

Multiparty Asynchronous Session Types



<http://mrg.doc.ic.ac.uk/>

Nobuko Yoshida
Imperial College London

Structure of Lectures

- Theory
 - Multiparty Session Types
- Summary: Practice and Programming using Scribble

Session Type Reading List

- Home Page <http://mrg.doc.ic.ac.uk/>
- [ESOP'98] Language Primitives and Type Disciplines for Structured Communication-based Programming, Honda, Vasconcelos and Kubo
- [SecRet'06] Language Primitives and Type Disciplines for Structured Communication-based Programming *Revisited*, Yoshida and Vasconcelos, ENTCS.
- [SFM'15] Gentle Introduction to Multiparty Asynchronous Session Types, Coppo et al.

Selected Publications 2015/2016



- **[FASE'16]** Raymond Hu, Nobuko Yoshida: Hybrid Session Verification through Endpoint API Generation.
- **[TACAS'16]** Julien Lange, Nobuko Yoshida: Characteristic Formulae for Session Types.
- **[ESOP'16]** Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida: On the Relative Expressiveness of Higher-Order Session Processes.
- **[POPL'16]** Dominic Orchard, Nobuko Yoshida: Effects as sessions, sessions as effects .
- **[FSTTCS'15]** Romain Demangeon, Nobuko Yoshida: On the Expressiveness of Multiparty Session Types.
- **[OOPSLA'15]** Hugo A. López, Eduardo R. B. Marques, Francisco Martins, Nicholas Ng, César Santos, Vasco Thudichum Vasconcelos, Nobuko Yoshida: Protocol-Based Verification of Message-Passing Parallel Programs .
- **[CONCUR'15]** Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida: Characteristic Bisimulations for Higher-Order Session Processes .
- **[CONCUR'15]** Laura Bocchi, Julien Lange, Nobuko Yoshida: Meeting Deadlines Together.
- **[CONCUR'15]** Marco Carbone, Fabrizio Montesi, Carsten Schürmann, Nobuko Yoshida: Multiparty Session Types as Coherence Proofs.
- **[CC'15]** Nicholas Ng, Jose G.F. Coutinho, Nobuko Yoshida: Protocols by Default: Safe MPI Code Generation based on Session Types.
- **[PPoPP'15]** Tiago Cogumbreiro, Raymond Hu, Francisco Martins, Nobuko Yoshida: Dynamic deadlock verification for general barrier synchronisation.
- **[POPL'15]** Julien Lange, Emilio Tuosto, Nobuko Yoshida: From communicating machines to graphical choreographies.

Selected Publications 2015/2016



- **[FASE'16]** Raymond Hu, Nobuko Yoshida: Hybrid Session Verification through Endpoint API Generation.
- **[TACAS'16]** Julien Lange, Nobuko Yoshida: Characteristic Formulae for Session Types.
- **[ESOP'16]** Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida: On the Relative Expressiveness of Higher-Order Session Processes.
- **[POPL'16]** Dominic Orchard, Nobuko Yoshida: Effects as sessions, sessions as effects .
- **[FSTTCS'15]** Romain Demangeon, Nobuko Yoshida: On the Expressiveness of Multiparty Session Types.
- **[OOPSLA'15]** Hugo A. López, Eduardo R. B. Marques, Francisco Martins, Nicholas Ng, César Santos, Vasco Thudichum Vasconcelos, Nobuko Yoshida: Protocol-Based Verification of Message-Passing Parallel Programs .
- **[CONCUR'15]** Dimitrios Kouzapas, Jorge A. Pérez, Nobuko Yoshida: Characteristic Bisimulations for Higher-Order Session Processes .
- **[CONCUR'15]** Laura Bocchi, Julien Lange, Nobuko Yoshida: Meeting Deadlines Together.
- **[CONCUR'15]** Marco Carbone, Fabrizio Montesi, Carsten Schürmann, Nobuko Yoshida: Multiparty Session Types as Coherence Proofs.
- **[CC'15]** Nicholas Ng, Jose G.F. Coutinho, Nobuko Yoshida: Protocols by Default: Safe MPI Code Generation based on Session Types.
- **[PPoPP'15]** Tiago Cogumbreiro, Raymond Hu, Francisco Martins, Nobuko Yoshida: Dynamic deadlock verification for general barrier synchronisation.
- **[POPL'15]** Julien Lange, Emilio Tuosto, Nobuko Yoshida: From communicating machines to graphical choreographies.

