Multiparty Session Types

BETTY Summer School 2016

Laura Bocchi, University of Kent, l.bocchi@kent.ac.uk

Agenda

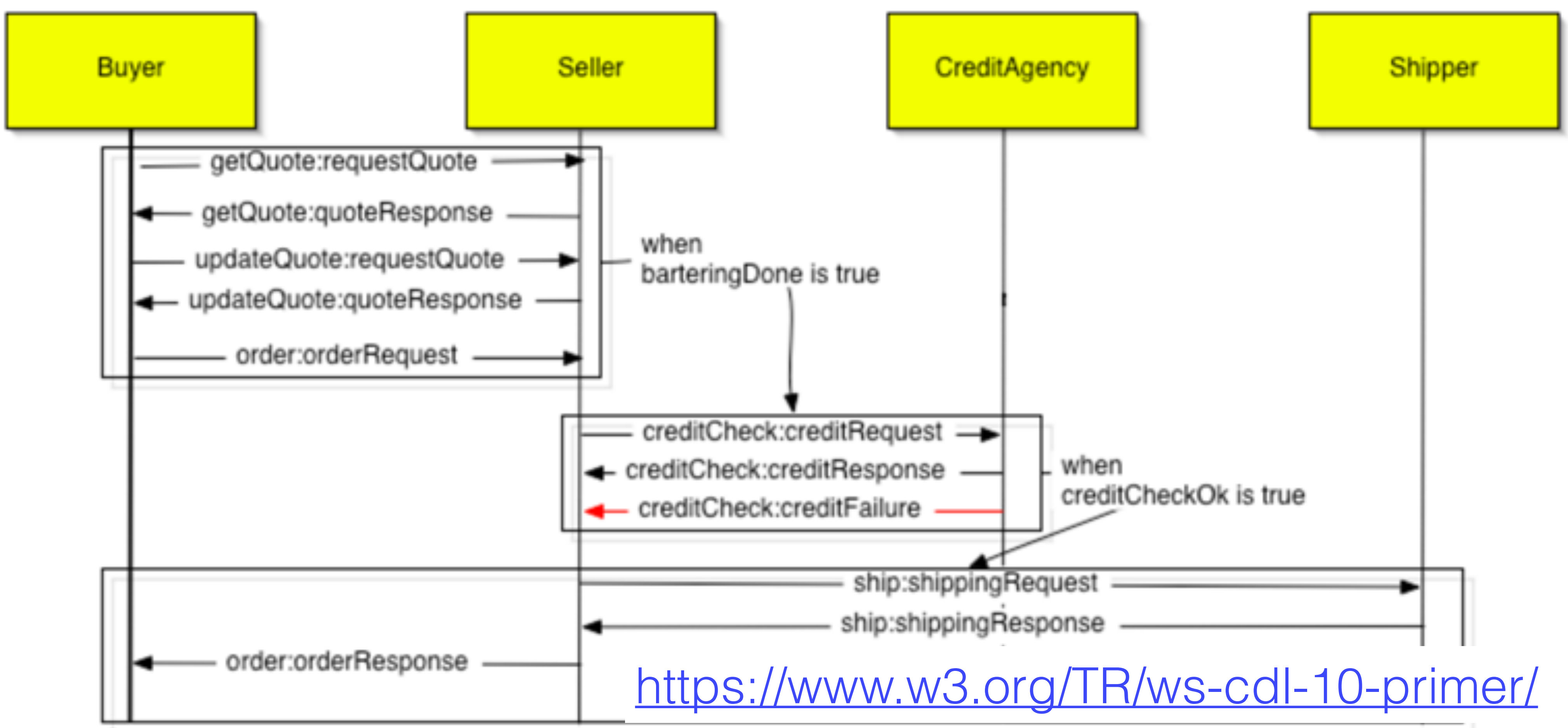
- Step 1 : the **types**
- Step 2 : the **processes**
- Step 3 : the **properties** of well-typed processes
- Step 4 : **advanced** topics *(delegation, static vs run-time verification, asserted session types, relationships with automata, session types for real-time systems)*

Motivation

- Why multiparty sessions?
- Why a theory of multiparty sessions?

Web service coordination

- Web services : not just web-based RMI
- Choreographies (WS-CDL Version 1.0 in 2005)



https://www.w3.org/TR/ws-cdl-10-primer/

global protocols

modular software development

Choreography & realisation

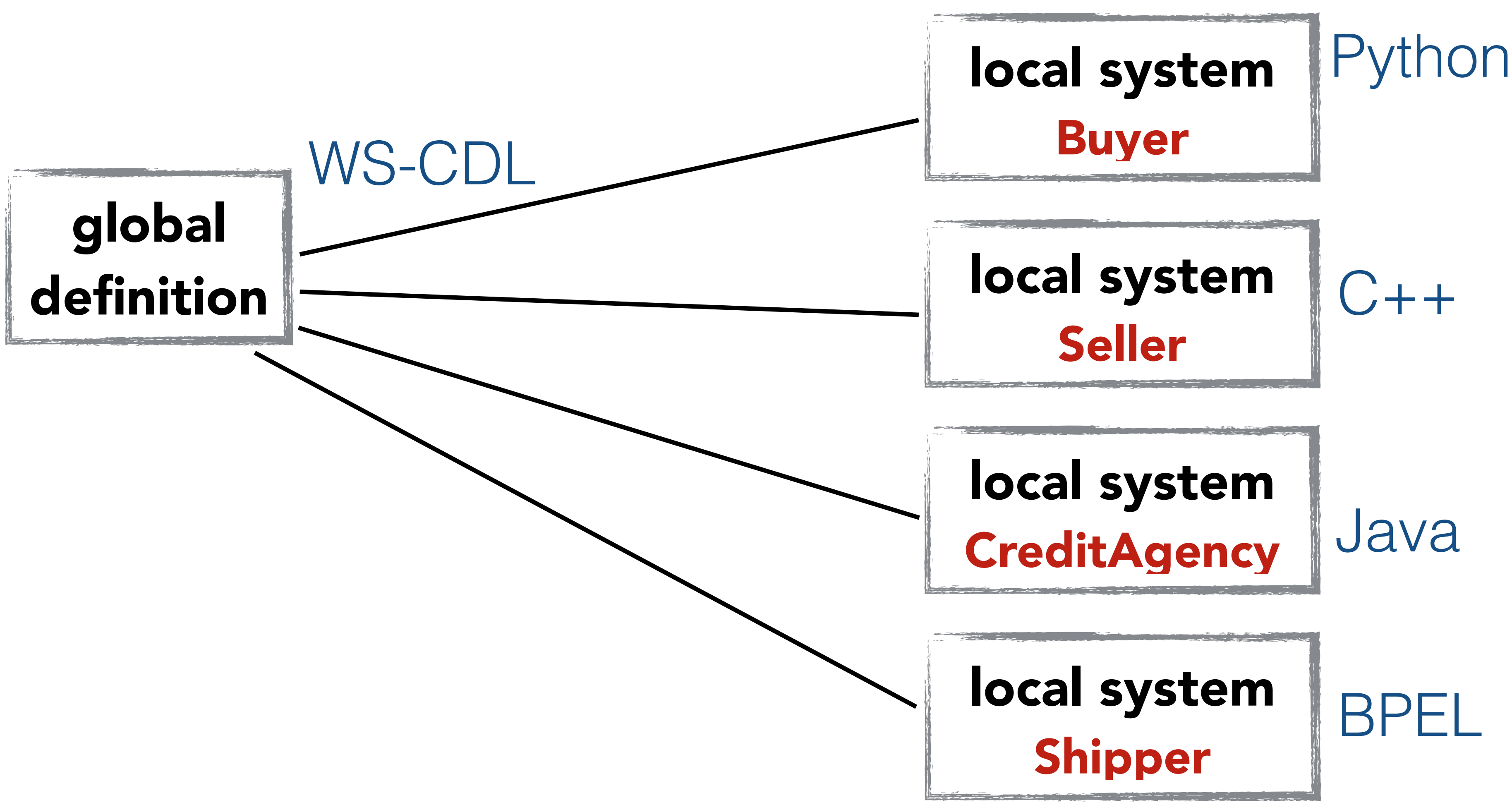
"Using the Web Services Choreography specification, a contract containing a **global definition** of the common ordering conditions and constraints under which messages are exchanged, is produced [...].
Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems […]" [W3C's WS-CDL]

WS-CDL

global definition

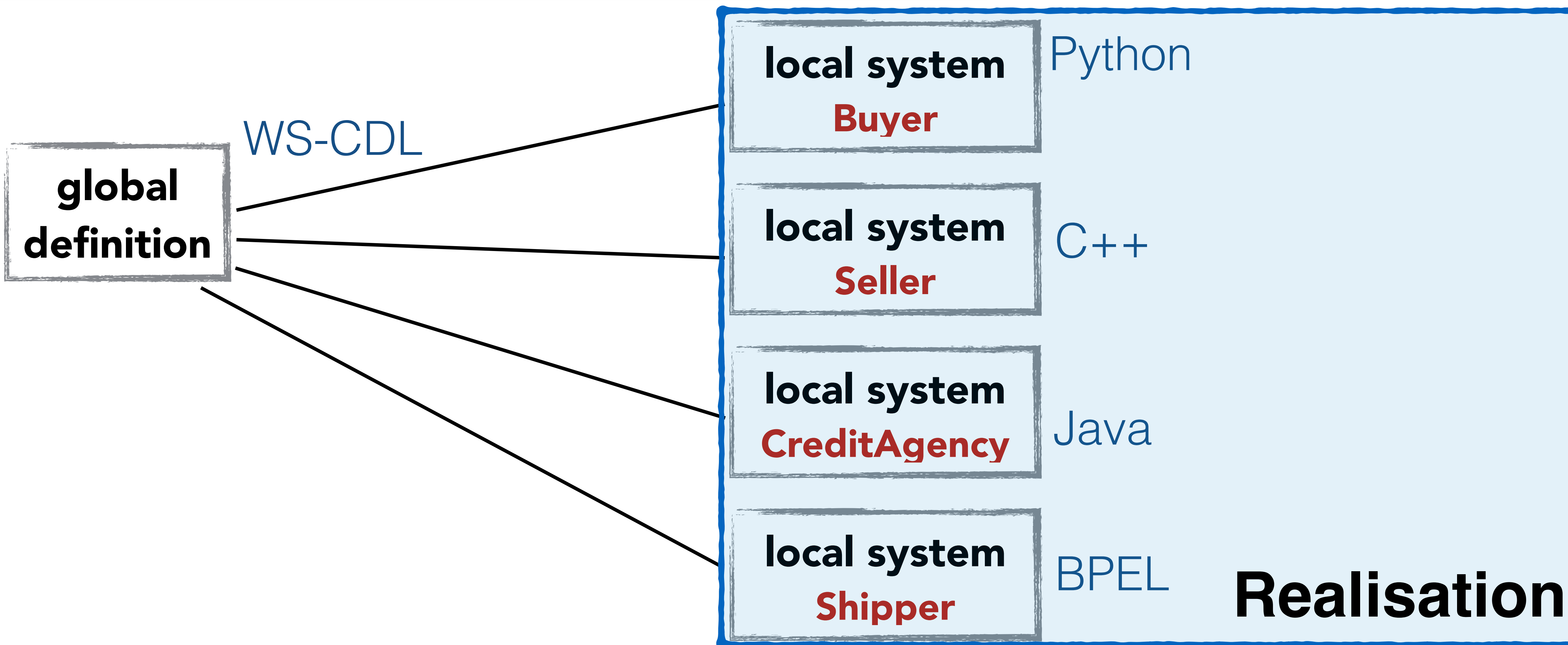
Choreography & realisation

"Using the Web Services Choreography specification, a contract containing a **global definition** of the common ordering conditions and constraints under which messages are exchanged, is produced [...]. Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]" [W3C's WS-CDL]



Choreography & realisation

"Using the Web Services Choreography specification, a contract containing a **global definition** of the common ordering conditions and constraints under which messages are exchanged, is produced [...].
Each party can then use the global definition to build and test solutions that conform to it. The global specification is in turn realised by combination of the resulting local systems [...]" [W3C's WS-CDL]



Origin of Multiparty Session Types

Binary Session Types [CONCUR'93,PARLE'94,ESOP'98]

⇩

Milner, Honda and Yoshida joined W3C WS-CDL (2002)

⇩

Formalisation of WS-CDL [ESOP'07]

⇩

Scribble at π4 Technology

⇩

Multiparty Session Types [POPL'08]

⇩

OOI OCEAN OBSERVATORIES INITIATIVE · Scribble · redhat · SAVARA · Cognizant

WS-CDL & Global types & Scribble

```
package HelloWorld {
    roleType YouRole, WorldRole;
    participantType You{YouRole}, World{WorldRole};
    relationshipType YouWorldRel between YouRole and WorldRole;
    channelType WorldChannelType with roleType WorldRole;

    choreography Main {
        WorldChannelType worldChannel;

        interaction operation=hello from=YouRole to=WorldRole
                relationship=YouWorldRel channel=worldChannel {
            request messageType=Hello;
        }
    }
}
```

By Nobuko Yoshida (Bertinoro SFM) in 2015 <— by Dr Gary Brown (Pi4 Tech) in 2007

WS-CDL & Global types & Scribble

$$\texttt{You} \rightarrow \texttt{World} : \langle \mathit{Hello} \rangle .\ \texttt{end}$$

