$	$X\langle e,y\rangle$	Process call
	$\mid$	$y\oplus l;P$	Selection				
	$\mid$	$y\&\{l_i : P_i\}_{i\in I}$	Branching	e	$::=$	$v \mid x$	
u	$::=$	$x \mid a$	Identifier		$\mid$	e and e' $\mid$ not e ...	Expression
v	$::=$	$a \mid$ true $\mid$ false	Value	D	$::=$	$X(x,y)=P$	Declaration

Table I. Syntax for user-defined processes

ticipants of a session. Session communications (communications that take place inside an established session) are performed using the next three pairs of primitives: the sending and receiving of a value; the session delegation and reception (where the former delegates to the latter the capability to participate in a session by passing a channel associated with the session); and the selection and branching (where the former chooses one of the branches offered by the latter). The rest of the syntax is standard from [Honda et al. 1998].

2.4 Global Types

A *global type*, ranged over by $G, G', ..$ describes the whole conversation scenario of a multiparty session as a type signature. Its grammar is given below:

$$\begin{array}{lllllllll}
\text{Global} & G & ::= & \mathtt{p}\to\Pi : \langle U\rangle.G' & \qquad & \text{Exchange} & U & ::= & S \mid T \\
 & & \mid & \mathtt{p}\to\Pi : \{l_i : G_i\}_{i\in I} & & \text{Sorts} & S & ::= & \mathsf{bool} \mid \ldots \mid G \\
 & & \mid & \mu\mathbf{t}.G \mid \mathbf{t} \mid \mathsf{end} & & & & &
\end{array}$$

We simplify the syntax in [Honda et al. 2008] by eliminating channels and parallel compositions, while preserving the original expressivity (see § 7).

The global type $\mathtt{p}\to\Pi : \langle U\rangle.G'$ says that participant $\mathtt{p}$ multicasts a message of type U to participants $\mathtt{p}_k$ ($\mathtt{p}_k \in \Pi$) and then interactions described in G' take place. *Exchange types* $U, U', ...$ consist of *sorts* types $S, S', \ldots$ for values (either base types or global types), and *action* types $T, T', \ldots$ for channels (discussed in §4). Type $\mathtt{p}\to\Pi : \{l_i : G_i\}_{i\in I}$ says participant $\mathtt{p}$ multicasts one of the labels l_i to participants $\mathtt{p}_k$ ($\mathtt{p}_k \in \Pi$). If l_j is sent, interactions described in G_j take place. Type $\mu\mathbf{t}.G$ is a recursive type, assuming type variables ($\mathbf{t}, \mathbf{t}', \ldots$) are guarded in the standard way, i.e., type variables only appear under some prefix. We take an *equi-recursive* view of recursive types, not distinguishing between $\mu\mathbf{t}.G$ and its unfolding $G\{\mu\mathbf{t}.G/\mathbf{t}\}$ [Pierce 2002, §21.8]. We assume that G in the grammar of sorts is closed, i.e., without free type variables. Type end represents the termination of the session. We often write $\mathtt{p}\to\mathtt{p}'$ for $\mathtt{p}\to\{\mathtt{p}'\}$.

2.5 Runtime Syntax

User defined processes equipped with global types are executed through a translation into runtime processes. The runtime syntax (Table II) differs from the syntax of Table I since the input/output operations (including the delegation ones) specify the sender and the receiver, respectively. Thus, $c!\langle\Pi, e\rangle$ sends a value to all the participants in Π; accordingly, $c?(\mathtt{p}, x)$ denotes the intention of receiving a value from the participant $\mathtt{p}$. The same holds for

P	$::=$	$c!\langle \Pi, e\rangle; P$	Value sending	$\mid$	$c \oplus \langle \Pi, l\rangle; P$	Selection
	$\mid$	$c?(\mathtt{p}, x); P$	Value reception	$\mid$	$c\&(\mathtt{p}, \{l_i : P_i\}_{i\in I})$	Branching
	$\mid$	$c!\langle\langle \mathtt{p}, c'\rangle\rangle; P$	Session delegation	$\mid$	$(\nu s)P$	Hiding session
	$\mid$	$c?((\mathtt{q}, y)); P$	Session reception	$\mid$	$s : h$	Named queue
				$\mid$	...	
c	$::=$	$y \mid s[\mathtt{p}]$	Channel			
m	$::=$	$(\mathtt{q}, \Pi, v) \mid (\mathtt{q}, \mathtt{p}, s[\mathtt{p}']) \mid (\mathtt{q}, \Pi, l)$	Message in transit			
h	$::=$	$m \cdot h \mid \varnothing$	Queue			

Table II. Runtime syntax: the other syntactic forms are as in Table I

delegation/reception (but the receiver is only one) and selection/branching.

We call $s[\mathtt{p}]$ a *channel with role*: it represents the channel of the participant $\mathtt{p}$ in the session s. We use c to range over variables and channels with roles. As in [Honda et al. 2008], in order to model TCP-like asynchronous communications (message order preservation and sender-non-blocking), we use the queues of messages in a session, denoted by h; a message in a queue can be a value message, $(\mathtt{q}, \Pi, v)$, indicating that the value v was sent by the participant $\mathtt{q}$ and the recipients are all the participants in Π; a channel message (delegation), $(\mathtt{q}, \mathtt{p}', s[\mathtt{p}])$, indicating that $\mathtt{q}$ delegates to $\mathtt{p}'$ the role of $\mathtt{p}$ on the session s (represented by the channel with role $s[\mathtt{p}]$); and a label message, $(\mathtt{q}, \Pi, l)$ (similar to a value message). The empty queue is denoted by $\varnothing$. With some abuse of notation we will write $h \cdot m$ to denote that m is the last element included in h and $m \cdot h$ to denote that m is the head of h. By $s : h$ we denote the queue h of the session s. In $(\nu s)P$ all occurrences of $s[\mathtt{p}]$ and the queue s are bound. Queues and channels with role are generated by the operational semantics (described later).

We present the translation of Bob ($\mathtt{B}$) in the first three-buyer protocol with the runtime syntax: the only difference is that all input/output operations specify also the sender and the receiver, respectively.

$$
\begin{array}{l}
\mathtt{B} = a[2](y_2).y_2?(3, \mathit{quote}); y_2?(1, \mathit{contrib});\\
\quad \mathsf{if}\ (\mathit{quote}\ \text{-}\ \mathit{contrib} < 100)\ \mathsf{then}\ \ y_2 \oplus \langle\{1,3\}, \mathsf{ok}\rangle; y_2!\langle\{3\}, \texttt{"Address"}\rangle; y_2?(3, \mathit{date}); \mathbf{0}\\
\quad \mathsf{else}\ \overline{b}[2](z_2).z_2!\langle\{1\}, \mathit{quote}\ \text{-}\ \mathit{contrib}\ \text{-}\ 99\rangle; z_2!\langle\langle 1, y_2\rangle\rangle; z_2\&(1, \{\mathsf{ok} : \mathbf{0},\ \mathsf{quit} : \mathbf{0}\}).
\end{array}
$$

It should be clear from this example that starting from a global type and user-defined processes respecting the global type it is possible to add sender and receivers to each communication obtaining in this way processes written in the runtime syntax. We call *pure* a process which does not contain message queues.

3. OPERATIONAL SEMANTICS

The operational semantics consists of some reduction rules and some structural equivalence rules that permit rearranging the terms in order to apply a specific reduction rule.

Table III shows the rules of the process reduction relation $P \longrightarrow P'$ (we use $\longrightarrow^*$ and $\not\longrightarrow$ with the expected meanings). Rule [Link] describes the initiation of a new session among n participants that synchronise over the service name a. The last participant $\overline{a}[n](y_n).P_n$, distinguished by the overbar on the service name, specifies the number n of participants. For this reason we call it the *initiator* of the session. Obviously each session must have a unique initiator. After the connection, the participants will share the private session name s, and the queue associated to s, which is initialized as empty. The variables $y_\mathtt{p}$ in each par-

$$a[1](y_1).P_1 \mid \ldots \mid \overline{a}[n](y_n).P_n \longrightarrow (\nu s)(P_1\{s[1]/y_1\} \mid \ldots \mid P_n\{s[n]/y_n\} \mid s : \varnothing) \quad \text{[Link]}$$

$$s[\mathtt{p}]!\langle \Pi, e\rangle; P \mid s : h \longrightarrow P \mid s : h \cdot (\mathtt{p}, \Pi, v) \quad (e{\downarrow}v) \quad \text{[Send]}$$

$$s[\mathtt{p}]!\langle\langle \mathtt{q}, s'[\mathtt{p}']\rangle\rangle; P \mid s : h \longrightarrow P \mid s : h \cdot (\mathtt{p}, \mathtt{q}, s'[\mathtt{p}']) \quad \text{[Deleg]}$$

$$s[\mathtt{p}] \oplus \langle \Pi, l\rangle; P \mid s : h \longrightarrow P \mid s : h \cdot (\mathtt{p}, \Pi, l) \quad \text{[Label]}$$

$$s[\mathtt{p}_j]?(\mathtt{q}, x); P \mid s : (\mathtt{q}, \{j\}, v) \cdot h \longrightarrow P\{v/x\} \mid s : h \quad \text{[Recv]}$$

$$s[\mathtt{p}]?((\mathtt{q}, y)); P \mid s : (\mathtt{q}, \mathtt{p}, s'[\mathtt{p}']) \cdot h \longrightarrow P\{s'[\mathtt{p}']/y\} \mid s : h \quad \text{[Srec]}$$

$$s[\mathtt{p}_j]\&(\mathtt{q}, \{l_i : P_i\}_{i \in I}) \mid s : (\mathtt{q}, \{j\}, l_{i_0}) \cdot h \longrightarrow P_{i_0} \mid s : h \quad (i_0 \in I) \quad \text{[Branch]}$$

$$\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \longrightarrow P \quad (e \downarrow \mathsf{true}) \qquad \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \longrightarrow Q \quad (e \downarrow \mathsf{false}) \quad \text{[If-T, If-F]}$$

$$\mathsf{def}\ X(x,y) = P\ \mathsf{in}\ (X\langle e, s[\mathtt{p}]\rangle \mid Q) \longrightarrow \mathsf{def}\ X(x,y) = P\ \mathsf{in}\ (P\{v/x\}\{s[\mathtt{p}]/y\} \mid Q) \quad (e \downarrow v) \quad \text{[Def]}$$

$$P \longrightarrow P' \quad \Rightarrow \quad (\nu r)P \longrightarrow (\nu r)P' \qquad P \longrightarrow P' \quad \Rightarrow \quad P \mid Q \longrightarrow P' \mid Q \quad \text{[Scop,Par]}$$

$$P \longrightarrow P' \quad \Rightarrow \quad \mathsf{def}\ D\ \mathsf{in}\ P \longrightarrow \mathsf{def}\ D\ \mathsf{in}\ P' \quad \text{[Defin]}$$

$$P \equiv P' \text{ and } P' \longrightarrow Q' \text{ and } Q \equiv Q' \quad \Rightarrow \quad P \longrightarrow Q \quad \text{[Str]}$$

Table III. Reduction rules

ticipant p will then be replaced with the corresponding channel with role, $s[\mathtt{p}]$. The output rules [Send], [Deleg] and [Label] push values, channels and labels, respectively, into the queue of the session s (in rule [Send], $e \downarrow v$ denotes the evaluation of the expression e to the value v). The rules [Recv], [Srec] and [Branch] perform the corresponding complementary operations. Note that these operations check that the sender matches, and also that the message is actually meant for the receiver (in particular, for [Recv], we need to remove the receiving participant from the set of the receivers in order to avoid reading the same message more than once).

Processes are considered modulo structural equivalence, denoted by $\equiv$ (Table IV); besides the standard rules [Milner 1999], we have a rule for rearranging messages when the senders or the receivers are not the same, and also splitting a message for multiple recipients and the rules for garbage-collecting messages that have already been read by all the intended recipients. By using structural rules we could write the [Recv] and [Branch] rules in a simpler way, instead of their more verbose versions:

$$s[\mathtt{p}_j]?(\mathtt{q}, x); P \mid s : (\mathtt{q}, \Pi, v) \cdot h \longrightarrow P\{v/x\} \mid s : (\mathtt{q}, \Pi \setminus j, v) \cdot h \quad (\mathtt{p}_j \in \Pi)$$

$$s[\mathtt{p}_j]\&(\mathtt{q}, \{l_i : P_i\}_{i \in I}) \mid s : (\mathtt{q}, \Pi, l_{i_0}) \cdot h \longrightarrow P_{i_0} \mid s : (\mathtt{q}, \Pi \setminus j, l_{i_0}) \cdot h \quad (\mathtt{p}_j \in \Pi,\ i_0 \in I)$$

We conclude this section by showing some reduction steps using the example of the three buyer protocol of Section 2; we will consider a simplified version of the example (i.e., the Buyer3 always selects the ok label, without the if ... then ... else ...) and we will concentrate on the part involving delegation. Thus, we assume that the seller and the first two buyers (Alice and Bob) have already established a connection (the session name is s^a) and that Bob is about to establish a connection with Carol; the first line represents the server that is waiting to conclude the transaction with participant 2. We give some reduction steps in Table V. In the computation, Carol plays the role of Bob (participant 2 in the session s^a) transparently to the seller.

$$P \mid \mathbf{0} \equiv P \quad P \mid Q \equiv Q \mid P \quad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$$

$$(\nu r)P \mid Q \equiv (\nu r)(P \mid Q) \quad \text{if } r \notin \mathrm{fn}(Q)$$

$$(\nu r r')P \equiv (\nu r' r)P \quad (\nu r)\mathbf{0} \equiv \mathbf{0} \quad \mathsf{def}\ D\ \mathsf{in}\ \mathbf{0} \equiv \mathbf{0}$$

$$\mathsf{def}\ D\ \mathsf{in}\ (\nu r)P \equiv (\nu r)\mathsf{def}\ D\ \mathsf{in}\ P \quad \text{if } r \notin \mathrm{fn}(D)$$

$$(\mathsf{def}\ D\ \mathsf{in}\ P) \mid Q \equiv \mathsf{def}\ D\ \mathsf{in}\ (P \mid Q) \quad \text{if } \mathrm{dpv}(D) \cap \mathrm{fpv}(Q) = \emptyset$$

$$\mathsf{def}\ D\ \mathsf{in}\ (\mathsf{def}\ D'\ \mathsf{in}\ P) \equiv \mathsf{def}\ D\ \mathsf{and}\ D'\ \mathsf{in}\ P \quad \text{if } \mathrm{dpv}(D) \cap \mathrm{dpv}(D') = \emptyset$$

$$s : (\mathsf{q}, \emptyset, v) \cdot h \equiv s : h \qquad s : (\mathsf{q}, \emptyset, l) \cdot h \equiv s : h$$

$$s : (\mathsf{q}, \Pi, z) \cdot (\mathsf{q}', \Pi', z') \cdot h \equiv s : (\mathsf{q}', \Pi', z') \cdot (\mathsf{q}, \Pi, z) \cdot h \quad \text{if } \Pi \cap \Pi' = \emptyset \text{ or } \mathsf{q} \neq \mathsf{q}'$$

$$s : (\mathsf{q}, \Pi, z) \cdot h \equiv s : (\mathsf{q}, \Pi', z) \cdot (\mathsf{q}, \Pi'', z) \cdot h \quad \text{where } \Pi = \Pi' \cup \Pi'' \text{ and } \Pi' \cap \Pi'' = \emptyset$$

Table IV. Structural equivalence (r ranges over a, s and z ranges over v, $s[\mathsf{p}]$ and l.)

$(\nu s^a)(s^a[3] \triangleright (2, \{\mathsf{ok} : s^a[3]?(2, \mathit{address}); s^a[3]!\langle\{2\}, \mathsf{date}\rangle; \mathbf{0},\ \mathsf{quit} : \mathbf{0}\}) \mid$
$b[1](z_1).z_1!\langle\{2\}, \mathsf{quote\ div}\ 2 - 99\rangle; z_2!\langle\langle 2, s^a[2]\rangle\rangle; \ldots) \mid$
$\overline{b}[2](z_2).z_2?(1, x); z_2?((1, t)); z_2 \triangleleft (\{1\}, \mathsf{ok}); t \triangleleft (\{1, 3\}, \mathsf{ok}); t!\langle\{3\}, \ldots\rangle; t?(3, \mathit{date})$

$\longrightarrow$ by using [Link] (and the structural congruence for scope extrusion)

$(\nu s^a s^b)(\ldots \text{as above} \ldots \mid s^b[1]!\langle\{2\}, \mathsf{quote\ div}\ 2 - 99\rangle; s^b[1]!\langle\langle 2, s^a[2]\rangle\rangle; \ldots \mid$
$s^b[2]?(1, x); s^b[2]?((1, t)); s^b[2] \triangleleft (\{1\}, \mathsf{ok}); t \triangleleft (\{1, 3\}, \mathsf{ok}); t!\langle\{3\}, \ldots\rangle; t?(3, \mathit{date}))$

$\longrightarrow^*$ by using [Send] and [Recv] the result of $\mathsf{quote\ div}\ 2 - 99$ is communicated

$(\nu s^a s^b)(\ldots \text{as above} \ldots \mid s^b[1]!\langle\langle 2, s^a[2]\rangle\rangle; s^b[1] \triangleright (2, \{\mathsf{ok} : \mathbf{0},\ \mathsf{quit} : \mathbf{0}\}) \mid$
$s^b[2]?((1, t)); s^b[2] \triangleleft (\{1\}, \mathsf{ok}); t \triangleleft (\{1, 3\}, \mathsf{ok}); t!\langle\{3\}, \ldots\rangle; t?(3, \mathit{date}))$

$\longrightarrow^*$ by using [Deleg] and [Srec]

$(\nu s^a s^b)(\ldots \text{as above} \ldots \mid s^b[1] \triangleright (2, \{\mathsf{ok} : \mathbf{0},\ \mathsf{quit} : \mathbf{0}\}) \mid$
$s^b[2] \triangleleft (\{1\}, \mathsf{ok}); s^a[2] \triangleleft (\{1, 3\}, \mathsf{ok}); s^a[2]!\langle\{1\}, \ldots\rangle; s^a[2]?(3, \mathit{date}))$

$\longrightarrow^*$ by using [Label] and [Branch]

$(\nu s^a s^b)(s^a[3] \triangleright (2, \{\mathsf{ok} : s^a[3]?(2, \mathit{address}); s^a[3]!\langle\{2\}, \mathsf{date}\rangle; \mathbf{0},\ \mathsf{quit} : \mathbf{0}\}) \ \mid\ \mathbf{0}\ \mid$
$s^a[2] \triangleleft (\{1, 3\}, \mathsf{ok}); s^a[2]!\langle\{3\}, \ldots\rangle; s^a[2]?(3, \mathit{date}))$

Table V. Example of reduction

4. COMMUNICATION TYPE SYSTEM

The previous section defines the syntax and the global types. This section introduces the communication type system, by which we can check type soundness of the communications.

4.1 Types and Typing Rules for Pure Runtime Processes

We first define the local types of pure processes, called *action types*. While global types represent the whole protocol, action types correspond to the communication actions, representing sessions from the view-points of single participants.

$$\begin{array}{llllll}
\text{Action} & T ::= & !\langle\Pi,U\rangle;T & \textit{send} & \mid\ \mu\mathbf{t}.T & \textit{recursive}\\
 & \mid & ?(\mathrm{p},U);T & \textit{receive} & \mid\ \mathbf{t} & \textit{variable}\\
 & \mid & \oplus\langle\Pi,\{l_i:T_i\}_{i\in I}\rangle & \textit{selection} & \mid\ \mathsf{end} & \textit{end}\\
 & \mid & \&(\mathrm{p},\{l_i:T_i\}_{i\in I}) & \textit{branching} & &
\end{array}$$

The *send type* $!\langle\Pi,U\rangle;T$ expresses the sending to all p_k for $\mathrm{p}_k\in\Pi$ of a value or of a channel of type U, followed by the communications of T. The *selection type* $\oplus\langle\Pi,\{l_i:T_i\}_{i\in I}\rangle$ represents the transmission to all p_k for $\mathrm{p}_k\in\Pi$ of a label l_i chosen in the set $\{l_i \mid i\in I\}$ followed by the communications described by T_i. The *receive* and *branching* are dual and only need one sender. Other types are standard.

The relation between action and global types is formalised by the notion of projection as in [Honda et al. 2008]. The *projection of G onto* q ($G\restriction\mathrm{q}$) is defined by induction on G:

$$(\mathrm{p}\to\Pi:\langle U\rangle.G')\restriction\mathrm{q}=\begin{cases}!\langle\Pi,U\rangle;(G'\restriction\mathrm{q}) & \text{if } \mathrm{q}=\mathrm{p},\\ ?(\mathrm{p},U);(G'\restriction\mathrm{q}) & \text{if } \mathrm{q}=\mathrm{p}_k \text{ for some } \mathrm{p}_k\in\Pi,\\ G'\restriction\mathrm{q} & \text{otherwise.}\end{cases}$$

$$(\mathrm{p}\to\Pi:\{l_i:G_i\}_{i\in I})\restriction\mathrm{q}=$$

$$\begin{cases}\oplus(\Pi,\{l_i:G_i\restriction\mathrm{q}\}_{i\in I}) & \text{if } \mathrm{q}=\mathrm{p}\\ \&(\mathrm{p},\{l_i:G_i\restriction\mathrm{q}\}_{i\in I}) & \text{if } \mathrm{q}=\mathrm{p}_k \text{ for some } \mathrm{p}_k\in\Pi\\ G_1\restriction\mathrm{q} & \text{if } \mathrm{q}\neq\mathrm{p},\mathrm{q}\neq\mathrm{p}_k\forall\mathrm{p}_k\in\Pi \text{ and}\\ & G_i\restriction\mathrm{q}=G_j\restriction\mathrm{q} \text{ for all } i,j\in I.\end{cases}$$

$$(\mu\mathbf{t}.G)\restriction\mathrm{q}=\mu\mathbf{t}.(G\restriction\mathrm{q})\quad \mathbf{t}\restriction\mathrm{q}=\mathbf{t}\quad \mathsf{end}\restriction\mathrm{q}=\mathsf{end}.$$

As an example, we list two of the projections of the global types G_a and G_b of the three-buyer protocol:

$$\begin{aligned}G_a\restriction 3 &= ?\langle 1,\mathsf{string}\rangle;!\langle\{1,2\},\mathsf{int}\rangle;\&(2,\{\mathsf{ok}:?\langle 2,\mathsf{string}\rangle;!\langle\{2\},\mathsf{date}\rangle;\mathsf{end},\mathsf{quit}:\mathsf{end}\})\\ G_b\restriction 1 &= ?\langle 2,\mathsf{int}\rangle;?\langle 2,T\rangle;\oplus\langle\{2\},\{\mathsf{ok}:\mathsf{end},\mathsf{quit}:\mathsf{end}\}\rangle\end{aligned}$$

where $T=\oplus\langle\{1,3\},\{\mathsf{ok}:!\langle\{3\},\mathsf{string}\rangle;?\langle 3,\mathsf{date}\rangle;\mathsf{end},\ \mathsf{quit}:\mathsf{end}\}\rangle$.

The typing judgements for expressions and pure processes are of the shape:

$$\Gamma\vdash e:S \quad\text{and}\quad \Gamma\vdash P\triangleright\Delta$$

where Γ is the *standard environment* which associates variables to sort types, service names to global types and process variables to pairs of sort types and action types; Δ is the *session environment* which associates channels to action types. Formally we define:

$$\Gamma::=\emptyset\mid\Gamma,u:S\mid\Gamma,X:S\,T \quad\text{and}\quad \Delta::=\emptyset\mid\Delta,c:T$$

assuming that we can write $\Gamma,u:S$ only if u does not occur in Γ, briefly $u\notin dom(\Gamma)$ ($dom(\Gamma)$ denotes the domain of Γ, i.e., the set of identifiers which occur in Γ). We use the same convention for $X:S\,T$ and Δ (thus we can write Δ,Δ' only if $dom(\Delta)\cap dom(\Delta')=\emptyset$).

Table VI presents the typing rules for pure processes. Rule $\lfloor$MCAST$\rfloor$ permits to type a service initiator identified by u, if the type of y is the n-th projection of the global type G of u and the number of participants in G (denoted by $\mathrm{pn}(G)$) is n. Rule $\lfloor$MACC$\rfloor$ permits to type the p-th participant identified by u, which uses the channel y, if the type of y is the p-th projection of the global type G of u. The successive six rules associate the

$$\Gamma, u : S \vdash u : S \ \lfloor\textsc{Name}\rfloor \qquad \Gamma \vdash \mathsf{true}, \mathsf{false} : \mathsf{bool} \qquad \dfrac{\Gamma \vdash e_i : \mathsf{bool}}{\Gamma \vdash e_1 \ \mathsf{and}\ e_2 : \mathsf{bool}} \ \lfloor\textsc{Bool}\rfloor, \lfloor\textsc{And}\rfloor$$

$$\dfrac{\Gamma \vdash u : \langle G\rangle \quad \Gamma \vdash P \triangleright \Delta, y : G \upharpoonright n \quad \mathsf{pn}(G) \leq n}{\Gamma \vdash \overline{u}[n](y).P \triangleright \Delta} \ \lfloor\textsc{MCast}\rfloor \qquad \dfrac{\Gamma \vdash u : \langle G\rangle \quad \Gamma \vdash P \triangleright \Delta, y : G \upharpoonright \mathtt{p}}{\Gamma \vdash u[\mathtt{p}](y).P \triangleright \Delta} \ \lfloor\textsc{MAcc}\rfloor$$

$$\dfrac{\Gamma \vdash e : S \quad \Gamma \vdash P \triangleright \Delta, c : T}{\Gamma \vdash c!\langle \Pi, e\rangle; P \triangleright \Delta, c : !\langle \Pi, S\rangle; T} \ \lfloor\textsc{Send}\rfloor \qquad \dfrac{\Gamma, x : S \vdash P \triangleright \Delta, c : T}{\Gamma \vdash c?(\mathtt{q}, x); P \triangleright \Delta, c : ?(\mathtt{q}, S); T} \ \lfloor\textsc{Rcv}\rfloor$$

$$\dfrac{\Gamma \vdash P \triangleright \Delta, c : T}{\Gamma \vdash c!\langle\langle \mathtt{p}, c'\rangle\rangle; P \triangleright \Delta, c : !\langle \mathtt{p}, T'\rangle; T, c' : T'} \ \lfloor\textsc{Deleg}\rfloor \qquad \dfrac{\Gamma \vdash P \triangleright \Delta, c : T, y : T'}{\Gamma \vdash c?((\mathtt{q}, y)); P \triangleright \Delta, c : ?(\mathtt{q}, T'); T} \ \lfloor\textsc{SRec}\rfloor$$

$$\dfrac{\Gamma \vdash P \triangleright \Delta, c : T_j \quad j \in I}{\Gamma \vdash c \oplus \langle \Pi, l_j\rangle; P \triangleright \Delta, c : \oplus\langle \Pi, \{l_i : T_i\}_{i\in I}\rangle} \ \lfloor\textsc{Sel}\rfloor \ \dfrac{\Gamma \vdash P_i \triangleright \Delta, c : T_i \quad \forall i \in I}{\Gamma \vdash c \& (\mathtt{p}, \{l_i : P_i\}_{i\in I}) \triangleright \Delta, c : \&(\mathtt{p}, \{l_i : T_i\}_{i\in I})} \ \lfloor\textsc{Branch}\rfloor$$

$$\dfrac{\Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash Q \triangleright \Delta'}{\Gamma \vdash P \mid Q \triangleright \Delta, \Delta'} \ \lfloor\textsc{Conc}\rfloor$$

$$\dfrac{\Gamma \vdash e : \mathsf{bool} \quad \Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash Q \triangleright \Delta}{\Gamma \vdash \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \triangleright \Delta} \ \lfloor\textsc{If}\rfloor \qquad \dfrac{\Delta\ \mathsf{end\ only}}{\Gamma \vdash \mathbf{0} \triangleright \Delta} \ \lfloor\textsc{Inact}\rfloor \qquad \dfrac{\Gamma, a : \langle G\rangle \vdash P \triangleright \Delta}{\Gamma \vdash (\nu a)P \triangleright \Delta} \ \lfloor\textsc{NRes}\rfloor$$

$$\dfrac{\Gamma \vdash e : S \quad \Delta\ \mathsf{end\ only}}{\Gamma, X : S\,T \vdash X\langle e, c\rangle \triangleright \Delta, c : T} \ \lfloor\textsc{Var}\rfloor \qquad \dfrac{\Gamma, X : S\,T, x : S \vdash P \triangleright y : T \qquad \Gamma, X : S\,T \vdash Q \triangleright \Delta}{\Gamma \vdash \mathsf{def}\ X(x, y) = P\ \mathsf{in}\ Q \triangleright \Delta} \ \lfloor\textsc{Def}\rfloor$$

Table VI. Typing rules for pure processes

input/output processes to the input/output types in the expected way. Note that, according to our notational convention on environments, in rule $\lfloor\textsc{Deleg}\rfloor$ the channel which is sent cannot appear in the session environment of the premise, i.e., $c' \notin dom(\Delta) \cup \{c\}$. Rule $\lfloor\textsc{Conc}\rfloor$ permits to put in parallel two processes only if their sessions environments have disjoint domains. For example we can derive:

$$\vdash t \oplus \langle \{1, 3\}, \mathsf{ok}\rangle; t!\langle \{3\}, \texttt{"Address"}\rangle; t?(3, \mathit{date}); \mathbf{0} \triangleright \{t : T\}$$

where $T = \oplus\langle \{1, 3\}, \{\mathsf{ok} : !(\{3\}, \mathsf{string}); ?\langle 3, \mathsf{date}\rangle; \mathsf{end},\ \mathsf{quit} : \mathsf{end}\}\rangle$.

In the typing of the example of the three-buyer protocol the types of the channels y_3 and z_1 are the third projection of G_a and the first projection of G_b, respectively. By applying rule $\lfloor\textsc{MCast}\rfloor$ we can then derive $a : G_a \vdash \mathrm{S} \triangleright \emptyset$. Similarly by applying rule $\lfloor\textsc{MAcc}\rfloor$ we can derive $b : G_b \vdash \mathrm{C} \triangleright \emptyset$.

4.2 Types and Typing Rules for Runtime Processes

We now extend the communication type system to processes containing queues.

— $\natural\langle\emptyset,Z\rangle;T \approx T$
— $\natural\langle\Pi,Z\rangle;\ \natural'\langle\Pi',Z\rangle;T \approx \natural'\langle\Pi',Z\rangle;\ \natural\langle\Pi,Z\rangle;T \quad \text{if } \Pi\cap\Pi' = \emptyset$
— $\natural\langle\Pi,Z\rangle;T \approx \natural\langle\Pi',Z\rangle;\ \natural\langle\Pi'',Z\rangle;T \quad \text{if } \Pi = \Pi'\cup\Pi'', \Pi'\cap\Pi'' = \emptyset$

Table VII. Equivalence relation rules on message types.

$$\begin{array}{llcll}
\text{Message} & \mathtt{T} & ::= & !\langle\Pi,U\rangle & \textit{message send}\\
 & & \mid & \oplus\langle\Pi,l\rangle & \textit{message selection}\\
 & & \mid & \mathtt{T};\mathtt{T}' & \textit{message sequence}\\
 & & & & \\
\text{Generalised} & \mathsf{T} & ::= & T & \textit{action}\\
 & & \mid & \mathtt{T} & \textit{message}\\
 & & \mid & \mathtt{T};T & \textit{continuation}
\end{array}$$

Message types are the types for queues: they represent the messages contained in the queues. The *message send type* $!\langle\Pi,U\rangle$ expresses the communication to all p_k for $\mathrm{p}_k\in\Pi$ of a value or of a channel of type U. The *message selection type* $\oplus\langle\Pi,l\rangle$ represents the communication to all p_k for $\mathrm{p}_k\in\Pi$ of the label l and $\mathtt{T};\mathtt{T}'$ represents sequencing of message types (we assume associativity for ;). For example $\oplus\langle\{1,3\},\mathsf{ok}\rangle$ is the message type for the message $(2,\{1,3\},\mathsf{ok})$. A *generalised type* is either an action type, or a message type, or a message type followed by an action type. Type $\mathtt{T};T$ represents the continuation of the type $\mathtt{T}$ associated to a queue with the type T associated to a pure process. An example of generalised type is $\oplus\langle\{1,3\},\mathsf{ok}\rangle;!\langle\{3\},\mathsf{string}\rangle;?\langle 3,\mathsf{date}\rangle;\mathsf{end}$.

In order to take into account the structural congruence between queues (see Table IV) we consider message types modulo the equivalence relation $\approx$ induced by the rules shown in Table VII (with $\natural\in\{!,\oplus\}$ and $Z\in\{U,l\}$).

We start by defining the typing rules for single queues, in which the turnstile $\vdash$ is decorated with $\{s\}$ (where s is the session name of the current queue) and the session environments are mappings from channels to message types. The empty queue has empty session environment. Each message adds an output type to the current type of the channel which has the role of the message sender. Table VIII lists the typing rules for queues, where ; is defined by:

$$\Delta;\{s[\mathrm{q}]:\mathtt{T}\} = \begin{cases}\Delta',s[\mathrm{q}]:\mathtt{T}';\mathtt{T} & \text{if } \Delta=\Delta',s[\mathrm{q}]:\mathtt{T}',\\ \Delta,s[\mathrm{q}]:\mathtt{T} & \text{otherwise.}\end{cases}$$

For example we can derive $\vdash_{\{s\}} s:(\mathsf{ok},\{1,2\},3)\triangleright\{s[3]:\oplus\langle\{1,2\},\mathsf{ok}\rangle\}$.

In order to type pure processes in parallel with queues, we need to use generalised types in session environments and further typing rules. Table IX lists the typing rules for processes containing queues. The judgement $\Gamma\vdash_\Sigma P\triangleright\Delta$ means that P contains the queues whose session names are in Σ. Rule $\lfloor\text{GINIT}\rfloor$ promotes the typing of a pure process to the typing of an arbitrary process, since a pure process does not contain queues. When two arbitrary processes are put in parallel (rule $\lfloor\text{GPAR}\rfloor$) we need to require that each session name is associated to at most one queue (condition $\Sigma\cap\Sigma'=\emptyset$). In composing the two session environments we want to put in sequence a message type and an action type for the same channel with role. For this reason we define the composition $*$ between local types as:

$$\frac{}{\Gamma \vdash_{\{s\}} s : \varnothing \triangleright \emptyset}\ \lfloor\text{QINIT}\rfloor \qquad \frac{\Gamma \vdash_{\{s\}} s : h \triangleright \Delta \qquad \Gamma \vdash v : S}{\Gamma \vdash_{\{s\}} s : h \cdot (\mathsf{q}, \Pi, v) \triangleright \Delta; \{s[\mathsf{q}] : !\langle \Pi, S\rangle\}}\ \lfloor\text{QSEND}\rfloor$$

$$\frac{\Gamma \vdash_{\{s\}} s : h \triangleright \Delta}{\Gamma \vdash_{\{s\}} s : h \cdot (\mathsf{q}, \mathsf{p}, s'[\mathsf{p}']) \triangleright \Delta, s'[\mathsf{p}'] : T'; \{s[\mathsf{q}] : !\langle \mathsf{p}, T'\rangle\}}\ \lfloor\text{QDELEG}\rfloor$$

$$\frac{\Gamma \vdash_{\{s\}} s : h \triangleright \Delta}{\Gamma \vdash_{\{s\}} s : h \cdot (\mathsf{q}, \Pi, l) \triangleright \Delta; \{s[\mathsf{q}] : \oplus\langle \Pi, l\rangle\}}\ \lfloor\text{QSEL}\rfloor$$

Table VIII. Typing rules for queues

$$\frac{\Gamma \vdash P \triangleright \Delta}{\Gamma \vdash_{\emptyset} P \triangleright \Delta}\ \lfloor\text{GINIT}\rfloor \qquad \frac{\Gamma \vdash_{\Sigma} P \triangleright \Delta \quad \Delta' \text{end only}}{\Gamma \vdash_{\Sigma} P \triangleright \Delta * \Delta'}\ \lfloor\text{WEAK}\rfloor$$

$$\frac{\Gamma \vdash_{\Sigma} P \triangleright \Delta \quad \Gamma \vdash_{\Sigma'} Q \triangleright \Delta' \quad \Sigma \cap \Sigma' = \emptyset}{\Gamma \vdash_{\Sigma \cup \Sigma'} P \mid Q \triangleright \Delta * \Delta'}\ \lfloor\text{GPAR}\rfloor$$

$$\frac{\Gamma \vdash_{\Sigma} P \triangleright \Delta \quad \mathsf{co}(\Delta, s)}{\Gamma \vdash_{\Sigma \setminus s} (\nu s) P \triangleright \Delta \setminus s}\ \lfloor\text{GSRES}\rfloor \qquad \frac{\Gamma, a : \langle G\rangle \vdash_{\Sigma} P \triangleright \Delta}{\Gamma \vdash_{\Sigma} (\nu a) P \triangleright \Delta}\ \lfloor\text{GNRES}\rfloor$$

$$\frac{\Gamma, X : S\,T, x : S \vdash P \triangleright \{y : T\} \qquad \Gamma, X : S\,T \vdash_{\Sigma} Q \triangleright \Delta}{\Gamma \vdash_{\Sigma} \mathsf{def}\ X(x, y) = P\ \mathsf{in}\ Q \triangleright \Delta}\ \lfloor\text{GDEF}\rfloor$$

Table IX. Typing rules for processes

$$\mathsf{T} * \mathsf{T}' = \begin{cases} \mathsf{T}; \mathsf{T}' & \text{if } \mathsf{T} \text{ is a message type,} \\ \mathsf{T}'; \mathsf{T} & \text{if } \mathsf{T}' \text{ is a message type,} \\ \bot & \text{otherwise} \end{cases}$$

where $\bot$ represents failure of typing.

We extend $*$ to session environments as expected:

$$\Delta * \Delta' = \Delta \setminus dom(\Delta') \cup \Delta' \setminus dom(\Delta) \cup \{c : \mathsf{T} * \mathsf{T}' \mid c : \mathsf{T} \in \Delta \ \&\ c : \mathsf{T}' \in \Delta'\}.$$

Note that $*$ is commutative, i.e., $\Delta * \Delta' = \Delta' * \Delta$. Also if we can derive message types only for channels with roles, we consider the channel variables in the definition of $*$ for session environments since we want to get for example $\{y : \mathsf{end}\} * \{y : \mathsf{end}\} = \bot$. An example of derivable judgement is:

$\vdash_{\{s\}} P \mid s : (3, \{1,2\}, \mathsf{ok}) \triangleright \{s[3] : \oplus\langle \{1,2\}, \mathsf{ok}\rangle; !\langle \{1\}, \mathsf{string}\rangle; ?\langle 1, \mathsf{date}\rangle; \mathsf{end}\}$
where $P = s[3]!\langle \{1\}, \texttt{"Address"}\rangle; s[3]?(1, date); \mathbf{0}$.

4.3 More on Communication Type System

Definition 4.1. The *projection of the generalised local type* T *onto* q, denoted by $\mathsf{T} \upharpoonright \mathsf{q}$, is defined by:

$$(\ !\langle \Pi, U\rangle; T') \upharpoonright \mathsf{q} = \begin{cases} !U; T' \upharpoonright \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}_k \text{ for some } \mathsf{p}_k \in \Pi, \\ T' \upharpoonright \mathsf{q} & \text{otherwise.} \end{cases}$$

$$(?(\Pi,U);T') \upharpoonright \mathsf{q} = \begin{cases} ?U;T' \upharpoonright \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}_k \text{ for some } \mathsf{p}_k \in \Pi, \\ T' \upharpoonright \mathsf{q} & \text{otherwise.} \end{cases}$$

$$(\oplus\langle\Pi,\{l_i : T_i\}_{i\in I}\rangle) \upharpoonright \mathsf{q} = \begin{cases} \oplus\{l_i : T_i \upharpoonright \mathsf{q}\}_{i\in I} & \text{if } \mathsf{q} = \mathsf{p}_k \text{ for some } \mathsf{p}_k \in \Pi, \\ T_1 \upharpoonright \mathsf{q} & \text{if } \mathsf{q} \neq \mathsf{p}_k\ \forall \mathsf{p}_k \in \Pi \text{ and} \\ & T_i \upharpoonright \mathsf{q} = T_j \upharpoonright \mathsf{q} \\ & \text{for all } i,j \in I. \end{cases}$$

$$(\&(\mathsf{p},\{l_i : T_i\}_{i\in I})) \upharpoonright \mathsf{q} = \begin{cases} \&\{l_i : T_i \upharpoonright \mathsf{q}\}_{i\in I} & \text{if } \mathsf{q} = \mathsf{p}, \\ T_1 \upharpoonright \mathsf{q} & \text{if } \mathsf{q} \neq \mathsf{p} \\ & \forall \mathsf{p}_k \in \Pi \text{ and} \\ & T_i \upharpoonright \mathsf{q} = T_j \upharpoonright \mathsf{q} \\ & \text{for all } i,j \in I. \end{cases}$$

$$(\oplus\langle\Pi,l\rangle;\mathsf{T}) \upharpoonright \mathsf{q} = \begin{cases} \oplus l;\mathsf{T} \upharpoonright \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}_k \text{ for some } \mathsf{p}_k \in \Pi, \\ \mathsf{T} \upharpoonright \mathsf{q} & \text{otherwise.} \end{cases}$$

$$(\mu\mathbf{t}.T) \upharpoonright \mathsf{q} = \mu\mathbf{t}.(T \upharpoonright \mathsf{q}) \quad \mathbf{t} \upharpoonright \mathsf{q} = \mathbf{t} \quad \mathsf{end} \upharpoonright \mathsf{q} = \mathsf{end}$$

Definition 4.2. The *duality relation* between projections of generalised local types is the minimal symmetric relation which satisfies:

$$\mathsf{end} \bowtie \mathsf{end} \qquad \mathbf{t} \bowtie \mathbf{t} \qquad T \bowtie T' \implies \mu\mathbf{t}.T \bowtie \mu\mathbf{t}.T' \qquad !U;T \bowtie ?U;T'$$
$$\forall i \in I\ T_i \bowtie T_i' \implies \oplus\{l_i : T_i\}_{i\in I} \bowtie \&\{l_i : T_i'\}_{i\in I}$$
$$\exists i \in I\ l = l_i\ \&\ \mathsf{T} \bowtie \mathsf{T}_i \implies \oplus l;\mathsf{T} \bowtie \&\{l_i : \mathsf{T}_i\}_{i\in I}$$

Definition 4.3. A session environment Δ is *coherent for the session* s (notation $\mathsf{co}(\Delta,s)$) if $s[\mathsf{p}] : T \in \Delta$ and $T \upharpoonright \mathsf{q} \neq \mathsf{end}$ imply $s[\mathsf{q}] : T' \in \Delta$ and $T \upharpoonright \mathsf{q} \bowtie T' \upharpoonright \mathsf{p}$. A session environment Δ is *coherent* if it is coherent for all sessions which occur in it.

4.4 Subject Reduction

Since session environments represent the forthcoming communications, by reducing processes session environments can change. This can be formalised as in [Honda et al. 2008] by introducing the notion of reduction of session environments, whose rules are:

—$\{s[\mathsf{p}] : !\langle\{j\},U\rangle;\ T, s[\mathsf{p}_j] : ?(\mathsf{p},U);T'\} \Rightarrow \{s[\mathsf{p}] : T, s[\mathsf{p}_j] : T'\}$
—$\{s[\mathsf{p}] : T;\oplus\langle\Pi,\{l_i : T_i\}_{i\in I}\rangle\} \Rightarrow \{s[\mathsf{p}] : T;\oplus\langle\Pi,l_i\rangle;T_i\}$
—$\{s[\mathsf{p}] : \oplus\langle\{j\},l\rangle;T, s[\mathsf{p}_j] : \&(\mathsf{p},\{l_i : T_i\}_{i\in I})\} \Rightarrow \{s[\mathsf{p}] : T, s[\mathsf{p}_j] : T_i\}$ if $l = l_i$
—$\{s[\mathsf{p}] : !\langle\emptyset,U\rangle;\ T\} \Rightarrow \{s[\mathsf{p}] : T\}$ $\qquad \{s[\mathsf{p}] : \oplus\langle\emptyset,l\rangle;T\} \Rightarrow \{s[\mathsf{p}] : T\}$
—$\Delta \cup \Delta'' \Rightarrow \Delta' \cup \Delta''$ if $\Delta \Rightarrow \Delta'$.

The first rule corresponds to the reception of a value or channel by the participant p_j, the second rule corresponds to the choice of the label l_i and the third rule corresponds to the reception of the label l by the participant p_j. The fourth and the fifth rules garbage collect read messages.

Using the above notion we can state type preservation under reduction as follows:

THEOREM 4.4 TYPE PRESERVATION. *If* $\Gamma \vdash_\Sigma P \triangleright \Delta$ *and* $P \longrightarrow^* P'$, *then* $\Gamma \vdash_\Sigma P' \triangleright \Delta'$ *for some* Δ' *such that* $\Delta \Rightarrow^* \Delta'$.

Note that the communication safety [Honda et al. 2008, Theorem 5.5] is a corollary of this theorem. Thus the user-defined processes with the global types can safely communicate since their runtime translation is typable by the communication type system.

4.5 From User Syntax to Runtime Syntax via Types

Given a user process P and the set of global types associated to the service identifiers which occur free or bound in P we can add the sender and the receivers to each communication, by getting in this way a process in the runtime syntax. We define two mappings with domain the set of user processes: the first one (denote by $\lfloor G \dagger u \rfloor$) depends on a global type G and on a service identifier u, while the second one (denote by $\lfloor T \ddagger y \rfloor$) depends on an action type T and on a channel variable y. The mapping $\lfloor G \dagger u \rfloor$ (Table X) calls the other mapping with the appropriate projection and channel variable when it is applied to a session initiation on the identifier u, and leaves the process unchanged otherwise. The mapping $\lfloor T \ddagger y \rfloor$ (Table XI) adds the sender or the receiver to the communications which use the channel y and it does not affect the other processes. An interesting clause is the fifth one, in which $\lfloor T' \ddagger y' \rfloor$ is applied to the body of the channel reception y' (T' is the action type of y'). In the last but one clause T' is the unique type such that $\lfloor T' \ddagger y \rfloor(X(e\,y))$ occurs in (the evaluation of) $\lfloor T \ddagger y \rfloor(Q)$. More precisely we evaluate this type by applying to Q the mapping $\lfloor T \natural y \natural X \rfloor()$ defined in Table XII.

In order to get the runtime version of an user process P we need to apply to P the mapping $\lfloor G \dagger u \rfloor$, for each service identifier u which occurs free or bound in P, where G is the global type of u. Note that when u is a bound variable we need to apply $\lfloor G \dagger x \rfloor$ only to the scope of x.

We say that a closed user process $P = \mathscr{C}[y_1?(x_1);Q_1]\dots[y_m?(x_m);Q_m]$ with bound service identifiers $x_1,\dots,x_m$ and service names a_ℓ with $\ell \in L$ is a correct implementation of the protocols described by $G_1,\dots,G_m$ and G'_ℓ for $\ell \in L$ if we can derive

$$\lfloor G'_\ell \dagger a_\ell \rfloor_{\ell \in L}(\mathscr{C}[y_1?(x_1);\lfloor G_1 \dagger x_1 \rfloor(Q_1)]\dots[y_m?(x_m);\lfloor G_m \dagger x_m \rfloor(Q_m)]) \triangleright \emptyset$$

from $\{a_\ell : G'_\ell \mid \ell \in L\}$.

$\lfloor G \dagger u \rfloor(\overline{u}[n](y).P) = \overline{u}[n](y).\lfloor G \upharpoonright 1 \ddagger y \rfloor(P)$	
$\lfloor G \dagger u \rfloor(u[\mathtt{p}](y).P) = u[\mathtt{p}](y).\lfloor G \upharpoonright \mathtt{p} \ddagger y \rfloor(P)$	
$\lfloor G \dagger u \rfloor(\mathsf{pref};P) = \mathsf{pref};\lfloor G \dagger u \rfloor(P)$	$u \notin \mathsf{pref}$
$\lfloor G \dagger u \rfloor(\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q) = \mathsf{if}\ e\ \mathsf{then}\ \lfloor G \dagger u \rfloor(P)\ \mathsf{else}\ \lfloor G \dagger u \rfloor(Q)$	
$\lfloor G \dagger u \rfloor(P \mid Q) = \lfloor G \dagger u \rfloor(P) \mid \lfloor G \dagger u \rfloor(Q)$	
$\lfloor G \dagger u \rfloor(\mathbf{0}) = \mathbf{0}$	
$\lfloor G \dagger u \rfloor((\nu a)P) = (\nu a)\lfloor G \dagger u \rfloor(P)$	
$\lfloor G \dagger u \rfloor(\mathsf{def}\ X(x\,y) = P\ \mathsf{in}\ Q) = \mathsf{def}\ X(x\,y) = \lfloor G \dagger u \rfloor(P)\ \mathsf{in}\ \lfloor G \dagger u \rfloor(Q)$	
$\lfloor G \dagger u \rfloor(X\langle e\,y \rangle) = X\langle e\,y \rangle$	

where pref is any session initialization or communication command.

Table X. Application of a global type and a service identifier to a user process.

$$\begin{array}{ll}
\lfloor\, !\langle \Pi, S\rangle; T \ddagger y\rfloor(y!\langle e\rangle; P) = y!\langle \Pi, e\rangle; \lfloor T \ddagger y\rfloor(P) & \\
\lfloor ?(\mathsf{p}, S); T \ddagger y\rfloor(y?(x); P) = y?(\mathsf{p}, x); \lfloor T \ddagger y\rfloor(P) & \\
\lfloor\, !\langle \Pi, T'\rangle; T \ddagger y\rfloor(y!\langle\langle y'\rangle\rangle; P) = y!\langle\langle \Pi, y'\rangle\rangle; \lfloor T \ddagger y\rfloor(P) & \\
\lfloor T \ddagger y\rfloor(y'!\langle\langle y\rangle\rangle; P) = y'!\langle\langle y\rangle\rangle; P & \\
\lfloor ?(\mathsf{p}, T'); T \ddagger y\rfloor(y?((y')); P) = y?((\mathsf{p}, y')); \lfloor T \ddagger y\rfloor(\lfloor T' \ddagger y'\rfloor(P)) & \\
\lfloor \oplus\langle \Pi, \{l_i : T_i\}_{i\in I}\rangle \ddagger y\rfloor(y \oplus l_j; P) = y \oplus \langle \mathsf{p}, l_j\rangle; \lfloor T_j \ddagger y\rfloor(P) & j \in I \\
\lfloor \&(\mathsf{p}, \{l_i : T_i\}_{i\in I}) \ddagger y\rfloor(y\&\{l_i : P_i\}_{i\in I}) = y\&(\mathsf{p}, \{l_i : \lfloor T \ddagger y\rfloor(P_i)\}_{i\in I}) & \\
\lfloor T \ddagger y\rfloor(\mathsf{pref}; P) = \mathsf{pref}; \lfloor T \ddagger y\rfloor(P) & y \notin \mathsf{pref} \\
\lfloor T \ddagger y\rfloor(\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q) = \mathsf{if}\ e\ \mathsf{then}\ \lfloor T \ddagger y\rfloor(P)\ \mathsf{else}\ \lfloor T \ddagger y\rfloor(Q) & \\
\lfloor T \ddagger y\rfloor(P \mid Q) = \lfloor T \ddagger y\rfloor(P) \mid Q & y \notin Q \\
\lfloor T \ddagger y\rfloor(P \mid Q) = P \mid \lfloor T \ddagger y\rfloor(Q) & y \notin P \\
\lfloor \mathsf{end} \ddagger y\rfloor(\mathbf{0}) = \mathbf{0} & \\
\lfloor T \ddagger y\rfloor((\nu a)P) = (\nu a)\lfloor T \ddagger y\rfloor(P) & \\
\lfloor T \ddagger y\rfloor(\mathsf{def}\ X(x\,y') = P\ \mathsf{in}\ Q) = \mathsf{def}\ X(x\,y') = \lfloor T' \ddagger y'\rfloor(P)\ \mathsf{in}\ \lfloor T \ddagger y\rfloor(Q) & \\
\quad \text{where } T' = \lfloor T \natural y \natural X\rfloor(Q) & \\
\lfloor T \ddagger y\rfloor(X\langle e\,y'\rangle) = X\langle e\,y'\rangle &
\end{array}$$

Table XI. Application of a local type and a channel variable to a user process.

$$\begin{array}{ll}
\lfloor\, !\langle \Pi, S\rangle; T \natural y \natural X\rfloor(y!\langle e\rangle; P) = \lfloor T \natural y \natural X\rfloor(P) & \\
\lfloor ?(\mathsf{p}, S); T \natural y \natural X\rfloor(y?(x); P) = \lfloor T \natural y \natural X\rfloor(P) & \\
\lfloor\, !\langle \Pi, T'\rangle; T \natural y \natural X\rfloor(y!\langle\langle y'\rangle\rangle; P) = \lfloor T \natural y \natural X\rfloor(P) & \\
\lfloor ?(\mathsf{p}, T'); T \natural y \natural X\rfloor(y?((y')); P) = \lfloor T \natural y \natural X\rfloor(P) & \\
\lfloor \oplus\langle \Pi, \{l_i : T_i\}_{i\in I}\rangle \natural y \natural X\rfloor(y \oplus l_j; P) = \lfloor T_j \natural y \natural X\rfloor(P) & j \in I \\
\lfloor \&(\mathsf{p}, \{l_i : T_i\}_{i\in I}) \natural y \natural X\rfloor(y\&\{l_i : P_i\}_{i\in I}) = \lfloor T_j \natural y \natural X\rfloor(P_j) & j \in I \;\&\; X \in P_j \\
\lfloor T \natural y \natural X\rfloor(\mathsf{pref}; P) = \lfloor T \natural y \natural X\rfloor(P) & y \notin \mathsf{pref} \\
\lfloor T \natural y \natural X\rfloor(\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q) = \lfloor T \natural y \natural X\rfloor(P) & X \in P \\
\lfloor T \natural y \natural X\rfloor(\mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q) = \lfloor T \natural y \natural X\rfloor(Q) & X \in Q \\
\lfloor T \natural y \natural X\rfloor(P \mid Q) = \lfloor T \natural y \natural X\rfloor(P) & X \in P \\
\lfloor T \natural y \natural X\rfloor(P \mid Q) = \lfloor T \natural y \natural X\rfloor(Q) & X \in Q \\
\lfloor T \natural y \natural X\rfloor((\nu a)P) = \lfloor T \natural y \natural X\rfloor(P) & \\
\lfloor T \natural y \natural X\rfloor(\mathsf{def}\ X'(x\,y') = P\ \mathsf{in}\ Q) = \lfloor T \natural y \natural X\rfloor(Q) & X \neq X' \\
\lfloor T \natural y \natural X\rfloor(X\langle e\,y'\rangle) = T & \\
\lfloor T \natural y \natural X\rfloor(\mathbf{0}) = \mathsf{end} &
\end{array}$$

Table XII. Application of a local type and a channel variable and a process variable to a user process.

5. PROGRESS

5.1 Progress Property and Channel Relations

This section studies progress: informally, we say that a process has the progress property if it can never reach a deadlock state, i.e., if it never reduces to a process which contains open sessions (this amounts to containing channels with roles) and which is irreducible in any inactive context (represented by another inactive process running in parallel).

Definition 5.1 *Progress*. A process P has the *progress property* if $P \longrightarrow^* P'$ implies that either P' does not contain channels with roles or $P' \mid Q \longrightarrow$ for some Q such that $P' \mid Q$ is well typed and $Q \not\longrightarrow$.

We will give an interaction type system which ensures that the typable processes always have the progress property. The crucial point to prove the progress property is to assure that a process, seen as a parallel composition of single threaded processes and queues, cannot be blocked in a configuration in which:

(1) there are no thread ready for a session initialization (i.e., of the form $\overline{a}[n](y).P$ or $a[\mathrm{p}](y).P$). Otherwise the process could be reactivated by providing it with the right partners;

(2) all subprocesses are either non-empty queues or processes waiting to perform an input action on a channel whose associated queue does not offer an appropriate message.

Progress inside a single service is assured by the communication typing rules in § 4. This will follow as an immediate corollary of Theorem 5.3. For ensuring progress, we need to introduce a few relations which statically analyse causalities and usage of channels.

Let us say that a *channel qualifier* is either a channel with role or a channel variable. Let c be a channel, its channel qualifier $\ell(c)$ is defined by: (1) if $c = y$, then $\ell(c) = y$; (2) else if $c = s[\mathrm{p}]$, then $\ell(c) = s$. Let Λ, ranged over by λ, denote the set of all service names and all channel qualifiers.

The progress property will be analysed via three finite sets: two sets $\mathscr{N}$ and $\mathscr{B}$ of service names and a set $\mathscr{R} \subseteq \Lambda \cup (\Lambda \times \Lambda)$. The set $\mathscr{N}$ collects the service names which are interleaved following the nesting policy. The set $\mathscr{B}$ collects the service names which can be bound. The Cartesian product $\Lambda \times \Lambda$, whose elements are denoted $\lambda \prec \lambda'$, represents a transitive relation. The meaning of $\lambda \prec \lambda'$ is that an input action involving a channel (qualified by) λ or belonging to service λ could block a communication action involving a channel (qualified by) λ' or belonging to service λ'. Moreover $\mathscr{R}$ includes all channel qualifiers and all service names which do not belong to $\mathscr{N}$ or $\mathscr{B}$ and which occur free in the current process. This will be useful to easily extend $\mathscr{R}$ in the assignment rules, as it will be pointed out below. We call $\mathscr{N}$ *nested service set*, $\mathscr{B}$ *bound service set* and $\mathscr{R}$ *channel relation* (even if only a subset of it is, strictly speaking, a relation). Let us give now some related definitions.

Definition 5.2. [Mario: I eliminated the syntax for $\mathscr{R}$. Already defined, moreover this is not syntax]

$$(1)\quad \mathscr{B} \overline{\cup} \{e\} \;=\; \begin{cases} \mathscr{B} \cup \{a\} & \text{if } e = a \text{ is a service name} \\ \mathscr{B} & \text{otherwise.} \end{cases}$$

$$(2)\quad \mathscr{R} \setminus \lambda \;=\; \{\lambda_1 \prec \lambda_2 \mid \lambda_1 \prec \lambda_2 \in \mathscr{R} \;\&\; \lambda_1 \neq \lambda \;\&\; \lambda_2 \neq \lambda\} \cup \{\lambda' \mid \lambda' \in \mathscr{R} \;\&\; \lambda' \neq \lambda\}$$

$$(3)\quad \mathscr{R} \setminus\!\!\setminus \lambda \;=\; \begin{cases} \mathscr{R} \setminus \lambda & \text{if } \lambda \text{ is minimal in } \mathscr{R} \\ \perp & \text{otherwise.} \end{cases}$$

$$(4)\quad \mathsf{pre}(\ell(c), \mathscr{R}) \;=\; (\mathscr{R} \cup \{\ell(c)\} \cup \{\ell(c) \prec \lambda \mid \lambda \in \mathscr{R} \;\&\; \ell(c) \neq \lambda\})^{+}$$

where $\mathscr{R}^{+}$ is the transitive closure of (the relation part of) $\mathscr{R}$ and λ is *minimal* in $\mathscr{R}$ if $\nexists \lambda' \prec \lambda \in \mathscr{R}$.

A channel relation is *well formed* if it is irreflexive, and does not contain cycles. A channel relation $\mathscr{R}$ is *channel free* ($\mathsf{cf}(\mathscr{R})$) if it contains only service names.

The channel relation is essentially defined to analyse the interactions between services: this is why in the definition of $\mathsf{pre}(\ell(c), \mathscr{R})$ we put the condition $\ell(c) \neq \lambda$. A basic point is that a loop in $\mathscr{R}$ represents the possibility of a deadlock state. For instance take the processes:

$$P_1 = b[1](y_1).\overline{a}[2](z_2).y_1?(2, x); z_2!\langle 1, \mathsf{false}\rangle; \mathbf{0}$$
$$P_2 = \overline{b}[2](y_2).a[1](z_1).z_1?(2, x'); y_2!\langle 1, \mathsf{true}\rangle; \mathbf{0}.$$

In process P_1 we have that an input action on service b can block an output action on service a and this determines $b \prec a$. In process P_2 the situation is inverted, determining $a \prec b$. In $P_1 \mid P_2$ we will then have a loop $a \prec b \prec a$. In fact $P_1 \mid P_2$ reduces to

$$Q \;=\; (\nu s)(\nu r)\;\; (s[1]?(2,x);r[1]!\langle 2,\mathsf{false}\rangle;\mathbf{0} \mid r[2]?(1,x');s[2]!\langle 1,\mathsf{true}\rangle;\mathbf{0})$$

which is stuck. It is easy to see that services a and b have the same types, thus we could change b in a in P_1 and P_2 obtaining:

$$P_1' = a[1](y_1).\overline{a}[2](z_2).y_1?(2,x);z_2!\langle 1,\mathsf{false}\rangle;\mathbf{0}$$
$$P_2' = \overline{a}[2](y_2).a[1](z_1).z_1?(2,x');y_2!\langle 1,\mathsf{true}\rangle;\mathbf{0}.$$

with two instances of service a and a relation $a \prec a$. But also $P_1' \mid P_2'$ would reduce to Q. Hence we must forbid also loops on single service names (i.e., the channel relation cannot be reflexive).