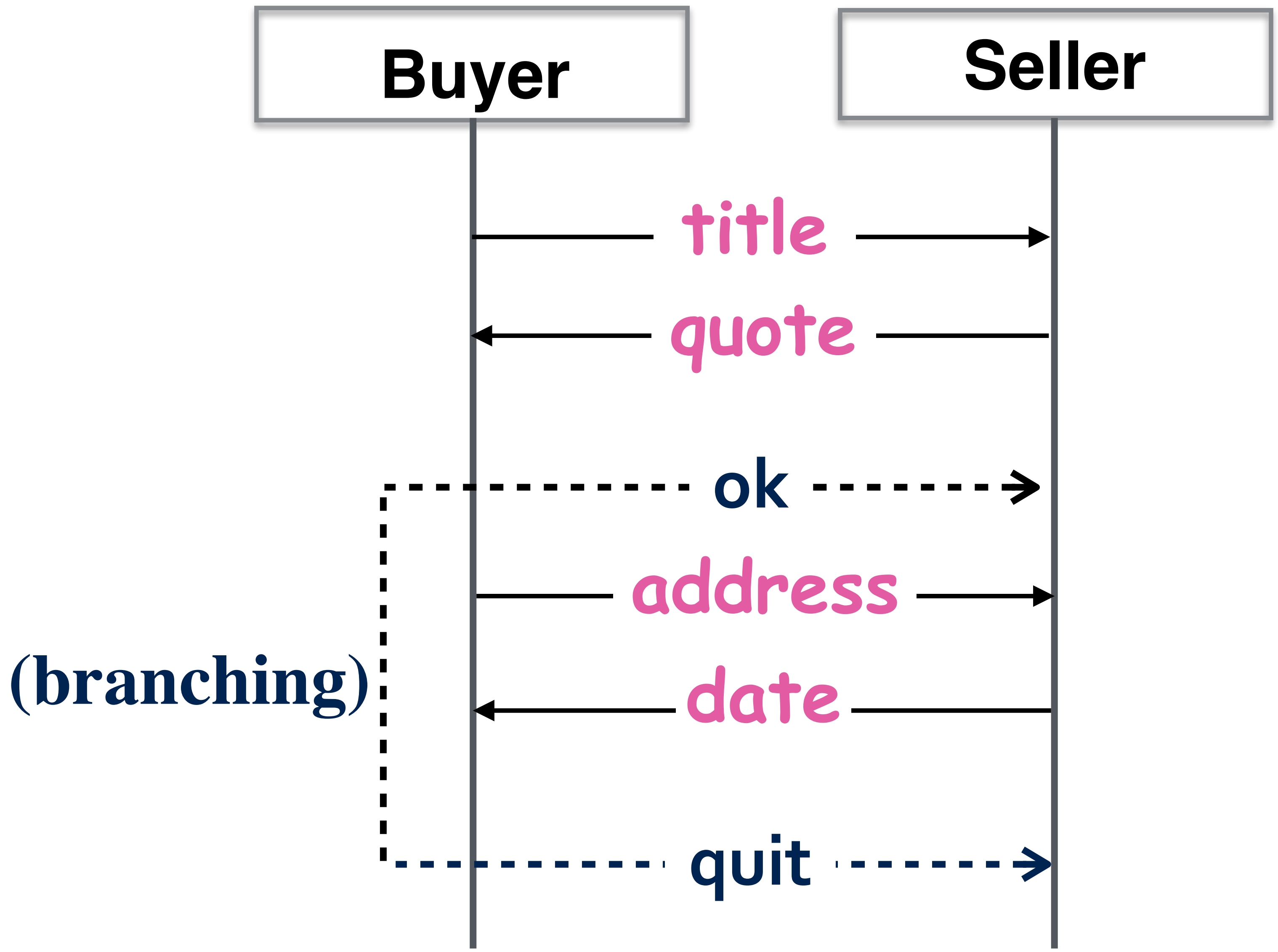
```
protocol HelloWorld {
    role You, World;
    Hello from You to World;
}
```

By Nobuko Yoshida (Bertinoro SFM) in 2015

Motivation

- Why multiparty sessions?
- Why a theory of multiparty sessions?

Binary Sessions (recap)

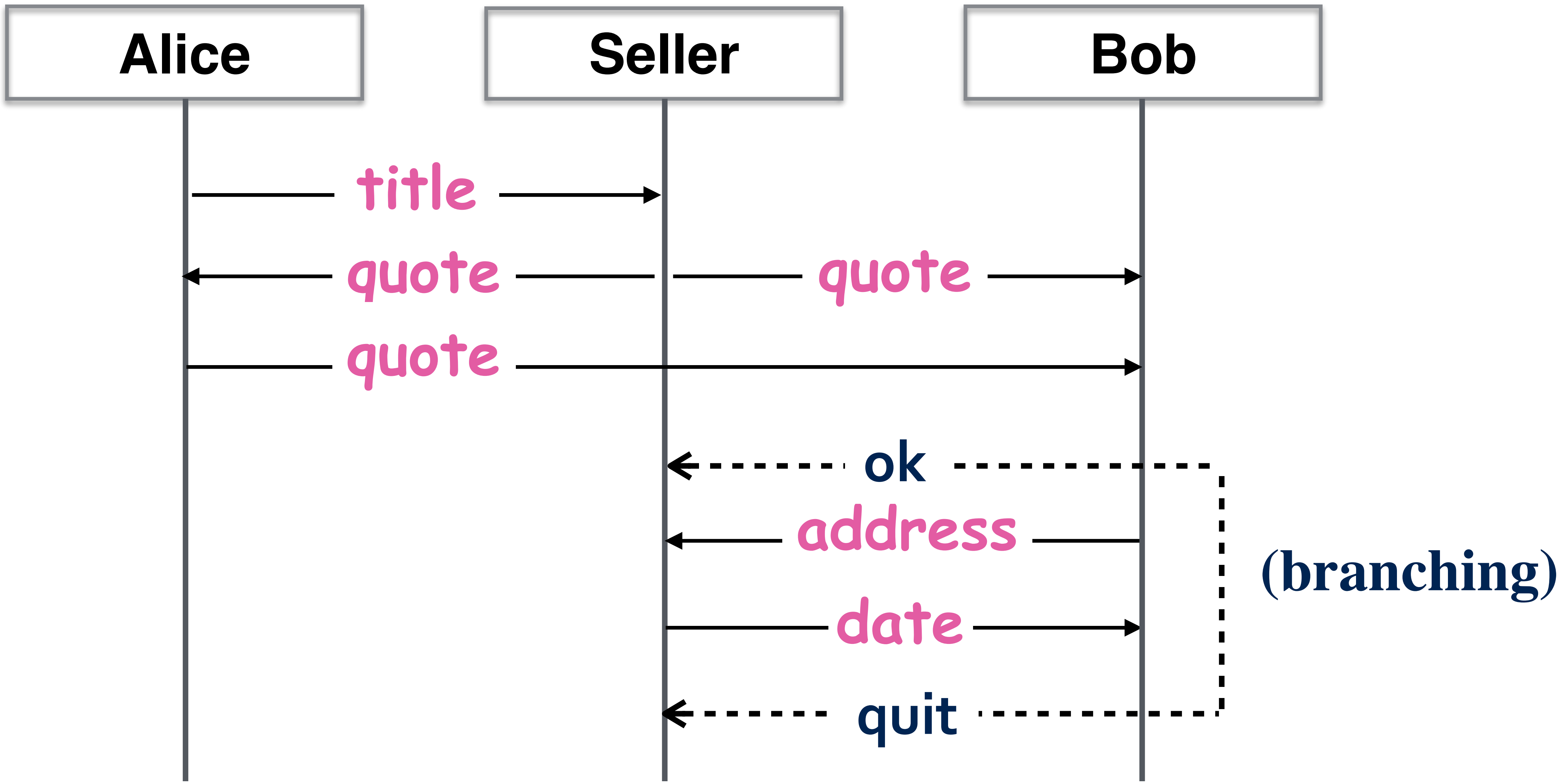


$$T = !String; ?Int; \oplus\{\mathsf{ok} : !String; ?Date; \mathtt{end}, \ \mathsf{quit} : \mathtt{end}\}$$

$$\overline{T} = ?String; !Int; \&\{\mathsf{ok} : ?String; !Date; \mathtt{end}, \ \mathsf{quit} : \mathtt{end}\}$$

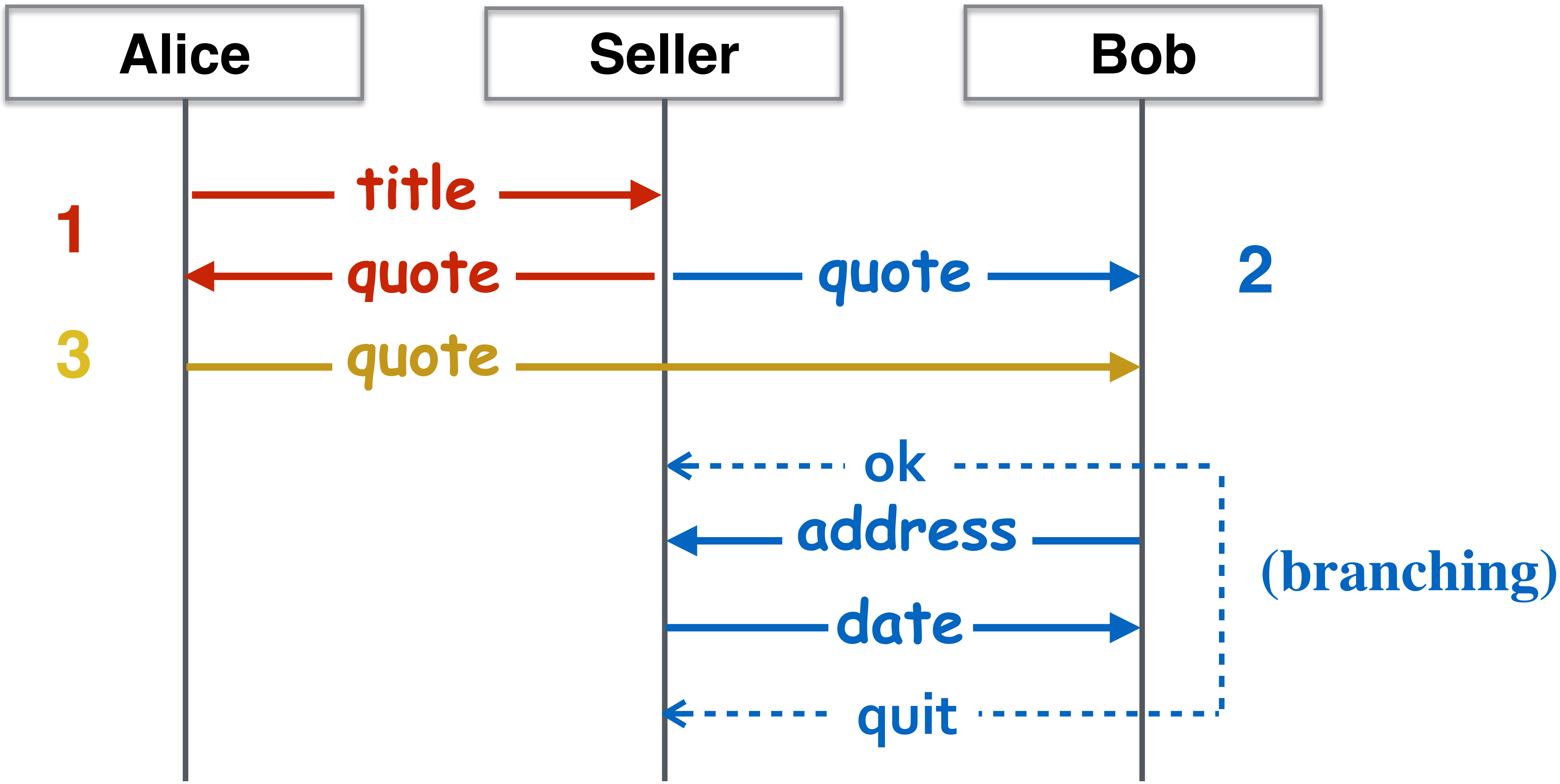
If P and Q have dual types (e.g., T and $\overline{T}$, respectively) then $P \mid Q$ is typable.

Binary vs Multiparty



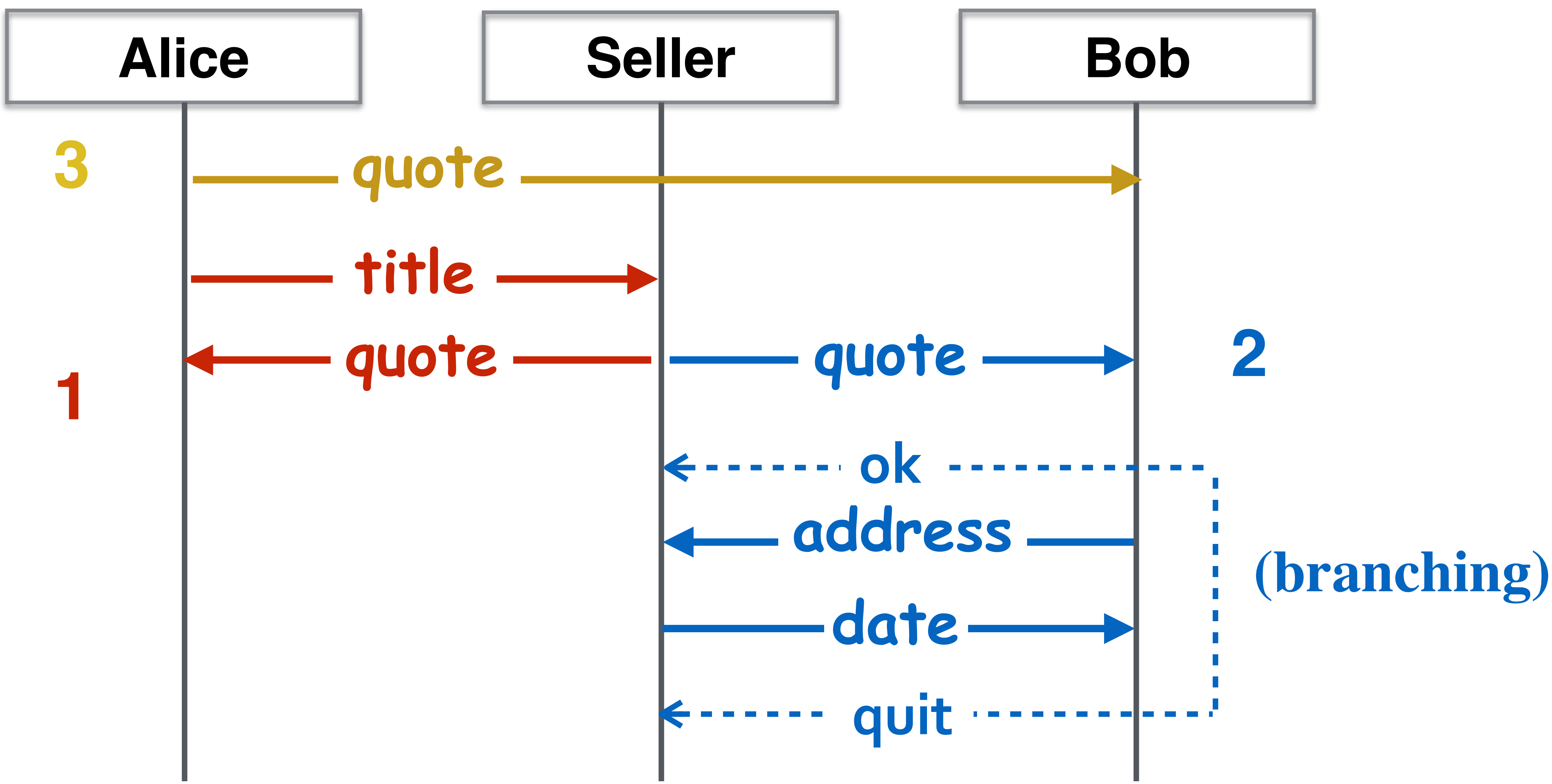
Can you model this using binary types?