5.2 The Interaction Typing System

We will give now an informal account of the interaction typing rules, through a set of examples. It is understood that all processes introduced in the examples can be typed with the communication typing rules given in the previous section.

Tables XIII and XIV give the interaction typing rules. The judgements are of the shape: $\Theta \vdash P \;\blacktriangleright\; \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$ where Θ is a set of *assumptions* of the shape $X[y] \;\blacktriangleright \mathscr{R}\;;\;\mathscr{N}\;;\;\mathscr{B}$ (for recursive definitions) with the variable y representing the channel parameter of X.

We say that a judgement $\Theta \vdash P \;\blacktriangleright\; \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$ is *coherent* if: (1) $\mathscr{R}$ is well formed; (2) $\mathscr{R} \cap (\mathscr{N} \cup \mathscr{B}) = \emptyset$. We assume that the typing rules are applicable if and only if *the judgements in the conclusion are coherent.*

Initialisation (1): $\{\text{MCAST}\}$ and $\{\text{MACC}\}$. Three different sets of rules handle service initialisations. In rules $\{\text{MCAST}\}$ and $\{\text{MACC}\}$, which are liberal on the occurrences of the channel y in P, the service name a replaces y in $\mathscr{R}$. The addition of a is needed when y does not occur in $\mathscr{R}$. By combining with $\{\text{CONC}\}$ later, we can detect that the parallel composition $P_1 \mid P_2$ in § 5.1 creates a circular dependency of channels. Also we can immediately exclude circular dependencies involving the same service name as in $P_1' \mid P_2'$ in § 5.1. Both $P_1 \mid P_2$ and $P_1' \mid P_2'$ reduce to deadlock processes.

Initialisation (2): $\{\text{MCASTN}\}$ and $\{\text{MACCN}\}$. Rules $\{\text{MCASTN}\}$ and $\{\text{MACCN}\}$ can be applied only if the channel y associated to a is minimal in $\mathscr{R}$. This implies that once a is initialised in P all communication actions on the channel with role instantiating y must be performed before any input communication action on a different channel in P. For example, in the following, R_1 is untypable using rules $\{\text{MCASTN}\}$ and $\{\text{MACCN}\}$, while R_2 is typable since the sessions are correctly nested.

$$R_1 \;=\; a[1](y).\overline{b}[1](z).y?(2,x);z!\langle 2,\mathsf{false}\rangle;\mathbf{0}$$
$$R_2 \;=\; a[1](y).\overline{b}[1](z).z?(2,x);y!\langle 2,\mathsf{false}\rangle;\mathbf{0}$$

Note that R_2 can be composed with $R_3 = \overline{a}[2](y).b[2](z).z!\langle 1,\mathsf{true}\rangle;y?(1,x);\mathbf{0}$, which is also typable by $\{\text{MCASTN}\}$ and $\{\text{MACCN}\}$. The name a is added to the nested service set. Remarkably, via rules $\{\text{MCASTN}\}$-$\{\text{MACCN}\}$ we can prove progress when services are nested, generalising the typing strategy of [Coppo et al. 2007].

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash \overline{a}[n](y).P \blacktriangleright \mathscr{R}\{a/y\} \cup \{a\} ; \mathscr{N} ; \mathscr{B}}\ \{\text{MC}\textsc{ast}\} \qquad \frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash a[\mathsf{p}](y).P \blacktriangleright \mathscr{R}\{a/y\} \cup \{a\} ; \mathscr{N} ; \mathscr{B}}\ \{\text{MA}\textsc{cc}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash \overline{a}[n](y).P \blacktriangleright \mathscr{R} \backslash\backslash y ; \mathscr{N} \cup \{a\} ; \mathscr{B}}\ \{\text{MC}\textsc{astn}\} \qquad \frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash a[\mathsf{p}](y).P \blacktriangleright \mathscr{R} \backslash\backslash y ; \mathscr{N} \cup \{a\} ; \mathscr{B}}\ \{\text{MA}\textsc{ccn}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \quad \mathsf{cf}(\mathscr{R} \backslash\backslash y)}{\Theta \vdash \overline{u}[n](y).P \blacktriangleright \mathscr{R} \backslash\backslash y ; \mathscr{N} ; \mathscr{B} \mathbin{\overline{\cup}} \{u\}}\ \{\text{MC}\textsc{astb}\} \qquad \frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \quad \mathsf{cf}(\mathscr{R} \backslash\backslash y)}{\Theta \vdash u[\mathsf{p}](y).P \blacktriangleright \mathscr{R} \backslash\backslash y ; \mathscr{N} ; \mathscr{B} \mathbin{\overline{\cup}} \{u\}}\ \{\text{MA}\textsc{ccb}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash c!\langle \Pi, e\rangle ; P \blacktriangleright \{\ell(c)\} \cup \mathscr{R} ; \mathscr{N} ; \mathscr{B} \mathbin{\overline{\cup}} \{e\}}\ \{\text{S}\textsc{end}\} \qquad \frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash c?(\mathsf{q}, x) ; P \blacktriangleright \mathsf{pre}(\ell(c), \mathscr{R}) ; \mathscr{N} ; \mathscr{B}}\ \{\text{R}\textsc{cv}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash c!\langle\langle \mathsf{p}', c'\rangle\rangle ; P \blacktriangleright (\{\ell(c), \ell(c'), \ell(c) \prec \ell(c')\} \cup \mathscr{R})^{+} ; \mathscr{N} ; \mathscr{B}}\ \{\text{D}\textsc{eleg}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \qquad \mathscr{R} \subseteq \{\ell(c), y, \ell(c) \prec y\}}{\Theta \vdash c?((\mathsf{q}, y)) ; P \blacktriangleright \{\ell(c)\} ; \mathscr{N} ; \mathscr{B}}\ \{\text{SR}\textsc{ec}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash c \oplus \langle \Pi, l\rangle ; P \blacktriangleright \{\ell(c)\} \cup \mathscr{R} ; \mathscr{N} ; \mathscr{B}}\ \{\text{S}\textsc{el}\}$$

$$\frac{\Theta \vdash P_i \blacktriangleright \mathscr{R}_i ; \mathscr{N}_i ; \mathscr{B}_i \quad \forall i \in I}{\Theta \vdash c\&(\mathsf{p}, \{l_i : P_i\}_{i \in I}) \blacktriangleright \mathsf{pre}(\ell(c), \bigcup_{i \in I} \mathscr{R}_i) ; \bigcup_{i \in I} \mathscr{N}_i ; \bigcup_{i \in I} \mathscr{B}_i}\ \{\text{B}\textsc{ranch}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \quad \Theta \vdash Q \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'}{\Theta \vdash P \mid Q \blacktriangleright (\mathscr{R} \cup \mathscr{R}')^{+} ; \mathscr{N} \cup \mathscr{N}' ; \mathscr{B} \cup \mathscr{B}'}\ \{\text{C}\textsc{onc}\} \qquad \frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \quad a \notin \mathscr{R} \cup \mathscr{N}}{\Theta \vdash (\nu a)P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \setminus a}\ \{\text{NR}\textsc{es}\}$$

$$\frac{}{\Theta, X[y] \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \vdash X\langle e, c\rangle \blacktriangleright \mathscr{R}\{\ell(c)/y\} ; \mathscr{N} ; \mathscr{B} \mathbin{\overline{\cup}} \{e\}}\ \{\text{V}\textsc{ar}\}$$

$$\frac{\Theta, X[y] \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \qquad \Theta, X[y] \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \vdash Q \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'}{\Theta \vdash \mathsf{def}\ X(x, y) = P\ \mathsf{in}\ Q \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'}\ \{\text{D}\textsc{ef}\}$$

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B} \quad \Theta \vdash Q \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'}{\Theta \vdash \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \blacktriangleright (\mathscr{R} \cup \mathscr{R}')^{+} ; \mathscr{N} \cup \mathscr{N}' ; \mathscr{B} \cup \mathscr{B}'}\ \{\text{I}\textsc{f}\} \qquad \frac{}{\Theta \vdash \mathbf{0} \blacktriangleright \emptyset ; \emptyset ; \emptyset}\ \{\text{I}\textsc{nact}\}$$

Table XIII. Interaction typing rules I

Initialisation (3): $\{\text{MC}\textsc{astb}\}$ **and** $\{\text{MA}\textsc{ccb}\}$. The rules $\{\text{MC}\textsc{astb}\}$ and $\{\text{MA}\textsc{ccb}\}$ add u to the bound service set whenever u is a service name. These rules are much more restrictive: they require that y is the only free channel in P and that it is minimal. Thus no interaction with other channels or services is possible. This safely allows u to be a variable (since nothing is known about it before execution except its type) or a restricted name (since no channel with role can be made inaccessible at runtime by a restriction on

$$\frac{}{\Theta \vdash s : \varnothing \blacktriangleright \emptyset;\, \emptyset;\, \emptyset}\{\textsc{QInit}\} \qquad \frac{\Theta \vdash s : h \blacktriangleright \mathscr{R};\, \emptyset;\, \mathscr{B}}{\Theta \vdash s : h \cdot (\mathsf{q}, \Pi, v) \blacktriangleright \mathscr{R};\, \emptyset;\, \mathscr{B} \overline{\cup} \{v\}}\{\textsc{QAddVal}\}$$

$$\frac{\Theta \vdash s : h \blacktriangleright \mathscr{R};\, \emptyset;\, \mathscr{B}}{\Theta \vdash s : h \cdot (\mathsf{q}, \mathsf{p}, s'[\mathsf{p}']) \blacktriangleright (\{s, s', s \prec s'\} \cup \mathscr{R})^{+};\, \emptyset;\, \mathscr{B}}\{\textsc{QAddSess}\}$$

$$\frac{\Theta \vdash s : h \blacktriangleright \mathscr{R};\, \emptyset;\, \mathscr{B}}{\Theta \vdash s : h \cdot (\mathsf{q}, \Pi, l) \blacktriangleright \mathscr{R};\, \emptyset;\, \mathscr{B}}\{\textsc{QSel}\} \qquad \frac{\Theta \vdash P \blacktriangleright \mathscr{R};\, \mathscr{N};\, \mathscr{B}}{\Theta \vdash (\nu s)P \blacktriangleright \mathscr{R} \setminus s;\, \mathscr{N};\, \mathscr{B}}\{\textsc{SRes}\}$$

Table XIV. Interaction typing rules II

u). Note that rule $\{\textsc{NRes}\}$ requires that a occurs neither in $\mathscr{R}$ nor in $\mathscr{N}$. By these rules, the following dead-locked process is untypable:

$$(\nu a)(a[\mathsf{p}](y).z!\langle \Pi, e\rangle;\mathbf{0})$$

The sets $\mathscr{N}$ and $\mathscr{B}$ include all service names of a process P whose initialisations is typed with $\{\textsc{MCastN}\}$-$\{\textsc{MAccN}\}$, $\{\textsc{MCastB}\}$-$\{\textsc{MAccB}\}$, respectively. Note that for a service name which will replace a variable this is assured by the (conditional) addition of e to $\mathscr{B}$ in the conclusion of rule $\{\textsc{Send}\}$. The sets $\mathscr{N}$ and $\mathscr{B}$ are used to assure, via the coherence condition $\mathscr{R} \cap (\mathscr{N} \cup \mathscr{B}) = \emptyset$, that *all* participants to the same service are typed either by the first two rules or by the remaining four. This is crucial to assure progress. Take for instance the processes P_1 and P_2 above. If we type the session initialisation on b using rule $\{\textsc{MAccN}\}$ or $\{\textsc{MAccB}\}$ in P_1 and rule $\{\textsc{MCast}\}$ in P_2 no inconsistency would be detected. But this is impossible since rule $\{\textsc{Conc}\}$ does not type $P_1 \mid P_2$ owing to the coherence condition. Instead if we use $\{\textsc{MAcc}\}$ in P_1, we detect the loop $a \prec b \prec a$. Note that we could not use $\{\textsc{MCastN}\}$ or $\{\textsc{MCastB}\}$ for b in P_2 since y_2 is not minimal.

Since all four rules $\{\textsc{MCastN}\}$,$\{\textsc{MAccN}\}$, $\{\textsc{MCastB}\}$,$\{\textsc{MAccB}\}$ only allow to type nested sessions, we call them *nesting rules*.

Sending and Receiving. Rule $\{\textsc{Rcv}\}$ asserts that the input action can block all other actions in P, while rule $\{\textsc{Send}\}$ simply adds $\ell(c)$ to $\mathscr{R}$ to register the presence of a communication action in P. In fact output is asynchronous, thus it can be always performed. If the sent value is a session name, then it is added to the set $\mathscr{B}$ of names which can could be restricted. This is essential in order to force its initialisation with one of the nesting rules.

Delegation. Rule $\{\textsc{Deleg}\}$ is similar to $\{\textsc{Send}\}$ but it asserts that a use of $\ell(c)$ must precede a use of $\ell(c')$: the relation $\ell(c) \prec \ell(c')$ needs to be registered since an action blocking $\ell(c)$ also blocks $\ell(c')$. Rule $\{\textsc{SRec}\}$ avoids to create a process where two different roles in the same session are put in sequence. Following [Dezani-Ciancaglini et al. 2006; Yoshida and Vasconcelos 2007] we call this phenomenon self-delegation. As an example consider the processes

$$P_1 = b[1](z_1).a[1](y_1).y_1!\langle\langle 2, z_1\rangle\rangle;\mathbf{0}$$
$$P_2 = \overline{b}[2](z_2).\overline{a}[2](y_2).y_2?((1,x));x?(2,w);z_2!\langle 1, \mathsf{false}\rangle;\mathbf{0}$$

and note that $P_1 \mid P_2$ reduces to $(\nu s)(\nu r)(s[1]?(2,w);s[2]!\langle 1, \mathsf{false}\rangle;\mathbf{0})$ which is stuck. Note that $P_1 \mid P_2$ is typable by the communication type system but P_2 is not typable by the interaction type system, since by typing $y_2?((1,x));x?(2,w);z_2!\langle 1, \mathsf{false}\rangle;\mathbf{0}$ we get $y_2 \prec z_2$

which is forbidden by rule {SREC}.

We note that rules {MCASTN}-{MACCN} are useful for typing delegation. An example is process B of the three-buyer protocol, in which the typing of the subprocess

$$z_2!\langle\{1\}, quote\text{ - }contrib\text{ - }99\rangle; z_2!\langle\langle 1, y_2\rangle\rangle; z_2\&(1, \{\mathsf{ok}: \mathbf{0},\ \mathsf{quit}: \mathbf{0}\})$$

gives $z_2 \prec y_2$. So by using rule {MCAST} we would get first $b \prec y_2$ and then the cycle $y_2 \prec b \prec y_2$. Instead using rule {MCASTN} for b we get in the final typing of B either $\{a\};\{b\};\emptyset$ or $\emptyset;\{a,b\};\emptyset$ according to we use either {MCAST} or {MCASTN} for a. We will show in details, in the following, the typing rules to prove that the whole process B is well typed.

Branching and Selection. Rule {SEL} is similar with rule {SEND}, while rule {BRANCH} needs to record the causality as a union of channel relations of all branchings.

Other Rules for User Syntax. Rule {CONC} is the key to calculate the channel relation among composed processes has no circularity as explained before. Rule {NRES} checks that a is neither in the channel relation nor the nested service set (see {MCASTB} and {MACCB}). Rules {VAR} and {DEF} relates a process variable and the corresponding channel relation. In particular, {VAR} replaces $\ell(c)$ to y and records the expression in the bound service set (like {SEND}). Rules {IF} and {INACT} are standard.

Other Rules for Queues. The first four rules can be understood as rules {INACT}, {SEND}, {DELEG} and {SEL}, respectively. {SRES} deletes the channel from the channel relation, where session names can only appear (instead of the set $\mathscr{B}$ in {NRES}).

An example. As an example we consider again the process B of the three-buyer protocol, which we report here for convenience:

$$\begin{array}{l}\mathsf{B} = a[2](y_2).y_2?(3, quote); y_2?(1, contrib);\\ \quad \mathsf{if}\ (quote\text{ - }contrib < 100)\ \mathsf{then}\ \ y_2 \oplus \langle\{1,3\}, \mathsf{ok}\rangle; y_2!\langle\{3\}, \texttt{"Address"}\rangle; y_2?(3, date); \mathbf{0}\\ \quad \mathsf{else}\ \overline{b}[2](z_2).z_2!\langle\{1\}, quote\text{ - }contrib\text{ - }99\rangle; z_2!\langle\langle 1, y_2\rangle\rangle; z_2\&(1, \{\mathsf{ok}: \mathbf{0},\ \mathsf{quit}: \mathbf{0}\}).\end{array}$$

In the following we will show the rules of the interaction type system to prove that this process has the progress property (in particular we illustrate the contents of the sets $\mathscr{R}$, $\mathscr{N}$ and $\mathscr{B}$):Mariangiola says: the original example is commented [Mario: I changed a little the setting to make it more readable. Moreover, this is not an algorithmic system so the notion of backtrack is not appropriate. Mariangiola's version is commented]

Process	$\mathscr{R}$	$\mathscr{N}$	$\mathscr{B}$	Rule
$z_2\&(1,\{\mathsf{ok}:\mathbf{0},\ \mathsf{quit}:\mathbf{0}\})$	z_2	$\emptyset$	$\emptyset$	$\{\textsc{Branch}\}$
$z_2!\langle\langle 1,y_2\rangle\rangle$	$\{z_2,y_2,z_2\prec y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{Deleg}\}$
$z_2!\langle\{1\},\mathit{quote}\text{ - }\mathit{contrib}\text{ - }99\rangle$	$\{z_2,y_2,z_2\prec y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{Send}\}$
$\overline{b}[2](z_2)$	$\{y_2\}$	$\{b\}$	$\emptyset$	$\{\textsc{MCastN}\}$ (*)
$y_2?(3,\mathit{date})\mathbf{0}$	$\{y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{Rcv}\}$
$y_2!\langle\{3\},\texttt{"Address"}\rangle$	$\{y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{Send}\}$
$y_2\oplus\langle\{1,3\},\mathsf{ok}\rangle$	$\{y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{Sel}\}$
$\mathsf{if}\ (\ldots)$	$\{y_2\}$	$\{b\}$	$\emptyset$	$\{\textsc{If}\}$
The two $\{\textsc{Rcv}\}$ do not change the sets				
$a[2](y_2)$	$\{a\}$	$\{b\}$	$\emptyset$	$\{\textsc{MCast}\}$
Or, alternatively				
$a[2](y_2)$	$\emptyset$	$\{a,b\}$	$\emptyset$	$\{\textsc{MCastN}\}$

Note that choosing to apply rule $\{\textsc{MCast}\}$ at point (*) we get

Process	$\mathscr{R}$	$\mathscr{N}$	$\mathscr{B}$	Rule
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\overline{b}[2](z_2);$	$\{b,y_2,b\prec y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{MCast}\}$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$\mathsf{if}\ (\ldots)$	$\{b,y_2,b\prec y_2\}$	$\emptyset$	$\emptyset$	$\{\textsc{If}\}$

And at that point we cannot apply rule $\{\textsc{Rcv}\}$ (to add the input action $y_2?(1,\mathit{contrib})$) since we would get $\mathscr{R}=\{b,y_2,y_2\prec b,b\prec y_2,y_2\prec y_2\}$ which contains a loop.

5.3 Progress Theorem and Proofs

A closed runtime process P is *initial* if it is typable both in the communication and in the interaction type systems. The progress property is assured for all computations that are generated from an initial process.

THEOREM 5.3 PROGRESS. *All initial processes have the progress property.*

It is easy to verify that the (runtime) version of the three-buyer protocol can be typed in the interaction type system with $\{a\};\{b\};\emptyset$ and $\emptyset;\{a,b\};\emptyset$ according to which typing rules we use for the initialisation actions on the service name a. Therefore we get

COROLLARY 5.4. *The three-buyer protocol has the progress property.*

5.4 Proof of the Progress Theorem

In the following definitions and proofs we assume that all considered processes are well typed with the communication type system of Section 4.

LEMMA 5.5. *If* $\Theta\vdash s:h\cdot m\ \blacktriangleright\ \mathscr{R};\ \emptyset;\ \mathscr{B}$ *then* $\Theta\vdash s:m\cdot h\ \blacktriangleright\ \mathscr{R};\ \emptyset;\ \mathscr{B}$.

PROOF. By induction on h. $\square$

LEMMA 5.6 SUBSTITUTION LEMMA. *Let* $\Theta\vdash P\ \blacktriangleright\ \mathscr{R};\mathscr{N};\mathscr{B}$.

(*1*) *Let* $v \notin \mathscr{R}$. *Then* $\Theta \vdash P\{v/x\} \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}'$ *where* $\mathscr{B}' = \mathscr{B} \cup \{v\}$*;*

(*2*) $\Theta \vdash P\{s[\mathrm{p}]/y\} \;\blacktriangleright\; \mathscr{R}\{s/y\}\,;\,\mathscr{N}\,;\,\mathscr{B}$.

PROOF. By induction on $\Theta \vdash P \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}$.

(1) By induction on P. The only interesting case is when v is a service name a, thus, $P \equiv \overline{x}[n](y).P'$ or $P \equiv x[n](y).P'$ and the last applied rules are $\{\text{MCASTB}\}$ or $\{\text{MACCB}\}$, respectively. Let us consider $P \equiv \overline{x}[n](y).P'$ (the other case is similar). From $\{\text{MCASTB}\}$ we have that $\Theta \vdash P' \;\blacktriangleright\; \mathscr{R}'\,;\,\mathscr{N}\,;\,\mathscr{B}$ such that $\mathsf{cf}(\mathscr{R}' \setminus\!\!\setminus y)$ and $\mathscr{R} = \mathscr{R}' \setminus\!\!\setminus y$. Now, $P\{a/x\} = \overline{a}[n](y).P'$. Since, by hypothesis, $\mathsf{cf}(\mathscr{R}' \setminus\!\!\setminus y)$, thus we can apply $\{\text{MCASTB}\}$, obtaining $\Theta \vdash \overline{a}[n](y).P' \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B} \cup \{a\}$. Note that this judgements is coherent since by hypothesis $a \notin \mathscr{R}$.

(2) Easily follows from the definition of $\ell(c)$.

□

THEOREM 5.7 TYPE PRESERVATION UNDER EQUIVALENCE. *If P is well typed and* $\Theta \vdash P \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}$ *and* $P \equiv P'$*, then* $\Theta \vdash P' \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}$.

PROOF. Standard induction on $\equiv$. □

THEOREM 5.8 TYPE PRESERVATION UNDER REDUCTION. *If P is well typed and* $\Theta \vdash P \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}$ *and* $P \longrightarrow^* P'$*, then* $\Theta \vdash P' \;\blacktriangleright\; \mathscr{R}'\,;\,\mathscr{N}'\,;\,\mathscr{B}'$ *for some* $\mathscr{R}' \subseteq \mathscr{R}$*,* $\mathscr{N}' \subseteq \mathscr{N}$ *and* $\mathscr{B}' \subseteq \mathscr{B}$.

PROOF. By induction on $\longrightarrow$ and by cases on the last applied rule.

- [Link]. By hypothesis

$$\Theta \vdash a[1](y_1).P_1 \mid \ldots \mid a[n-1](y_{n-1}).P_{n-1} \mid \overline{a}[n](y_n).P_n \;\blacktriangleright\; \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}.$$

This judgement is obtained by applying rule $\{\text{CONC}\}$ to the subprocesses $a[1](y_1).P_1, \ldots, a[n-1](y_{n-1}).P_{n-1}, \overline{a}[n](y_n).P_n$. Then we have:
—$\Theta \vdash a[1](y_1).P_1 \;\blacktriangleright\; \mathscr{R}_1\,;\,\mathscr{N}_1\,;\,\mathscr{B}_1$
—...
—$\Theta \vdash a[n-1](y_{n-1}).P_{n-1} \;\blacktriangleright\; \mathscr{R}_{n-1}\,;\,\mathscr{N}_{n-1}\,;\,\mathscr{B}_{n-1}$
—$\Theta \vdash \overline{a}[n](y_n).P_n \;\blacktriangleright\; \mathscr{R}_n\,;\,\mathscr{N}_n\,;\,\mathscr{B}_n$
where $\mathscr{R} = (\bigcup_{1\le i\le n} \mathscr{R}_i)^+$ and $\mathscr{N} = \bigcup_{1\le i\le n} \mathscr{N}_i$ and $\mathscr{B} = \bigcup_{1\le i\le n} \mathscr{B}_i$. Point 2. of the the coherence condition (see page 19) implies that the rules $\{\text{MCAST}\}$, $\{\text{MACC}\}$ cannot be used for the same session name with the rules $\{\text{MCASTN}\}$, $\{\text{MACCN}\}$, $\{\text{MCASTB}\}$, $\{\text{MACCB}\}$.
We consider the case in which P_n has been typed with rule $\{\text{MCASTN}\}$ or $\{\text{MCASTB}\}$ and each P_{p} $(1 \le \mathrm{p} \le n-1)$ with $\{\text{MACCN}\}$ or $\{\text{MACCB}\}$.
Then for each i $(1 \le i \le n)$ we must have $\Theta \vdash P_i \;\blacktriangleright\; \mathscr{R}_i'\,;\,\mathscr{N}_i'\,;\,\mathscr{B}_i'$ such that $\mathscr{R}_i = \mathscr{R}_i' \setminus\!\!\setminus y_i$, $\mathscr{N}_i' \subseteq \mathscr{N}_i$, $\mathscr{B}_i' \subseteq \mathscr{B}_i$ (y_i is minimal in $\mathscr{R}_i'$). By Lemma 5.6(2) we have

$$\Theta \vdash P_i\{s[i]/y_i\} \;\blacktriangleright\; \mathscr{R}_i'\{s/y_i\}\,;\,\mathscr{N}_i'\,;\,\mathscr{B}_i'.$$

By using $\{\text{CONC}\}$ (and $\{\text{QINIT}\}$) we have

$$\Theta \vdash P_1\{s[1]/y_1\}|\ldots|P_n\{s[n]/y_n\}|s:\varnothing \;\blacktriangleright\; \mathscr{R}'\,;\,\mathscr{N}'\,;\,\mathscr{B}'$$

where $\mathscr{R}' = (\bigcup_{1\leq i\leq n} \mathscr{R}'_i\{s/y_i\})^+, \mathscr{N}' = \bigcup_{1\leq i\leq n} \mathscr{N}_i{}'$ and $\mathscr{B}' = \bigcup_{1\leq i\leq n} \mathscr{B}'_i$. Note that this judgement is coherent since s must be minimal in $\mathscr{R}'$ and $\mathscr{R}' \cap (\mathscr{N}' \cup \mathscr{B}') = \emptyset$. By using $\{\text{SRES}\}$,

$$\Theta \vdash (\nu s)(P_1\{s[1]/y_1\}|...|P_n\{s[n]/y_n\}|s:\varnothing) \blacktriangleright \mathscr{R}' \setminus s\,;\ \mathscr{N}'\,;\ \mathscr{B}'$$

Finally it is easy to see that $\mathscr{R}' \setminus s = \mathscr{R}$ (by the minimality of the y_i in $\mathscr{R}'_i$ and of s in $\mathscr{R}'$), $\mathscr{N}' \subseteq \mathscr{N}$ and $\mathscr{B}' \subseteq \mathscr{B}$.

- [Send]. By hypothesis, $\Theta \vdash s[\mathrm{p}]!\langle \Pi, e\rangle; P \mid s : h \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$, which is obtained by applying rule $\{\text{CONC}\}$. Thus,

$$\Theta \vdash s[\mathrm{p}]!\langle \Pi, e\rangle; P \blacktriangleright \mathscr{R}_1\,;\ \mathscr{N}\,;\ \mathscr{B}_1 \qquad \Theta \vdash s:h \blacktriangleright \mathscr{R}_2\,;\ \emptyset;\ \mathscr{B}_2$$

where $\mathscr{R} = (\mathscr{R}_1 \cup \mathscr{R}_2)^+$ and $\mathscr{B} = \mathscr{B}_1 \cup \mathscr{B}_2$. The first judgement can only be obtained by $\{\text{SEND}\}$, i.e., $\Theta \vdash P \blacktriangleright \mathscr{R}'_1\,;\ \mathscr{N}\,;\ \mathscr{B}'_1$ such that $\mathscr{R}_1 = \{s\} \cup \mathscr{R}'_1$ and $\mathscr{B}_1 = \mathscr{B}'_1 \overline{\cup} \{v\}$. By using rules $\{\text{QADDVAL}\}$ and $\{\text{CONC}\}$ we obtain

$$\Theta \vdash P \mid s : h \cdot (\mathrm{p}, \Pi, v) \blacktriangleright (\mathscr{R}'_1 \cup \mathscr{R}_2)^+\,;\ \mathscr{N}\,;\ \mathscr{B}'_1 \cup (\mathscr{B}_2 \overline{\cup} \{v\}).$$

Now note that $(\mathscr{R}'_1 \cup \mathscr{R}_2)^+ \subseteq \mathscr{R}$ and $\mathscr{B}'_1 \cup (\mathscr{B}_2 \overline{\cup} \{v\}) = \mathscr{B}$.

- [Deleg]. Proceed as in the previous case, thus obtaining

$$\Theta \vdash s[\mathrm{p}]!\langle\langle \mathrm{q}, s'[\mathrm{p}']\rangle\rangle; P \blacktriangleright \mathscr{R}_1\,;\ \mathscr{N}\,;\ \mathscr{B}_1 \qquad \Theta \vdash s:h \blacktriangleright \mathscr{R}_2\,;\ \emptyset;\ \mathscr{B}_2$$

where $\mathscr{R} = (\mathscr{R}_1 \cup \mathscr{R}_2)^+$ and $\mathscr{B} = \mathscr{B}_1 \cup \mathscr{B}_2$. By inverting rule $\{\text{DELEG}\}$ we obtain $\Theta \vdash P \blacktriangleright \mathscr{R}'_1\,;\ \mathscr{N}\,;\ \mathscr{B}_1$ where $\mathscr{R}_1 = (\{s, s', s \prec s'\} \cup \mathscr{R}'_1)^+$. By using rules $\{\text{QADDSESS}\}$ and $\{\text{CONC}\}$ we have

$$\Theta \vdash P \mid s : h \cdot (\mathrm{q}, \mathrm{p}, s'[\mathrm{p}']) \blacktriangleright (\mathscr{R}'_1 \cup \{s, s', s \prec s'\} \cup \mathscr{R}_2)^+\,;\ \mathscr{N}\,;\ \mathscr{B}_1 \cup \mathscr{B}_2.$$

- [Label]. Similar to [Send] but simpler (using rule $\{\text{QSEL}\}$ instead of $\{\text{QADDVAL}\}$).

- [Recv]. By hypothesis, $\Theta \vdash s[\mathrm{p}_j]?(\mathrm{q}, x); P \mid s : (\mathrm{q}, \Pi, v) \cdot h \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$. Proceed as in the case of rule [Send], thus obtaining

$$\Theta \vdash s[\mathrm{p}_j]?(\mathrm{q}, x); P \blacktriangleright \mathscr{R}_1\,;\ \mathscr{N}\,;\ \mathscr{B}_1 \qquad \Theta \vdash s : (\mathrm{q}, \Pi, v) \cdot h \blacktriangleright \mathscr{R}_2\,;\ \emptyset;\ \mathscr{B}_2$$

where $\mathscr{R} = (\mathscr{R}_1 \cup \mathscr{R}_2)^+$ and $\mathscr{B} = \mathscr{B}_1 \cup \mathscr{B}_2$. By inverting rule $\{\text{RECV}\}$ we obtain $\Theta \vdash P \blacktriangleright \mathscr{R}'_1\,;\ \mathscr{N}\,;\ \mathscr{B}_1$ where $\mathscr{R}_1 = \mathsf{pre}(s, \mathscr{R}'_1)$. By Lemma 5.6(1) we obtain $\Theta \vdash P\{v/x\} \blacktriangleright \mathscr{R}'_1\,;\ \mathscr{N}\,;\ \mathscr{B}_1 \overline{\cup} \{v\}$. Moreover we have $\Theta \vdash s : h \blacktriangleright \mathscr{R}_2\,;\ \emptyset;\ \mathscr{B}'_2$ where $\mathscr{B}_2 = \mathscr{B}'_2 \overline{\cup} \{v\}$. Applying $\{\text{CONC}\}$ we get

$$(1)\ \ \Theta \vdash P\{v/x\} \mid s : (\mathrm{q}, \Pi \setminus j, v) \cdot h \blacktriangleright (\mathscr{R}'_1 \cup \mathscr{R}_2)^+\,;\ \mathscr{N}\,;\ \mathscr{B}_1 \overline{\cup} \{v\} \cup \mathscr{B}'_2.$$

and note that $(\mathscr{R}'_1 \cup \mathscr{R}_2)^+ \subseteq (\mathscr{R}_1 \cup \mathscr{R}_2)^+$ and $\mathscr{B}_1 \overline{\cup} \{v\} \cup \mathscr{B}'_2 = \mathscr{B}$.
If $v = a$ is a service name, then $a \in \mathscr{B}_2$ implies that $a \notin (\mathscr{R}_1 \cup \mathscr{R}_2)^+$ and so $a \notin (\mathscr{R}'_1 \cup \mathscr{R}_2)^+$. Then (1) is coherent.

- [Srec]. By hypothesis, $\Theta \vdash s[\mathsf{p}]?((\mathsf{q},y));P \mid s:(\mathsf{q},\mathsf{p},s'[\mathsf{p}'])\cdot h \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$. Proceeding as before,

  $$\Theta \vdash s[\mathsf{p}]?((\mathsf{q},y));P \blacktriangleright \{s\}\,;\mathscr{N}\,;\mathscr{B}_1 \qquad \Theta \vdash s:(\mathsf{q},\mathsf{p},s'[\mathsf{p}'])\cdot h \blacktriangleright \mathscr{R}_2\,;\emptyset;\mathscr{B}_2$$

  where $\mathscr{R} = (\{s\}\cup\mathscr{R}_2)^+$ and $\mathscr{B} = \mathscr{B}_1\cup\mathscr{B}_2$. In particular (inverting rule $\{\text{SREC}\}$) we have $\Theta \vdash P \blacktriangleright \mathscr{R}'_1\,;\mathscr{N}\,;\mathscr{B}_1$ where $\mathscr{R}'_1 \subseteq \{s,y,s\prec y\}$. Moreover, by $\{\text{QADDSESS}\}$ (and Lemma 5.5) we have that $\Theta \vdash s:h \blacktriangleright \mathscr{R}'_2\,;\emptyset;\mathscr{B}_2$ such that $\mathscr{R}_2 = (\{s,s',s\prec s'\}\cup\mathscr{R}'_2)^+$. By Lemma 5.6(2), we have $\Theta \vdash P\{s'[\mathsf{p}']/y\} \blacktriangleright \mathscr{R}''_1\,;\mathscr{N}\,;\mathscr{B}_1$ where $\mathscr{R}''_1 \subseteq \{s,s',s\prec s'\}$. By applying rule $\{\text{CONC}\}$ we obtain

  $$\Theta \vdash P\{s'[\mathsf{p}']/y\} \mid s:h \blacktriangleright (\mathscr{R}''_1\cup\mathscr{R}'_2)^+\,;\mathscr{N}\,;\mathscr{B}_1\cup\mathscr{B}_2.$$

  Lastly it is easy to see that this statement is coherent and that $(\mathscr{R}''_1\cup\mathscr{R}'_2)^+ \subseteq \mathscr{R}$.

- [Branch]. By hypothesis, $\Theta \vdash s[\mathsf{p}_j]\&(\mathsf{q},\{l_i:P_i\}_{i\in I}) \mid s:(\mathsf{q},\Pi,l_{i_0})\cdot h \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$. By inverting the rules we have
  —$\Theta \vdash P_i \blacktriangleright \mathscr{R}_i\,;\mathscr{N}_i\,;\mathscr{B}_i \quad \forall i\in I$
  —$\Theta \vdash s:(\mathsf{q},\Pi,l_{i_0})\cdot h \blacktriangleright \mathscr{R}'\,;\emptyset;\mathscr{B}'$
  —$\mathscr{R} = (\mathsf{pre}(s,\bigcup_{i\in I}\mathscr{R}_i)\cup\mathscr{R}')^+$, $\mathscr{N} = \bigcup_{i\in I}\mathscr{N}_i$, $\mathscr{B} = \bigcup_{i\in I}\mathscr{B}_i\cup\mathscr{B}'$.
  By applying rule $\{\text{CONC}\}$ to the reduced process we obtain

  $$\Theta \vdash P_{i_0} \mid s:(\mathsf{q},\Pi\setminus j,l_{i_0})\cdot h \blacktriangleright (\mathscr{R}_{i_0}\cup\mathscr{R}')^+\,;\mathscr{N}_{i_0}\,;\mathscr{B}_{i_0}\cup\mathscr{B}'$$

  and the result follows easily.

- [If-T, If-F]. Straightforward.

- [Def]. Let's assume $\Theta \vdash \mathsf{def}\ X(x,y)=P\ \mathsf{in}\ (X\langle e,s[\mathsf{p}]\rangle \mid Q) \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$. Note that by rule $\lfloor\text{DEF}\rfloor$ y is the only free channel which can occur P. By inspecting the inference rule, as before, we must have:
  (a) $\Theta' = \Theta, X[y] \blacktriangleright \mathscr{R}'\,;\mathscr{N}'\,;\mathscr{B}'$;
  (b) $\Theta' \vdash P \blacktriangleright \mathscr{R}'\,;\mathscr{N}'\,;\mathscr{B}'$;
  (c) $\Theta' \vdash X\langle e,s[\mathsf{p}]\rangle \blacktriangleright \mathscr{R}'\{s/y\}\,;\mathscr{N}'\,;\mathscr{B}'\overline{\cup}\{e\}$;
  (d) $\Theta' \vdash Q \blacktriangleright \mathscr{R}''\,;\mathscr{N}''\,;\mathscr{B}''$;
  where $\mathscr{R} = (\mathscr{R}'\{s/y\}\cup\mathscr{R}'')^+, \mathscr{N} = \mathscr{N}'\cup\mathscr{N}'', \mathscr{B} = \mathscr{B}'\overline{\cup}\{e\}\cup\mathscr{B}''$.
  By Lemma 5.6 we have $\Theta' \vdash P\{v/x\}\{s[\mathsf{p}]/y\} \blacktriangleright \mathscr{R}\{s/y\}\,;\mathscr{N}\,;\mathscr{B}'\overline{\cup}\{v\}$ and then by rule $\{\text{CONC}\}$ $\Theta' \vdash (P\{v/x\}\{s[\mathsf{p}]/y\} \mid Q) \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$ since $e\downarrow v$ implies $\mathscr{B}'\overline{\cup}\{e\} = \mathscr{B}'\overline{\cup}\{v\}$. By rule $\{\text{DEF}\}$ we conclude $\Theta \vdash \mathsf{def}\ X(x,y)=P\ \mathsf{in}\ (P\{v/x\}\{s[\mathsf{p}]/y\} \mid Q) \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$.

- [Scop, Pat, Defin, Str]. For the congruence rules the thesis follows from the induction hypothesis.

□

LEMMA 5.9. *If* $\Gamma \vdash_\Sigma P \triangleright \Delta$ *and* $\Theta \vdash P \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$*, then:*
(1) $s[\mathsf{p}]:T \in \Delta$ *and* $T \neq \mathsf{end}$ *imply* $s\in\mathscr{R}$*;*
(2) $s\in\mathscr{R}$ *implies* $\Delta(s[\mathsf{p}]) \neq \mathsf{end}$ *for some* p.
PROOF. Standard by induction on P. □

LEMMA 5.10. *If* $\Theta \vdash P \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$ *and* $a \notin \mathscr{R} \cup \mathscr{N}$ *and* $P \equiv \overline{a}[n](y).P'$ *or* $P \equiv a[\mathrm{p}](y).P'$*, then no channel with role occurs in* $\mathscr{R}$*.*

PROOF. The last applied rule must be $\{\text{MCASTB}\}$ or $\{\text{MACCB}\}$ and then we must have $\Theta \vdash P' \blacktriangleright \mathscr{R}'\,;\mathscr{N}\,;\mathscr{B}$ and $\mathscr{R} = \mathscr{R}' \setminus\!\setminus y$. Note that the condition $\mathsf{cf}(\mathscr{R}' \setminus\!\setminus y)$ prevents channels with roles to occur in $\mathscr{R}'$. □

In the following definition we use $C[\;]$ to denote a context with a hole defined in the standard way.

Definition 5.11 *Precedence*. (1) The channel *c precedes* c' in the process P if one of the following condition holds:
—$P = C[c?(\mathrm{q},x);Q]$ and c' occurs in Q;
—$P = C[c!\langle\langle \mathrm{p},c'\rangle\rangle;Q]$;
—$P = C[c?((\mathrm{q},y));Q]$ and c' occurs in Q;
—$P = C[c\&(\mathrm{q},\{l_i : P_i\}_{i\in I})]$ and c' occurs in P_i for some $i \in I$;
—$P = C[s : h \cdot (\mathrm{q},\mathrm{p},s'[\mathrm{p}']) \cdot h']$ and $c = s[\mathrm{p}]$ and $c' = s'[\mathrm{p}']$.

(2) The channel *c weakly precedes* c' in the process P if either c precedes c' in P or one of the following condition holds:
—$P = C[c!\langle \Pi, e\rangle;Q]$ and c' occurs in Q;
—$P = C[c!\langle\langle \mathrm{p},c_0\rangle\rangle;Q]$ and c' occurs in Q.

LEMMA 5.12. *If* $\Theta \vdash P \blacktriangleright \mathscr{R}\,;\mathscr{N}\,;\mathscr{B}$ *and* $s[\mathrm{p}]$ *precedes* $s'[\mathrm{p}']$ *in* P *and* $s \neq s'$*, then* $s \prec s' \in \mathscr{R}$*.*

PROOF. By induction on P. □

LEMMA 5.13. *Let* P *be initial and* $P \longrightarrow^* P'$*.*

(*1*) *If* $s[\mathrm{p}]$ *weakly precedes* $s'[\mathrm{q}]$ *in* P'*, then either* $s \neq s'$ *or* $\mathrm{p} = \mathrm{q}$*;*

(*2*) *If* $P \equiv P' \mid s : h' \cdot (\mathrm{q},\mathrm{p},s'[\mathrm{p}']) \cdot h$ *then* $s' \neq s$*.*

PROOF. We show both points simultaneously by induction on $\longrightarrow^*$. In an initial P there are no channels with roles. As for the induction step we discuss the more interesting cases.
- Rule [Link] creates a new channel with a unique distinguished role for each parallel process. Both 1. and 2. follow trivially by the induction hypothesis.
- When the reduction step is obtained by rule [Srec] we must have $s : (\mathrm{q},\mathrm{p},s'[\mathrm{p}']) \cdot h$. By induction hypothesis we must have $s \neq s'$. By Theorem 5.8 we can derive a channel relation for the left hand side of the reduction rule [Srec] using the interaction typing rule $\{\text{SREC}\}$. Therefore $s[\mathrm{p}]$ and $s'[\mathrm{p}']$ are the only channels with role in $P\{s'[\mathrm{p}']/y\}$ and point 1. follows immediately.
- When the reduction step is obtained by rule [Deleg] note that the session delegation command must have been typed by rule $\lfloor\text{DELEG}\rfloor$. For this reason we get $s[\mathrm{p}] \neq s'[\mathrm{p}']$. Since $s[\mathrm{p}]$ precedes $s'[\mathrm{p}']$ in the session delegation command, then by induction $s = s'$ implies $\mathrm{p} = \mathrm{p}'$. We then conclude $s \neq s'$.

□

Definition 5.14. Define $\propto$ between processes, message queues and local types, as follows:

$$\begin{array}{c}
c!\langle\Pi,e\rangle;P \propto\, !\langle\Pi,S\rangle;T \quad c?(\mathsf{q},x);P \propto ?(\mathsf{q},S);T \\
c!\langle\langle\mathsf{p}',c'\rangle\rangle;P \propto\, !\langle\Pi,T\rangle;T \quad c?((\mathsf{q},y));P \propto ?(\mathsf{q},T);T \\
c\oplus\langle\Pi,l_i\rangle;P \propto \oplus\langle\Pi,\{l_i:T_i\}_{i\in I}\rangle \quad c\&(\mathsf{q},\{l_i:P_i\}_{i\in I}) \propto \&(\mathsf{p},\{l_i:T_i\}_{i\in I}) \\
(\mathsf{q},\Pi,v)\cdot h \propto\, !\langle\Pi,S\rangle;T \quad (\mathsf{q},\mathsf{p}',s[\mathsf{p}])\cdot h \propto\, !\langle\Pi,T\rangle;T \\
(\mathsf{q},\Pi,l)\cdot h \propto \oplus\langle\Pi,\{l_i:T_i\}_{i\in I}\rangle \quad X\langle e,c\rangle \propto T
\end{array}$$

where $i \in I$.

Definition 5.15. A process P is *ready* in a process Q if one of the following conditions holds:

—$Q \equiv P$;

—$Q \equiv P \mid R$ for some R;

—$Q \equiv (\nu a)R$ and P is ready in R, for some R, a;

—$Q \equiv (\nu s)R$ and P is ready in R, for some R, s;

—$Q \equiv \mathsf{def}\ D$ in R and P is ready in R, for some R, D.

Definition 5.16.

—An *input process* is a value sending, session delegation or label selection.

—An *output process* is a value reception, session reception or label branching.

—The identifier u is the *subject* of $\overline{u}[n](y).P$ and $u[\mathsf{p}](y).P$.

—The channel c is the *subject* of $c!\langle\Pi,e\rangle;P$, $c?(\mathsf{q},x);P$, $c!\langle\langle\mathsf{p}',c'\rangle\rangle;P$, $c?((\mathsf{q},y));P$, $c\oplus\langle\Pi,l\rangle;P$ and $c\&(\mathsf{q},\{l_i:P_i\}_{i\in I})$.

—An *output* type is a type of the shape $!\langle\Pi,U\rangle;\mathsf{T}$, $\oplus\langle\Pi,\{l_i:T_i\}_{i\in I}\rangle$, or $\oplus\langle\Pi,l\rangle;\mathsf{T}$.

—An *input* type is a type of the shape $?(\Pi,U);T$, or $\&(\mathsf{p},\{l_i:T_i\}_{i\in I})$.

LEMMA 5.17. *Assume that*

—$\Theta \vdash P \blacktriangleright \mathscr{R};\mathscr{N};\mathscr{B}$;

—*$\mathscr{R}$ contains service names which are not bigger than channels with roles and less than at least one channel with role;*

—*no ready process in P is an output or a conditional or a process call or a session initialisation on a variable.*

Then P contains one ready session initialisation on a free service name which belongs to $\mathscr{R}\cup\mathscr{N}$.

PROOF. If P is a session initialisation on a free service name which belongs to $\mathscr{R}\cup\mathscr{N}$ there is nothing to prove. Otherwise the proof is by induction on P.

P cannot be a session initialisation on a free session name which does not belong to $\mathscr{R}\cup\mathscr{N}$, since otherwise $\mathscr{R}$ could not contain channels with roles by Lemma 5.10.

P cannot be an input process since otherwise by Lemma 5.12 a channel with role would be less than all channels with roles which occur in $\mathscr{R}$.

If $P \equiv P_1 \mid P_2$, then $\mathscr{R} = (\mathscr{R}_1\cup\mathscr{R}_2)^+$ and $\Theta \vdash P_1 \blacktriangleright \mathscr{R}_1;\mathscr{N}_1;\mathscr{B}_1$ and $\Theta \vdash P_2 \blacktriangleright \mathscr{R}_2;\mathscr{N}_2;\mathscr{B}_2$ for some $\mathscr{R}_1,\mathscr{R}_2$, since the last applied rule for deriving $\Theta \vdash P \blacktriangleright \mathscr{R};\mathscr{N};\mathscr{B}$ must be {CONC}. Note that at least one between $\mathscr{R}_1$ and $\mathscr{R}_2$ must contain session names which are not bigger than channels with roles and less than at least one channel with role. Therefore by induction either P_1 or P_2 contains a ready session initialisation on a free service name which belongs to $\mathscr{R}\cup\mathscr{N}$.

If $P \equiv \text{def } X(x,y) = P'$ in Q, then $\Theta, X[y] \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}' \vdash Q \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ since the last applied rule for deriving $\Theta \vdash P' \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'$ must be $\{\text{DEF}\}$. Therefore by induction Q contains a ready session initialisation on a free service name which belongs to $\mathscr{R} \cup \mathscr{N}$.

If $P \equiv (\nu a)P'$, then $\Theta \vdash P' \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}'$ where $\mathscr{B}' = \mathscr{B} \setminus a$ and $a \notin \mathscr{R} \cup \mathscr{N}$, since the last applied rule for deriving $\Theta \vdash (\nu a)P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ must be $\{\text{NRES}\}$. Therefore by induction P' contains a ready session initialisation on a free service name which belongs to $\mathscr{R} \cup \mathscr{N}$.

□

LEMMA 5.18. *Assume that*

—$\Gamma \vdash_\Sigma P \triangleright \Delta$;

—$\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ *is proved without using rule* $\{\text{SRES}\}$*;*

—*s is minimal in $\mathscr{R}$;*

—*no $s[\mathrm{p}]$ precedes $s[\mathrm{q}]$ with $\mathrm{p} \neq \mathrm{q}$ in P;*

—*no ready process in P is an output, a conditional, a process call, a session initialisation on a free channel or on a variable.*

Then:

(1) *if $\Delta(s[\mathrm{p}])$ is an input type then P contains a ready input process Q with subject $s[\mathrm{p}]$ such that $Q \propto \Delta(s[\mathrm{p}])$;*

(2) *if $\Delta(s[\mathrm{p}])$ is an output type then P contains the queue $s : h$ and $h \propto \Delta(s[\mathrm{p}])$.*

PROOF. The proof of both points is by induction on P. Note that P cannot be a session initialisation on a bound channel, i.e. we cannot have $P \equiv (\nu a)Q$ where Q is a session initialisation on the channel a, since in that case the channel relation for Q should contain $a \prec s$ and this is impossible by Lemma 5.10.

(1). If P is an input process, then by Lemmas 5.12 and 5.13 the subject of P must be $s[\mathrm{p}]$: obviously P is ready. Note that P is a user process and then $\Gamma \vdash P \triangleright \Delta$ by Lemma A.2(1). We get $P \propto \Delta(s[\mathrm{p}])$ by Lemma A.1(8), (10) and (A.1).

If $P \equiv P_1 \mid P_2$, then by Lemma A.2(6) $\Sigma = \Sigma_1 \cup \Sigma_2$ and $\Delta = \Delta_1 * \Delta_2$ and $\Gamma \vdash_{\Sigma_1} P_1 \triangleright \Delta_1$ and $\Gamma \vdash_{\Sigma_2} P_2 \triangleright \Delta_2$. Since an input type is never a message type we have either $\Delta(s[\mathrm{p}]) = \Delta_1(s[\mathrm{p}])$ or $\Delta(s[\mathrm{p}]) = \Delta_2(s[\mathrm{p}])$. Assume $\Delta(s[\mathrm{p}]) = \Delta_1(s[\mathrm{p}])$. Moreover, since the last applied rule must be $\{\text{CONC}\}$, $\Theta \vdash P_1 \blacktriangleright \mathscr{R}_1 ; \mathscr{N}_1 ; \mathscr{B}_1$ and $\Theta \vdash P_2 \blacktriangleright \mathscr{R}_2 ; \mathscr{N}_2 ; \mathscr{B}_2$ and $\mathscr{R}$=$(\mathscr{R}_1 \cup \mathscr{R}_2)^+$. Note that by Lemma 5.9 $\mathscr{R}_1$ contains s. Moreover s is minimal in $\mathscr{R}_1$ since $\mathscr{R}_1 \subseteq \mathscr{R}$. Therefore by induction P_1 contains a ready input process Q with subject $s[\mathrm{p}]$ such that $Q \propto \Delta(s[\mathrm{p}])$.

If $P \equiv \text{def } X(x,y) = P'$ in Q, then by Lemma A.2(9) $\Gamma, X : S\ T, x : S \vdash P \triangleright y : T$ and $\Gamma, X : S\ T \vdash_\Sigma Q \triangleright \Delta$. Moreover $\Theta, X[y] \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}' \vdash Q \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$, since the last applied rule for deriving $\Theta \vdash P' \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'$ must be $\{\text{DEF}\}$. Therefore by induction Q contains a ready input process Q with subject $s[\mathrm{p}]$ such that $Q \propto \Delta(s[\mathrm{p}])$.

If $P \equiv (\nu a)P'$, then by Lemma A.2(8) $\Gamma, a : \langle G \rangle \vdash_\Sigma P' \triangleright \Delta$. Moreover, since the last applied rule for deriving $\Theta \vdash (\nu a)P' \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ must be $\{\text{NRES}\}$, $\Theta \vdash P' \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}'$ where $\mathscr{B} = \mathscr{B}' \setminus a$ and $a \notin \mathscr{R} \cup \mathscr{N}$. Therefore by induction P' contains a ready input process Q with subject $s[\mathrm{p}]$ such that $Q \propto \Delta(s[\mathrm{p}])$.

(2). If P is a queue, then it must be the queue s and the result follows from Lemma A.3.

If $P \equiv P_1 \mid P_2$, then by Lemma A.2(6) $\Sigma = \Sigma_1 \cup \Sigma_2$ and $\Delta = \Delta_1 * \Delta_2$ and $\Gamma \vdash_{\Sigma_1} P_1 \triangleright \Delta_1$ and $\Gamma \vdash_{\Sigma_2} P_2 \triangleright \Delta_2$. We consider the case $\Delta(s[\mathrm{p}]) = \Delta_1(s[\mathrm{p}]);\Delta_2(s[\mathrm{p}])$, the other cases being similar or simpler. As in the proof of (1) we get $\mathscr{R} = (\mathscr{R}_1 \cup \mathscr{R}_2)^+$ and $\Theta \vdash P_1 \blacktriangleright \mathscr{R}_1 ; \mathscr{N}_1 ; \mathscr{B}_1$. Note that by Lemma 5.9(1) $\mathscr{R}_1$ contain s. Therefore by induction P_1 contains the queue $s{:}h$ and $h \propto \Delta(s[\mathsf{p}])$.

If $P \equiv \mathsf{def}\ P_1\ \mathsf{in}\ P_2$ or $P \equiv (\nu a)P'$, the proof proceeds as in the case of (1).

□

Proof of Theorem 5.3 [Progress].
Let P_0 be initial and $P_0 \longrightarrow^* P$.

If P does not contain channels with roles there is nothing to prove.

If a ready sub-process of P is an output process, then P is reducible.

If a ready process in P is a conditional, then P would reduce, since P is closed (being P_0 closed) and any closed boolean value is either true or false. Similarly if a ready process of P is a process call it can be reduced.

No ready process in P is an accept/request on a variable since P is closed.

If one ready process in P is an accept/request on a free channel a, then a must be in the domain of the standard environment Γ used to type P_0 and P. Even if in P there are not enough partners to apply rule [Link], using $\Gamma(a)$ we can build a process Q containing the missing partners which are necessary in order to apply it to $P \mid Q$.

Otherwise let $P \equiv (\nu \tilde{s})Q$, where $\tilde{s}$ is the set of all session names which occur in P. By the Type Preservation Theorems A.6 and 5.8 P is well typed both in the communication and in the interaction type systems. This implies $\vdash Q \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ for some $\mathscr{R}, \mathscr{N}, \mathscr{B}$. Let Δ be the session environment of Q. Note that by construction we do not use rule $\{\text{SRES}\}$ for deriving $\mathscr{R}$. All minimals in $\mathscr{R}$ cannot be service names names since otherwise P would contain one ready initialisation on a free service name by Lemma 5.17. So there must be a session name s which is minimal. By Lemma 5.9(2) and the coherence of Δ there must be p, q such that $\Delta(s[\mathsf{p}]) = T$, $\Delta(s[\mathsf{q}]) = T'$ and $T \upharpoonright \mathsf{q} \bowtie T' \upharpoonright \mathsf{p}$. Without loss of generality we can assume that T is an input type and T' is an output type. Then Lemma 5.18 implies that Q contains a ready input process R such that $R \propto T$ and the queue $s : h$ with $h \propto T'$. Therefore P reduces by rule [Recv].

6. INFERENCE

In this section we introduce a deterministic, compositional type inference algorithm, defined via a set natural semantics rules, to check if a given process can be typed with the interaction typing rules of Section 5, thus assuring that the considered process has the progress property.

The inference rules define, on the structure of a process P, a judgement of the form:

$$\Theta\,; P \mapsto \mathsf{R}\,; \mathsf{C}; \mathsf{N}; \mathsf{B}$$

where R, C, N, B are inferred as defined below and they carry on the necessary informations to check if the process P is typable with the interaction rules. Instead the process variable environment Θ is a set of progress assumptions for process variables of the form $X[y] \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ that must be supplied by the user. The reason of this choice are twofold. On the one hand, it seems reasonable that a programmer be aware of the interaction properties of the recursive processes that represents the most critical part of a system: indeed the interactions properties of these components should be kept as simple as possible to allow an easy control of the global interaction. On the other hand the automatic inference of the process variable environment would introduce, to keep the completeness property, a very heavy fixpoint construction that would make much more difficult (and in the end unpractical) the inference algorithm.

The set $\mathsf{R} \subseteq \Lambda \cup (\Lambda \times \Lambda)$ denotes, informally, an order between channels and services as the set $\mathscr{R}$ in Section 5. Nevertheless R is *not* kept closed under transitivity. Then a pair $\lambda \prec \lambda' \in \mathsf{R}$ registers the fact that a communication via λ' *immediately* precedes a communication via λ. We will prove that all possible sets $\mathscr{R}$ that can be deduced in the interaction type system for a given process can be obtained by taking the transitive closure of suitable subsets of the set R inferred for this process.

Let $\mathscr{S}$ be the set of service names. The set C ia a partial function in $\mathscr{S} \rightarrow_{fin} \mathscr{P}^{fin}(\mathscr{S})$ represented as a set of pairs $(a, \mathscr{A})$ where $a \in \mathscr{S}$ and $\mathscr{A} \subseteq \mathscr{S}$. The service names occurring in C are intended to represent a subset of the service names in $dom(\mathsf{R})$ which could (potentially) be closed with a nesting rule. In particular in a pair $(a, \mathscr{A}) \in \mathsf{C}$ the set $\mathscr{A}$ represents the union of all subsets of service names that immediately preceded one occurrence of a in the channel relation, at the moment in which that occurrence was introduced in R. Recall that, in case of $\overline{a}[n](y)$ or $a[\mathrm{p}](y)$, the nesting rules require that the channel y is minimal in the channel relation. This means that if for some reason we realize that, in all deductions of the interaction system, a needs to be closed with a nesting rule (which implies that y must be minimal in the channel relation when the rule is applied) then we must register that all the services in $\mathscr{A}$ must also be closed with a nesting rule and propagate this along C.

The service names that do not belong to the domain of C can be closed exclusively with the rules {MCAST}, {MACC}. The inference rules, whenever applicable, guarantee that the transitive closure of R minus the domain of C is loop free.

The set $\mathsf{N} \subseteq \mathscr{S}$ contains the service names that *must* be closed with a nesting rule while $\mathsf{B} \subseteq \mathscr{S}$ denotes a set of service names that *may* be closed by rule {MCASTB}, {MACCB}. We will prove that all possible values of $\mathscr{N} \cup \mathscr{B}$ in a deduction in the interaction system for a given process must include the set N. As can be expected by their meaning R and N are disjoint sets while B can have a non empty intersection both with N and with R (in particular with the domain of C).

By the above discussion we then have that:

—the names in $dom(\mathsf{R}) \setminus dom(\mathsf{C})$ *must* be closed by rules {MCAST}, {MACC};

—the names in $dom(\mathsf{C}) \setminus \mathsf{B}$ are candidates to be closed by rules {MCAST}, {MACC} or {MCASTN}, {MACCN};

—the names in $dom(\mathsf{C}) \cap \mathsf{B}$ are candidates to be be closed by all rules;

—the names in $\mathsf{N} \setminus \mathsf{B}$ must be closed by rules {MCASTN}, {MACCN};

—the names in $\mathsf{N} \cap \mathsf{B}$ are candidates to be closed by rules {MCASTN}, {MACCN} or {MCASTB}, {MACCB}.

Let us introduce now some more preliminary definitions. Mariangiola says: Add explanations!

—If $\mathscr{A}$ is a set, then $\mathsf{R} \setminus \mathscr{A}$ extends Definition 5.2(2) in the obvious way.

—$\mathsf{R} \downarrow y = \{\lambda \mid \lambda \prec y \in \mathsf{R}\}$

—$\mathsf{C}(a)$ and $dom(\mathsf{C})$ have the obvious definition, $\mathsf{C}(\mathscr{A}) = \bigcup_{a \in \mathscr{A}} \mathsf{C}(a)$, where $\mathscr{A} \subseteq \mathscr{S}$.