Binary vs Multiparty



… for example by combining binary session …

Binary vs Multiparty



No. Causalities may not be preserved across sessions

Therefore…

- Having session involving more than two roles allows to model protocols that we could not model with binary session types
- The challenges of Multiparty Session types:
    - asynchrony + distribution
    - network configuration
    - extending duality to multiparty specifications
    - realizability

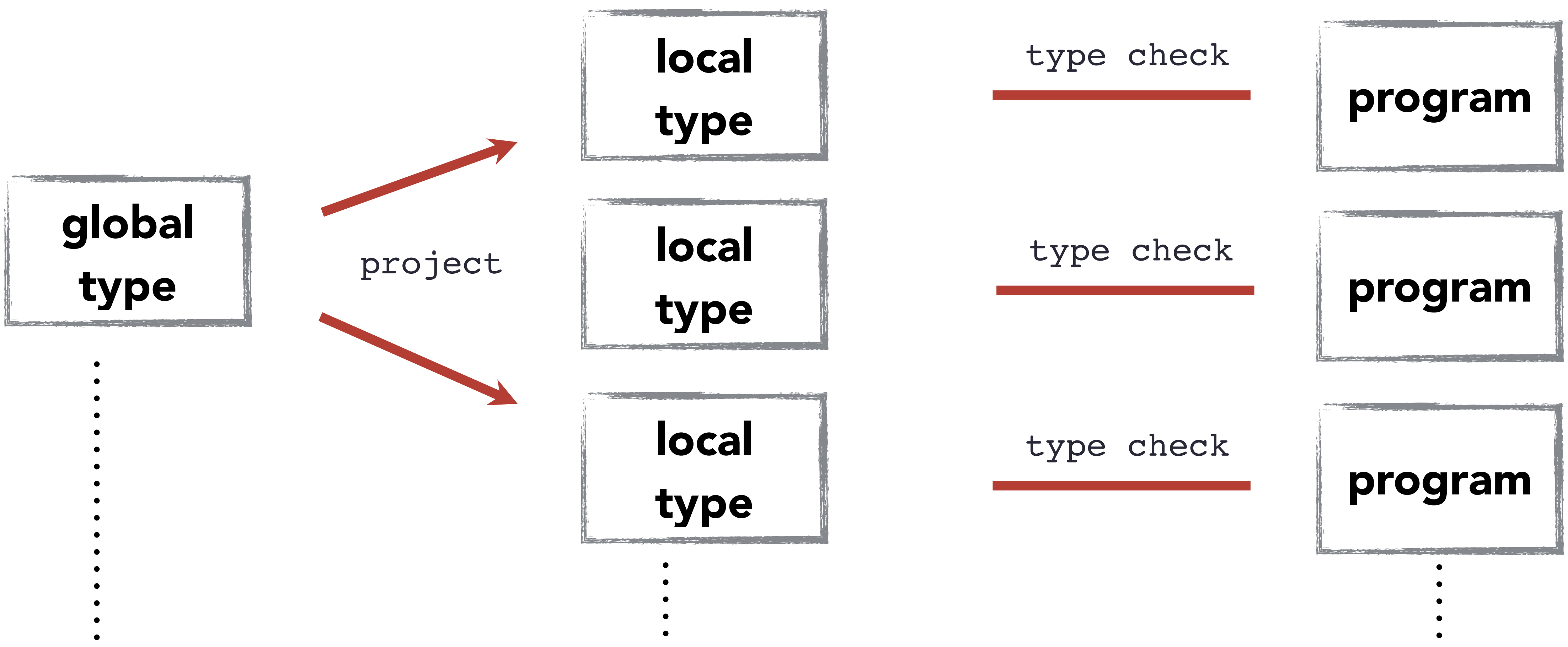
Essential reading list

Honda,Yoshida,Carbone@POPL'08
"*Multiparty asynchronous session types*"

Bettini et al.@CONCUR'08
"*Global Progress in Dynamically Interleaved Multiparty Sessions*"

Coppo,Dezani-Ciancaglini,Padovani,Yoshida@SFM'15
"*A Gentle Introduction to Multiparty Asynchronous Session Types*"

Multiparty Session Types : the method

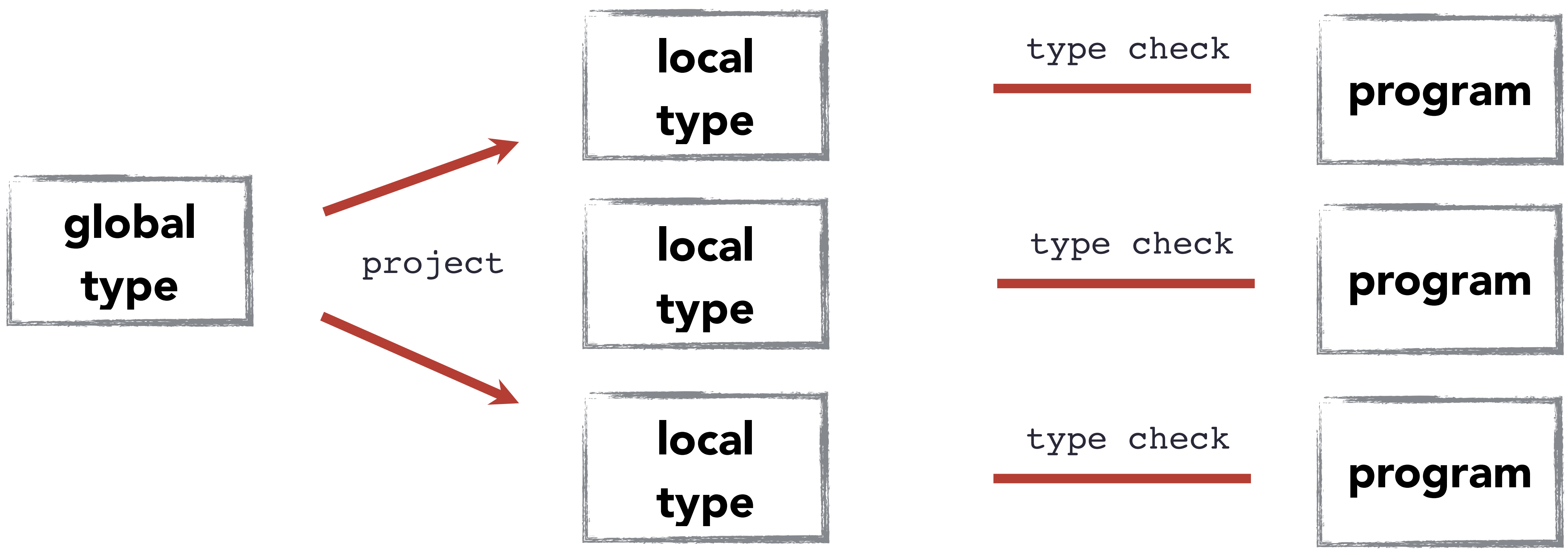


$\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle.$
$\mathtt{S} \to \mathtt{A} : \langle \mathit{Int} \rangle.$
$\mathtt{S} \to \mathtt{B} : \langle \mathit{Int} \rangle.$
$\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle.$
$\ldots$

$!\mathtt{S}\langle \mathit{String} \rangle.$
$?\mathtt{S}(\mathit{Int}).$
$!\mathtt{B}\langle \mathit{Int} \rangle.$
$\ldots$

$s[\mathtt{B1}]!\langle \mathtt{S}, \text{“Crime and punishment”} \rangle.$
$s[\mathtt{B1}]?(\mathtt{S}, x).$
if $x < 7$
then $s[\mathtt{B1}]!\langle \mathtt{B2}, x \rangle.\ \ldots$
else $s[\mathtt{B1}]!\langle \mathtt{B2}, 7 \rangle.\ \ldots$

Some considerations …



- **Top down** approach (global types known/agreed beforehand)
- **Local** verification of **global** properties: session fidelity, communication safety, progress
- **Tractability**

Global types : syntax

G	$::=$	$\mathtt{p} \to \mathtt{q} : \langle U \rangle . G$	Interaction
	$\mid$	$\mathtt{p} \to \mathtt{q} : \{\mathsf{l}_i : G_i\}_{i \in I}$	Branching
	$\mid$	$\mu \mathtt{t}.\ G$	Recursion
	$\mid$	$\mathtt{t}$	Type variable
	$\mid$	$\mathtt{end}$	End

$$U \quad ::= \quad \mathit{Int} \mid \mathit{String} \mid \mathit{Bool} \mid \ldots \mid \mathsf{T} \mid G$$

Three buyers protocol: global type

$$
\begin{aligned}
&\mathtt{A} \to \mathtt{S} : \langle String \rangle. \\
&\mathtt{S} \to \mathtt{A} : \langle Int \rangle. \\
&\mathtt{S} \to \mathtt{B} : \langle Int \rangle. \\
&\mathtt{A} \to \mathtt{B} : \langle Int \rangle. \\
&\mathtt{B} \to \mathtt{S} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle String \rangle. \\
&\qquad\qquad\qquad \mathtt{S} \to \mathtt{B} : \langle Date \rangle.\mathbf{end}, \\
&\qquad\quad \mathsf{quit} : \mathbf{end}\}
\end{aligned}
$$

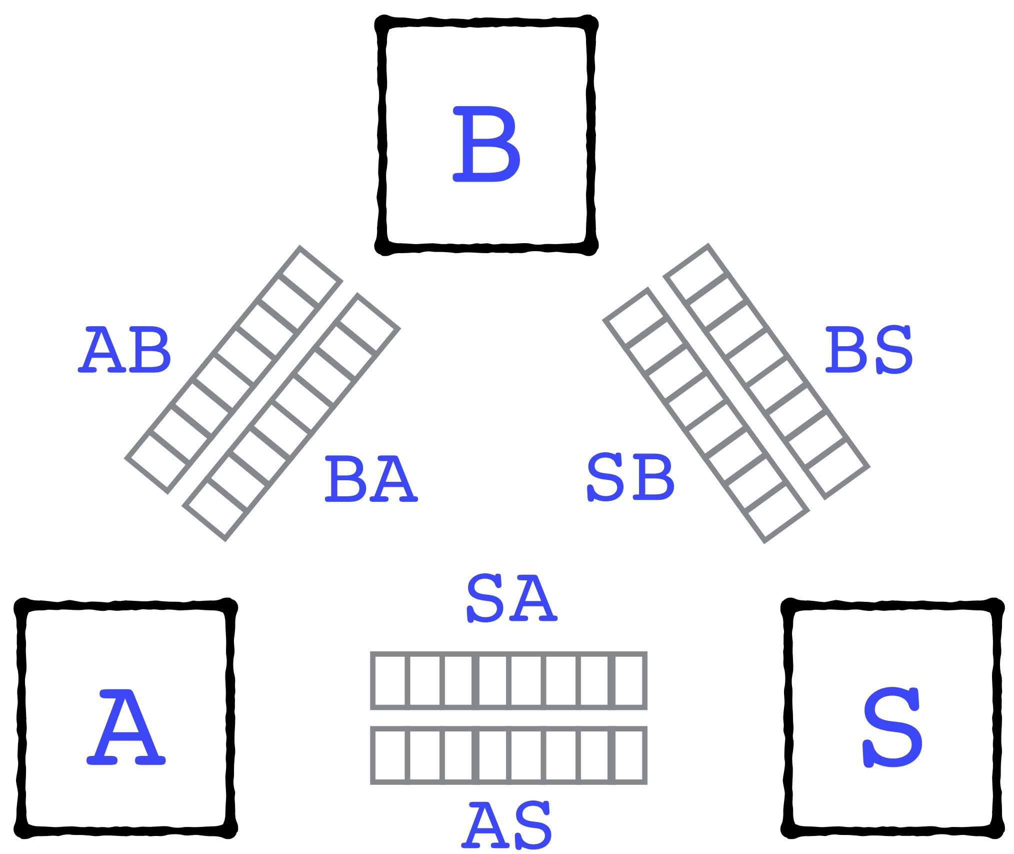
Semantics of global types

- Not directly given in [Honda,Yoshida, Carbone@POPL'08] and [Bettini et al.@CONCUR'08] (but via local types)
- Given in [Deniélou&Yoshida@ICALP'13]
- Key characteristics: **asynchrony** and **distribution**

Deniélou,Yoshida@ICALP'13
"*Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types.*"

Communication channels

- Messages in transit stored in FIFO unbounded queues
  - assume two channels for each pair of roles*



* [Honda,Yoshida, Carbone@POPL'08] gives explicit account of channel's usage and requires additional checks to prevent races. Above we follow [Bettini et al.@CONCUR'08]

Asynchrony

- Each interaction is made of a **send** action and a (separated) **receive** action

- We will use the following (transition) labels

$$\ell \quad ::= \quad \mathtt{pq}!U \quad | \quad \mathtt{pq}?U \quad | \quad \mathtt{pq}!\,\mathsf{l} \quad | \quad \mathtt{pq}?\,\mathsf{l}$$

$\mathtt{p}$ sends $\mathtt{q}$ a message of type U

$\mathtt{q}$ receives from $\mathtt{p}$ a message of type U

… and the same with branching labels…

Asynchrony : example

$$\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle.$$
$$\mathtt{S} \to \mathtt{A} : \langle \mathit{Int} \rangle.$$
$$\mathtt{S} \to \mathtt{B} : \langle \mathit{Int} \rangle.$$
$$\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle.$$
$$G'$$