—$\mathsf{C} \setminus \mathscr{A}$ is the function obtained from C by restricting the domain of C to the names which do not belong to $\mathscr{A}$

—$\mathsf{C} \Cup \mathsf{C}' = \{(a, \mathsf{C}(a) \cup \mathsf{C}'(a) \mid a \in dom(\mathsf{C}) \cup dom(\mathsf{C}')\}$ assuming $\mathsf{C}(a) = \emptyset$ if $a \notin dom(\mathsf{C})$

—$\mathsf{cl}(\mathsf{C}, a) = \{b \mid b \in \mathsf{C}(a)$ or $b \in \mathsf{C}(c)$ for some $c \in \mathsf{cl}(\mathsf{C}, a)\}$;

—$\mathsf{rcl}(\mathsf{C}, a) = \begin{cases} \mathsf{cl}(\mathsf{C}, a) \cup \{a\} & \text{if } a \in dom(\mathsf{C}), \\ \emptyset & \text{otherwise.} \end{cases}$

—$\mathsf{rcl}^{\Uparrow}(\mathsf{C}, a) = \{a\} \cup \{b \mid a \in \mathsf{cl}(\mathsf{C}, b)\}$

—$\mathsf{rcl}(\mathsf{C}, \mathscr{A}) = \bigcup_{a \in \mathscr{A}} \mathsf{rcl}(\mathsf{C}, a)$

The *closure* $\mathsf{cl}(\mathsf{C}, a)$ of the mapping C over a is intended to represent all the service names that must be closed with a nesting rule to allow a to be closed with a nesting rule. Note that $\mathsf{C}(a)$ only represents the service names that immediately precede a in R. The set $\mathsf{rcl}^{\Uparrow}(\mathsf{C}, a)$ represent all the service names that can be closed with a nesing rule only if a is closed with a nesting rule.

Note that "$\setminus$" is somewhat overloaded since it is used to denote both set difference, restriction on a relation ($\mathscr{R}$, R) and elimination from a domain of a finite map (in our case C).

Let's first prove a useful property.

LEMMA 6.1. *If $\mathscr{H} \subseteq \mathscr{K}$, then* $\mathsf{rcl}(\mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathscr{H}), \mathscr{K}) \cup \mathsf{rcl}(\mathsf{C}, \mathscr{H}) = \mathsf{rcl}(\mathsf{C}, \mathscr{K})$.

PROOF. $\underline{(\subseteq)}$ Proof of $\mathsf{rcl}(\mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathscr{H}), \mathscr{K}) \cup \mathsf{rcl}(\mathsf{C}, \mathscr{H}) \subseteq \mathsf{rcl}(\mathsf{C}, \mathscr{K})$. We have $\mathsf{rcl}(\mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathscr{H}), \mathscr{K}) \subseteq \mathsf{rcl}(\mathsf{C}, \mathscr{K})$ by definition of rcl and $\mathsf{rcl}(\mathsf{C}, \mathscr{H}) \subseteq \mathsf{rcl}(\mathsf{C}, \mathscr{K})$ by definition of rcl being $\mathscr{H} \subseteq \mathscr{K}$.
$\underline{(\supseteq)}$ Proof of $\mathsf{rcl}(\mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathscr{H}), \mathscr{K}) \cup \mathsf{rcl}(\mathsf{C}, \mathscr{H}) \supseteq \mathsf{rcl}(\mathsf{C}, \mathscr{K})$. If $a \in \mathsf{rcl}(\mathsf{C}, \mathscr{K})$ and $a \notin \mathsf{rcl}(\mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathscr{H}), \mathscr{K})$, then by definition of rcl there are $b \in \mathscr{K}$ and $c \in \mathsf{rcl}(\mathsf{C}, \mathscr{H})$ such that $c \in \mathsf{rcl}(\mathsf{C}, b)$ and $a \in \mathsf{rcl}(\mathsf{C}, c)$. We conclude that $a \in \mathsf{rcl}(\mathsf{C}, \mathscr{H})$. $\square$

The inference rules are given in Tables XV, XVI and XVII. The most crucial rules are $\{$MCAST-I$\}$, $\{$MACC-I$\}$. In both rules we use by the function m defined in Table XV. In this function we have to distinguish several cases. A first crucial point is weather the service name a involved in the rule is already contained in the sets R or N or not (conditions (μ), (ν) or (ρ). In all cases an other crucial condition is the position of the set $\mathsf{R} \downarrow y$ in R.

If $\mathsf{R} \downarrow y$ completely contained in the domain of C (cases (γ), (ν)) we can leave open the possibility that a could be closed with a nesting rule (case (γ)), since we can make y minimal.

If a is already an element of N as a service that must be closed with a nesting rule (case (ν)) then we must add to N all the services which must be closed by a nesting rule in order to allow the closure of a with a nesting rule, removing them from R, C.

If $\mathsf{R} \downarrow y$ is not included in the domain of C (case $(\overline{\gamma})$), then the service a cannot be closed with a nesting rule in any valid deduction of the interaction type system (since there is no way of making y minimal in any possible deduction). So if a already belongs to R (case $(\rho)(\alpha)(\overline{\gamma})$) we must eliminate from C and B both a and all the service names that could be closed by a nesting rule only if a also is closed by a nesting rule, checking that this restriction of C does not determine a loop in $\mathsf{R} \setminus dom(\mathsf{C})$.

The function $\mathsf{m}(u,\mathsf{R},\mathsf{C},\mathsf{N},\mathsf{B})$ is defined closing in boxes the list of returned values for the various cases.

- $u=a$ is a service name.
  (μ) $a \notin dom(\mathsf{R}) \cup \mathsf{N}$
    (γ) $\mathsf{R} \downarrow y \subseteq dom(\mathsf{C})$
      (ϕ) $\mathsf{cf}(\mathsf{R}\{a/y\})$ $\boxed{\mathsf{R}\{a/y\} \cup \{a\}; \mathsf{C} \uplus \{(a, \mathsf{R} \downarrow y)\}; \mathsf{N}; \mathsf{B} \cup \{\mathsf{a}\}}$
      $(\overline{\phi})$ $\neg\mathsf{cf}(\mathsf{R}\{a/y\})$ $\boxed{\mathsf{R}\{a/y\} \cup \{a\}; \mathsf{C} \uplus \{(a, \mathsf{R} \downarrow y)\}; \mathsf{N}; \mathsf{B}}$
    $(\overline{\gamma})$ $\mathsf{R} \downarrow y \not\subseteq dom(\mathsf{C})$ $\boxed{\mathsf{R}\{a/y\}; \mathsf{C}; \mathsf{N}; \mathsf{B}}$
  (ν) $a \in \mathsf{N}$ $\mathsf{R} \downarrow y \subseteq dom(\mathsf{C})$
    (ϕ) $\mathsf{cf}(\mathsf{R}\{a/y\})$ $\boxed{\mathsf{R} \setminus (\mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y) \cup \{y\}); \mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y); \mathsf{N} \cup \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow \mathsf{y}); \mathsf{B}}$
    $(\overline{\phi})$ $\neg\mathsf{cf}(\mathsf{R}\{a/y\})$ $\boxed{\mathsf{R} \setminus (\mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y) \cup \{y\}); \mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y); \mathsf{N} \cup \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow \mathsf{y}); \mathsf{B} \setminus \{\mathsf{a}\}}$
  (ρ) $a \in dom(\mathsf{R})$
    (α) $a \in dom(\mathsf{C})$
      (γ) $\mathsf{R} \downarrow y \subseteq dom(\mathsf{C})$
        (ϕ) $\mathsf{cf}(\mathsf{R}\{a/y\})$ $\boxed{\mathsf{R}\{a/y\}; \mathsf{C} \uplus \{(a, \mathsf{R} \downarrow y)\}; \mathsf{N}; \mathsf{B}}$
        $(\overline{\phi})$ $\neg\mathsf{cf}(\mathsf{R}\{a/y\})$ $\boxed{\mathsf{R}\{a/y\}; \mathsf{C} \uplus \{(a, \mathsf{R} \downarrow y)\}; \mathsf{N}; \mathsf{B} \setminus \{\mathsf{a}\}}$
      $(\overline{\gamma})$ $\mathsf{R} \downarrow y \not\subseteq dom(\mathsf{C})$ and $(\mathsf{R}\{a/y\} \setminus dom(\mathsf{C} \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}, a)))^{+}$ is loop free
      $\boxed{\mathsf{R}\{a/y\}; \mathsf{C} \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}, a); \mathsf{N}; \mathsf{B} \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}, \mathsf{a})}$
    $(\overline{\alpha})$ $a \notin dom(\mathsf{C})$ and $(\mathsf{R}\{a/y\} \setminus dom(\mathsf{C}))^{+}$ is loop free $\boxed{\mathsf{R}\{a/y\}; \mathsf{C}; \mathsf{N}; \mathsf{B}}$
- $u=x$ is a variable, $\mathsf{R} \downarrow y \subseteq dom(\mathsf{C})$, $\mathsf{cf}(\mathsf{R} \setminus \{y\})$
  $\boxed{\mathsf{R} \setminus (\mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y) \cup \{y\});\ \mathsf{C} \setminus \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y); \mathsf{N} \cup \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow \mathsf{y}); \mathsf{B}}$

Table XV. Inference function for services

The treatment of the set B is somewhat simpler: just note that a service name a belongs to B only if it can be closed by a nesting rule and each subprocess starting with an accept-request on a has an associated channel relation that does not contain free channel names.

Another crucial rule is $\{$CONC-I$\}$. In this rule we must check that the two processes P_1 and P_2 that are put in parallel are compatible with respect to the interaction properties. The crucial part is the treatment of the names that occur both in P_1 and P_2. Note that the sets D_i contains the service names that cannot be closed with the nesting rules in the respective process. The set K then contains the services that must be excluded from the C since it contains both sets D_i and the set of services whose closure with the nesting rules would imply that a service in some D_i should also be closed with a nesting rule.

Rule $\{$IF-I$\}$ has the same structure of $\{$CONC-I$\}$. Rule $\{$BRANCH-I$\}$ is a generalization of $\{$IF-I$\}$.

We need four different versions of rule $\{$SEND-I$\}$ in order to to take into account the different form of the communicated value. Note in particular that if a service name is sent we must force it to be closed by a nesting rule (if this is allowed by R and C).

An an example consider the process:

$$P_3 = a[1](y_1).b[1](z).\,(P_1 \mid P_2)$$

$$\frac{}{\Theta\,;\mathbf{0} \mapsto\!\!\!\Rightarrow\ \emptyset\,;\,\emptyset;\,\emptyset;\,\emptyset}\ \{\textsc{Inact-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}{\Theta\,;\overline{u}[n](y).P \mapsto\!\!\!\Rightarrow\ \mathsf{R}'\,;\,\mathsf{C}';\,\mathsf{N}';\,\mathsf{B}'}\ \{\textsc{MCast-I}\} \qquad \frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}{\Theta\,;u[\mathrm{p}](y).P \mapsto\!\!\!\Rightarrow\ \mathsf{R}'\,;\,\mathsf{C}';\,\mathsf{N}';\,\mathsf{B}'}\ \{\textsc{MAcc-I}\}$$

where $\mathsf{R}';\mathsf{C}';\mathsf{N}';\mathsf{B}' = \mathsf{m}(u,\mathsf{R},\mathsf{C},\mathsf{N},\mathsf{B})$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \qquad \mathsf{e} \neq \mathsf{a}}{\Theta\,;y!\langle\Pi,e\rangle;P \mapsto\!\!\!\Rightarrow\ \{y\}\cup\mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}\ \{\textsc{Send1-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \qquad \mathsf{e} = \mathsf{a} \qquad \mathsf{a} \notin \mathsf{dom}(\mathsf{R})\cup\mathsf{N}}{\Theta\,;y!\langle\Pi,e\rangle;P \mapsto\!\!\!\Rightarrow\ \{y\}\cup\mathsf{R}\,;\,\mathsf{C};\,\mathsf{N}\cup\{\mathsf{a}\};\,\mathsf{B}\cup\{\mathsf{a}\}}\ \{\textsc{Send2-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \quad \mathsf{e} = \mathsf{a} \quad \mathsf{a} \in \mathsf{dom}(\mathsf{C})}{\Theta\,;y!\langle\Pi,e\rangle;P \mapsto\!\!\!\Rightarrow\ \{y\}\cup\mathsf{R}\setminus\mathsf{rcl}(\mathsf{C},a)\,;\,\mathsf{C}\setminus\mathsf{rcl}(\mathsf{C},a);\,(\mathsf{N}\cup\mathsf{rcl}(\mathsf{C},\mathsf{a}));\,\mathsf{B}}\ \{\textsc{Send3-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \quad \mathsf{e} = \mathsf{a} \quad \mathsf{a} \in \mathsf{N}}{\Theta\,;y!\langle\Pi,e\rangle;P \mapsto\!\!\!\Rightarrow\ \{y\}\cup\mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}\ \{\textsc{Send4-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \qquad \mathsf{R}\downarrow\mathsf{y} \subseteq \mathsf{dom}(\mathsf{C})}{\Theta\,;y?(\mathsf{q},x);P \mapsto\!\!\!\Rightarrow\ \mathsf{pre}(y,\mathsf{R}\setminus(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y)))\,;\,\mathsf{C}\setminus(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y));\,\mathsf{N}\cup(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow\mathsf{y}));\,\mathsf{B}}\ \{\textsc{Rcv-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}{\Theta\,;y!\langle\langle\mathrm{p}',y'\rangle\rangle;P \mapsto\!\!\!\Rightarrow\ \{y,y',y\prec y'\}\cup\mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}\ \{\textsc{Deleg-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \qquad \mathsf{R}\setminus\mathsf{dom}(\mathsf{C}) \subseteq \{\mathsf{y},\mathsf{y}',\mathsf{y}\prec\mathsf{y}'\}}{\Theta\,;y?((\mathsf{q},y'));P \mapsto\!\!\!\Rightarrow\ \{y\}\,;\,\emptyset;\,\mathsf{N}\cup\mathsf{dom}(\mathsf{C});\,\mathsf{B}}\ \{\textsc{SRec-I}\}$$

$$\frac{\Theta\,;P \mapsto\!\!\!\Rightarrow\ \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}{\Theta\,;y\oplus\langle\Pi,l\rangle;P \mapsto\!\!\!\Rightarrow\ \{y\}\cup\mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}\ \{\textsc{Sel-I}\}$$

$$\frac{\Theta\,;P_i \mapsto\!\!\!\Rightarrow\ \mathsf{R}_i\,;\,\mathsf{C}_i;\,\mathsf{N_i};\,\mathsf{B_i} \quad \mathsf{i}\in\mathsf{I} \qquad \mathsf{K}\cap\mathsf{N} = \emptyset}{\Theta\,;y\&(\mathrm{p},\{l_i : P_i\}_{i\in I}) \mapsto\!\!\!\Rightarrow\ \mathsf{pre}(y,\mathsf{R})\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}\ \{\textsc{Branch-I}\}$$

where

$\mathsf{D}_i = dom(\mathsf{R}_i)\setminus dom(\mathsf{C}_i) \qquad i\in I$

$\mathsf{N} = \bigcup_{\mathsf{i}\in\mathsf{I}}\mathsf{N_i}\cup\mathsf{cl}(\uplus_{\mathsf{i}\in\mathsf{I}}\mathsf{C_i},\bigcup_{\mathsf{i}\in\mathsf{I}}\mathsf{N_i})$

$\mathsf{K} = \mathsf{rcl}^{\Uparrow}(\uplus_{i\in I}\mathsf{C}_i,\bigcup_{i\in I}\mathsf{D}_i)$

$\mathsf{R} = (\bigcup_{i\in I}\mathsf{R}_i)\setminus\mathsf{N}$

$\mathsf{C} = (\uplus_{i\in I}\mathsf{C}_i)\setminus(\mathsf{N}\cup\mathsf{K})$

$\mathsf{B} = \{\mathsf{a} \mid \exists\mathsf{i}\in\mathsf{I}.\ \mathsf{a}\in\mathsf{B_i}\ \wedge\ \forall\mathsf{i}\in\mathsf{I}.\ (\mathsf{a}\in\mathsf{B_i}\vee\mathsf{a}\notin\mathsf{dom}(\mathsf{R_i})\cup\mathsf{N_i})\}$

Table XVI. Inference for Interaction I

$$\frac{\Theta\,;P_1 \Mapsto \mathsf{R}_1\,;\,\mathsf{C}_1\,;\,\mathsf{N}_1\,;\,\mathsf{B}_1 \qquad \Theta\,;\mathsf{P}_2 \Mapsto \mathsf{R}_2\,;\,\mathsf{C}_2\,;\,\mathsf{N}_2\,;\,\mathsf{B}_2 \qquad \mathsf{K}\cap\mathsf{N}=\emptyset}{\Theta\,;P_1 \mid P_2 \Mapsto \mathsf{R}\,;\,\mathsf{C}\,;\,\mathsf{N}\,;\,\mathsf{B}}\ \{\textsc{Conc-I}\}$$

where

$\mathsf{D}_i = dom(\mathsf{R}_i)\setminus dom(\mathsf{C}_i) \qquad i\in\{1,2\}$

$\mathsf{N} = \mathsf{N}_1\cup\mathsf{N}_2\cup\mathsf{cl}(\mathsf{C}_1\uplus\mathsf{C}_2,\mathsf{N}_1\cup\mathsf{N}_2)$

$\mathsf{K} = \mathsf{rcl}^{\Uparrow}(\mathsf{C}_1\uplus\mathsf{C}_2,\mathsf{D}_1\cup\mathsf{D}_2)$

$\mathsf{R} = (\bigcup_{i=1,2}\mathsf{R}_i)\setminus\mathsf{N}$

$\mathsf{C} = (\mathsf{C}_1\uplus\mathsf{C}_2)\setminus(\mathsf{N}\cup\mathsf{K})$

$\mathsf{B} = (\mathsf{B}_1\cap\mathsf{B}_2)\cup(\bigcup_{\mathsf{i}=1,2}\{\mathsf{a}\in\mathsf{B}_\mathsf{i}\mid \mathsf{a}\notin\mathsf{dom}(\mathsf{R}_{\bar{\mathsf{i}}})\cup\mathsf{N}_{\bar{\mathsf{i}}}\})$

$$\frac{X[y]\ \blacktriangleright\ \mathscr{R};\mathscr{N};\mathscr{B}\in\Theta}{\Theta\ ;\ X\langle e,y\rangle \Mapsto \mathscr{R}\{y/y\}\,;\,\emptyset;\ \mathscr{N}\overline{\cup}\{e\}\cup\mathscr{B};\ \mathscr{B}\overline{\cup}\{e\}}\ \{\textsc{Var-I}\}$$

$$\frac{\begin{array}{c}\Theta,X[y]\ \blacktriangleright\ \mathscr{R};\mathscr{N};\mathscr{B}\ ;\ P\Mapsto \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}\\ \mathscr{R}=(\mathsf{R}\setminus(\mathscr{N}\cup\mathscr{B}))^{+} \qquad \mathscr{N}\cup\mathscr{B}=\mathsf{rcl}(\mathsf{C},\mathscr{N}\cup\mathscr{B})\cup\mathsf{N} \qquad \mathscr{B}\setminus\mathscr{N}\subseteq\mathsf{B}\\ \Theta\,;Q\Mapsto \mathsf{R}'\,;\,\mathsf{C}';\,\mathsf{N}';\,\mathsf{B}'\end{array}}{\Theta\,;\mathsf{def}\ X(x,y)=P\ \mathsf{in}\ Q\Mapsto \mathsf{R}'\,;\,\mathsf{C}'\,;\,\mathsf{N}';\,\mathsf{B}'}\ \{\textsc{Def-I}\}$$

$$\frac{\Theta\,;P\Mapsto \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}\quad \text{either } \mathsf{a}\in\mathsf{B} \text{ or } \mathsf{a}\notin\mathsf{dom}(\mathsf{R})\cup\mathsf{N}}{\Theta\,;(\nu a)P\Mapsto \mathsf{R}\setminus\mathsf{rcl}(\mathsf{C},a)\,;\,\mathsf{C}\setminus\mathsf{rcl}(\mathsf{C},a);\,\mathsf{N}\setminus\{\mathsf{a}\}\cup\mathsf{cl}(\mathsf{C},\mathsf{a});\,\mathsf{B}\setminus\{\mathsf{a}\}}\ \{\textsc{NRes-I}\}$$

$$\frac{\Theta\,;P_1 \Mapsto \mathsf{R}_1\,;\,\mathsf{C}_1\,;\,\mathsf{N}_1\,;\,\mathsf{B}_1 \quad \Theta\,;\mathsf{P}_2 \Mapsto \mathsf{R}_2\,;\,\mathsf{C}_2\,;\,\mathsf{N}_2\,;\,\mathsf{B}_2 \qquad \mathsf{K}\cap\mathsf{N}=\emptyset}{\Theta\,;\mathsf{if}\ e\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \Mapsto \mathsf{R}\,;\,\mathsf{C}\,;\,\mathsf{N}\,;\,\mathsf{B}}\ \{\textsc{If-I}\}$$

where

$\mathsf{D}_i = dom(\mathsf{R}_i)\setminus dom(\mathsf{C}_i) \qquad i\in\{1,2\}$

$\mathsf{N} = \mathsf{N}_1\cup\mathsf{N}_2\cup\mathsf{cl}(\mathsf{C}_1\uplus\mathsf{C}_2,\mathsf{N}_1\cup\mathsf{N}_2)$

$\mathsf{K} = \mathsf{rcl}^{\Uparrow}(\mathsf{C}_1\uplus\mathsf{C}_2,\mathsf{D}_1\cup\mathsf{D}_2)$

$\mathsf{R} = (\bigcup_{i=1,2}\mathsf{R}_i)\setminus\mathsf{N}$

$\mathsf{C} = (\mathsf{C}_1\uplus\mathsf{C}_2)\setminus(\mathsf{N}\cup\mathsf{K})$

$\mathsf{B} = (\mathsf{B}_1\cap\mathsf{B}_2)\cup(\bigcup_{\mathsf{i}=1,2}\{\mathsf{a}\in\mathsf{B}_\mathsf{i}\mid \mathsf{a}\notin\mathsf{dom}(\mathsf{R}_{\bar{\mathsf{i}}})\cup\mathsf{N}_{\bar{\mathsf{i}}}\})$

Table XVII. Inference for Interaction II

where $P_1 = a[2](y_2).y_2?(4,x_1);z?(5,x_2);\mathbf{0}$ and $P_2 = y_1?(6,x_3);a[3](y_3).y_3?(7,x_4);\mathbf{0}$.

Applying the inference rules we get:

$$\begin{array}{l}
\emptyset \;;\; y_2?(4,x_1); z?(5,x_2); \mathbf{0} \mapsto \{y_2 \prec z, y_2, z\} \;;\; \emptyset;\; \emptyset;\; \emptyset \\
\emptyset \;;\; a[2](y_2).y_2?(4,x_1); z?(5,x_2); \mathbf{0} \mapsto \{a \prec z, a, z\} \;;\; \{(a,\emptyset)\};\; \emptyset;\; \emptyset \\
\emptyset \;;\; a[3](y_3).y_3?(7,x_4); \mathbf{0} \mapsto \{a\} \;;\; \{(a,\emptyset)\};\; \emptyset;\; \{a\} \\
\emptyset \;;\; y_1?(6,x_3); a[3](y_3).y_3?(7,x_4); \mathbf{0} \mapsto \{y_1 \prec a, y_1, a\} \;;\; \{(a,\emptyset)\};\; \emptyset;\; \{a\} \\
\emptyset \;;\; P_1 \mid P_2 \mapsto \{a \prec z, y_1 \prec a, a, z, y_1\} \;;\; \{(a,\emptyset)\};\; \emptyset;\; \emptyset \\
\emptyset \;;\; b[1](z).\,(P_1 \mid P_2) \mapsto \{a \prec b, y_1 \prec a, a, b, y_1\} \;;\; \{(a,\emptyset),(b,a)\};\; \emptyset;\; \emptyset \\
\emptyset \;;\; a[1](y_1).b[1](z).\,(P_1 \mid P_2) \mapsto \{a \prec b, a \prec a, a, b\} \;;\; \{(a,\emptyset),(b,a)\};\; \emptyset;\; \emptyset
\end{array}$$

With a simple induction on deductions we can show the following basic facts.

LEMMA 6.2. *Let* $\Theta \;;\; P \mapsto \mathsf{R};\, \mathsf{C};\, \mathsf{N};\, \mathsf{B}$. *Then*

(*1*) $dom(\mathsf{R}) \cap \mathsf{N} = \emptyset$.
(*2*) $dom(\mathsf{C}) \subseteq dom(\mathsf{R})$.
(*3*) $\mathsf{B} \subseteq \mathsf{dom}(\mathsf{C}) \cup \mathsf{N}$.
(*4*) $range(\mathsf{C}) \subseteq dom(\mathsf{C})$.
(*5*) $\mathsf{cl}(\mathsf{C}, \mathscr{A}) \subseteq dom(\mathsf{C})$.
(*6*) $\mathsf{rcl}(\mathsf{C}, \mathscr{A}) \subseteq dom(\mathsf{C})$.
(*7*) $\mathsf{rcl}^{\Uparrow}(\mathsf{C}, \mathscr{A}) \subseteq dom(\mathsf{C})$.
(*8*) $(\mathsf{R} \setminus dom(\mathsf{C}))^+$ *is loop free.*

LEMMA 6.3. *If* $\mathsf{R} \downarrow y \subseteq dom(\mathsf{C})$, *then y is minimal in* $(\mathsf{R} \setminus \mathsf{rcl}(\mathsf{C}, \mathsf{R} \downarrow y))^+$.

The following Soundness and Completeness Theorem shows that he above rules fully characterize all possible interference typing of a process. correctness of rule (Def-I).

THEOREM 6.4 SOUNDNESS. *If* $\Theta \;;\; P \mapsto \mathsf{R};\, \mathsf{C};\, \mathsf{N};\, \mathsf{B}$ *and for some* $\mathscr{R}, \mathscr{N}, \mathscr{B}$:

(*1*) $\mathscr{R} = (\mathsf{R} \setminus (\mathscr{N} \cup \mathscr{B}))^+$ *is loop free;*
(*2*) $\mathscr{N} \cup \mathscr{B} = \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) \cup \mathsf{N}$;
(*3*) $\mathscr{B} \setminus \mathscr{N} \subseteq \mathsf{B}$;

then $\Theta \vdash P \blacktriangleright \mathscr{R};\, \mathscr{N};\, \mathscr{B}$.

PROOF. By induction on the proof of $\Theta \;;\; P \mapsto \mathsf{R};\, \mathsf{C};\, \mathsf{N};\, \mathsf{B}$.
<u>Rule {MCAST-I}</u>

$$\frac{\Theta\,; P \mapsto \mathsf{R}';\, \mathsf{C}';\, \mathsf{N}';\, \mathsf{B}'}{\Theta\,; \overline{u}[n](y).P \mapsto \mathsf{R};\, \mathsf{C};\, \mathsf{N};\, \mathsf{B}}\;\{\text{MCAST-I}\}$$

where $\mathsf{R};\mathsf{C};\mathsf{N};\mathsf{B} = \mathsf{m}(\mathsf{u}, \mathsf{R}', \mathsf{C}', \mathsf{N}', \mathsf{B}')$.

We define:

- $\mathscr{R}' = (\mathsf{R}' \setminus (\mathscr{N}' \cup \mathscr{B}'))^+$,
- $\mathscr{N}' = \begin{cases} \mathscr{N} \setminus u & \text{if } u \notin dom(\mathsf{R}') \cup \mathsf{N}', \\ \mathscr{N} & \text{otherwise.} \end{cases}$,

- $\mathscr{B}' = \begin{cases} \mathscr{B} \setminus u & \text{if } u \notin dom(\mathsf{R}') \cup \mathsf{N}', \\ \mathscr{B} & \text{otherwise.} \end{cases}$

We need to show that:

(1′) $\mathscr{R}' = (\mathsf{R}' \setminus (\mathscr{N}' \cup \mathscr{B}'))^+$;

(2′) $\mathscr{N}' \cup \mathscr{B}' = \mathsf{rcl}(\mathsf{C}', \mathscr{N}' \cup \mathscr{B}') \cup \mathsf{N}'$;

(3′) $\mathscr{B}' \setminus \mathscr{N}' \subseteq \mathsf{B}'$.

Points (1′), (2′), and (3′) imply by induction hypothesis that $\Theta \vdash P \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'$. Moreover we must show that one rule between {MCAST}, {MCASTN} and {MCASTB} can be applied to $\Theta \vdash P \blacktriangleright \mathscr{R}' ; \mathscr{N}' ; \mathscr{B}'$ giving as conclusion $\Theta \vdash \overline{u}[n](y).P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$.

Point (1′) holds by definition. Point (3′) follows from Point (3) since $\mathscr{B}' \setminus \mathscr{N}' = \mathscr{B} \setminus \mathscr{N}$ and in all cases of the definition of m but $(\mu)(\gamma)(\phi)$ we have $\mathsf{B} \subseteq \mathsf{B}'$. In case $(\mu)(\gamma)(\phi)$ we have $\mathsf{B} = \mathsf{B}' \cup \{\mathsf{a}\}$ and $a \notin dom(\mathsf{R}') \cup \mathsf{N}'$, which implies $a \notin \mathscr{B}'$. So also in this case Point (3′) holds.

To prove Point (2′) and the derivability of $\Theta \vdash \overline{u}[n](y).P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ we distinguish if u is a service name or a variable. For u service name we consider different cases according to

—how R is obtained from R′ by rule {MCAST-I};

—whether u is an element of $\mathscr{R}$, $\mathscr{N}$ or $\mathscr{B} \setminus \mathscr{N}$.

<u>u is a service name</u>. Let $u = a$. To show derivability of $\Theta \vdash \overline{u}[n](y).P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$ and Point (2′) we distinguish some subcases.

$\mathsf{R} = \mathsf{R}'\{a/y\}$ or $\mathsf{R} = \mathsf{R}'\{a/y\} \cup \{a\}$. Note that these two cases are different only when both y and a do not occur in R′. This case corresponds to cases (μ) and (ρ) of the definition of m. In both cases $a \notin \mathsf{N}'$: this holds by hypothesis in case (μ) and it follows from $a \in dom(\mathsf{R}')$ and Lemma 6.2(1) in case (ρ). Moreover we get $\mathsf{N} = \mathsf{N}'$ by definition of m, which implies $a \notin \mathsf{N}$. We distinguish the following cases.

$a \in \mathscr{R}$. We can only apply rule {MCAST} to obtain $\Theta \vdash \overline{u}[n](y).P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}$; this is possible being $\mathscr{R}$ loop free by hypothesis. For Point (2′) from Point (1) we get $a \notin \mathscr{N} \cup \mathscr{B}$ and then $\mathscr{N}' = \mathscr{N}$ and $\mathscr{B}' = \mathscr{B}$ by definition. We distinguish some further subcases according to the definition of C.