$\xrightarrow{\mathtt{AS}!\mathit{String}}$

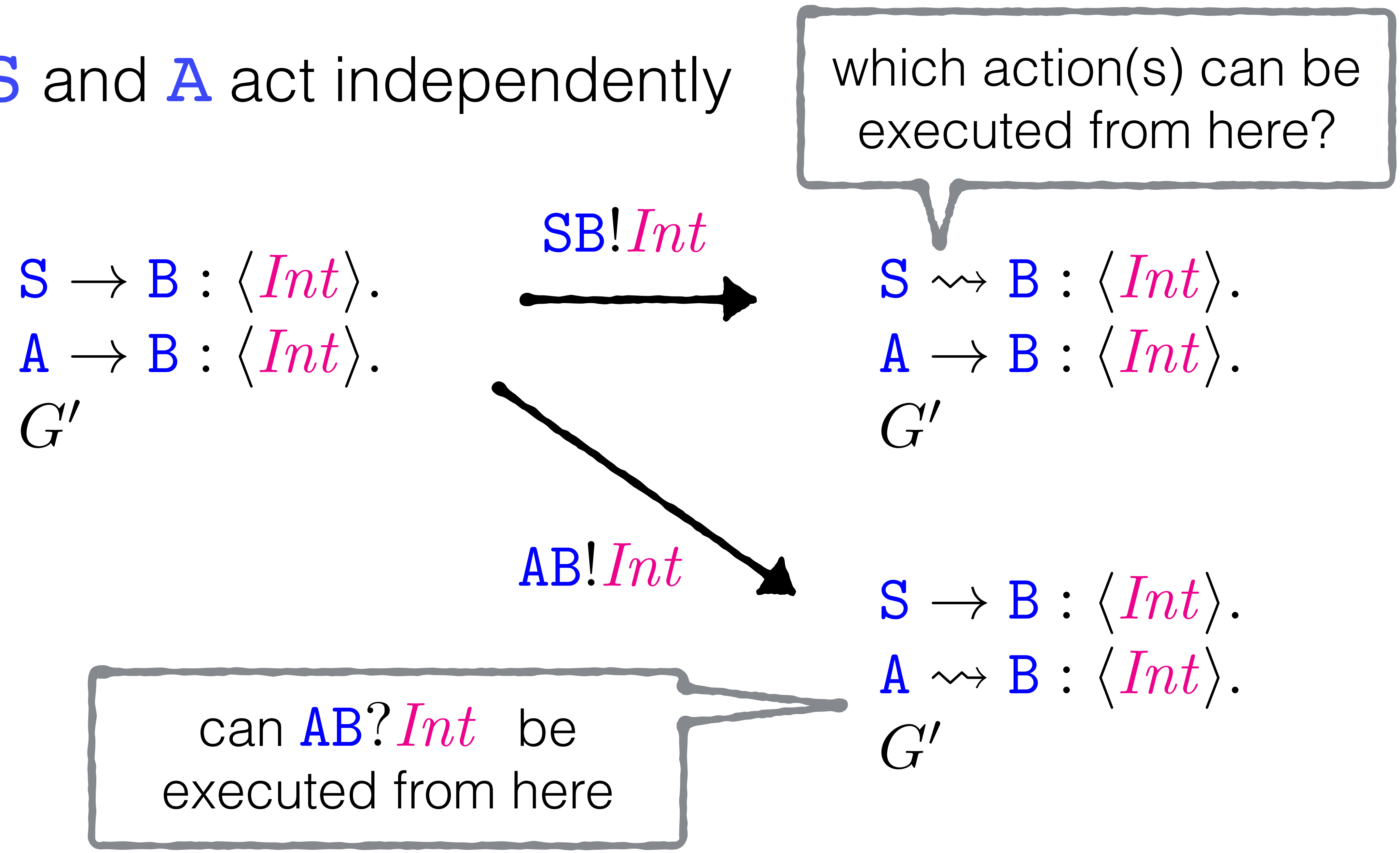
$$\mathtt{A} \rightsquigarrow \mathtt{S} : \langle \mathit{String} \rangle.$$
$$\mathtt{S} \to \mathtt{A} : \langle \mathit{Int} \rangle.$$
$$\mathtt{S} \to \mathtt{B} : \langle \mathit{Int} \rangle.$$
$$\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle.$$
$$G'$$

$\downarrow \; \mathtt{AS}?\mathit{String}$

$$\mathtt{S} \to \mathtt{A} : \langle \mathit{Int} \rangle.$$
$$\mathtt{S} \to \mathtt{B} : \langle \mathit{Int} \rangle.$$
$$\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle.$$
$$G'$$

Distribution

- Participants have a limited knowledge of the global state
- Below, **S** and **A** act independently



On the order of actions

- The **subject** of an action is the role who performs that action

$$subj(\mathtt{pq}!\,U) = \mathtt{p}$$
$$subj(\mathtt{pq}?\,U) = \mathtt{q}$$

- In an execution, it is possible to swap two contiguous actions **unless**:

1. they have the same subject

2. they are corresponding send and receive actions (e.g., of the form $\mathtt{pq}!\,U$ and $\mathtt{pq}?\,U$)

Exercise

$$\mathtt{p} \to \mathtt{q} : \langle U \rangle .\ \mathtt{r} \to \mathtt{s} : \langle U \rangle .\ \mathtt{end}$$

- Which executions are possible?

a. $\mathtt{pq}!U \quad \mathtt{pq}?U \quad \mathtt{rs}!U \quad \mathtt{rs}?U$ ✓

b. $\mathtt{pq}?U \quad \mathtt{pq}!U \quad \mathtt{rs}!U \quad \mathtt{rs}?U$ ✗

c. $\mathtt{pq}!U \quad \mathtt{rs}!U \quad \mathtt{pq}?U \quad \mathtt{rs}?U$ ✓

d. $\mathtt{rs}!U \quad \mathtt{pq}!U \quad \mathtt{pq}?U \quad \mathtt{rs}?U$ ✓

Exercise

- Enumerate all the possible executions of the global type below:

$$\mathtt{p} \to \mathtt{q} : \langle U \rangle .\ \mathtt{q} \to \mathtt{r} : \langle U \rangle .\ \mathtt{end}$$

Multicast vs point to point

$$\begin{aligned}
&\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\mathtt{S} \to \mathtt{A} : \langle \mathit{Int} \rangle. \\
&\mathtt{S} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
&\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
&\mathtt{B} \to \{\mathtt{S}, \mathtt{A}\} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\qquad\qquad\qquad\qquad \mathtt{S} \to \mathtt{B} : \langle \mathit{Date} \rangle.\mathbf{end}, \\
&\qquad\qquad\quad \mathsf{quit} : \mathbf{end}\}
\end{aligned}$$

$$\begin{aligned}
&\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\mathtt{S} \to \{\mathtt{A}, \mathtt{B}\} : \langle \mathit{Int} \rangle. \\
&\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
&\mathtt{B} \to \{\mathtt{S}, \mathtt{A}\} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\qquad\qquad\qquad\qquad \mathtt{S} \to \mathtt{B} : \langle \mathit{Date} \rangle.\mathbf{end}, \\
&\qquad\qquad\quad \mathsf{quit} : \mathbf{end}\}
\end{aligned}$$

Unbounded buffers

$$
\begin{array}{lcl}
G & ::= & \mathtt{p} \to \mathtt{q} : \langle U \rangle . G \\
 & | & \mathtt{p} \to \mathtt{q} : \{ \mathsf{l}_i : G_i \}_{i \in I} \\
 & | & \mu \mathtt{t}.\ G \\
 & | & \mathtt{t} \\
 & | & \mathtt{end}
\end{array}
$$

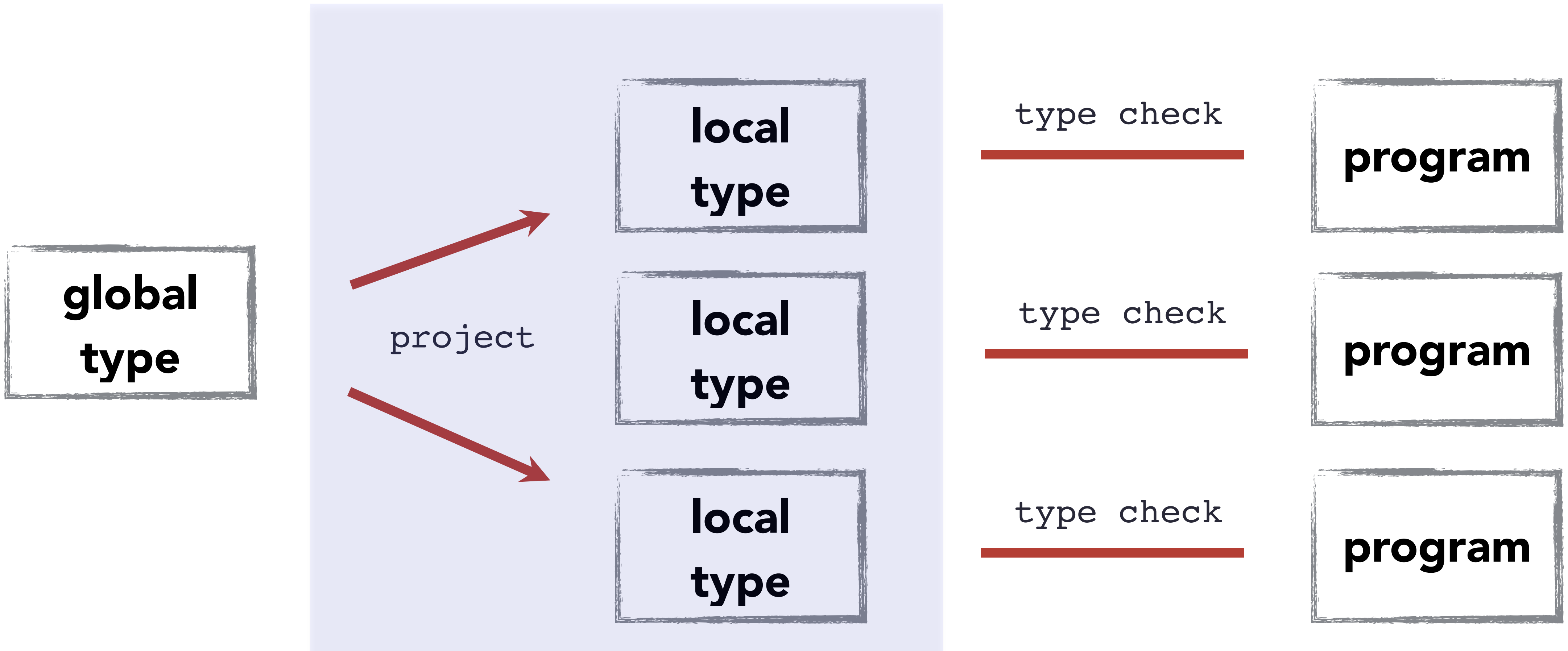
Exercise: think of an example of unbound buffers (use the rules below)

$$
\frac{G[\mu \mathtt{t}.G/\mathtt{t}] \quad \stackrel{\ell}{\to} \quad G'}{\mu \mathtt{t}.G \quad \stackrel{\ell}{\to} \quad G'}
\qquad
\frac{G \quad \stackrel{\ell}{\longrightarrow} \quad G' \qquad \mathtt{p}, \mathtt{q} \notin subj(\ell)}{\mathtt{p} \rightsquigarrow \mathtt{q} : \langle U \rangle . G \quad \stackrel{\ell}{\longrightarrow} \quad \mathtt{p} \rightsquigarrow \mathtt{q} : \langle U \rangle . G'}
$$

Wrapping up

- Global types come with a concise notation
  - … but the syntactic order may differ from the execution order

Multiparty Session Types : the method



- Projection yields a local type for each role
- Why bother?
  - formal model for (modular) verification
  - check that the global type is realisable

Local types

Global Types

$$
\begin{array}{rcl}
G & ::= & \mathsf{p} \to \mathsf{q} : \langle U \rangle . G \\
& \mid & \mathsf{p} \to \mathsf{q} : \{ \mathsf{l}_i : G_i \}_{i \in I} \\
& \mid & \mu \mathtt{t}.\ G \\
& \mid & \mathtt{t} \\
& \mid & \mathtt{end}
\end{array}
$$

Local Types

$$
\begin{array}{rcll}
T & ::= & !\langle \mathsf{p}, U \rangle . T & \text{Send} \\
& \mid & ?\langle \mathsf{p}, U \rangle . T & \text{Receive} \\
& \mid & \oplus \langle \mathsf{p}, \{ \mathsf{l}_i : T_i \}_{i \in I} \rangle & \text{Select} \\
& \mid & \& \langle \mathsf{p}, \{ \mathsf{l}_i : T_i \}_{i \in I} \rangle & \text{Branch} \\
& \mid & \mu \mathtt{t}. T & \text{Rec} \\
& \mid & \mathtt{t} & \text{Call} \\
& \mid & \mathtt{end} & \text{End}
\end{array}
$$

Projection

$$(\mathsf{p} \to \mathsf{p}' : \langle U \rangle . G') \upharpoonright \mathsf{q} = \begin{cases} !\langle \mathsf{p}', U \rangle . G' \upharpoonright \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}, \\ ?(\mathsf{p}, U) . G' \upharpoonright \mathsf{q} & \text{if } \mathsf{q} = \mathsf{p}', \\ G' \upharpoonright \mathsf{q} & \text{otherwise.} \end{cases}$$

$$(\mathsf{p} \to \mathsf{p}' : \{l_i : G_i\}_{i \in I}) \upharpoonright \mathsf{q} = \begin{cases} \oplus \langle \mathsf{p}', \{\mathsf{l}_i : G_i \upharpoonright \mathsf{q}\}_{i \in I} \rangle & \text{if } \mathsf{q} = \mathsf{p}, \\ \&(\mathsf{p}, \{\mathsf{l}_i : G_i \upharpoonright \mathsf{q}\}_{i \in I}) & \text{if } \mathsf{q} = \mathsf{p}', \\ G_{i0} \upharpoonright \mathsf{q} & \text{if } \mathsf{p},\ \mathsf{p}' \neq \mathsf{q},\ i0 \in I,\ \text{and} \\ & G_i \upharpoonright \mathsf{q} = G_j \upharpoonright \mathsf{q} \text{ for all } i, j \in I. \end{cases}$$

$$(\mu \mathtt{t}.G) \upharpoonright \mathsf{q} = \begin{cases} \mu \mathtt{t}.(G \upharpoonright \mathsf{q}) & \text{if } G \upharpoonright \mathsf{q} \neq \mathtt{t}, \\ \mathtt{end} & \text{otherwise.} \end{cases}$$

$$\mathtt{t} \upharpoonright \mathsf{q} = \mathtt{t}$$

$$\mathtt{end} \upharpoonright \mathsf{q} = \mathtt{end}$$

If none of the conditions above applies, then G is **not projectable**

Projection: example

$$
\begin{aligned}
G \quad = \quad & \mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
& \mathtt{S} \to \mathtt{A} : \langle \mathit{Int} \rangle. \\
& \mathtt{S} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
& \mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
& \mathtt{B} \to \mathtt{S} : \{ \mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
& \qquad\qquad\quad \mathtt{S} \to \mathtt{B} : \langle \mathit{Date} \rangle.\mathtt{end}, \\
& \qquad\quad \mathsf{quit} : \mathtt{end} \}
\end{aligned}
$$

$$G \upharpoonright \mathtt{A} =$$

Projection : exercise

$$
\begin{aligned}
G \quad ::= \quad & \mathtt{M} \to \mathtt{W} : \langle \mathit{task} \rangle. \\
& \mathtt{W} \to \mathtt{A} \{ \ \mathsf{ok} : \mathtt{W} \to \mathtt{A} : \langle \mathit{data} \rangle.\ \mathtt{A} \to \mathtt{M} : \langle \mathit{result} \rangle.\mathtt{end}, \\
& \qquad \mathsf{stop} : \ \mathtt{A} \to \mathtt{M} : \langle \mathit{error\_code} \rangle.\mathtt{end} \ \}
\end{aligned}
$$

$G \upharpoonright \mathtt{W} \quad ::=$

$G \upharpoonright \mathtt{M} \quad ::=$

Exercise : branching

- Is G projectable on M?

$$G \quad ::= \quad \mathtt{M} \to \mathtt{W} : \langle task \rangle.$$
$$\mathtt{W} \to \mathtt{A}\{\ \mathsf{ok} : \mathtt{W} \to \mathtt{A} : \langle data \rangle.\ \mathtt{A} \to \mathtt{M} : \{\mathsf{ok} : \mathtt{A} \to \mathtt{M} : \langle result \rangle.\mathtt{end}\},$$
$$\mathsf{stop} :\ \mathtt{A} \to \mathtt{M} : \{\mathsf{stop} :\ \mathtt{A} \to \mathtt{M} : \langle error\_code \rangle.\mathtt{end}\}\ \}$$

- Not with the projection rules we have seen

- But it is projectable with a more powerful definition of projection which include "**mergeability**" [Deniélou&Yoshida@POPL'11]
    - in all branches the "third party" must receive a **label** from the **same sender**
    - **different labels** are merged
    - **same labels** need to have **same-continuation**

Three buyer protocol with notification

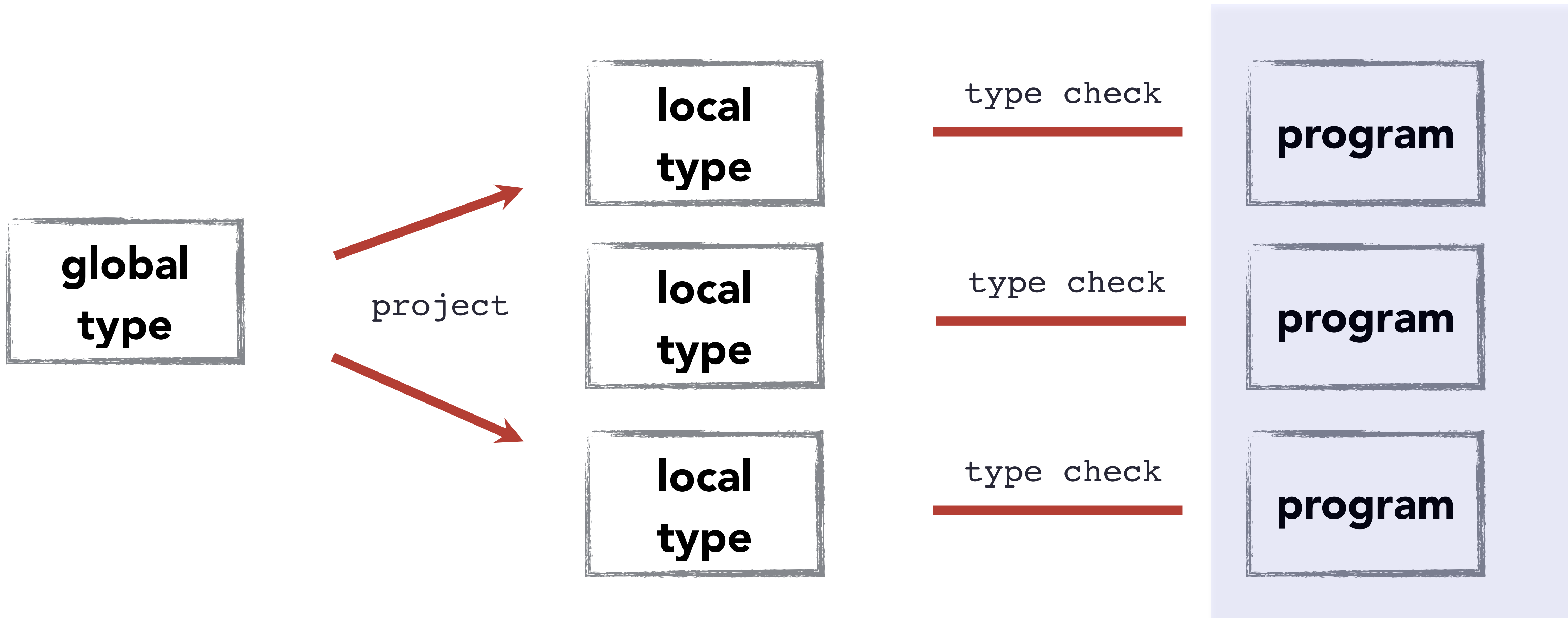
$$
\begin{aligned}
&\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\mathtt{S} \to \{\mathtt{A}, \mathtt{B}\} : \langle \mathit{Int} \rangle. \\
&\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
&\mathtt{B} \to \{\mathtt{S}, \mathtt{A}\} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\qquad\qquad\qquad\qquad \mathtt{S} \to \mathtt{B} : \langle \mathit{Date} \rangle.\mathbf{end}, \\
&\qquad\qquad\quad \mathsf{quit} : \mathbf{end}\}
\end{aligned}
$$

....

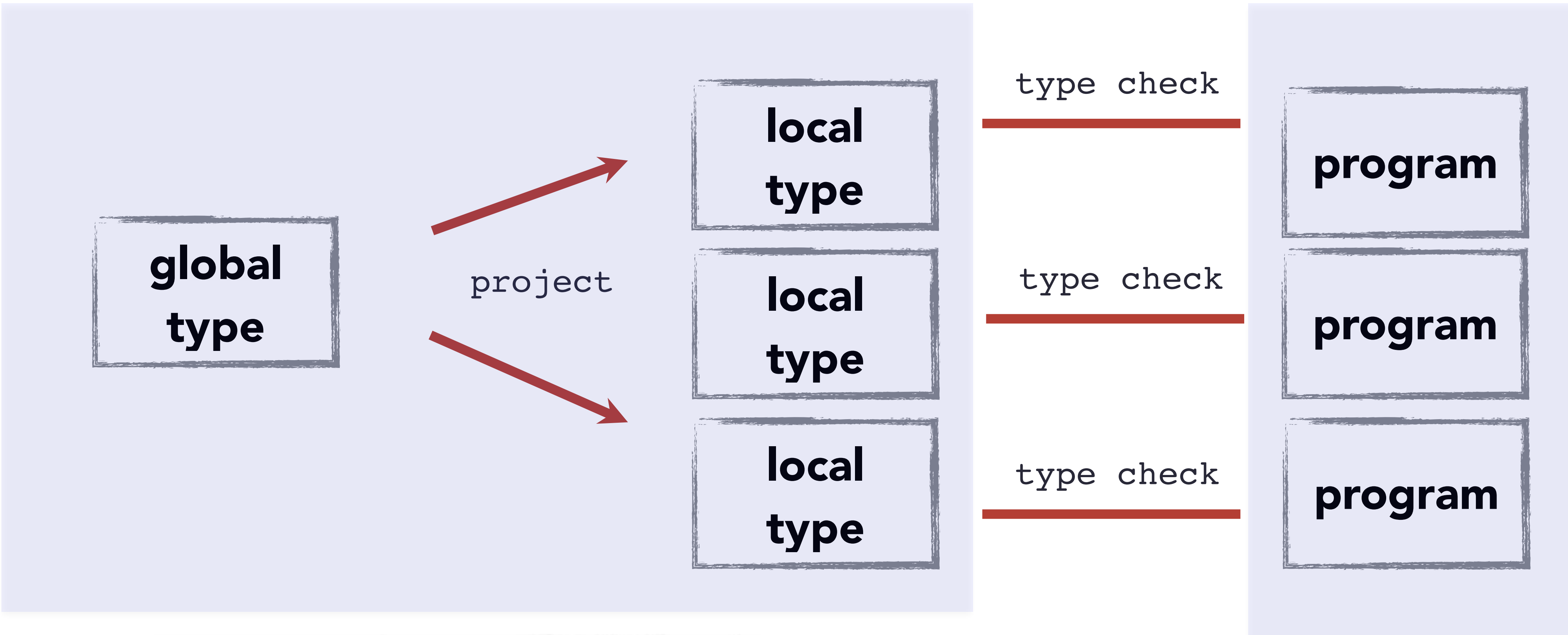
$$
\begin{aligned}
&\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\mathtt{S} \to \{\mathtt{A}, \mathtt{B}\} : \langle \mathit{Int} \rangle. \\
&\mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle. \\
&\mathtt{B} \to \mathtt{S} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{A} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle \mathit{String} \rangle. \\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathtt{S} \to \mathtt{B} : \langle \mathit{Date} \rangle.\mathbf{end}\}, \\
&\qquad\quad \mathsf{quit} : \mathtt{B} \to \mathtt{A} : \{\mathsf{quit} : \mathbf{end}\}\}
\end{aligned}
$$

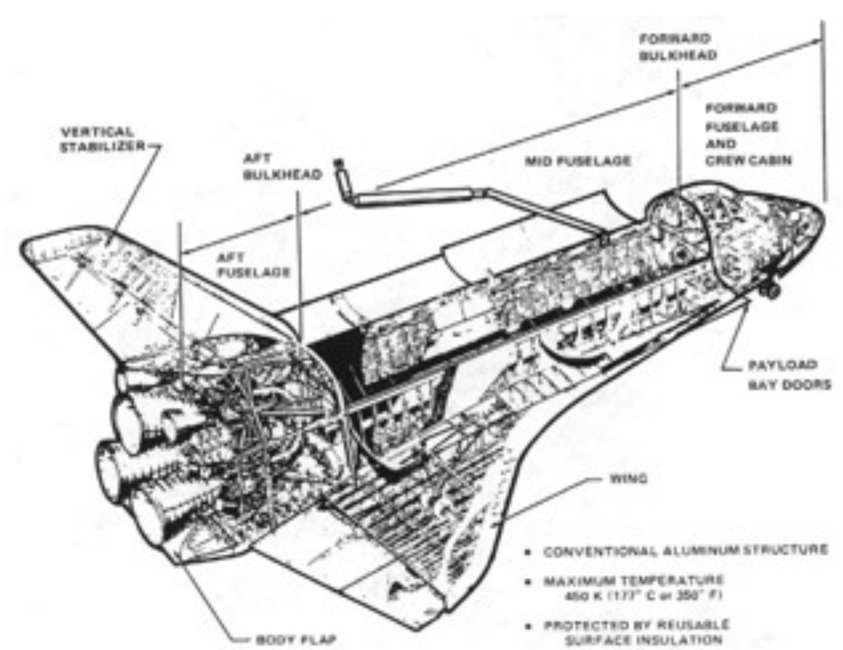
see extras for more on merge…

What's next


global
type
project
local
type
local
type
local
type
type check
type check
type check
program
program
program

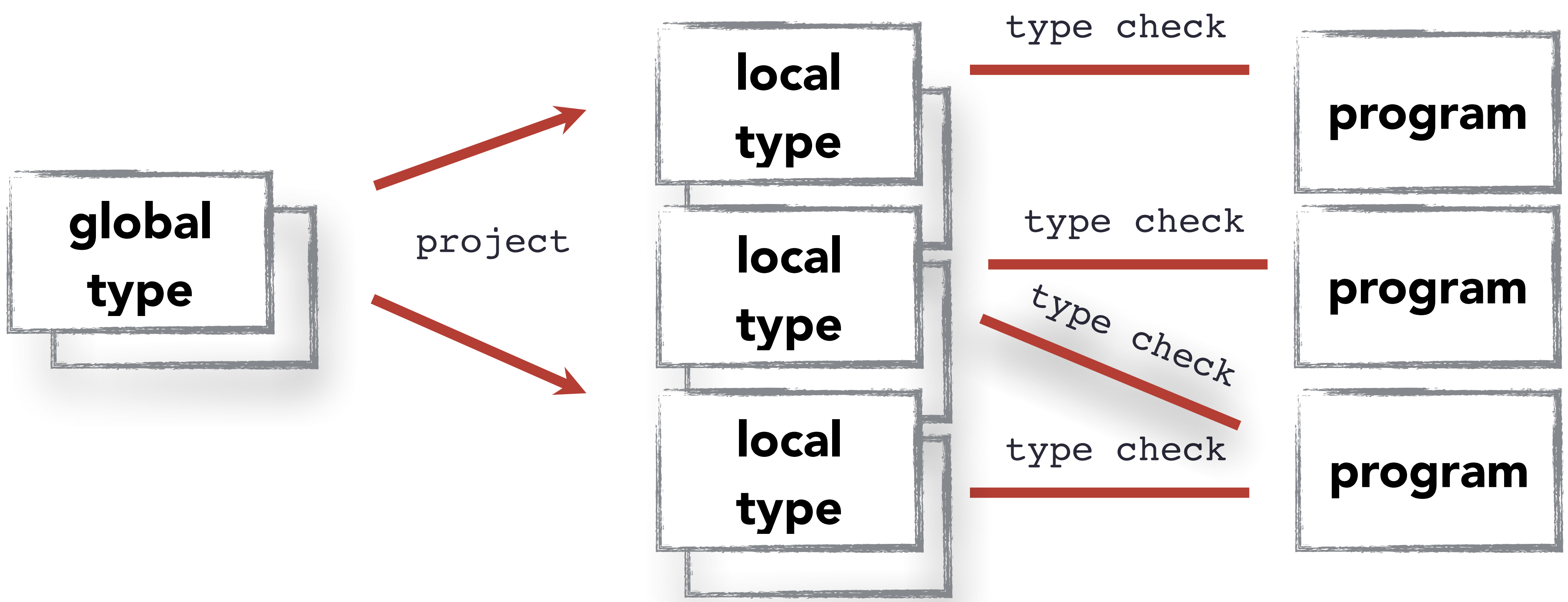
Recap





photos by http://nasa.gov/

Recap



- **Global and local types** specify types of sessions
- **Programs** are
  - code implementing roles in one or more sessions
  - executed —> **processes**