$\mathsf{C} = \mathsf{C}' \uplus \{(a, \mathsf{R}' \downarrow y)\}$. We are in cases $(\mu)(\gamma)$ and $(\rho)(\alpha)(\gamma)$. We get $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B}) = \mathsf{rcl}(\mathsf{C}', \mathscr{N}' \cup \mathscr{B}')$, since $a \notin \mathscr{N} \cup \mathscr{B} = \mathscr{N}' \cup \mathscr{B}'$. Since $\mathsf{N} = \mathsf{N}'$, Point (2′) follows from Point (2).

$\mathsf{C} = \mathsf{C}' \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}', a)$. We are in case $(\rho)(\alpha)(\overline{\gamma})$. We get $\mathsf{rcl}^{\Uparrow}(\mathsf{C}', a) \cap dom(\mathsf{C}) = \emptyset$ by definition of C. Moreover $\mathsf{rcl}^{\Uparrow}(\mathsf{C}', a) \cap \mathsf{N}' = \emptyset$ by Lemma 6.2(1) and (2), being $\mathsf{rcl}^{\Uparrow}(\mathsf{C}', a) \subseteq dom(\mathsf{C}')$ by Lemma 6.2(7). Point (2) implies $\mathscr{N} \cup \mathscr{B} \subseteq dom(\mathsf{C}) \cup \mathsf{N}$, so we get $\mathscr{N}' \cup \mathscr{B}' \subseteq dom(\mathsf{C}) \cup \mathsf{N}'$ since $\mathscr{N} \cup \mathscr{B} = \mathscr{N}' \cup \mathscr{B}'$ and we conclude that $\mathsf{rcl}^{\Uparrow}(\mathsf{C}', a) \cap (\mathscr{N}' \cup \mathscr{B}') = \emptyset$. This implies $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) = \mathsf{rcl}(\mathsf{C}', \mathscr{N}' \cup \mathscr{B}')$ and Point (2′) follows from Point (2), being $\mathsf{N} = \mathsf{N}'$.

$\mathsf{C} = \mathsf{C}'$. We are in cases $(\mu)(\overline{\gamma})$ and $(\rho)(\overline{\alpha})$. Point (2′) coincides with Point (2).

$a \in \mathscr{N}$. By Point (2) $\mathsf{rcl}(\mathsf{C}, a) \subseteq \mathscr{N} \cup \mathscr{B}$, which implies $a \in dom(\mathsf{C})$, since $a \notin \mathsf{N}$. From $a \in dom(\mathsf{C})$ we can exclude the following cases of the definition of m:

- case $(\mu)(\overline{\gamma})$, since $a \notin dom(\mathsf{R}') \cup \mathsf{N}'$ and $\mathsf{C} = \mathsf{C}'$;
- case $(\rho)(\alpha)(\overline{\gamma})$, since $a \in dom(\mathsf{C}')$ and $\mathsf{C} = \mathsf{C}' \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}', a)$;

- case $(\rho)(\overline{\alpha})$, since $a \notin dom(\mathsf{C}')$ and $\mathsf{C} = \mathsf{C}'$.

Therefore we can only be in cases $(\mu)(\gamma)$ or $(\rho)(\alpha)(\gamma)$ of the definition of m. In both cases condition (γ) holds, i.e. $\mathsf{R}' \downarrow y \subseteq dom(\mathsf{C}')$, and $\mathsf{C} = \mathsf{C}' \uplus \{(a, \mathsf{R}' \downarrow y)\}$. Since $a \in \mathscr{N}$ and $\mathsf{R}' \downarrow y \subseteq \mathsf{C}(a)$ by definition of C, we get by Point (2)

$$\mathsf{R}' \downarrow y \subseteq \mathscr{N} \cup \mathscr{B} \qquad (\#)$$

In case $(\mu)(\gamma)$ we have $\mathscr{N}' \cup \mathscr{B}' \cup \{a\} = \mathscr{N} \cup \mathscr{B}$ by definition of $\mathscr{N}', \mathscr{B}'$. Moreover condition (μ) implies $a \notin \mathsf{R}' \downarrow y$ (since $a \notin dom(\mathsf{R}')$ is part of condition (μ)). Then (#) implies $\mathsf{R}' \downarrow y \subseteq \mathscr{N}' \cup \mathscr{B}'$. So $\mathsf{rcl}(\mathsf{C}', \mathscr{N}' \cup \mathscr{B}') \cup \{a\} = \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$ by definition of C and we get from Point (2) $\mathsf{rcl}(\mathsf{C}', \mathscr{N}' \cup \mathscr{B}') \cup \mathsf{N}' = \mathscr{N}' \cup \mathscr{B}'$, i.e. Point $(2')$.
In case $(\rho)(\alpha)(\gamma)$, condition (ρ) is $a \in dom(\mathsf{R}')$ and then $\mathscr{N} \cup \mathscr{B} = \mathscr{N}' \cup \mathscr{B}'$ by definition of $\mathscr{N}', \mathscr{B}'$. We get $\mathsf{R}' \downarrow y \subseteq \mathscr{N}' \cup \mathscr{B}'$ from (#), which implies $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) = \mathsf{rcl}(\mathsf{C}', \mathscr{N}' \cup \mathscr{B}')$ by definition of C. By Point (2) then we get immediately Point $(2')$.
In both cases, from $\mathsf{R}' \downarrow y \subseteq \mathscr{N}' \cup \mathscr{B}'$, Point $(2')$ and Lemma 6.2(1) we get $\mathsf{R}' \downarrow y \subseteq dom(\mathsf{C}')$. This implies y is minimal in $(\mathsf{R}' \setminus \mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow y))^+$ by Lemma 6.3. From $\mathsf{R}' \downarrow y \subseteq \mathscr{N}' \cup \mathscr{B}'$, Points $(1')$ and $(2')$ we have that y is minimal in $\mathscr{R}'$ and then we can apply rule {MCastN}.

$a \in \mathscr{B} \setminus \mathscr{N}$. As in previous case we can exclude cases $(\mu)(\overline{\gamma})$, $(\rho)(\alpha)(\overline{\gamma})$, and $(\rho)(\overline{\alpha})$ of the definition of m. Point $(2')$ is proved reasoning as in the previous case, and this implies that y is minimal in $\mathscr{R}'$. By Point (3) we get $a \in \mathsf{B}$ and this implies we are in case $(\mu)(\gamma)(\phi)$ or $(\rho)(\alpha)(\gamma)(\phi)$ of the definition of m. Condition (ϕ) is $\mathsf{cf}(\mathsf{R}'\{a/y\})$ and this implies $\mathsf{cf}(\mathscr{R}' \setminus\setminus y)$ by Point $(1')$. So rule {MCastB} can be applied.

$\mathsf{R} = \mathsf{R}' \setminus (\mathsf{rcl}(\mathsf{C}', (\mathsf{R}' \downarrow y)) \cup \{y\})$. This correspond only to case (ν) of the definition of m. The condition $a \in \mathsf{N}$ implies $a \in \mathscr{N} \cup \mathscr{B}$ by Point (2). Moreover we have by definition $\mathscr{N} = \mathscr{N}'$ and $\mathscr{B} = \mathscr{B}'$. We distinguish two subcases:

$a \in \mathscr{N}$. We have $\mathsf{C} = \mathsf{C}' \setminus \mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow y)$ and $\mathsf{N} = \mathsf{N}' \cup \mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow \mathsf{y})$. By Point (2) $\mathsf{N} \subseteq \mathscr{N} \cup \mathscr{B}$, and this implies $\mathsf{R}' \downarrow y \subseteq \mathscr{N} \cup \mathscr{B}$ by definitions of N and rcl, being by hypothesis $\mathsf{R}' \downarrow y \subseteq dom(\mathsf{C}')$. By Lemma 6.1 we get $\mathsf{rcl}(\mathsf{C}' \setminus \mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow y), \mathscr{N} \cup \mathscr{B}) \cup \mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow y) = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B})$. i.e. $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) \cup \mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow y) = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B})$, which implies $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) \cup \mathsf{N} = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B}) \cup \mathsf{N}'$ by definitions of C and N. So Point (2) implies Point $(2')$.
The proof of y minimal in $\mathscr{R}'$ is as in case $\mathsf{R} = \mathsf{R}'\{a/y\}$ or $\mathsf{R} = \mathsf{R}'\{a/y\} \cup \{a\}$ and $a \in \mathscr{N}$. So rule {MCastN} can be applied to obtain the typing in the progress system.

$a \in \mathscr{B} \setminus \mathscr{N}$. The proofs of Point $(2')$ and of y minimal in $\mathscr{R}'$ are as in previous case. By Point (3) we get $a \in \mathsf{B}$ and this implies we are in case $(\nu)(\phi)$ of the definition of m. Condition (ϕ), i.e. $\mathsf{cf}(\mathsf{R}'\{a/y\})$, implies $\mathsf{cf}(\mathscr{R}'\{a/y\})$ by Point $(1')$. So we can apply rule {MCastB}.

<u>u is a variable</u>. The proof is similar to that of case $u = a$ and $\mathsf{R} = \mathsf{R}' \setminus (\mathsf{rcl}(\mathsf{C}', \mathsf{R}' \downarrow y) \cup \{y\})$.

Rule $\{\text{CONC-I}\}$.

$$\frac{\Theta\,;P_1 \mapsto \mathsf{R}_1\,;\,\mathsf{C}_1\,;\,\mathsf{N}_1\,;\,\mathsf{B}_1 \qquad \Theta\,;P_2 \mapsto \mathsf{R}_2\,;\,\mathsf{C}_2\,;\,\mathsf{N}_2\,;\,\mathsf{B}_2 \qquad \mathsf{K}\cap\mathsf{N}=\emptyset}{\Theta\,;P_1 \mid P_2 \mapsto \mathsf{R}\,;\,\mathsf{C}\,;\,\mathsf{N}\,;\,\mathsf{B}}\ \{\text{CONC-I}\}$$

where

$$\begin{aligned}
&\mathsf{D}_i = dom(\mathsf{R}_i)\setminus dom(\mathsf{C}_i) \qquad i\in\{1,2\}\\
&\mathsf{N} = \mathsf{N}_1\cup\mathsf{N}_2\cup\mathsf{cl}(\mathsf{C}_1\uplus\mathsf{C}_2,\mathsf{N}_1\cup\mathsf{N}_2)\\
&\mathsf{K} = \mathsf{rcl}^{\Uparrow}(\mathsf{C}_1\uplus\mathsf{C}_2,\mathsf{D}_1\cup\mathsf{D}_2)\\
&\mathsf{R} = (\textstyle\bigcup_{i=1,2}\mathsf{R}_i)\setminus\mathsf{N}\\
&\mathsf{C} = (\mathsf{C}_1\uplus\mathsf{C}_2)\setminus(\mathsf{N}\cup\mathsf{K})\\
&\mathsf{B} = (\mathsf{B}_1\cap\mathsf{B}_2)\cup(\textstyle\bigcup_{i=1,2}\{a\in\mathsf{B}_i \mid a\notin dom(\mathsf{R}_{\bar{i}})\cup\mathsf{N}_{\bar{i}}\})
\end{aligned}$$

We define:
- $\mathscr{R}_i = (\mathsf{R}_i\setminus(\mathscr{N}\cup\mathscr{B}))^+$,
- $\mathscr{N}_i = \mathscr{N}\cap(dom(\mathsf{C}_i)\cup\mathsf{N}_i)$,
- $\mathscr{B}_i = \mathscr{B}\cap(dom(\mathsf{C}_i)\cup\mathsf{N}_i)$.

We need to show that rule $\{\text{CONC}\}$ can be applied to $\Theta\vdash P_1 \blacktriangleright \mathscr{R}_1\,;\,\mathscr{N}_1\,;\,\mathscr{B}_1$ and $\Theta\vdash P_2 \blacktriangleright \mathscr{R}_2\,;\,\mathscr{N}_2\,;\,\mathscr{B}_2$, giving as result $\Theta\vdash P_1\mid P_2 \blacktriangleright \mathscr{R}\,;\,\mathscr{N}\,;\,\mathscr{B}$, i.e. (being $\mathscr{R}$ loop free by hypothesis) that:

(a) $\mathscr{R} = (\mathscr{R}_1\cup\mathscr{R}_2)^+$;

(b) $\mathscr{N} = \mathscr{N}_1\cup\mathscr{N}_2$;

(c) $\mathscr{B} = \mathscr{B}_1\cup\mathscr{B}_2$;

(d) $\mathscr{R}_i\cap(\mathscr{N}_{\bar{i}}\cup\mathscr{B}_{\bar{i}}) = \emptyset$;

and moreover that:

(1′) $\mathscr{R}_i = (\mathsf{R}_i\setminus(\mathscr{N}_i\cup\mathscr{B}_i))^+$;

(2′) $\mathscr{N}_i\cup\mathscr{B}_i = \mathsf{rcl}(\mathsf{C}_i,\mathscr{N}_i\cup\mathscr{B}_i)\cup\mathsf{N}_i$;

(3′) $\mathscr{B}_i\setminus\mathscr{N}_i\subseteq\mathsf{B}_i$.

We first show that:

$$\mathsf{K}\cap(\mathscr{N}\cup\mathscr{B}) = \emptyset \qquad (\flat)$$

By definition of C we get $\mathsf{K}\cap dom(\mathsf{C}) = \emptyset$. By applicability of rule $\{\text{CONC-I}\}$ we get $\mathsf{K}\cap\mathsf{N}=\emptyset$. By lemma 6.2(6) we have $\mathsf{rcl}(\mathsf{C},\mathscr{N}\cup\mathscr{B})\subseteq dom(\mathsf{C})$, which gives by Point (2) $\mathscr{N}\cup\mathscr{B}\subseteq dom(\mathsf{C})\cup\mathsf{N}$.

—Proof of Point (a).

$$\begin{aligned}
\mathscr{R} &= (\mathsf{R} \setminus (\mathscr{N} \cup \mathscr{B}))^+ \\
&\quad \text{by Point (1)} \\
&= ((\mathsf{R}_1 \cup \mathsf{R}_2) \setminus \mathsf{N}) \setminus (\mathscr{N} \cup \mathscr{B}))^+ \\
&\quad \text{by definition of } \mathsf{R} \\
&= ((\mathsf{R}_1 \cup \mathsf{R}_2) \setminus (\mathscr{N} \cup \mathscr{B}))^+ \\
&\quad \text{since } \mathsf{N} \subseteq \mathscr{N} \cup \mathscr{B} \text{ by Point (2)} \\
&= ((\mathsf{R}_1 \setminus (\mathscr{N} \cup \mathscr{B})) \cup (\mathsf{R}_2 \setminus (\mathscr{N} \cup \mathscr{B})))^+ \\
&= (\mathscr{R}_1 \cup \mathscr{R}_2)^+.
\end{aligned}$$

—Proof of Point (b).
($\supseteq$). Immediate by definition of $\mathscr{N}_i$.
($\subseteq$). If $a \in \mathscr{N}$, then by Point (2) either $a \in \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$ or $a \in \mathsf{N}$.
By Lemma 6.2(6) $a \in \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$ implies $a \in dom(\mathsf{C})$, and this gives $a \in dom(\mathsf{C}_i)$ for $i = 1$ or $i = 2$ by definition of C. From $a \in \mathscr{N} \cap dom(\mathsf{C}_i)$ we conclude $a \in \mathscr{N}_i$ by definition of $\mathscr{N}_i$.
If $a \in \mathsf{N}$, then either $a \in \mathsf{cl}(\mathsf{C}_1 \uplus \mathsf{C}_2, \mathsf{N}_1 \cup \mathsf{N}_2)$ or $a \in \mathsf{N_i}$ for $i = 1$ or $i = 2$ by definition of N. In the first case $a \in dom(\mathsf{C}_i)$ by Lemma 6.2(5) and we can conclude as before. In the second case we have $a \in \mathscr{N} \cap \mathsf{N_i}$ and we conclude $a \in \mathscr{N}_i$ by definition of $\mathscr{N}_i$.

—Proof of Point (c). Similar to the proof of Point (b).

—Proof of Point (d). By definition of $\mathscr{R}_i$ and Points (b) and (c).

—Proof of Point (1′).
($\subseteq$). Immediate by definition of $\mathscr{R}_i$ and Points (b) and (c) since $\mathscr{N}_i \cup \mathscr{B}_i \subseteq \mathscr{N} \cup \mathscr{B}$.
($\supseteq$). We start by proving

$$\mathsf{R}_i \setminus dom(\mathsf{C}_i) \subseteq \mathsf{R}_i \setminus (\mathscr{N} \cup \mathscr{B}) \qquad (\star)$$

By definition $dom(\mathsf{R}_i) \setminus dom(\mathsf{C}_i) = \mathsf{D}_i \subseteq \mathsf{K}$ and this implies $\mathsf{D}_i \cap (\mathscr{N} \cup \mathscr{B}) = \emptyset$ by ($\flat$).
We get then $\mathsf{D}_i \subseteq dom(\mathsf{R}_i) \setminus (\mathscr{N} \cup \mathscr{B})$, which in turn implies ($\star$).

$$\begin{aligned}
\mathsf{R}_i \setminus (\mathscr{N}_i \cup \mathscr{B}_i) &= \mathsf{R}_i \setminus ((\mathscr{N} \cup \mathscr{B}) \cap (dom(\mathsf{C}_i) \cup \mathsf{N_i})) && \text{by definition of } \mathscr{N}_i, \mathscr{B}_i \text{ and distributivity} \\
&= \mathsf{R}_i \setminus ((\mathscr{N} \cup \mathscr{B}) \cap dom(\mathsf{C}_i)) && \text{since } dom(\mathsf{R}_i) \cap \mathsf{N_i} = \emptyset \text{ by Lemma 6.2(1)} \\
&= (\mathsf{R}_i \setminus (\mathscr{N} \cup \mathscr{B})) \cup (\mathsf{R}_i \setminus dom(\mathsf{C}_i)) \\
&\subseteq \mathsf{R}_i \setminus (\mathscr{N} \cup \mathscr{B}) && \text{by } (\star).
\end{aligned}$$

—Proof of Point (2′).
($\subseteq$). If $a \in \mathscr{N}_i \cup \mathscr{B}_i$, then $a \in \mathscr{N} \cup \mathscr{B}$ and either $a \in dom(\mathsf{C}_i)$ or $a \in \mathsf{N_i}$ by definition of $\mathscr{N}_i$ and $\mathscr{B}_i$. From $a \in \mathscr{N}_i \cup \mathscr{B}_i$ and $a \in dom(\mathsf{C}_i)$ we get $a \in \mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i)$ by definition of rcl and this concludes the proof.
($\supseteq$). If $a \in \mathsf{N_i}$, then $a \in \mathsf{N}$ by definition of N, and this implies $a \in \mathscr{N} \cup \mathscr{B}$ by Point (2). From $a \in (\mathscr{N} \cup \mathscr{B}) \cap \mathsf{N_i}$ we conclude $a \in \mathscr{N}_i \cup \mathscr{B}_i$ by definition of $\mathscr{N}_i$ and $\mathscr{B}_i$.
If $a \in \mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i)$, then $a \in dom(\mathsf{C}_i)$ by Lemma 6.2(6). We prove

$$\mathsf{rcl}(\mathsf{C}_i, \mathscr{N} \cup \mathscr{B}) \subseteq \mathscr{N} \cup \mathscr{B} \qquad (\natural)$$

by induction on the definition of rcl. Since $\mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i) \subseteq \mathsf{rcl}(\mathsf{C}_i, \mathscr{N} \cup \mathscr{B})$ ($\natural$) implies $\mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i) \subseteq \mathscr{N} \cup \mathscr{B}$. Then we have $a \in (\mathscr{N} \cup \mathscr{B}) \cap dom(\mathsf{C}_i)$ and we conclude $a \in \mathscr{N}_i \cup \mathscr{B}_i$ by definition of $\mathscr{N}_i$ and $\mathscr{B}_i$.

For the induction step of the proof of ($\natural$) let $b \in \mathsf{C}_i(a)$ and $a \in \mathscr{N} \cup \mathscr{B}$. By definition of C we get either $b \in \mathsf{C}(a)$, or $a \in \mathsf{N} \cup \mathsf{K}$.
If $b \in \mathsf{C}(a)$, then $b \in \mathscr{N} \cup \mathscr{B}$ by Point (2).
If $a \in \mathsf{N}$, then either $a \in \mathsf{N}_1 \cup \mathsf{N}_2$ or $a \in \mathsf{cl}(\mathsf{C}_1 \uplus \mathsf{C}_2, \mathsf{N}_1 \cup \mathsf{N}_2)$ by definition of N. In both cases $b \in \mathsf{rcl}(\mathsf{C}_1 \uplus \mathsf{C}_2, \mathsf{N}_1 \cup \mathsf{N}_2)$ by definition of rcl. This implies $b \in \mathsf{N}$ again by definition of N, so we conclude $b \in \mathscr{N} \cup \mathscr{B}$ by Point (2).
If $a \in \mathscr{N} \cup \mathscr{B}$, then $a \notin \mathsf{K}$ by ($\flat$).

—Proof of Point ($3'$).

$$\begin{array}{rcl}
\mathscr{B}_i \setminus \mathscr{N}_i & = & (\mathscr{B} \cap (dom(\mathsf{C}_i) \cup \mathsf{N}_i)) \setminus (\mathscr{N} \cap (\mathsf{dom}(\mathsf{C}_i) \cup \mathsf{N}_i)) \\
& & \text{by definition} \\
& = & (\mathscr{B} \setminus \mathscr{N}) \cap (dom(\mathsf{C}_i) \cup \mathsf{N}_i) \\
& \subseteq & \mathsf{B} \cap (\mathsf{dom}(\mathsf{C}_i) \cup \mathsf{N}_i) \\
& & \text{by Point (3)} \\
& \subseteq & \mathsf{B} \cap (\mathsf{dom}(\mathsf{R}_i) \cup \mathsf{N}_i) \\
& & \text{by Lemma 6.2(2)} \\
& = & [(\mathsf{B}_1 \cap \mathsf{B}_2) \cup (\bigcup_{j=1,2}\{\mathsf{a} \in \mathsf{B}_j \mid \mathsf{a} \notin \mathsf{dom}(\mathsf{R}_{\bar{j}}) \cup \mathsf{N}_{\bar{j}}\})] \cap (\mathsf{dom}(\mathsf{R}_i) \cup \mathsf{N}_i) \\
& & \text{by definition of B} \\
& \subseteq & \mathsf{B}_i \cup \{\mathsf{a} \in \mathsf{B}_i \mid \mathsf{a} \notin \mathsf{dom}(\mathsf{R}_{\bar{i}}) \cup \mathsf{N}_{\bar{i}}\}) \\
& = & \mathsf{B}_i.
\end{array}$$

□

Take for instance the process P_3 defined above. A possible choices for $\mathscr{R}$, $\mathscr{N}$, $\mathscr{B}$ is: $\mathscr{R}_1 = \emptyset$, $\mathscr{N}_1 = \{a, b\}$, $\mathscr{B}_1 = \emptyset$. However also the choice $\mathscr{R}_2 = \{b\}$, $\mathscr{N}_2 = \{a\}$, $\mathscr{B}_2 = \emptyset$ represent a valid typing for P_3 in the interaction system

In a valid deduction $(\mathsf{R} \setminus (\mathscr{N} \cup \mathscr{B}))^+$ is always loop free so if for some process P we can deduce $\Theta \ ; \ P \mapsto \ \mathsf{R} \ ; \mathsf{C}; \ \mathsf{N}; \ \mathsf{B}$, a safe choice of $\mathscr{R}$, $\mathscr{N}$, $\mathscr{B}$ is to take $\mathscr{R} = (\mathsf{R} \setminus dom(\mathsf{C}))^+$ $\mathscr{N} = \mathsf{N} \cup \mathsf{dom}(\mathsf{C})$ and $\mathscr{B} = \mathsf{B}$.

Note the use of C. Looking only at the set R we see a loop $a \prec a$ which would forbid to type the process. However in C we recover the information that at the point in which a was closed it was minimal in the order, and then it was suitable to be closed with a nesting rule. The service b, on the contrary, can be closed both with a nesting rule and with $\{$MCast$\}$, $\{$MAcc$\}$. This could be relevant if P_3 is a subterm of a bigger process: our type analysis is fully compositional. Indeed the inference rules fully characterize all the possible typings of a process in the interaction type system.Mariangiola says: modified in accordance with the new example

Theorem 6.5 Completeness. *Let* $\Theta \vdash P \blacktriangleright \mathscr{R}\, ; \mathscr{N}\, ; \mathscr{B}$. *Then* $\Theta \ ; \ P \mapsto \mathsf{R}\, ; \mathsf{C}\, ; \mathsf{N}\, ; \mathsf{B}$ *and:*

(1) $\mathscr{R} = (\mathsf{R} \setminus (\mathscr{N} \cup \mathscr{B}))^+$;
(2) $\mathscr{N} \cup \mathscr{B} = \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) \cup \mathsf{N}$;
(3) $\mathscr{B} \setminus \mathscr{N} \subseteq \mathsf{B}$.

Proof. By induction on the deduction of $\Theta \vdash P \ \blacktriangleright \ \mathscr{R}\, ; \ \mathscr{N}\, ; \ \mathscr{B}$. We show the most interesting cases.

Rule $\{$MCast$\}$.

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash \overline{a}[n](y).P \blacktriangleright \mathscr{R}\{a/y\} \cup \{a\} ; \mathscr{N} ; \mathscr{B}} \;\{\text{MCAST}\}$$

By induction hypothesis $\Theta \;;\; P \mapsto \mathsf{R} \,;\, \mathsf{C};\, \mathsf{N};\, \mathsf{B}$ where Points (1), (2) and (3) hold. We need to show that rule {MCAST-I} can be applied to $\Theta \;;\; P \mapsto \mathsf{R} \,;\, \mathsf{C};\, \mathsf{N};\, \mathsf{B}$ getting $\Theta \;;\; \overline{a}[n](y).P \mapsto \mathsf{R}' \,;\, \mathsf{C}';\, \mathsf{N}';\, \mathsf{B}'$ and that:

(1′) $\mathscr{R}\{a/y\} \cup \{a\} = (\mathsf{R}' \setminus (\mathscr{N} \cup \mathscr{B}))^+$;
(2′) $\mathscr{N} \cup \mathscr{B} = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B}) \cup \mathsf{N}'$;
(3′) $\mathscr{B} \setminus \mathscr{N} \subseteq \mathsf{B}'$.

Applicability of rule {MCAST} requires $a \notin \mathscr{N} \cup \mathscr{B}$. From $a \notin \mathscr{N} \cup \mathscr{B}$ we get $a \notin \mathsf{N}$ by Point (2). Therefore we are in cases (μ) or (ρ) of the definition of m. Since in these cases either $\mathsf{R}' = \mathsf{R}\{a/y\}$ and $a \in \mathsf{R}\{a/y\}$ or $\mathsf{R} = \mathsf{R}'\{a/y\} \cup \{a\}$, from Point (1) we get $\mathscr{R}\{a/y\} \cup \{a\} = (\mathsf{R}' \setminus (\mathscr{N} \cup \mathscr{B}))^+$, i.e. Point (1′).

In all cases of the definition of m but $(\mu)(\gamma)(\phi)$ and $(\rho)(\alpha)(\overline{\gamma})$ we have either $\mathsf{B}' = \mathsf{B}$ or $\mathsf{B}' = \mathsf{B} \cup \{\mathsf{a}\}$, so from Point (3) we get $\mathscr{B} \setminus \mathscr{N} \subseteq \mathsf{B}'$, i.e. Point (3′). In case $(\mu)(\gamma)(\phi)$ we have $\mathsf{B}' = \mathsf{B} \setminus \{\mathsf{a}\}$; being $a \notin \mathscr{B}$, Point (3) implies Point (3′). In case $(\rho)(\alpha)(\overline{\gamma})$ we have $\mathsf{B}' = \mathsf{B} \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}, \mathsf{a})$: since $a \notin \mathscr{N} \cup \mathscr{B}$ implies $\mathsf{rcl}^{\Uparrow}(\mathsf{C}, a) \cap (\mathscr{N} \cup \mathscr{B}) = \emptyset$ by Point (2), Point (3) implies Point (3′).

To show applicability of rule {MCAST-I} and Point (2′) we consider some subcases.

$\underline{a \notin \mathscr{R} \text{ or } a \in \mathscr{R} \text{ and } \mathsf{R} \downarrow y \cup \{a\} \subseteq dom(\mathsf{C})}$. We are respectively in cases (μ) or $(\rho)(\alpha)(\gamma)$ of the definition of m, so rule {MCAST-I} can be applied. In both cases C' differs from C only in the value at a and, since $a \notin \mathscr{N} \cup \mathscr{B}$, Point (2) implies that $a \notin \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$. Then $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B})$, so Point (2′) follows immediately by Point (2).

$\underline{a \in \mathscr{R},\ a \in dom(\mathsf{C}),\ \mathsf{R} \downarrow y \not\subseteq dom(\mathsf{C})}$. We are in case $(\rho)(\alpha)(\overline{\gamma})$ where $\mathsf{C}' = \mathsf{C} \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}, a)$. From $a \notin \mathscr{N} \cup \mathscr{B}$ we have $\mathsf{rcl}^{\Uparrow}(\mathsf{C}, a) \cap (\mathscr{N} \cup \mathscr{B}) = \emptyset$ by Point (2) and this implies $\mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B}) = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B})$ showing, as in the previous case, Point (2′). By applicability of {MCAST} we get $\mathscr{R}\{a/y\}$ loop free, which implies $(\mathsf{R}\{a/y\} \setminus (\mathscr{N} \cup \mathscr{B}))^+$ loop free by Point (1′). We have

$$\begin{array}{rcll}
\mathsf{R}\{a/y\} \setminus dom(\mathsf{C}') & = & \mathsf{R}\{a/y\} \setminus (dom(\mathsf{C}') \cup \mathsf{N}) & \text{by Lemma 6.2(1)} \\
& \subseteq & \mathsf{R}\{a/y\} \setminus (\mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B}) \cup \mathsf{N}) & \text{by Lemma 6.2(6)} \\
& = & \mathsf{R}\{a/y\} \setminus (\mathscr{N} \cup \mathscr{B}) & \text{by Point (2′).}
\end{array}$$

This implies that $(\mathsf{R}\{a/y\} \setminus dom(\mathsf{C} \setminus \mathsf{rcl}^{\Uparrow}(\mathsf{C}, a)))^+$ is loop free, justifying the applicability of rule {MCAST-I} in this case.

$\underline{a \in \mathscr{R} \text{ and } a \notin dom(\mathsf{C})}$. We are in case $(\rho)(\overline{\alpha})$. Point (2′) is straightforward since $\mathsf{C}' = \mathsf{C}$ and $\mathsf{N} = \mathsf{N}'$. As in previous case we have that $(\mathsf{R}\{a/y\} \setminus (\mathscr{N} \cup \mathscr{B}))^+$ is loop free and we can show $\mathsf{R}\{a/y\} \setminus dom(\mathsf{C}') \subseteq \mathsf{R}\{a/y\} \setminus (\mathscr{N} \cup \mathscr{B})$. So we conclude that $(\mathsf{R}\{a/y\} \setminus dom(\mathsf{C}))^+$ is loop free and then rule {MCAST-I} can be applied.