Programs

$$
\begin{array}{rcl}
P & ::= & \overline{u}[\mathtt{p}](y).P \\
 & \mid & u[\mathtt{p}](y).P \\
 & \mid & c!\langle \mathtt{p}, e\rangle.P \\
 & \mid & c?(\mathtt{p}, x).P \\
 & \mid & c \oplus \langle \mathtt{p}, \mathsf{l}\rangle.P \\
 & \mid & c\&(\mathtt{p}, \{\mathsf{l}_i : P_i\}_{i\in I}) \\
 & \mid & c!\langle\langle \mathtt{p}, s'[\mathtt{p}']\rangle\rangle.P \\
 & \mid & c?((\mathtt{p}, y)).P \\
 & \mid & \mathsf{if}\ e\ \mathsf{then}\ P\ \mathsf{else}\ Q \\
 & \mid & P \mid Q \\
 & \mid & \mathbf{0} \\
 & \mid & (\nu a)P \\
 & \mid & \mathsf{def}\ D\ \mathsf{in}\ P \\
 & \mid & X\langle e, c\rangle
\end{array}
$$

The system is a parallel composition of processes

Programs

$$
\begin{array}{lll}
P & ::= & \overline{u}[\mathtt{p}](y).P \\
 & \mid & u[\mathtt{p}](y).P \\
 & \mid & c!\langle \mathtt{p}, e\rangle.P \\
 & \mid & c?(\mathtt{p}, x).P \\
 & \mid & c \oplus \langle \mathtt{p}, \mathsf{l}\rangle.P \\
 & \mid & c\&(\mathtt{p}, \{\mathsf{l}_i : P_i\}_{i \in I}) \\
 & \mid & c!\langle\langle \mathtt{p}, s'[\mathtt{p}']\rangle\rangle.P \\
 & \mid & c?((\mathtt{p}, y)).P \\
 & \mid & \text{if } e \text{ then } P \text{ else } Q \\
 & \mid & P \mid Q \\
 & \mid & \mathbf{0} \\
 & \mid & (\nu a)P \\
 & \mid & \text{def } D \text{ in } P \\
 & \mid & X\langle e, c\rangle
\end{array}
$$

establish sessions

interact in sessions

other constructs

Establishing sessions

service channels $a, a', \ldots$

- used to initiate sessions
- shared

session channels : $s, s', \ldots$

- for session communication
- private

$$[\text{Init}] \quad \overline{a}[\mathtt{n}](y).P_n \mid a[\mathtt{1}](y).P_1 \mid \ldots \mid a[\mathtt{n}-\mathtt{1}](y).P_{n-1}$$

$$\to \quad (\nu s)(P_1[s[1]/y] \mid P_2[s[2]/y] \mid \ldots \mid P_n[s[n]/y] \mid s : \emptyset)$$

Coppo et al.@SFM'15

Processes

$$
\begin{array}{rcll}
P & ::= & \overline{u}[\mathsf{p}](y).P & \text{programs} \\
& | & u[\mathsf{p}](y).P & \\
& | & c!\langle \mathsf{p}, e\rangle.P & \\
& | & c?(\mathsf{p}, x).P & \\
& | & c \oplus \langle \mathsf{p}, \mathsf{l}\rangle.P & \\
& | & c\&(\mathsf{p}, \{\mathsf{l}_i : P_i\}_{i\in I}) & \\
& | & c!\langle\langle \mathsf{p}, s'[\mathsf{p}']\rangle\rangle.P & \\
& | & c?((\mathsf{p}, y)).P & \\
& | & \text{if } e \text{ then } P \text{ else } Q & \\
& | & P \mid Q & \\
& | & \mathbf{0} & \\
& | & (\nu a)P & \\
& | & \text{def } D \text{ in } P & \\
& | & X\langle e, c\rangle & \\
& | & (\nu s)P & \text{run-time processes} \\
& | & s : h &
\end{array}
$$

queue of messages in transit:

$(\mathsf{q}, \mathsf{p}, v)$

$(\mathsf{q}, \mathsf{p}, \mathsf{l})$

Processes

$$
\begin{array}{lcl}
P & ::= & \overline{u}[\mathtt{p}](y).P \\
 & \mid & u[\mathtt{p}](y).P \\
 & \mid & c!\langle \mathtt{p}, e\rangle.P \\
 & \mid & c?(\mathtt{p}, x).P \\
 & \mid & c \oplus \langle \mathtt{p}, \mathsf{l}\rangle.P \\
 & \mid & c\&(\mathtt{p}, \{\mathsf{l}_i : P_i\}_{i\in I}) \\
 & \mid & c!\langle\langle \mathtt{p}, s'[\mathtt{p}']\rangle\rangle.P \\
 & \mid & c?((\mathtt{p}, y)).P \\
 & \mid & \text{if } e \text{ then } P \text{ else } Q \\
 & \mid & P \mid Q \\
 & \mid & \mathbf{0} \\
 & \mid & (\nu a)P \\
 & \mid & \text{def } D \text{ in } P \\
 & \mid & X\langle e, c\rangle
\end{array}
$$

interact in sessions

Sending a message

- What you need
  - a process that is ready to send/select
  - an established session (hence a queue)

$$s[\texttt{p}]!\langle \texttt{q}, e\rangle.P \mid s : h$$

Sending a message

- What you need
  - a process that is ready to send/select
  - an established session (hence a queue)

$$s[\mathtt{p}]!\langle \mathtt{q}, e\rangle.P \mid s : h \quad \rightarrow \quad P \mid s : h \cdot (\mathtt{p}, \mathtt{q}, v) \qquad (e \downarrow v)$$

- What you get
  - a message is placed in the queue
  - the process reduces to its continuation

Communications within sessions (1/2)

$$[\text{Send}] \quad s[\mathtt{p}]!\langle \mathtt{q}, e\rangle.P \mid s : h \quad \rightarrow \quad P \mid s : h \cdot (\mathtt{p}, \mathtt{q}, v) \qquad (e \downarrow v)$$

$$[\text{Rcv}] \quad s[\mathtt{p}]?(\mathtt{q}, x).P \mid s : (\mathtt{q}, \mathtt{p}, v) \cdot h \quad \rightarrow \quad P[v/x] \mid s : h$$

Coppo et al.@SFM'15

- Exercise: give the reduction of the process below

$$s[\mathtt{p}]?(\mathtt{q}, x).s[\mathtt{p}]!\langle \mathtt{q}, x+1\rangle.0 \mid s : (\mathtt{q}, \mathtt{p}, 5)$$

Communications within sessions (2/2)

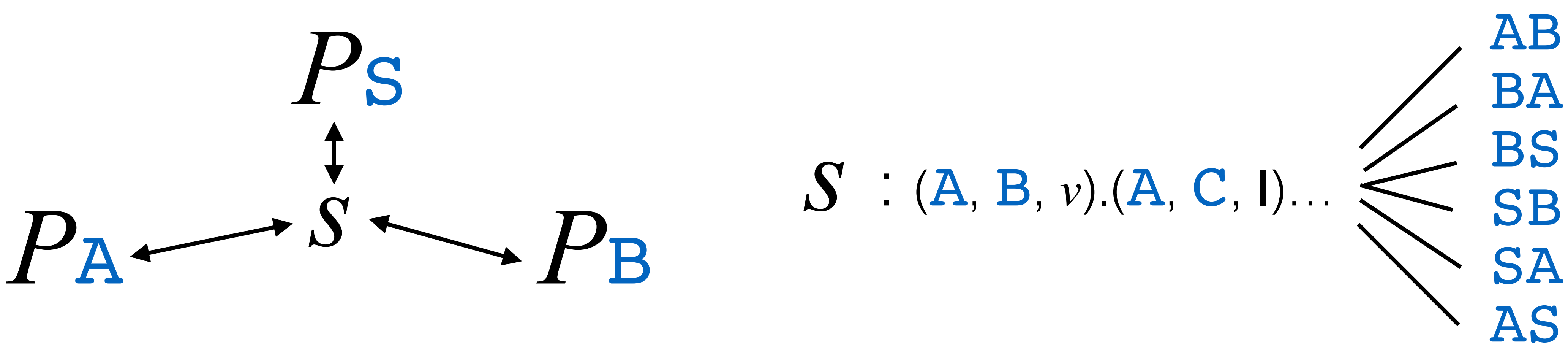
[Sel] $s[\mathtt{p}] \oplus \langle \mathtt{q}, \mathsf{l} \rangle.P \mid s : h \quad \rightarrow \quad P \mid s : h \cdot (\mathtt{p}, \mathtt{q}, \mathsf{l})$

[Bra] $s[\mathtt{p}] \& (\mathtt{q}, \{\mathsf{l}_i : P_i\}_{i \in I}) \mid s : (\mathtt{q}, \mathtt{p}, \mathsf{l}_j) \cdot h \quad \rightarrow \quad P_j \mid s : h \qquad (j \in I)$

Coppo et al.@SFM'15

Recall (queues)

- Recall: we assume two queues for each pair of roles



- FIFO must be preserved for messages “in the same queue”
- The others may need to be swapped to allow reduction

message permutation

$$s : h_1 \cdot (\mathtt{p}, \mathtt{q}, v) \cdot (\mathtt{p}', \mathtt{q}', v') \cdot h_2 \equiv s : h_1 \cdot (\mathtt{p}', \mathtt{q}', v') \cdot (\mathtt{p}, \mathtt{q}, v) \cdot h_2$$
$$\text{if } \mathtt{p} \neq \mathtt{p}' \text{ or } \mathtt{q} \neq \mathtt{q}'.$$

Example (1/2)

$$s[\mathtt{A}]!\langle \mathtt{C}, \text{“}chocolate\text{”}\rangle.s[\mathtt{A}]!\langle \mathtt{C}, 10\rangle.\mathbf{0}$$
$$\mid \quad s[\mathtt{B}] \oplus \langle \mathtt{C}, \mathsf{l}_1\rangle.\mathbf{0}$$
$$\mid \quad s[\mathtt{C}]?(\mathtt{A}, x).s[\mathtt{C}]?(\mathtt{A}, y).s[\mathtt{C}]\&(\mathtt{B}, \{l_i : \mathbf{0}\}_{i\in\{1,2,3\}})$$
$$\mid \quad s : \emptyset$$

$$s[\mathtt{A}]!\langle \mathtt{C}, 10\rangle.\mathbf{0}$$
$$\mid \quad s[\mathtt{C}]?(\mathtt{A}, x).s[\mathtt{C}]?(\mathtt{A}, y).s[\mathtt{C}]\&(\mathtt{B}, \{\mathsf{l}_i : \mathbf{0}\}_{i\in\{1,2,3\}})$$
$$\mid \quad s : (\mathtt{B}, \mathtt{C}, \mathsf{l}_1) \cdot (\mathtt{A}, \mathtt{C}, \text{“}chocolate\text{”})$$

$$s[\mathtt{C}]?(\mathtt{A}, x).s[\mathtt{C}]?(\mathtt{A}, y).s[\mathtt{C}]\&(\mathtt{B}, \{\mathsf{l}_i : \mathbf{0}\}_{i\in\{1,2,3\}})$$
$$\mid \quad s : (\mathtt{B}, \mathtt{C}, \mathsf{l}_1) \cdot (\mathtt{A}, \mathtt{C}, \text{“}chocolate\text{”}) \cdot (\mathtt{A}, \mathtt{C}, 10)$$

Example (2/2)

$$s[\texttt{C}]?(\texttt{A}, x).s[\texttt{C}]?(\texttt{A}, y).s[\texttt{C}]\&(\texttt{B}, \{\mathsf{l}_i : \mathbf{0}\}_{i\in\{1,2,3\}})$$
$$\mid \quad s : (\texttt{B}, \texttt{C}, \mathsf{l}_1) \cdot (\texttt{A}, \texttt{C}, \text{“}\mathit{chocolate}\text{”}) \cdot (\texttt{A}, \texttt{C}, 10)$$

- which of the following permutations are allowed?

$$s : (\texttt{A}, \texttt{C}, \text{“}\mathit{chocolate}\text{”}) \cdot (\texttt{B}, \texttt{C}, \mathsf{l}_1) \cdot (\texttt{A}, \texttt{C}, 10)$$ ✓

$$s : (\texttt{A}, \texttt{C}, \text{“}\mathit{chocolate}\text{”}) \cdot (\texttt{A}, \texttt{C}, 10) \cdot (\texttt{B}, \texttt{C}, \mathsf{l}_1)$$ ✓

$$s : (\texttt{A}, \texttt{C}, 10) \cdot (\texttt{A}, \texttt{C}, \text{“}\mathit{chocolate}\text{”}) \cdot (\texttt{B}, \texttt{C}, \mathsf{l}_1)$$ ✗

Programs

$$
\begin{array}{lll}
P & ::= & \overline{u}[\mathtt{p}](y).P \\
& | & u[\mathtt{p}](y).P \\
& | & c!\langle \mathtt{p}, e\rangle.P \\
& | & c?(\mathtt{p}, x).P \\
& | & c \oplus \langle \mathtt{p}, \mathsf{l}\rangle.P \\
& | & c\&(\mathtt{p}, \{\mathsf{l}_i : P_i\}_{i\in I}) \\
& | & c!\langle\langle \mathtt{p}, s'[\mathtt{p}']\rangle\rangle.P \\
& | & c?((\mathtt{p}, y)).P \\
& | & \text{if } e \text{ then } P \text{ else } Q \\
& | & P \mid Q \\
& | & \mathbf{0} \\
& | & (\nu a)P \\
& | & \text{def } D \text{ in } P \\
& | & X\langle e, c\rangle
\end{array}
$$

other constructs

Example : recursive three buyer protocol

$$
\begin{array}{l}
\mathtt{A} \to \mathtt{S} : \langle \mathit{String} \rangle . \\
\mathtt{S} \to \{\mathtt{A}, \mathtt{B}\} : \langle \mathit{Int} \rangle . \\
\mu \mathtt{t}. \mathtt{A} \to \mathtt{B} : \langle \mathit{Int} \rangle . \\
\quad \mathtt{B} \to \{\mathtt{S}, \mathtt{A}\} : \{\mathsf{ok} : \mathtt{B} \to \mathtt{S} : \langle \mathit{String} \rangle . \\
\qquad\qquad \mathtt{S} \to \mathtt{B} : \langle \mathit{Date} \rangle . \mathtt{end}, \\
\qquad \mathsf{more} : \mathtt{t}, \\
\qquad \mathsf{quit} : \mathtt{end}\}
\end{array}
$$