Rule {MCASTN}.

$$\frac{\Theta \vdash P \blacktriangleright \mathscr{R} ; \mathscr{N} ; \mathscr{B}}{\Theta \vdash \overline{a}[n](y).P \blacktriangleright \mathscr{R} \backslash\backslash y ;\ \mathscr{N} \cup \{a\} ;\ \mathscr{B}} \;\{\text{MCASTN}\}$$

By induction hypothesis $\Theta\ ;\ P \mapsto\ \mathsf{R}\ ;\ \mathsf{C};\ \mathsf{N};\ \mathsf{B}$ and Points (1), (2) and (3) hold. From y minimal in $\mathscr{R}$ and Point (1) we get $\mathsf{R}\downarrow y \subseteq \mathscr{N}\cup\mathscr{B}$, which gives the condition $\mathsf{R}\downarrow y \subseteq dom(\mathsf{C})$ (i.e. condition (γ) of the definition of m) by Point (2) and Lemma 6.2(1). Applicability of rule $\{$MCastN$\}$ requires $a \notin \mathscr{R}$. From $a \notin \mathscr{R}$ by Point (1) we get either $a \notin dom(\mathsf{R})$ or $a \in \mathscr{N}\cup\mathscr{B}$, which imply either $a \notin dom(\mathsf{R})\cup\mathsf{N}$ or $a\in\mathsf{N}$ or $a \in dom(\mathsf{C})$ by Point (2). Therefore rule $\{$MCast-I$\}$ can be applied getting $\Theta\ ;\ \overline{a}[n](y).P \mapsto \mathsf{R}'\ ;\ \mathsf{C}';\ \mathsf{N}';\ \mathsf{B}'$ and we are in cases (μ)(γ) or (ν) or (ρ)(α)(γ) of the definition of m.

We need to show:

(1′) $\mathscr{R}\backslash\backslash y = (\mathsf{R}'\setminus(\mathscr{N}\cup\{a\}\cup\mathscr{B}))^{+}$;
(2′) $\mathscr{N}\cup\{a\}\cup\mathscr{B} = \mathsf{rcl}(\mathsf{C}',\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{N}'$;
(3′) $\mathscr{B}\setminus(\mathscr{N}\cup\{a\}) \subseteq \mathsf{B}'$.

In cases (μ)(γ) or (ρ)(α)(γ) we have either $\mathsf{R}' = \mathsf{R}\{a/y\}$ or $\mathsf{R}' = \mathsf{R}\{a/y\}\cup\{a\}$, $\mathsf{C}' = \mathsf{C}\uplus\{\overline{(a,\mathsf{R}\downarrow y)}\}$, $\mathsf{N}'=\mathsf{N}$, and $\mathsf{B}'=\mathsf{B}$, or $\mathsf{B}'=\mathsf{B}\cup\{\mathsf{a}\}$, or $\mathsf{B}'=\mathsf{B}\setminus\{\mathsf{a}\}$. We get immediately $\mathscr{R}\backslash\backslash y = (\mathsf{R}'\setminus(\mathscr{N}\cup\{a\}\cup\mathscr{B}))^{+}$, i.e. Point (1′), by Point (1).

We get $\mathscr{N}\cup\{a\}\cup\mathscr{B} \subseteq \mathsf{rcl}(\mathsf{C}',\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{N}$ by Point (2), since $a\in dom(\mathsf{C}')$ by definition of C'. Since $a\notin\mathscr{R}$ by applicability of rule $\{$MCastN$\}$ we have by Point (1) either $a\in\mathscr{N}\cup\mathscr{B}$ or $a\notin dom(\mathsf{R})$, which implies $a\notin dom(\mathsf{C})$ by Lemma 6.2(2). In both cases $\mathsf{C}(a)\subseteq\mathscr{N}\cup\mathscr{B}$, in the first case by Point (2) and in the second case since $\mathsf{C}(a)=\emptyset$. Then we get $\mathscr{N}\cup\{a\}\cup\mathscr{B}$=$\mathsf{rcl}(\mathsf{C}',\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{N}$, i.e. Point (2’), by Point (2) and definition of C', being $\mathsf{R}\downarrow y\subseteq\mathscr{N}\cup\mathscr{B}$ and $\mathsf{C}(a)\subseteq\mathscr{N}\cup\mathscr{B}$. We conclude $\mathscr{N}\cup\{a\}\cup\mathscr{B} = \mathsf{rcl}(\mathsf{C}',\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{N}$, i.e. Point (2′).
From Point (3) and the definition of B' we get $\mathscr{B}\setminus(\mathscr{N}\cup\{a\})\subseteq\mathsf{B}'$, i.e. Point (3′).

In case (ν) we have $\mathsf{R}'=\mathsf{R}\setminus(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y)\cup\{y\})$, $\mathsf{C}'=\mathsf{C}\setminus\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y)$, $\mathsf{N}'=\mathsf{N}\cup \mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow \mathsf{y})$ and either $\mathsf{B}'=\mathsf{B}$ or $\mathsf{B}'=\mathsf{B}\setminus\{\mathsf{a}\}$. Note that $a\in\mathsf{N}$ implies $a\notin dom(\mathsf{R})$ by Lemma 6.2(1), and then $a\notin dom(\mathsf{R}')$ by definition. From $\mathsf{R}\downarrow y\subseteq\mathscr{N}\cup\mathscr{B}$ we get $\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y)\subseteq(\mathscr{N}\cup\mathscr{B})$ by Point (2). Therefore from Point (1) and $a\notin dom(\mathsf{R}')$ we have $\mathscr{R}\backslash\backslash y = (\mathsf{R}'\setminus(\mathscr{N}\cup\{a\}\cup\mathscr{B}))^{+}$, i.e. Point (1′).

Since $\mathsf{R}\downarrow y\subseteq\mathscr{N}\cup\{a\}\cup\mathscr{B}$, by Lemma 6.1 $\mathsf{rcl}(\mathsf{C}\setminus\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y),\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y) = \mathsf{rcl}(\mathsf{C},\mathscr{N}\cup\{a\}\cup\mathscr{B})$, i.e. $\mathsf{rcl}(\mathsf{C}',\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y) = \mathsf{rcl}(\mathsf{C},\mathscr{N}\cup\{a\}\cup\mathscr{B})$, which implies $\mathsf{rcl}(\mathsf{C}',\mathscr{N}\cup\{a\}\cup\mathscr{B})\cup\mathsf{N}' = \mathsf{rcl}(\mathsf{C},\mathscr{N}\cup\{\mathsf{a}\}\cup\mathscr{B})\cup\mathsf{N}$ by definition of N'. From $a\notin dom(\mathsf{R})$ we get $a\notin dom(\mathsf{C})$ by Lemma 6.2(2), which implies $\mathsf{rcl}(\mathsf{C},a) = \mathsf{rcl}(\mathsf{C}',a)=\emptyset$ by definition of rcl. So Point (2) implies Point (2′).

From Point (3) and the definition of B' we get $\mathscr{B}\setminus(\mathscr{N}\cup\{a\})\subseteq\mathsf{B}'$, i.e. Point (3′).

Rule $\{$MCastB$\}$.

$$\frac{\Theta\vdash P\ \blacktriangleright\ \mathscr{R};\mathscr{N};\mathscr{B}\quad \mathsf{cf}(\mathscr{R}\backslash\backslash y)}{\Theta\vdash\overline{u}[n](y).P\ \blacktriangleright\ \mathscr{R}\backslash\backslash y;\mathscr{N};\mathscr{B}\,\overline{\cup}\,\{u\}}\ \{\text{MCastB}\}$$

We only consider the case $u=a$ is a service name, the proof for u variable being similar and simpler.

Note that the conditions for applicability of rule $\{$MCastB$\}$ are all those for applicability of rule $\{$MCastN$\}$ with the additional requirement that $\mathsf{cf}(\mathscr{R}\backslash\backslash y)$ holds, which implies by Point (1) $\mathsf{cf}(\mathsf{R}\{a/y\})$, i.e. condition ($\phi$). We are so in cases ($\mu$)($\gamma$)($\phi$), ($\nu$)($\phi$) and ($\rho$)($\alpha$)($\gamma$)($\phi$). We can then prove the applicability of rule $\{$MCast-I$\}$ as in the case of rule $\{$MCastN$\}$. If the result of the application of rule $\{$MCast-I$\}$ is $\Theta\ ;\ \overline{a}[n](y).P\mapsto \mathsf{R}'\ ;\ \mathsf{C}';\ \mathsf{N}';\ \mathsf{B}'$ we need to show:

(1′) $\mathscr{R} \backslash\backslash y = (\mathsf{R}' \setminus (\mathscr{N} \cup \mathscr{B} \cup \{a\}))^+$;
(2′) $\mathscr{N} \cup \mathscr{B} \cup \{a\} = \mathsf{rcl}(\mathsf{C}', \mathscr{N} \cup \mathscr{B} \cup \{a\}) \cup \mathsf{N}'$;
(3′) $\mathscr{B} \cup \{a\} \setminus \mathscr{N} \subseteq \mathsf{B}'$.

The proof of Points (1′) and (2′) is as in the case of rule {MCASTN}.

We get $\mathscr{B} \cup \{a\} \setminus \mathscr{N} \subseteq \mathsf{B}'$, i.e. Point (3′), in case $(\mu)(\gamma)(\phi)$ since $\mathsf{B}' = \mathsf{B} \cup \{\mathsf{a}\}$ and in the other cases since $\mathsf{B}' = \mathsf{B}$ and $a \in dom(\mathsf{R}) \cup \mathsf{N}$, $a \notin \mathscr{R}$ imply $a \in \mathscr{N} \cup \mathscr{B}$.

Rule {CONC}. Let

$$\frac{\Theta \vdash P_1 \blacktriangleright \mathscr{R}_1 ; \mathscr{N}_1 ; \mathscr{B}_1 \quad \Theta \vdash P_2 \blacktriangleright \mathscr{R}_2 ; \mathscr{N}_2 ; \mathscr{B}_2}{\Theta \vdash P_1 \mid P_2 \blacktriangleright (\mathscr{R}_1 \cup \mathscr{R}_2)^+ ; \mathscr{N}_1 \cup \mathscr{N}_2 ; \mathscr{B}_1 \cup \mathscr{B}_2} \{\textsc{Conc}\}$$

be the last applied rule. By induction hypothesis for $i = 1,2$ we have $\Theta \ ; \ P_i \mapsto \mathsf{R}_i ; \mathsf{C}_i ; \mathsf{N_i} ; \mathsf{B_i}$ and:

(1′) $\mathscr{R}_i = (\mathsf{R}_i \setminus (\mathscr{N}_i \cup \mathscr{B}_i))^+$ is loop free;
(2′) $\mathscr{N}_i \cup \mathscr{B}_i = \mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i) \cup \mathsf{N_i}$;
(3′) $\mathscr{B}_i \setminus \mathscr{N}_i \subseteq \mathsf{B_i}$.

We need to prove that rule {CONC-I} can be applied to $\Theta \ ; \ P_1 \mapsto \mathsf{R}_1 ; \mathsf{C}_1 ; \mathsf{N}_1 ; \mathsf{B}_1$ and $\Theta \ ; \ P_2 \mapsto \mathsf{R}_2 ; \mathsf{C}_2 ; \mathsf{N}_2 ; \mathsf{B}_2$ getting as result $\Theta \ ; \ P_1 \mid P_2 \mapsto \mathsf{R} ; \mathsf{C} ; \mathsf{N} ; \mathsf{B}$ and that Points (1), (2) and (3) hold with $\mathscr{R} = (\mathscr{R}_1 \cup \mathscr{R}_2)^+$, $\mathscr{N} = \mathscr{N}_1 \cup \mathscr{N}_2$, and $\mathscr{B} = \mathscr{B}_1 \cup \mathscr{B}_2$.

We first show that:

(a) $\mathsf{rcl}(\mathsf{C}_1 \uplus \mathsf{C}_2, \mathscr{N} \cup \mathscr{B}) \subseteq \mathscr{N} \cup \mathscr{B}$;
(b) $\mathsf{N} \subseteq \mathscr{N} \cup \mathscr{B}$;
(c) $\mathsf{K} \subseteq dom(\mathscr{R})$.

—Proof of Point (a).
The proof is by induction on the construction of rcl. For the induction step if $a \in \mathscr{N}_i \cup \mathscr{B}_i$, then $\mathsf{C}_i(a) \subseteq \mathscr{N}_i \cup \mathscr{B}_i$ by (2′). If $a \in \mathscr{N}_{\bar{i}} \cup \mathscr{B}_{\bar{i}}$ and $\mathsf{C}_i(a) \neq \emptyset$, then $a \in dom(\mathsf{C}_i)$. Lemma 6.2(2) implies $dom(\mathsf{C}_i) \subseteq dom(\mathsf{R}_i)$, and this gives $dom(\mathsf{C}_i) \subseteq dom(\mathscr{R}_i) \cup \mathscr{N}_i \cup \mathscr{B}_i$ by (1′). We have $dom(\mathscr{R}_i) \cap (\mathscr{N}_{\bar{i}} \cup \mathscr{B}_{\bar{i}}) = \emptyset$ by applicability of rule {CONC}. This implies $a \in \mathscr{N}_i \cup \mathscr{B}_i$ and we conclude as in previous case.

—Proof of Point (b).
By definition of N, since $\mathsf{N}_1 \cup \mathsf{N}_2 \subseteq \mathscr{N} \cup \mathscr{B}$ by (2′) and $\mathsf{rcl}(\mathsf{C}_1 \uplus \mathsf{C}_2, \mathscr{N} \cup \mathscr{B}) \subseteq \mathscr{N} \cup \mathscr{B}$ for $i = 1,2$ by Point (a).

—Proof of Point (c).
By definition $a \in \mathsf{K}$ implies $a \in \mathsf{D}_1$, or $a \in \mathsf{D}_2$ or there is $b \in \mathsf{D}_1 \cup \mathsf{D}_2$ such that $b \in \mathsf{cl}(\mathsf{C}_1 \uplus \mathsf{C}_2, a)$.

$$\begin{array}{rll} \mathsf{D}_i & = dom(\mathsf{R}_i) \setminus dom(\mathsf{C}_i) & \\ & \subseteq dom(\mathsf{R}_i) \setminus (\mathscr{N}_i \cup \mathscr{B}_i) & \text{by Point (2′) since } dom(\mathsf{R}_i) \cap \mathsf{N_i} = \emptyset \text{ by Lemma 6.2(1)} \\ & = dom(\mathscr{R}_i) & \text{by Point (1′)} \\ & \subseteq dom(\mathscr{R}) & \text{since } \mathscr{R} = (\mathscr{R}_1 \cup \mathscr{R}_2)^+. \end{array}$$

If there is $b \in \mathsf{D}_1 \cup \mathsf{D}_2$ such that $b \in \mathsf{cl}(\mathsf{C}_1 \uplus \mathsf{C}_2, a)$, then by above $b \in dom(\mathscr{R})$. Being $dom(\mathscr{R}) \cap (\mathscr{N} \cup \mathscr{B}) = \emptyset$ by applicability of rule {CONC}, this implies $b \notin \mathscr{N} \cup \mathscr{B}$. So we get $a \notin \mathscr{N} \cup \mathscr{B}$ by Point (a). Therefore we conclude $a \in dom(\mathscr{R})$.

—Applicability of rule $\{\textsc{Conc-I}\}$.

The condition $\mathsf{K} \cap \mathsf{N} = \emptyset$ immediately follows from Points (b) and (c).

—Proof of Point (1).

$$\begin{array}{rcl}
\mathscr{R} & = & (\mathscr{R}_1 \cup \mathscr{R}_2)^+ \\
 & & \text{by definition} \\
 & = & ((\mathscr{R}_1 \setminus (\mathscr{N}_2 \cup \mathscr{B}_2)) \cup (\mathscr{R}_2 \setminus (\mathscr{N}_1 \cup \mathscr{B}_1)))^+ \\
 & & \text{by applicability of rule } \{\textsc{Conc}\} \\
 & = & (((\mathsf{R}_1 \setminus (\mathscr{N}_1 \cup \mathscr{B}_1)) \setminus (\mathscr{N}_2 \cup \mathscr{B}_2)) \cup ((\mathsf{R}_2 \setminus (\mathscr{N}_2 \cup \mathscr{B}_2)) \setminus (\mathscr{N}_1 \cup \mathscr{B}_1)))^+ \\
 & & \text{by Point } (1') \\
 & = & ((\mathsf{R}_1 \cup \mathsf{R}_2) \setminus (\mathscr{N} \cup \mathscr{B}))^+ \\
 & = & (((\mathsf{R}_1 \setminus \mathsf{N}) \cup (\mathsf{R}_2 \setminus \mathsf{N})) \setminus (\mathscr{N} \cup \mathscr{B}))^+ \\
 & & \text{by Point (b)} \\
 & = & (\mathsf{R} \setminus (\mathscr{N} \cup \mathscr{B}))^+.
\end{array}$$

—Proof of Point (2).

$(\supseteq)$. The inclusion $\mathscr{N} \cup \mathscr{B} \supseteq \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$ follows immediately from Point (a) since by definition $\mathsf{C} \subseteq \mathsf{C}_1 \uplus \mathsf{C}_2$. By $(2')$ $\mathscr{N} \cup \mathscr{B} \supseteq \mathsf{N}_1 \cup \mathsf{N}_2$ and by Point (a) we have $\mathscr{N} \cup \mathscr{B} \supseteq \mathsf{cl}(\mathsf{C}_1 \uplus \mathsf{C}_2, \mathsf{N}_1 \cup \mathsf{N}_2)$, so we get $\mathscr{N} \cup \mathscr{B} \supseteq \mathsf{N}$.

$(\subseteq)$. We show that $a \in \mathscr{N}_i \cup \mathscr{B}_i$ implies either $a \in \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$ or $a \in \mathsf{N}$. By $(2')$ $a \in \mathscr{N}_i \cup \mathscr{B}_i$ implies either $a \in \mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i)$ or $a \in \mathsf{N_i}$. If $a \in \mathsf{N_i}$, then by definition of N we get $a \in \mathsf{N}$.

Let $a \in \mathsf{rcl}(\mathsf{C}_i, \mathscr{N}_i \cup \mathscr{B}_i)$. If $a \notin \mathsf{N} \cup \mathsf{K}$, then by definition of C we have $a \in \mathsf{rcl}(\mathsf{C}, \mathscr{N}_i \cup \mathscr{B}_i)$. Being $\mathsf{rcl}(\mathsf{C}, \mathscr{N}_i \cup \mathscr{B}_i) \subseteq \mathsf{rcl}(\mathsf{C}, \mathscr{N} \cup \mathscr{B})$ we are done.

If $a \in \mathsf{N}$ there is nothing to prove.

The case $a \in \mathsf{K}$ is impossible, since $\mathsf{K} \subseteq dom(\mathscr{R})$ by Point (c) and $\mathsf{rcl}(\mathsf{C}_1, \mathscr{N}_1 \cup \mathscr{B}_1) \subseteq (\mathscr{N}_1 \cup \mathscr{B}_1)$ by Point $(2')$ and $dom(\mathscr{R}) \cap (\mathscr{N}_1 \cup \mathscr{B}_1) = \emptyset$ by applicability of rule $\{\textsc{Conc}\}$.

—Proof of Point (3). Note that $\mathscr{B}_1 \setminus \mathscr{N}_1 \cup \mathscr{B}_2 \setminus \mathscr{N}_2 \subseteq (\mathscr{B}_1 \cup \mathscr{B}_2) \setminus (\mathscr{N}_1 \cup \mathscr{N}_2)$ and that

$(\mathscr{B}_1 \cup \mathscr{B}_2) \setminus (\mathscr{N}_1 \cup \mathscr{N}_2) = \mathscr{B}_1 \setminus (\mathscr{N}_1 \cup \mathscr{N}_2) \cup \mathscr{B}_2 \setminus (\mathscr{N}_1 \cup \mathscr{N}_2)$. Then we have:

$$\begin{array}{rcl}
(\mathscr{B}_1 \cup \mathscr{B}_2) \setminus (\mathscr{N}_1 \cup \mathscr{N}_2) & = & ((\mathscr{B}_1 \setminus \mathscr{N}_1) \cap (\mathscr{B}_2 \setminus \mathscr{N}_2)) \cup ((\mathscr{B}_1 \setminus (\mathscr{N}_1 \cup \mathscr{N}_2)) \cup \\
& & ((\mathscr{B}_2 \setminus (\mathscr{N}_1 \cup \mathscr{N}_2)) \\
& \subseteq & (\mathsf{B}_1 \cap \mathsf{B}_2) \cup \{\mathsf{a} \in \mathsf{B}_1 \cap \mathscr{B}_1 \mid \mathsf{a} \notin \mathscr{N}_2\} \cup \\
& & \{a \in \mathsf{B}_2 \cap \mathscr{B}_2 \mid \mathsf{a} \notin \mathscr{N}_1\} \\
& & \text{by Point } (3') \\
& = & (\mathsf{B}_1 \cap \mathsf{B}_2) \cup \{\mathsf{a} \in \mathsf{B}_1 \cap \mathscr{B}_1 \mid \mathsf{a} \notin \mathscr{R}_2 \cup \mathscr{N}_2\} \cup \\
& & \{a \in \mathsf{B}_2 \cap \mathscr{B}_2 \mid \mathsf{a} \notin \mathscr{R}_1 \cup \mathscr{N}_1\} \\
& & \text{since } \mathscr{B}_i \cap \mathscr{R}_{\overline{i}} = \emptyset \\
& \subseteq & (\mathsf{B}_1 \cap \mathsf{B}_2) \cup \{\mathsf{a} \in \mathsf{B}_1 \mid \mathsf{a} \notin \mathscr{R}_2 \cup \mathscr{N}_2\} \cup \\
& & \{a \in \mathsf{B}_2 \mid \mathsf{a} \notin \mathscr{R}_1 \cup \mathscr{N}_1\} \\
& = & (\mathsf{B}_1 \cap \mathsf{B}_2) \cup \{\mathsf{a} \in \mathsf{B}_1 \mid \mathsf{a} \notin \mathscr{R}_2 \cup \mathscr{N}_2 \cup \mathscr{B}_2\} \cup \\
& & \{a \in \mathsf{B}_2 \mid \mathsf{a} \notin \mathscr{R}_1 \cup \mathscr{N}_1 \cup \mathscr{B}_1\} \\
& & \text{since if } a \in \mathscr{B}_2 \text{ and } a \notin \mathscr{N}_2 \text{ then} \\
& & \text{by Point } (3')\ a \in \mathsf{B}_2 \text{ and then } a \in \mathsf{B}_1 \cap \mathsf{B}_2 \\
& = & (\mathsf{B}_1 \cap \mathsf{B}_2) \cup \{\mathsf{a} \in \mathsf{B}_1 \mid \mathsf{a} \notin \mathsf{dom}(\mathsf{R}_2) \cup \mathsf{N}_2\} \cup \\
& & \{a \in \mathsf{B}_2 \mid \mathsf{a} \notin \mathsf{dom}(\mathsf{R}_1) \cup \mathsf{N}_1\} \\
& & \text{by Points } (1'), (2') \text{ and} \\
& & \text{Lemma 6.2(2) and (6)} \\
& = & \mathsf{B}.
\end{array}$$

□

In the end we can state the following

COROLLARY 6.6. *A process P has the progress property iff the exists Θ such that $\Theta\ ;\ P \mapsto\!\!\!> \mathsf{R}\,;\,\mathsf{C}\,;\,\mathsf{N}\,;\,\mathsf{B}$ is provable for some* R, C, N, B.

7. CONCLUSIONS AND RELATED WORK

The programming framework presented in this paper relies on the concept of global types that can be seen as the language to describe the model of the distributed communications, i.e., an abstract high-level view of the protocol that all the participants will have to respect in order to communicate in a multiparty communication. The programmer will then write the program to implement this communication protocol; the system will use the global types (abstract model) and the program (implementation) to generate a runtime representation of the program which consists of the input/output operations decorated with explicit senders and receivers, according to the information provided in the global types. An alternative way could be that the programmer directly specifies the senders and the receivers in the communication operations as our low-level processes; the system could then infer the global types from the program. Our communication and interaction type systems will work as before in order to check the correctness and the progress of the program. Thus the programmer can choose between a top-down and a bottom-up style of programming, while relying on the same properties checked and guaranteed by the system.

We are currently designing and implementing a modelling and specification language with multiparty session types [Scribble 2008] for the standards of business and financial protocols with our industry collaborators [UNIFI 2002; Web Services Choreography Working Group 2002]. This consists of three layers: the first layer is a global type which corresponds to a signature of class models in UML; the second one is for conversation models where signatures and variables for multiple conversations are integrated; and the third layer includes extensions of the existing languages (such as Java [Hu et al. 2008]) which implement conversation models. We are currently considering to extend this modelling framework with our type discipline so that we can specify and ensure progress for executable conversations.

Multiparty sessions. The first papers on multiparty session types are [Bonelli and Compagnoni 2008] and [Honda et al. 2008]. The work [Bonelli and Compagnoni 2008] uses a distributed calculus where each channel connects a master end-point and one or more slave endpoints; instead of global types, they solely use (recursion-free) local types. In type checking, local types are projected to binary sessions, so that type safety is ensured using duality, but it loses sequencing information: hence progress in a session interleaved with other sessions is not guaranteed.

The present calculus is an essential improvement from [Honda et al. 2008]; both processes and types in [Honda et al. 2008] share a vector of channels and each communication uses one of these channels, while our user processes and global types are simpler and user-friendly without these channels. The global types in [Honda et al. 2008] have a parallel composition operator, but its projectability from global to local types limits to disjoint senders and receivers; hence it does not increase expressivity.

Our calculus is more liberal than the calculus of [Honda et al. 2008] in the use of declarations. For example

$$P_1 = \overline{a}[2](y_1).\overline{b}[2](y_2).\mathsf{def}\ X(x\,y) = P\ \mathsf{in}\ y_1?(2,x_1);X\langle x_1,y_1\rangle \mid y_2?(2,x_2);X\langle x_2,y_2\rangle$$

in parallel with

$$P_2 = a[2](z_1).b[2](z_2).z_1!\langle 1,\mathsf{true}\rangle;\mathbf{0} \mid z_2!\langle 1,\mathsf{false}\rangle;\mathbf{0}$$

reduces to

$$(\nu s)(\nu s')(\mathsf{def}\ X(x\,y) = P\ \mathsf{in}\ P\{\mathsf{true}/x\}\{s/y\} \mid P\{\mathsf{false}/x\}\{s'/y\} \mid s:\varnothing \mid s':\varnothing)$$

while the process in the calculus of [Honda et al. 2008] whose translation is $P_1 \mid P_2$ cannot be typed, since the channel variable must be the same in the definition and in the call. Similarly the delegation in [Honda et al. 2008] requires that the same channel is sent and received: this restriction is important for subject reduction as pointed out in [Yoshida and Vasconcelos 2007]. The present calculus solves this issue by having channels with roles, as in [Gay and Hole 2005] (see the example at page 21). This last example shows also that the original calculus has stuck processes whose translations in the present calculus are reducible. Note that these translations satisfy the interaction type system.

Different approaches to the description of service-oriented multiparty communications are taken in [Bravetti and Zavattaro 2007; Bruni et al. 2008]. In [Bravetti and Zavattaro 2007], the global and local views of protocols are described in two different calculi and the agreement between these views becomes a bisimulation between processes; [Bruni et al. 2008] proposes a distributed calculus which provides communications either inside sessions or inside locations, modelling merging running sessions. The type-safety and progress in interleaved sessions are left as an open problem in [Bruni et al. 2008].

Progress. The majority of papers on service-oriented calculi only assure that clients are never stuck inside a *single* session, see [Acciai and Boreale 2008; Dezani-Ciancaglini et al. 2008; Honda et al. 2008] for detailed discussions, including comparisons between the session-based and the traditional behavioural type systems of mobile processes, e.g. [Yoshida 1996; Kobayashi 2006]. We only say here that our interaction type system is inspired by deadlock-free typing systems [Kobayashi 1998; 2006; Yoshida 1996]. In [Acciai and Boreale 2008; Dezani-Ciancaglini et al. 2008; Honda et al. 2008], structured session primitives help to give simpler typing systems for progress.

The first papers considering progress for interleaved sessions required the nesting of sessions in Java [Dezani-Ciancaglini et al. 2006; Coppo et al. 2007] and SOC [Acciai and Boreale 2008; Lanese et al. 2007; Bruni and Mezzina 2008]. The present approach significantly improves the binary session system for progress in [Dezani-Ciancaglini et al. 2008] by treating the following points:

(1) asynchrony of the communication with queues, which enhances progress;
(2) a general mechanism of process recursion instead of the limited permanent accepts;
(3) a more liberal treatment of the channels which can be sent; and
(4) the standard semantics for the reception of channels with roles, which permits to get rid of process sequencing.

None of the previous work had treated progress across interfered, dynamically interleaved multiparty sessions.

Acknowledgements. We thank Kohei Honda and the Concur reviewers for their comments on an early version of this paper and Gary Brown for his collaboration on an implementation of multiparty session types.

REFERENCES

Acciai, L. and Boreale, M. 2008. A Type System for Client Progress in a Service-Oriented Calculus. In *Concurrency, Graphs and Models*. LNCS, vol. 5065. Springer, Berlin, Heidelberg, 642–658.

Bonelli, E. and Compagnoni, A. 2008. Multipoint Session Types for a Distributed Calculus. In *TGC'07*, G. Barthe and C. Fournet, Eds. LNCS, vol. 4912. Springer, Sophia-Antipolis, France, 240–256.

Bravetti, M. and Zavattaro, G. 2007. Towards a Unifying Theory for Choreography Conformance and Contract Compliance. In *Software Composition*. LNCS, vol. 4829. Springer, Braga, Portugal, 34–50.

Bruni, R., Lanese, I., Melgratti, H., and Tuosto, E. 2008. Multiparty Sessions in SOC. In *COORDINATION'08*. LNCS, vol. 5052. Springer, Oslo, Norway, 67–82.

Bruni, R. and Mezzina, L. G. 2008. A Deadlock Free Type System for a Calculus of Services and Sessions. http://www.di.unipi.it/ bruni/publications/ListOfDrafts.html.

Coppo, M., Dezani-Ciancaglini, M., and Yoshida, N. 2007. Asynchronous Session Types and Progress for Object-Oriented Languages. In *FMOODS'07*, M. Bonsangue and E. B. Johnsen, Eds. LNCS, vol. 4468. Springer, Paphos, Cyprus, 1–31.

Dezani-Ciancaglini, M., de' Liguoro, U., and Yoshida, N. 2008. On Progress for Structured Communications. In *TGC'07*, G. Barthe and C. Fournet, Eds. LNCS, vol. 4912. Springer, Sophia-Antipolis, France, 257–275.

Dezani-Ciancaglini, M., Mostrous, D., Yoshida, N., and Drossopoulou, S. 2006. Session Types for Object-Oriented Languages. In *ECOOP'06*. LNCS, vol. 4067. Springer, Nantes, France, 328–352.

Gay, S. and Hole, M. 2005. Subtyping for Session Types in the Pi-Calculus. *Acta Informatica 42,* 2/3, 191–225.

Honda, K. 1993. Types for Dyadic Interaction. In *CONCUR'93*. LNCS, vol. 715. Springer, Hildesheim, Germany, 509–523.

HONDA, K., VASCONCELOS, V. T., AND KUBO, M. 1998. Language Primitives and Type Disciplines for Structured Communication-based Programming. In *ESOP'98*, C. Hankin, Ed. LNCS, vol. 1381. Springer, Lisbon, Portugal, 22–138.

HONDA, K., YOSHIDA, N., AND CARBONE, M. 2008. Multiparty Asynchronous Session Types. In *POPL'08*, G. C. Necula and P. Wadler, Eds. ACM, San Francisco, USA, 273–284.

HU, R., YOSHIDA, N., AND HONDA, K. 2008. Session-Based Distributed Programming in Java. In *ECOOP'08*, J. Vitek, Ed. LNCS, vol. 5142. Springer, Paphos, Cyprus, 516–541.

KOBAYASHI, N. 1998. A Partially Deadlock-Free Typed Process Calculus. *ACM TOPLAS 20,* 2, 436–482.

KOBAYASHI, N. 2006. A New Type System for Deadlock-Free Processes. In *CONCUR'06*, C. Baier and H. Hermanns, Eds. LNCS, vol. 4137. Springer, Bonn, Germany, 233–247.

LANESE, I., VASCONCELOS, V. T., MARTINS, F., AND RAVARA, A. 2007. Disciplining Orchestration and Conversation in Service-Oriented Computing. In *SEFM'07*. IEEE Computer Society Press, London, UK, 305–314.

MILNER, R. 1999. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, Cambridge, England.

PIERCE, B. C. 2002. *Types and Programming Languages*. MIT Press, Cambridge, Massachusetts.

SCRIBBLE. 2008. Scribble Project. `www.scribble.org`.

UNIFI. 2002. International Organization for Standardization ISO 20022 UNIversal Financial Industry message scheme. `http://www.iso20022.org`.

WEB SERVICES CHOREOGRAPHY WORKING GROUP. 2002. Web Services Choreography Description Language. http://www.w3.org/2002/ws/chor/.

YOSHIDA, N. 1996. Graph Types for Monadic Mobile Processes. In *FSTTCS'96*. LNCS, vol. 1180. Springer, Hyderabad, India, 371–386.

YOSHIDA, N. AND VASCONCELOS, V. T. 2007. Language Primitives and Type Disciplines for Structured Communication-based Programming Revisited. In *SecRet'06*. ENTCS, vol. 171. Elsevier, Venice, Italy, 73–93.

A. PROOFS OF THE PROPERTIES OF THE COMMUNICATION TYPE SYSTEM

LEMMA A.1 INVERSION LEMMA FOR PURE PROCESSES. *(1) If* $\Gamma \vdash u : S$*, then* $u : S \in \Gamma$.

(2) *If* $\Gamma \vdash \mathsf{true} : S$*, then* $S = \mathsf{bool}$.

(3) *If* $\Gamma \vdash \mathsf{false} : S$*, then* $S = \mathsf{bool}$.

(4) *If* $\Gamma \vdash e_1\ \mathsf{and}\ e_2 : S$*, then* $\Gamma \vdash e_1, e_2 : \mathsf{bool}, S = \mathsf{bool}$.

(5) *If* $\Gamma \vdash \overline{a}[n](y).P \triangleright \Delta$*, then* $\Gamma \vdash a : \langle G \rangle$ *and* $\Gamma \vdash P \triangleright \Delta, y : G \upharpoonright 1$ *and* $\mathrm{pn}(G) \leq n$.

(6) *If* $\Gamma \vdash a[\mathsf{p}](y).P \triangleright \Delta$*, then* $\Gamma \vdash a : \langle G \rangle$ *and* $\Gamma \vdash P \triangleright \Delta, y : G \upharpoonright \mathsf{p}$.

(7) *If* $\Gamma \vdash c!\langle \Pi, e\rangle; P \triangleright \Delta$*, then* $\Delta = \Delta', c : !\langle \Pi, S\rangle; T$ *and* $\Gamma \vdash e : S$ *and* $\Gamma \vdash P \triangleright \Delta', c : T$.

(8) *If* $\Gamma \vdash c?(\mathsf{q}, x); P \triangleright \Delta$*, then* $\Delta = \Delta', c : ?(\mathsf{q}, S); T$ *and* $\Gamma, x : S \vdash P \triangleright \Delta', c : T$.

(9) *If* $\Gamma \vdash c!\langle\langle \mathsf{p}, c'\rangle\rangle; P \triangleright \Delta$*, then* $\Delta = \Delta', c : !\langle \mathsf{p}, T'\rangle; T, c' : T'$ *and* $\Gamma \vdash P \triangleright \Delta', c : T$.

(10) *If* $\Gamma \vdash c?((\mathsf{q}, y)); P \triangleright \Delta$*, then* $\Delta = \Delta', c : ?(\mathsf{q}, T'); T$ *and* $\Gamma \vdash P \triangleright \Delta', c : T, y : T'$.

(11) *If* $\Gamma \vdash c \oplus \langle \Pi, l_j\rangle; P \triangleright \Delta$*, then* $\Delta = \Delta', c : \oplus\langle \Pi, \{l_i : T_i\}_{i\in I}\rangle$ *and* $\Gamma \vdash P \triangleright \Delta', c : T_j$ *and* $j \in I$.

(12) *If* $\Gamma \vdash c\&(\mathsf{p}, \{l_i : P_i\}_{i\in I}) \triangleright \Delta$*, then* $\Delta = \Delta', c : \&(\mathsf{p}, \{l_i : T_i\}_{i\in I})$ *and* $\Gamma \vdash P_i \triangleright \Delta', c : T_i \quad \forall i \in I$.

(13) *If* $\Gamma \vdash P \mid Q \triangleright \Delta$*, then* $\Delta = \Delta' \cup \Delta''$ *and* $\Gamma \vdash P \triangleright \Delta'$ *and* $\Gamma \vdash Q \triangleright \Delta''$ *where* $dom(\Delta') \cap dom(\Delta'') = \emptyset$.

(14) *If* $\Gamma \vdash \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \triangleright \Delta$*, then* $\Gamma \vdash e : \mathsf{bool}$ *and* $\Gamma \vdash P \triangleright \Delta$ *and* $\Gamma \vdash Q \triangleright \Delta$.

(15) *If* $\Gamma \vdash \mathbf{0} \triangleright \Delta$*, then* Δ end *only.*

(16) *If* $\Gamma \vdash (\nu a)P \triangleright \Delta$*, then* $\Gamma, a : \langle G \rangle \vdash P \triangleright \Delta$.

(17) *If* $\Gamma, X : S\,T \vdash X\langle e, c\rangle \triangleright \Delta$*, then* $\Delta = \Delta', c : T$ *and* $\Gamma \vdash e : S$ *and* Δ' end *only.*

(18) *If* $\Gamma \vdash \mathsf{def}\ X(x, y) = P\ \mathsf{in}\ Q \triangleright \Delta$*, then* $\Gamma, X : S\,T, x : S \vdash P \triangleright \{y : T\}$ *and* $\Gamma, X : S\,T \vdash Q \triangleright \Delta$.

LEMMA A.2 INVERSION LEMMA FOR PROCESSES. *(1) If* $\Gamma \vdash_\Sigma P \triangleright \Delta$ *and* P *is a pure process, then* $\Sigma = \emptyset$ *and* $\Gamma \vdash P \triangleright \Delta$.

(2) *If* $\Gamma \vdash_{\{s\}} s : \varnothing \triangleright \Delta$*, then* $\Delta = \mathsf{end}$ *only.*

(3) *If* $\Gamma \vdash_{\{s\}} s : h \cdot (\mathsf{q}, \Pi, v) \triangleright \Delta$*, then* $\Delta = \Delta'; \{s[\mathsf{q}] : !\langle \Pi, S\rangle\}$ *and* $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$ *and* $\Gamma \vdash v : S$.

(4) *If* $\Gamma \vdash_{\{s\}} s : h \cdot (\mathsf{q}, \mathsf{p}, s'[\mathsf{p}']) \triangleright \Delta$*, then* $\Delta = \Delta'; \{s[\mathsf{q}] : !\langle \mathsf{p}, T'\rangle\}$ *and* $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$ *and* $s'[\mathsf{p}'] : T' \in \Delta$.

(5) *If* $\Gamma \vdash_{\{s\}} s : h \cdot (\mathsf{q}, \Pi, l) \triangleright \Delta$*, then* $\Delta = \Delta'; \{s[\mathsf{q}] : \oplus\langle \Pi, l\rangle\}$ *and* $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$.

(6) *If* $\Gamma \vdash_\Sigma P \mid Q \triangleright \Delta$*, then* $\Sigma = \Sigma_1 \cup \Sigma_2$ *and* $\Delta = \Delta_1 * \Delta_2$ *and* $\Gamma \vdash_{\Sigma_1} P \triangleright \Delta_1$ *and* $\Gamma \vdash_{\Sigma_2} Q \triangleright \Delta_2$.

(7) *If* $\Gamma \vdash_\Sigma (\nu s)P \triangleright \Delta$*, then* $\Sigma = \Sigma' \setminus s$ *and* $\Delta = \Delta' \setminus s$ *and* $\mathsf{co}(\Delta', s)$ *and* $\Gamma \vdash_{\Sigma'} P \triangleright \Delta'$.

(8) *If* $\Gamma \vdash_\Sigma (\nu a)P \triangleright \Delta$*, then* $\Gamma, a : \langle G \rangle \vdash_\Sigma P \triangleright \Delta$.

(9) *If* $\Gamma \vdash_\Sigma \mathsf{def}\ X(x, y) = P\ \mathsf{in}\ Q \triangleright \Delta$*, then* $\Gamma, X : S\,T, x : S \vdash P \triangleright y : T$ *and* $\Gamma, X : S\,T \vdash_\Sigma Q \triangleright \Delta$.

LEMMA A.3. *(1) If* $\Gamma \vdash_{\{s\}} s : (\mathsf{q}, \Pi, v) \cdot h \triangleright \Delta$*, then* $\Delta = \{s[\mathsf{q}] : !\langle \Pi, S\rangle\} * \Delta'$ *and* $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$ *and* $\Gamma \vdash v : S$.

(2) *If* $\Gamma \vdash_{\{s\}} s : (\mathsf{q}, \mathsf{p}, s'[\mathsf{p}']) \cdot h \triangleright \Delta$*, then* $\Delta = \{s[\mathsf{q}] : !\langle \mathsf{p}, T'\rangle\} * \Delta'$ *and* $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$ *and* $s'[\mathsf{p}'] : T' \in \Delta$.

(3) *If* $\Gamma \vdash_{\{s\}} s : (\mathsf{q}, \Pi, l) \cdot h \triangleright \Delta$*, then* $\Delta = \{s[\mathsf{q}] : \oplus\langle \Pi, l\rangle\} * \Delta'$ *and* $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$.

THEOREM A.4 TYPE PRESERVATION UNDER EQUIVALENCE. *If* $\Gamma \vdash_\Sigma P \triangleright \Delta$ *and* $P \equiv P'$, *then* $\Gamma \vdash_\Sigma P' \triangleright \Delta$.

PROOF. By induction on $\equiv$. We only consider some interesting cases (the other cases are straightforward).

$P \mid \mathbf{0} \equiv P$. First we assume $\Gamma \vdash_\Sigma P \triangleright \Delta$. By $\Gamma \vdash_\emptyset \mathbf{0} \triangleright \emptyset$ and by applying $\lfloor$GPAR$\rfloor$ to these two sequents we obtain $\Gamma \vdash_\Sigma P|\mathbf{0} \triangleright \Delta$.
For the converse direction assume $\Gamma \vdash_\Sigma P|\mathbf{0} \triangleright \Delta$. Using A.2(6) we obtain: $\Gamma \vdash_{\Sigma'} P \triangleright \Delta_1$, $\Gamma \vdash_{\Sigma''} \mathbf{0} \triangleright \Delta_2$ where $\Delta = \Delta_1 * \Delta_2$, $\Sigma = \Sigma' \cup \Sigma''$ and $\Sigma' \cap \Sigma'' = \emptyset$. Using A.2(1) we get $\Sigma'' = \emptyset$, which implies $\Sigma = \Sigma'$, and $\Gamma \vdash \mathbf{0} \triangleright \Delta_2$. Using A.1(15) we get Δ_2 end only and we conclude $\Gamma \vdash_\Sigma P \triangleright \Delta_1 * \Delta_2$ by applying $\lfloor$QWEAK$\rfloor$.

$P \mid Q \equiv Q \mid P$. By the symmetry of the rule we have only to show one direction. Suppose $\Gamma \vdash_\Sigma P \mid Q \triangleright \Delta$. Using A.2(6) we obtain $\Gamma \vdash_{\Sigma'} P \triangleright \Delta_1$, $\Gamma \vdash_{\Sigma''} Q \triangleright \Delta_2$ where $\Delta = \Delta_1 * \Delta_2$, $\Sigma = \Sigma' \cup \Sigma''$ and $\Sigma' \cap \Sigma'' = \emptyset$. Using $\lfloor$GPAR$\rfloor$ we get $\Gamma \vdash_\Sigma Q \mid P \triangleright \Delta_2 * \Delta_1$. Thanks to the commutativity of $*$, we get $\Delta_2 * \Delta_1 = \Delta$ and so we are done.

$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$. Suppose $\Gamma \vdash_\Sigma P \mid (Q \mid R) \triangleright \Delta$. Using A.2(6) we obtain $\Gamma \vdash_{\Sigma'} P \triangleright \Delta_1$, $\Gamma \vdash_{\Sigma''} Q \mid R \triangleright \Delta_2$ where $\Delta = \Delta_1 * \Delta_2$, $\Sigma = \Sigma' \cup \Sigma''$ and $\Sigma' \cap \Sigma'' = \emptyset$. Using A.2(6) we obtain $\Gamma \vdash_{\Sigma''_1} Q \triangleright \Delta_{21}$, $\Gamma \vdash_{\Sigma''_2} R \triangleright \Delta_{22}$ where $\Delta_2 = \Delta_{21} * \Delta_{22}$, $\Sigma'' = \Sigma''_1 \cap \Sigma''_2$ and $\Sigma''_1 \cup \Sigma''_2 = \emptyset$. Using [GPar] we get $\Gamma \vdash_{\Sigma' \cup \Sigma''_1} P \mid Q \triangleright \Delta_1 * \Delta_{21}$. Using $\lfloor$GPAR$\rfloor$ again we get $\Gamma \vdash_\Sigma (P \mid Q) \mid R \triangleright \Delta_1 * \Delta_{21} * \Delta_{22}$ and so we are done by the associativity of $*$. The proof for the other direction is similar.

$s : (\mathtt{q}, \emptyset, v) \cdot h \equiv s : h$. Using A.3(1) we obtain $\Gamma \vdash_s (\mathtt{q}, \emptyset, v) \cdot h \triangleright \Delta$, where $\Delta = \{s[\mathtt{q}] : !\langle \emptyset, S \rangle\} * \Delta'$ and $\Gamma \vdash_{\{s\}} s : h \triangleright \Delta'$ and $\Gamma \vdash v : S$. Using the equivalence relation on Δ we get $\{s[\mathtt{q}] : !\langle \emptyset, S \rangle\} * \Delta' \approx \Delta'$.

□

LEMMA A.5 SUBSTITUTION LEMMA. (*1*) *If* $\Gamma, x : S \vdash P \triangleright \Delta$ *and* $\Gamma \vdash v : S$, *then* $\Gamma \vdash P\{v/x\} \triangleright \Delta$.

(*2*) *If* $\Gamma \vdash P \triangleright \Delta, y : T$, *then* $\Gamma \vdash P\{s[\mathtt{p}]/y\} \triangleright \Delta, s[\mathtt{p}] : T$.

PROOF. Standard induction on the type derivation, with a case analysis on the last applied rule. □

THEOREM A.6 TYPE PRESERVATION UNDER REDUCTION. *If* $\Gamma \vdash_\Sigma P \triangleright \Delta$ *and* $P \longrightarrow^* P'$, *then* $\Gamma \vdash_\Sigma P' \triangleright \Delta'$ *for some* Δ' *such that* $\Delta \Rightarrow \Delta'$. *Moreover* Δ *coherent implies* Δ' *coherent and* Δ *closed implies* Δ' *closed.*

PROOF. By induction on a derivation of $P \longrightarrow^* P'$, with a case analysis on the final rule (using Theorem A.4 for the structural equivalence).

- **Case [Link]**
  $a[1](y_1).P_1 \mid ... \mid \overline{a}[n](y_n).P_n \longrightarrow (\nu s)(P_1\{s[1]/y_1\} \mid ... \mid P_n\{s[n]/y_n\} \mid s : \varnothing)$. By hypothesis $\Gamma \vdash_\Sigma \overline{a}[n](y_1).P_1 \mid a[2](y_2).P_2 \mid \ldots \mid a[n](y_n).P_n \triangleright \Delta$; then, since the redex is an

inital process, $\Sigma = \emptyset$ and $\Gamma \vdash \overline{a}[n](y_1).P_1 \mid a[2](y_2).P_2 \mid \ldots \mid a[n](y_n).P_n \triangleright \Delta$ by Lemma A.2(1). Using Lemma A.1(13) on all the processes in parallel we have

$$\Gamma \vdash \overline{a}[n](y_n).P_n \triangleright \Delta_n \tag{1}$$
$$\Gamma \vdash a[i](y_i).P_i \triangleright \Delta_i \quad (1 \leq i \leq n-1) \tag{2}$$

where $\Delta = \bigcup_{i=1}^{n} \Delta_i$. Using Lemma A.1(5) on (1) we have

$$\Gamma \vdash a : \langle G \rangle$$
$$\Gamma \vdash P_n \triangleright \Delta_n, y_n : G \restriction n \tag{3}$$

and $\mathrm{pn}(G) \leq n$. Using Lemma A.1(6) on (2) we have

$$\Gamma \vdash a : \langle G \rangle$$
$$\Gamma \vdash P_i \triangleright \Delta_i, y_i : G \restriction i \quad (1 \leq i \leq n-1). \tag{4}$$

Using Lemma A.5(2) on (3) and (4) we have

$$\Gamma \vdash_{\{s\}} P_i\{s[i]/y_i\} \triangleright \Delta_i, s[i] : G \restriction i \quad (1 \leq i \leq n). \tag{5}$$

Using $\lfloor$CONC$\rfloor$ on all the processes of (5) we have

$$\Gamma \vdash P_1\{s[1]/y_1\}|...|P_n\{s[n]/y_n\} \triangleright \bigcup_{i=1}^{n} (\Delta_i, s[i] : G \restriction i). \tag{6}$$

Note that $\bigcup_{i=1}^{n} (\Delta_i, s[i] : G \restriction i) = \Delta, s[1] : G \restriction 1, \ldots, s[n] : G \restriction n$. Using $\lfloor$GINIT$\rfloor$, $\lfloor$QINIT$\rfloor$ and $\lfloor$GPAR$\rfloor$ on (6) we have

$$\Gamma \vdash_{\{s\}} P_1\{s[1]/y_1\}|...|P_n\{s[n]/y_n\} \mid s : \varnothing \triangleright \Delta, s[1] : G \restriction 1, \ldots, s[n] : G \restriction n. \tag{7}$$

Using $\lfloor$QSCOPE$\rfloor$ on (7) we have

$$\Gamma \vdash_{\emptyset} (\nu s)(P_1\{s[1]/y_1\}|...|P_n\{s[n]/y_n\} \mid s : \varnothing) \triangleright \Delta \tag{8}$$

since $(\Delta, s[1] : G \restriction 1, \ldots, s[n] : G \restriction n) \setminus s = \Delta$.

- **Case [Send]** $s[\mathrm{p}]!\langle \Pi, e \rangle; P \mid s : h \longrightarrow P \mid s : h \cdot (\mathrm{p}, \Pi, v)$ $(e \downarrow v)$.
By hypothesis, $\Gamma \vdash_{\Sigma} s[\mathrm{p}]!\langle \Pi, e \rangle; P \mid s : h \triangleright \Delta$. Using Lemma A.2(1) and A.2(6) we have $\Sigma = \{s\}$ and

$$\Gamma \vdash s[\mathrm{p}]!\langle \Pi, e \rangle; P \triangleright \Delta_1 \tag{9}$$
$$\Gamma \vdash_{\{s\}} s : h \triangleright \Delta_2 \tag{10}$$

where $\Delta = \Delta_2 * \Delta_1$. Using A.1(7) on (9) we have

$$\begin{aligned} \Delta_1 = \Delta_1', s[\mathrm{p}] : !\langle \Pi, S\rangle; T & \\ \Gamma \vdash e : S & \quad (11) \\ \Gamma \vdash P \triangleright \Delta_1', s[\mathrm{p}] : T. & \quad (12) \end{aligned}$$

Using $\lfloor$QADDVAL$\rfloor$ on (10) and (11) we have

$$\Gamma \vdash_{\{s\}} s : h \cdot (\mathrm{q}, \Pi, v) \triangleright \Delta_2; \{s[\mathrm{p}] : !\langle \Pi, S\rangle\}. \quad (13)$$

Using $\lfloor$GINIT$\rfloor$ on (12) and then $\lfloor$GPAR$\rfloor$ on (12), (13) we get

$$\Gamma \vdash_{\{s\}} P \mid s : h \cdot (\mathrm{q}, \Pi, v) \triangleright (\Delta_2; \{s[\mathrm{p}] : !\langle \Pi, S\rangle\}) * (\Delta_1', s[\mathrm{p}] : T).$$

Note that $(\Delta_2; \{s[\mathrm{p}] : !\langle \Pi, S\rangle\}) * (\Delta_1', s[\mathrm{p}] : T) \Rightarrow \Delta_2 * (\Delta_1', s[\mathrm{p}] : !\langle \Pi, S\rangle; T)$.

- **Case [Recv]**
$s[\mathrm{p}_j]?(\mathrm{q}, x); P \mid s : (\mathrm{q}, \Pi, v) \cdot h \longrightarrow P\{v/x\} \mid s : (\mathrm{q}, \Pi \setminus j, v) \cdot h \quad (\mathrm{p}_j \in \Pi)$.
By hypothesis, $\Gamma \vdash_\Sigma s[\mathrm{p}_j]?(\mathrm{q}, x); P \mid s : (\mathrm{q}, \Pi, v) \cdot h \triangleright \Delta$. By A.2(1) and A.2(6) we have $\Sigma = \emptyset$ and

$$\begin{aligned} \Gamma \vdash s[\mathrm{p}_j]?(\mathrm{q}, x); P \triangleright \Delta_1 & \quad (14) \\ \Gamma \vdash_{\{s\}} s : (\mathrm{q}, \Pi, v) \cdot h \triangleright \Delta_2 & \quad (15) \end{aligned}$$

where $\Delta = \Delta_2 * \Delta_1$. Using Lemma A.1(8) on (14) we have

$$\begin{aligned} \Delta_1 = \Delta_1', s[\mathrm{p}_j] : ?(\mathrm{q}, S); T & \\ \Gamma, x : S \vdash P \triangleright \Delta_1', s[\mathrm{p}_j] : T & \quad (16) \end{aligned}$$

Using Lemma A.5(1) from (16) we get $\Gamma \vdash P\{v/x\} \triangleright \Delta_1', s[\mathrm{p}_j] : T$, which implies by rule $\lfloor$GINIT$\rfloor$

$$\Gamma \vdash_\emptyset P\{v/x\} \triangleright \Delta_1', s[\mathrm{p}_j] : T. \quad (17)$$

Using Lemma A.3(1) on (15) we have

$$\begin{aligned} \Delta_2 = \{s[\mathrm{q}] : !\langle \Pi, S\rangle\} * \Delta_2' & \\ \Gamma \vdash_{\{s\}} s : h \triangleright \Delta_2' & \quad (18) \\ \Gamma \vdash v : S. & \end{aligned}$$

Applying rule $\lfloor$QADDVAL$\rfloor$ on (18) we get

$$\Gamma \vdash_{\{s\}} (\mathrm{q}, \Pi \setminus j, v) \cdot h \triangleright \{s[\mathrm{q}] : !\langle \Pi \setminus j, S\rangle\} * \Delta_2' \quad (19)$$

Using rule $\lfloor$GPAR$\rfloor$ on (17) and (19) we get

$$\Gamma \vdash_{\{s\}} P\{v/x\} \mid (\mathrm{q}, \Pi \setminus j, v) \cdot h \triangleright (\{s[\mathrm{q}] : !\langle \Pi \setminus j, S\rangle\} * \Delta_2') * (\Delta_1', s[\mathrm{p}_j] : T).$$

Note that
$(\{s[\mathrm{q}] : !\langle \Pi, S\rangle\} * \Delta_2') * (\Delta_1', s[\mathrm{p}_j] : ?(\mathrm{q}, S); T) \Rightarrow (\{s[\mathrm{q}] : !\langle \Pi \setminus j, S\rangle\} * \Delta_2') * (\Delta_1', s[\mathrm{p}_j] : T)$.

- **Case [Label]**
$s[\mathrm{p}] \oplus \langle \Pi, l\rangle; P \mid s : h \longrightarrow P \mid s : h \cdot (\mathrm{p}, \Pi, l)$
By hypothesis, $\Gamma \vdash_\Sigma s[\mathrm{p}] \oplus \langle \Pi, l\rangle; P \mid s : h \triangleright \Delta$ Using Lemma A.2(1) and A.2(6) we have $\Sigma = \{s\}$ and

$$\Gamma \vdash s[\mathrm{p}] \oplus \langle \Pi, l\rangle; P \triangleright \Delta_1 \quad (20)$$
$$\Gamma \vdash_{\{s\}} s : h \triangleright \Delta_2 \quad (21)$$

where $\Delta = \Delta_2 * \Delta_1$. Using Lemma A.1(11) on (20) we have for $l = l_j \ (j \in I)$:

$$\Delta_1 = \Delta_1', s[\mathrm{p}] : \oplus\langle \Pi, \{l_i : T_i\}_{i\in I}\rangle$$
$$\Gamma \vdash P \triangleright \Delta_1', T_j. \quad (22)$$

Using rule $\lfloor$QSEL$\rfloor$ on (21) we have

$$\Gamma \vdash_{\{s\}} s : h \cdot (\mathrm{p}, \Pi, l) \triangleright \Delta_2; \{s[\mathrm{p}] : \oplus\langle \Pi, l\rangle\}. \quad (23)$$

Using $\lfloor$GPAR$\rfloor$ on (22) and (23) we have

$$\Gamma \vdash_{\{s\}} P \mid s : h \cdot (\mathrm{p}, \Pi, l) \triangleright (\Delta_2; \{s[\mathrm{p}] : \oplus\langle \Pi, l\rangle\}) * (\Delta_1', s[\mathrm{p}] : T_j).$$

Note that $\Delta_2 * (\Delta_1', s[\mathrm{p}] : \oplus\langle \Pi, \{l_i : T_i\}_{i\in I}\rangle) \Rightarrow (\Delta_2; \{s[p] : \oplus\langle \Pi, l\rangle\}) * (\Delta_1', s[\mathrm{p}] : T_j)$.

- **Case [Branch]**
$s[\mathrm{p}_j]\&(\mathrm{q}, \{l_i : P_i\}_{i\in I}) \mid s : (\mathrm{q}, \Pi, l_{i_0}) \cdot h \longrightarrow P_{i_0} \mid s : (\mathrm{q}, \Pi \setminus j, l_{i_0}) \cdot h \quad (\mathrm{p}_j \in \Pi) \quad (i_0 \in I)$
By hypothesis, $\Gamma \vdash_\Sigma s[\mathrm{p}_j]\&(\mathrm{q}, \{l_i : P_i\}_{i\in I}) \mid s : (\mathrm{q}, \Pi, l_{i_0}) \cdot h \triangleright \Delta$. Using Lemma A.2(1) and A.2(6) we have $\Sigma = \{s\}$ and

$$\Gamma \vdash s[\mathrm{p}_j]\&(\mathrm{q}, \{l_i : P_i\}_{i\in I}) \triangleright \Delta_1 \quad (24)$$
$$\Gamma \vdash_{\{s\}} s : (\mathrm{q}, \Pi, l_{i_0}) \cdot h \triangleright \Delta_2 \quad (25)$$

where $\Delta = \Delta_2 * \Delta_1 = \Delta_2 * \Delta_1$. Using Lemma A.1(12) on (24) we have

$$\Delta_1 = \Delta_1', s[\mathrm{p}_j] : \&(\mathrm{q}, \{l_i : T_i\}_{i\in I})$$
$$\Gamma \vdash P_i \triangleright \Delta_1', s[\mathrm{p}_j] : T_i \quad \forall i \in I. \quad (26)$$

Using Lemma A.3(3) on (25) we have

$$\Delta_2 = \{s[\mathrm{q}] : \oplus(\Pi, l_{i'})\} * \Delta_2'$$
$$\Gamma \vdash_{\{s\}} s : h \triangleright \Delta_2'. \quad (27)$$

Using $\lfloor$QSEL$\rfloor$ on (27) we get

$$\Gamma \vdash_{\{s\}} s : (\mathrm{q}, \Pi \setminus j, l_{i_0}) \cdot h \triangleright \{s[\mathrm{q}] : \oplus(\Pi \setminus j, l_{i'})\} * \Delta_2'. \quad (28)$$

Using $\lfloor \text{GP}\textsc{ar} \rfloor$ on (26) and (28) we have

$$\Gamma \vdash_{\{s\}} P_{i_0} \mid s : (\mathtt{q}, \Pi \setminus j, l_{i_0}) \cdot h \triangleright (\{s[\mathtt{q}] : \oplus(\Pi \setminus j, l_{i'})\} * \Delta'_2) * (\Delta'_1, s[\mathtt{p}_j] : T_{i_0}).$$

Note that

$$(\{s[\mathtt{q}] : \oplus(\Pi, l_{i'})\} * \Delta'_2) * (\Delta'_1, s[\mathtt{p}_j] : \&(\mathtt{q}, \{l_i : T_i\}_{i \in I})) \Rightarrow$$
$$(\{s[\mathtt{q}] : \oplus(\Pi \setminus j, l_{i'})\} * \Delta'_2) * (\Delta'_1, s[\mathtt{p}_j] : T_{i_0}).$$

□

Received M 20YY; revised M 20YY; accepted M 20YY