- Focusing on **Bob**

$$
G \upharpoonright \mathtt{B} = ?(\mathtt{S}, \mathit{Int}).\ \mu \mathtt{t}.\ ?(\mathtt{A}, \mathit{Int}). \oplus (\{\mathtt{S}, \mathtt{A}\}, \{\mathsf{ok} : T', \mathsf{more} : \mathtt{t}, \mathsf{quit} : \mathtt{end}\})
$$

$$
T' = !\langle \mathtt{S}, \mathit{String} \rangle . ?(\mathtt{S}, \mathit{Date}). \mathtt{end}
$$

An implementation of Bob

$$G \restriction \mathtt{B} = ?(\mathtt{S}, Int).\ \mu \mathtt{t}.\ ?(\mathtt{A}, Int). \oplus (\{\mathtt{S}, \mathtt{A}\}, \{\mathsf{ok} : T', \mathsf{more} : \mathtt{t}, \mathsf{quit} : \mathtt{end}\})$$

$$T' = !\langle \mathtt{S}, String\rangle.?(\mathtt{S}, Date).\mathtt{end}$$

$$
\begin{array}{lcl}
\mathtt{Bob} & = & a[\mathtt{B}](y').y'?(\mathtt{S}, x_{quote}). \\
 & & \mathtt{def}\ X(x, y) \ = P\ \mathtt{in}\ X\langle 0, y'\rangle \\
P & = & y?(\mathtt{A}, x_{contrib}). \\
 & & \mathtt{if}\ (x_{quote} - x_{contrib} \leq 100) \\
 & & \mathtt{then} \\
 & & \quad y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok}\rangle. \\
 & & \quad y!\langle \mathtt{S}, \text{“CT27NF”}\rangle. \\
 & & \quad y?(\mathtt{S}, x_{date}).\mathbf{0} \\
 & & \mathtt{else} \\
 & & \quad \mathtt{if}\ (x_{contrib} > x + 10)\ \mathtt{then}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit}\rangle.\mathbf{0} \\
 & & \quad \mathtt{else}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more}\rangle.X(x_{contrib}, y)
\end{array}
$$

An implementation of Bob

$$G \upharpoonright \mathtt{B} = ?(\mathtt{S}, Int).\ \mu\mathtt{t}.\ ?(\mathtt{A}, Int). \oplus (\{\mathtt{S}, \mathtt{A}\}, \{\mathsf{ok} : T', \mathsf{more} : \mathtt{t}, \mathsf{quit} : \mathtt{end}\})$$

$$T' = !\langle \mathtt{S}, String \rangle.?(\mathtt{S}, Date).\mathtt{end}$$

$$
\begin{array}{lcl}
\mathtt{Bob} & = & a[\mathtt{B}](y').y'?(\mathtt{S}, x_{quote}). \\
 & & \mathtt{def}\ X(x, y) \ = P\ \mathtt{in}\ X\langle 0, y' \rangle \\
P & = & y?(\mathtt{A}, x_{contrib}). \\
 & & \mathtt{if}\ (x_{quote} - x_{contrib} \leq 100) \\
 & & \mathtt{then} \\
 & & \quad y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok} \rangle. \\
 & & \quad y!\langle \mathtt{S}, \text{“CT27NF”} \rangle. \\
 & & \quad y?(\mathtt{S}, x_{date}).\mathbf{0} \\
 & & \mathtt{else} \\
 & & \quad \mathtt{if}\ (x_{contrib} > x + 10)\ \mathtt{then}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit} \rangle.\mathbf{0} \\
 & & \quad \mathtt{else}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more} \rangle.X(x_{contrib}, y)
\end{array}
$$

An implementation of Bob

$$G \upharpoonright \mathtt{B} = ?(\mathtt{S}, Int).\ \mu \mathtt{t}.\ ?(\mathtt{A}, Int). \oplus (\{\mathtt{S}, \mathtt{A}\}, \{\mathsf{ok} : T', \mathsf{more} : \mathtt{t}, \mathsf{quit} : \mathtt{end}\})$$

$$T' = !\langle \mathtt{S}, String \rangle.?(\mathtt{S}, Date).\mathtt{end}$$

$$
\begin{array}{lcl}
\mathtt{Bob} & = & a[\mathtt{B}](y').y'?(\mathtt{S}, x_{quote}). \\
 & & \mathtt{def}\ X(x, y) = P\ \mathtt{in}\ X\langle 0, y' \rangle \\
P & = & y?(\mathtt{A}, x_{contrib}). \\
 & & \mathtt{if}\ (x_{quote} - x_{contrib} \leq 100) \\
 & & \mathtt{then} \\
 & & \quad y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok} \rangle. \\
 & & \quad y!\langle \mathtt{S}, \text{“CT27NF”} \rangle. \\
 & & \quad y?(\mathtt{S}, x_{date}).\mathbf{0} \\
 & & \mathtt{else} \\
 & & \quad \mathtt{if}\ (x_{contrib} > x + 10)\ \mathtt{then}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit} \rangle.\mathbf{0} \\
 & & \quad \mathtt{else}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more} \rangle.X(x_{contrib}, y)
\end{array}
$$

An implementation of Bob

$$G \upharpoonright \mathtt{B} = ?(\mathtt{S}, Int).\ \mu \mathtt{t}.\ ?(\mathtt{A}, Int).\ \oplus(\{\mathtt{S},\mathtt{A}\}, \{\mathsf{ok} : T', \mathsf{more} : \mathtt{t}, \mathsf{quit} : \mathtt{end}\})$$

$$T' = !\langle \mathtt{S}, String\rangle.?(\mathtt{S}, Date).\mathtt{end}$$

$$\begin{array}{lcl}
\mathtt{Bob} & = & a[\mathtt{B}](y').y'?(\mathtt{S}, x_{quote}). \\
 & & \mathtt{def}\ X(x, y) = P\ \mathtt{in}\ X\langle 0, y'\rangle \\
P & = & y?(\mathtt{A}, x_{contrib}). \\
 & & \mathtt{if}\ (x_{quote} - x_{contrib} \leq 100) \\
 & & \mathtt{then} \\
 & & \quad y \oplus \langle\{\mathtt{S},\mathtt{A}\}, \mathsf{ok}\rangle. \\
 & & \quad y!\langle \mathtt{S}, \text{“CT27NF”}\rangle. \\
 & & \quad y?(\mathtt{S}, x_{date}).\mathbf{0} \\
 & & \mathtt{else} \\
 & & \quad \mathtt{if}\ (x_{contrib} > x + 10)\ \mathtt{then}\ y \oplus \langle\{\mathtt{S},\mathtt{A}\}, \mathsf{quit}\rangle.\mathbf{0} \\
 & & \quad \mathtt{else}\ y \oplus \langle\{\mathtt{S},\mathtt{A}\}, \mathsf{more}\rangle.X(x_{contrib}, y)
\end{array}$$

An implementation of Bob

$$G \upharpoonright \mathtt{B} = ?(\mathtt{S}, \mathit{Int}).\ \mu\mathtt{t}.\ ?(\mathtt{A}, \mathit{Int}).\oplus(\{\mathtt{S}, \mathtt{A}\}, \{\mathsf{ok} : T', \mathsf{more} : \mathtt{t}, \mathsf{quit} : \mathtt{end}\})$$

$$T' = !\langle \mathtt{S}, \mathit{String}\rangle.?(\mathtt{S}, \mathit{Date}).\mathtt{end}$$

$$
\begin{array}{lcl}
\mathtt{Bob} & = & a[\mathtt{B}](y').y'?(\mathtt{S}, x_{quote}). \\
& & \mathtt{def}\ X(x, y) = P\ \mathtt{in}\ X\langle 0, y'\rangle \\
P & = & y?(\mathtt{A}, x_{contrib}). \\
& & \mathtt{if}\ (x_{quote} - x_{contrib} \leq 100) \\
& & \mathtt{then} \\
& & \quad y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok}\rangle. \\
& & \quad y!\langle \mathtt{S}, \text{“CT27NF”}\rangle. \\
& & \quad y?(\mathtt{S}, x_{date}).\mathbf{0} \\
& & \mathtt{else} \\
& & \quad \mathtt{if}\ (x_{contrib} > x + 10)\ \mathtt{then}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit}\rangle.\mathbf{0} \\
& & \quad \mathtt{else}\ y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more}\rangle.X(x_{contrib}, y)
\end{array}
$$

A bit of semantics by examples

$$a[\texttt{B}](y').y'?(\texttt{S}, x_{quote}).\ \texttt{def}\ X(x,y) = P\ \texttt{in}\ X\langle 0, y'\rangle \mid \overline{a}[\texttt{A}](y').P_a \mid a[\texttt{S}](y').P_s$$

$\Downarrow^*$

$$(\nu s)\ s[\texttt{B}]?(\texttt{S}, x_{quote}).\ \texttt{def}\ X(x,y) = P\ \texttt{in}\ X\langle 0, s[\texttt{B}]\rangle \mid P'_a \mid P'_s \mid s : (\texttt{S}, \texttt{B}, 150)$$

$\Downarrow$

$$(\nu s)\ \texttt{def}\ X(x,y) = P[150/x_{quote}]\ \texttt{in}\ X\langle 0, s[\texttt{B}]\rangle \mid P'_a \mid P'_s \mid s : \emptyset$$

Semantics of recursion

[ProcCall] $\texttt{def } X(x,y) = P \texttt{ in } (X\langle e, s[\texttt{p}]\rangle \mid Q)$

$\rightarrow \texttt{ def } X(x,y) = P \texttt{ in } (P[v/x][s[\texttt{p}]/y] \mid Q)$

Coppo et al.@SFM'15

$\texttt{def } X(x,y) \ = P[150/x_{quote}] \texttt{ in } X\langle 0, s[\texttt{B}]\rangle$

$\rightarrow \texttt{def } X(x,y) \ = P[150/x_{quote}] \texttt{ in } P[150/x_{quote}][0/x][s[\texttt{B}]/y]$

Semantics of recursion

$$\texttt{def}\ X(x,y) = P[150/x_{quote}]\ \texttt{in}\ P[150/x_{quote}][0/x][s[\mathsf{B}]/y]$$

$s[\mathsf{B}]?(\mathtt{A}, x_{contrib}).$
$\texttt{if}\ (150 - x_{contrib} \leq 100)$
$\texttt{then}$
$\quad s[\mathsf{B}] \oplus \langle\{\mathtt{S}, \mathtt{A}\}, \mathsf{ok}\rangle.$
$\quad s[\mathsf{B}]!\langle \mathtt{S}, \text{“CT27NF”}\rangle.$
$\quad s[\mathsf{B}]?(\mathtt{S}, x_{date}).\mathbf{0}$
$\texttt{else}$
$\quad \texttt{if}\ (x_{contrib} > 0 + 10)\ \texttt{then}\ s[\mathsf{B}] \oplus \langle\{\mathtt{S}, \mathtt{A}\}, \mathsf{quit}\rangle.\mathbf{0}$
$\quad \texttt{else}\ s[\mathsf{B}] \oplus \langle\{\mathtt{S}, \mathtt{A}\}, \mathsf{more}\rangle.X(x_{contrib}, s[\mathsf{B}])$

…ignoring the context and assuming (A,B,5) *is in the queue…*

Semantics of conditional

$$
\begin{array}{l}
\texttt{if}\ (150 - 5 \leq 100) \\
\texttt{then} \\
\quad s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok} \rangle. \\
\quad s[\mathsf{B}]!\langle \mathtt{S}, \text{“CT27NF”} \rangle. \\
\quad s[\mathsf{B}]?(\mathtt{S}, x_{date}).\mathbf{0} \\
\texttt{else} \\
\quad \texttt{if}\ (5 > 0 + 10)\ \texttt{then}\ s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit} \rangle.\mathbf{0} \\
\quad \texttt{else}\ s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more} \rangle.X(5, s[\mathsf{B}])
\end{array}
$$

$$
\begin{array}{llll}
[\text{If\_T}] & \text{if } e \text{ then } P \text{ else } Q & \rightarrow & P \quad (e \downarrow \texttt{true}) \\
[\text{If\_F}] & \text{if } e \text{ then } P \text{ else } Q & \rightarrow & Q \quad (e \downarrow \texttt{false})
\end{array}
$$

Coppo et al.@SFM’15

Semantics of conditional

$$\texttt{if } (150 - 5 \leq 100) \\ \texttt{then} \\ \quad s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok}\rangle. \\ \quad s[\mathsf{B}]!\langle \mathtt{S}, \text{"CT27NF"}\rangle. \\ \quad s[\mathsf{B}]?(\mathtt{S}, x_{date}).\mathbf{0} \\ \texttt{else} \\ \quad \texttt{if } (5 > 0 + 10) \texttt{ then } s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit}\rangle.\mathbf{0} \\ \quad \texttt{else } s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more}\rangle.X(5, s[\mathsf{B}])$$

$$s[\mathsf{B}] \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more}\rangle.X(5, s[\mathsf{B}]) \quad \Rightarrow \quad X(5, s[\mathsf{B}])$$

...ignoring the context...

Back to recursion

$$\texttt{def } X(x,y) \;=\; P[150/x_{quote}] \texttt{ in } X(5, s[\mathtt{B}])$$

- P is executed again, within the same session s[B], but with a different value for x

$$
\begin{array}{lcl}
P & = & y?(\mathtt{A}, x_{contrib}). \\
 & & \texttt{if } (x_{quote} - x_{contrib} \leq 100) \\
 & & \texttt{then} \\
 & & \quad y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{ok} \rangle. \\
 & & \quad y!\langle \mathtt{S}, \text{“CT27NF”} \rangle. \\
 & & \quad y?(\mathtt{S}, x_{date}).\mathbf{0} \\
 & & \texttt{else} \\
 & & \quad \texttt{if } (x_{contrib} > x + 10) \texttt{ then } y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{quit} \rangle.\mathbf{0} \\
 & & \quad \texttt{else } y \oplus \langle \{\mathtt{S}, \mathtt{A}\}, \mathsf{more} \rangle.X(x_{contrib}, y)
\end{array}
$$

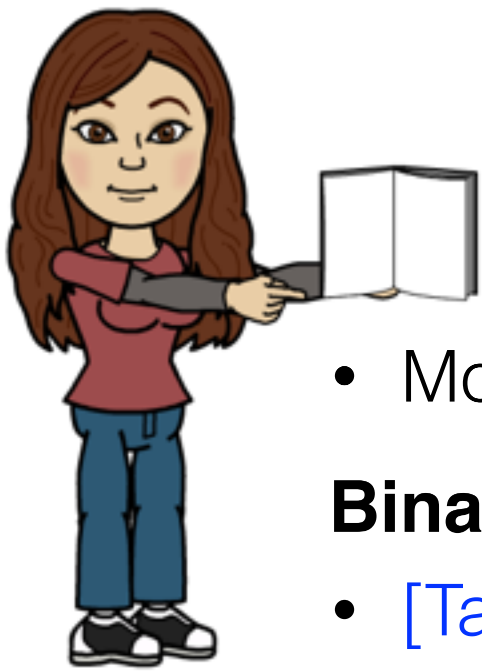
More on the semantics…

- See extra at the end for quick reference
  - See (better) the whole paper!

[Coppo et al.@SFM'15]

Wrapping up

- Today we have seen
  - types
  - processes
  - some hints on types-process “correspondence”
- Next time we will see
  - typing
  - properties guaranteed by typing



References

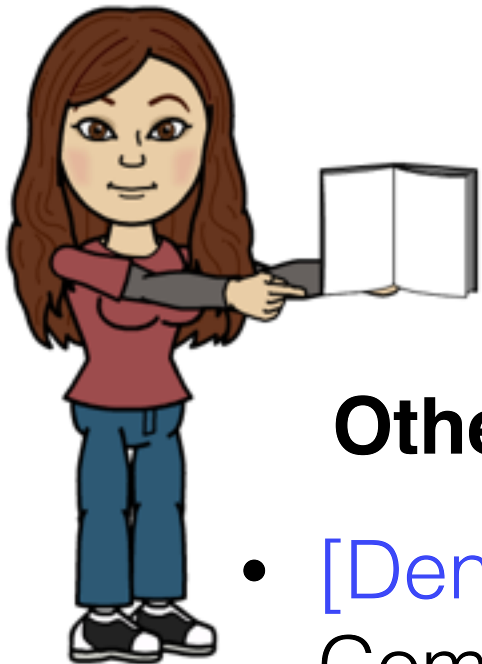
- Mobility Reading Group's home page **http://mrg.doc.ic.ac.uk**

Binary

- [Takeuchi,Honda,Kubo@PARLE'94] An Interaction-Based Language and its Typing System
- [Honda,Vasconcelos,Kubo@ESOP'98] Language Primitives and Type Disciplines for Structured Communication-based Programming

Multiparty

- [Honda,Yoshida,Carbone@POPL'08] Multiparty asynchronous session types
- [Bettini et al.@CONCUR'08] Global Progress in Dynamically Interleaved Multiparty Sessions
- [Castagna, Dezani-Ciancaglini, Padovani@FTDS'12] On Global Types and Multi-Party Sessions
- [Deniélou,Yoshida@POPL'11] Dynamic multirole session types
- [Coppo et al.@SFM'15] Gentle Introduction to Multiparty Asynchronous Session Types



References

Others

- [Deniélou,Yoshida@ICALP'13] Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types
- [Deniélou,Yoshida@POPL'11] Dynamic Multirole Session Types.
- [COB'14, TGC/13] The Scribble protocol language
- [Beljeri,Yoshida@PLACES'08] Synchronous Multiparty Session types
- [Kouzapas,Yoshida@CONCUR'13] Globally Governed Session Semantics

Extras

- Semantics of global types
- More on channels
- Semantics & structural equivalence of processes
- Exercise (processes)

Global semantics (part 1)

[Snd] $\mathtt{p} \to \mathtt{p}' : \langle U \rangle . G' \xrightarrow{\mathtt{pp}'!U} \mathtt{p} \rightsquigarrow \mathtt{p}' : \langle U \rangle . G'$

[Rcv] $\mathtt{p} \rightsquigarrow \mathtt{p}' : \langle U \rangle . G' \xrightarrow{\mathtt{pp}'?U} G'$

[Sel] $\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i \in I} \xrightarrow{\mathtt{pq}!l_i} \mathtt{p} \rightsquigarrow \mathtt{q} : \{l_i : G_i\}$

[Bra] $\mathtt{p} \rightsquigarrow \mathtt{q} : \{l_i : G_i\} \xrightarrow{\mathtt{pq}?l_i} G_i$

[Rec] $$\frac{G[\mu \mathtt{t}.G/\mathtt{t}] \xrightarrow{\ell} G'}{\mu \mathtt{t}.G \xrightarrow{\ell} G'}$$

adapted from
Deniélou, Yoshida@ICALP'13

Global semantics (part 2)

[Async1]

$$\frac{\forall j \in I \quad G_j \xrightarrow{\ell} G'_j \quad \mathtt{p}, \mathtt{q} \notin subj(\ell)}{\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i \in I} \xrightarrow{\ell} \mathtt{p} \to \mathtt{q} : \{l_i : G'_i\}_{i \in I}}$$

[Async1']

$$\frac{G \xrightarrow{\ell} G' \quad \mathtt{p}, \mathtt{q} \notin subj(\ell)}{\mathtt{p} \to \mathtt{q} : \langle U \rangle . G \xrightarrow{\ell} \mathtt{p} \to \mathtt{q} : \langle U \rangle . G'}$$

[Async2]

$$\frac{G_i \xrightarrow{\ell} G'_i \quad \mathtt{p}, \mathtt{q} \notin subj(\ell)}{\mathtt{p} \rightsquigarrow \mathtt{q} : \{l_i : G_i\} \xrightarrow{\ell} \mathtt{p} \rightsquigarrow \mathtt{q} : \{l_i : G'_i\}}$$

[Async2']

$$\frac{G \xrightarrow{\ell} G' \quad \mathtt{p}, \mathtt{q} \notin subj(\ell)}{\mathtt{p} \rightsquigarrow \mathtt{q} : \langle U \rangle . G \xrightarrow{\ell} \mathtt{p} \rightsquigarrow \mathtt{q} : \langle U \rangle . G'}$$

adapted from
Deniélou,Yoshida@ICALP'13

Applying the global semantics

[Async1]
$$\frac{\forall j \in I \qquad G_j \xrightarrow{\ell} G'_j \qquad \mathtt{p},\mathtt{q} \notin subj(\ell)}{\mathtt{p} \to \mathtt{q} : \{l_i : G_i\}_{i\in I} \xrightarrow{\ell} \mathtt{p} \to \mathtt{q} : \{l_i : G'_i\}_{i\in I}}$$

[Async1']
$$\frac{G \xrightarrow{\ell} G' \qquad \mathtt{p},\mathtt{q} \notin subj(\ell)}{\mathtt{p} \to \mathtt{q} : \langle U \rangle.G \xrightarrow{\ell} \mathtt{p} \to \mathtt{q} : \langle U \rangle.G'}$$

adapted from
Deniélou,Yoshida@ICALP'13

Exercise : *Which labels can be produced by applying rule* [Asynch1] *to **G** ?*

$$G = \mathtt{p} \to \mathtt{q} : \langle U \rangle.\ \mathtt{p} \to \mathtt{r} : \langle U \rangle.\ \mathtt{s} \to \mathtt{t} : \langle U \rangle.G'$$

Global semantics (part 3)

- A local semantics is also given
  - each local type can make a send/receive step by "interacting" with a queue
  - when one local type makes a step then the whole system make a step (interleaved semantics)

Theorem 3.1 (soundness and completeness). *Let G be a global type with participants $\mathcal{P}$ and let $\vec{T} = \{G \upharpoonright \mathtt{p}\}_{\mathtt{p}\in\mathcal{P}}$ be the local types projected from G. Then $G \approx (\vec{T}; \vec{\varepsilon})$.*

Deniélou,Yoshida@ICALP'13

More on channels

- In [Honda,Yoshida,Carbone@POPL'08] channels are defined explicitly and each can be used by different participants
- This creates risk of races on channel and requires extra checks that channels are used linearly (with no races).

Figure 5 Causality Analysis

(II) Good	(II) Bad	(IO) Good	(IO) Bad	(OO, II) Good	(OI) Bad
$A \to B : s$	$A \to B : s$	$A \to B : s$	$A \to B : s$	$A \to B : s$	$A \to B : s$
$C \to B : t$	$C \to B : s$	$B \to C : t$	$B \to C : s$	$A \to B : s$	$C \to A : s$
$s! \mid s?; t? \mid t!$	$s! \mid s?; s? \mid s!$	$s! \mid s?; t! \mid t?$	$s! \mid s?; s! \mid s?$	$s!; s! \mid s?; s?$	$s!; s? \mid s? \mid s!$

Honda,Yoshida,Carbone@POPL'08

Example: sending type G

$$((\nu a)a'[1](y).y!\langle 2, a\rangle.\overline{a}[2](y').P) \mid \overline{a'}[2](y).y?(1, x).x[2](y').Q)$$

$$\rightarrow$$

$$(\nu s)(((\nu a)s[1]!\langle 2, a\rangle.\overline{a}[2](y').P) \mid s[2]?(1, x).x[2](y').Q \mid s : \emptyset)$$

$$\rightarrow$$

$$(\nu s)(((\nu a)\overline{a}[2](y').P) \mid s[2]?(1, x).x[2](y').Q \mid s : (1, 2, a))$$

$$\rightarrow$$

$$(\nu s)(\nu a)\ \overline{a}[2](y').P) \mid a[2](y').Q) \mid s : \emptyset$$

$$\rightarrow$$

$$(\nu s)(\nu a)(\nu s')\ P[s'[2]/y'] \mid Q[s'[1]/y'] \mid s : \emptyset \mid s' : \emptyset$$

Mergeability

$$(\mathtt{p} \to \mathtt{p}' : \{l_i : G_i\}_{i\in I}) \upharpoonright \mathtt{q} = \begin{cases} \oplus\langle \mathtt{p}', \{l_i : G_i \upharpoonright \mathtt{q}\}_{i\in I}\rangle & \text{if } \mathtt{q} = \mathtt{p},\\ \&\langle \mathtt{p}, \{l_i : G_i \upharpoonright \mathtt{q}\}_{i\in I}\rangle & \text{if } \mathtt{q} = \mathtt{p}',\\ \sqcup_{i\in I} G_i \upharpoonright \mathtt{q} & \text{if } \mathtt{p},\ \mathtt{p}' \neq \mathtt{q} \end{cases}$$

merge!

- Basically:
  - the first action of **q** in all ***Gi*** **must** be receiving a label from a unique role
  - the merge will include the union of the labels in all ***Gi*** (each with the respective continuation)
  - if two or more ***Gi*** have the same label then their type (projected on **q**) **must** be the same

Deniélou,Yoshida@POPL'11

Equivalence

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

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

$$(\nu r)(\nu r')P \equiv (\nu r')(\nu r)P \quad (\nu a)\mathbf{0} \equiv \mathbf{0} \quad (\nu s)(s : \varnothing) \equiv \mathbf{0}$$

$$\text{where } r ::= a \mid s$$

$$\mathsf{def}\ D\ \mathsf{in}\ \mathbf{0} \equiv \mathbf{0} \quad \mathsf{def}\ D\ \mathsf{in}\ (\nu r)P \equiv (\nu r)\mathsf{def}\ D\ \mathsf{in}\ P \quad \text{if } r \notin \text{fn}(D)$$

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

$$\mathsf{def}\ D\ \mathsf{in}\ (\mathsf{def}\ D'\ \mathsf{in}\ P) \equiv \mathsf{def}\ D'\ \mathsf{in}\ (\mathsf{def}\ D\ \mathsf{in}\ P)$$
$$\text{if } (\text{dpv}(D) \cup \text{fpv}(D)) \cap \text{dpv}(D') = \text{dpv}(D) \cap (\text{dpv}(D') \cup \text{fpv}(D')) = \emptyset$$

$$s : h \cdot (\mathrm{q}, \mathrm{p}, \zeta) \cdot (\mathrm{q}', \mathrm{p}', \zeta') \cdot h' \equiv s : h \cdot (\mathrm{q}', \mathrm{p}', \zeta') \cdot (\mathrm{q}, \mathrm{p}, \zeta) \cdot h' \quad \text{if } \mathrm{p} \neq \mathrm{p}' \text{ or } \mathrm{q} \neq \mathrm{q}'$$

Table 3. Structural equivalence.

Coppo et al.@SFM'15

Semantics

$$a[1](y).P_1 \mid \dots \mid a[n-1](y).P_{n-1} \mid \overline{a}[n](y).P_n \longrightarrow$$
$$(\nu s)(P_1\{s[1]/y\} \mid \dots \mid P_{n-1}\{s[n-1]/y\} \mid P_n\{s[n]/y\} \mid s : \varnothing) \qquad \text{[Init]}$$

$$s[\mathrm{p}]!\langle \mathrm{q}, e\rangle.P \mid s : h \longrightarrow P \mid s : h \cdot (\mathrm{p}, \mathrm{q}, v) \quad (e{\downarrow}v) \qquad \text{[Send]}$$

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

$$s[\mathrm{p}] \oplus \langle l, \mathrm{q}\rangle.P \mid s : h \longrightarrow P \mid s : h \cdot (\mathrm{p}, \mathrm{q}, l) \qquad \text{[Sel]}$$

$$s[\mathrm{p}]?(\mathrm{q}, x).P \mid s : (\mathrm{q}, \mathrm{p}, v) \cdot h \longrightarrow P\{v/x\} \mid s : h \qquad \text{[Rcv]}$$

$$s[\mathrm{p}]?((\mathrm{q}, y)).P \mid s : (\mathrm{q}, \mathrm{p}, s'[\mathrm{p}']) \cdot h \longrightarrow P\{s'[\mathrm{p}']/y\} \mid s : h \qquad \text{[SRcv]}$$

$$s[\mathrm{p}]\&(\mathrm{q}, \{l_i : P_i\}_{i \in I}) \mid s : (\mathrm{q}, \mathrm{p}, l_j) \cdot h \longrightarrow P_j \mid s : h \quad (j \in I) \qquad \text{[Branch]}$$

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

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

$$P \longrightarrow P' \quad \Rightarrow \quad \mathscr{E}[P] \longrightarrow \mathscr{E}[P'] \qquad \text{[Ctxt]}$$

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

Table 4. Reduction rules.

Coppo et al.@SFM'15