Origin of Multiparty Session Types

Binary Session Types [PARL'94, ESOP'98]



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



Formalisation of W3C WS-CDL [ESOP'07]



Scribble at π^4 Technology

CDL Equivalent

- Basic example:

```
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;
        }
    }
}
```

Scribble Protocol

- *"Scribbling is necessary for architects, either physical or computing, since all great ideas of architectural construction come from that unconscious moment, when you do not realise what it is, when there is no concrete shape, only a whisper which is not a whisper, an image which is not an image, somehow it starts to urge you in your mind, in so small a voice but how persistent it is, at that point you start scribbling" - Kohei Honda 2007*
- **Basic example:**

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


Origin of Multiparty Session Types

Binary Session Types [PARL'94, ESOP'98]



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



Formalisation of W3C WS-CDL [ESOP'07]



Scribble at π 4 Technology



Multiparty Session Types [POPL'08]



Origin of Multiparty Session Types

Binary Session Types [PARL'94, ESOP'98]



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



Formalisation of W3C WS-CDL [ESOP'07]



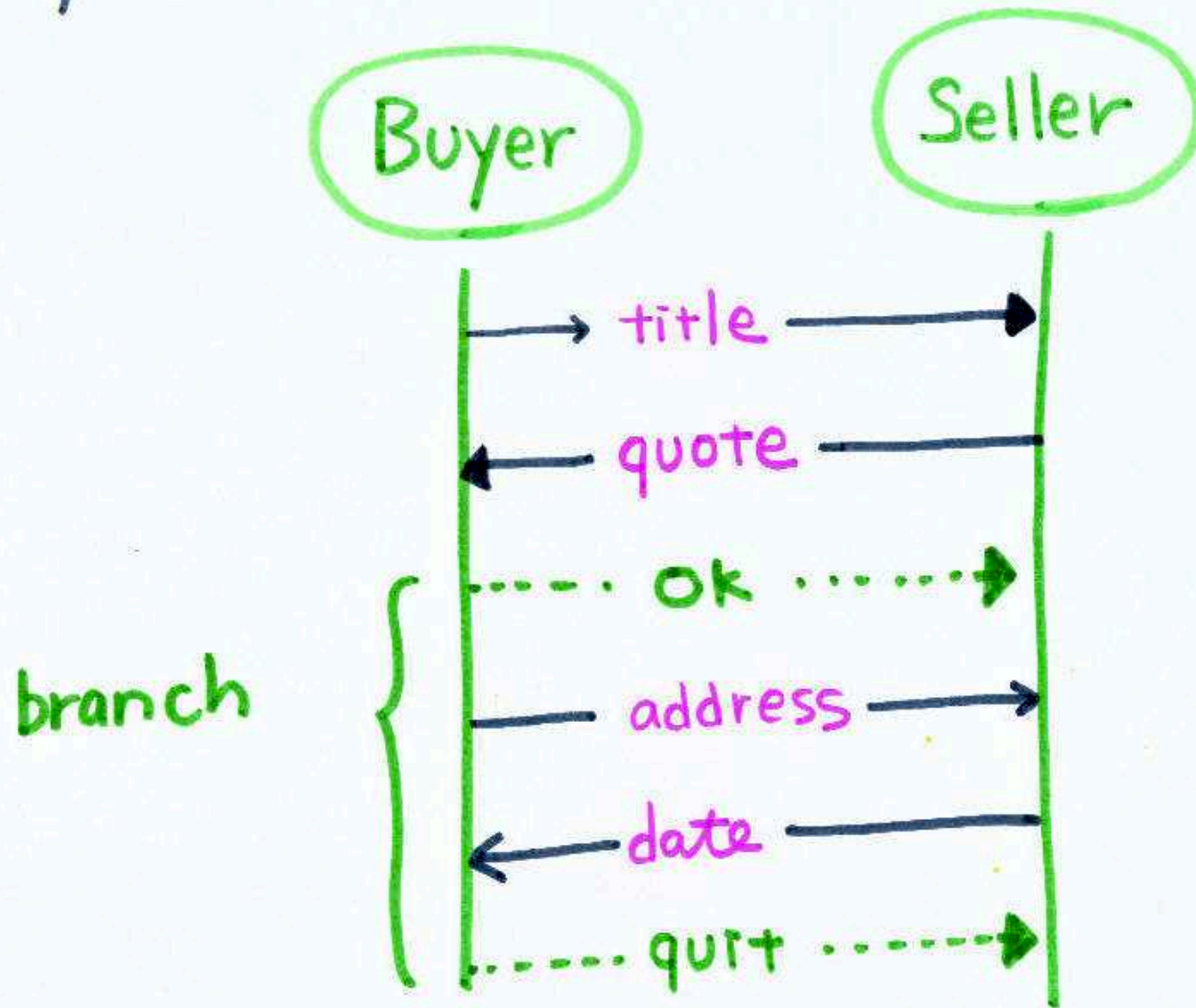
Scribble at π 4 Technology

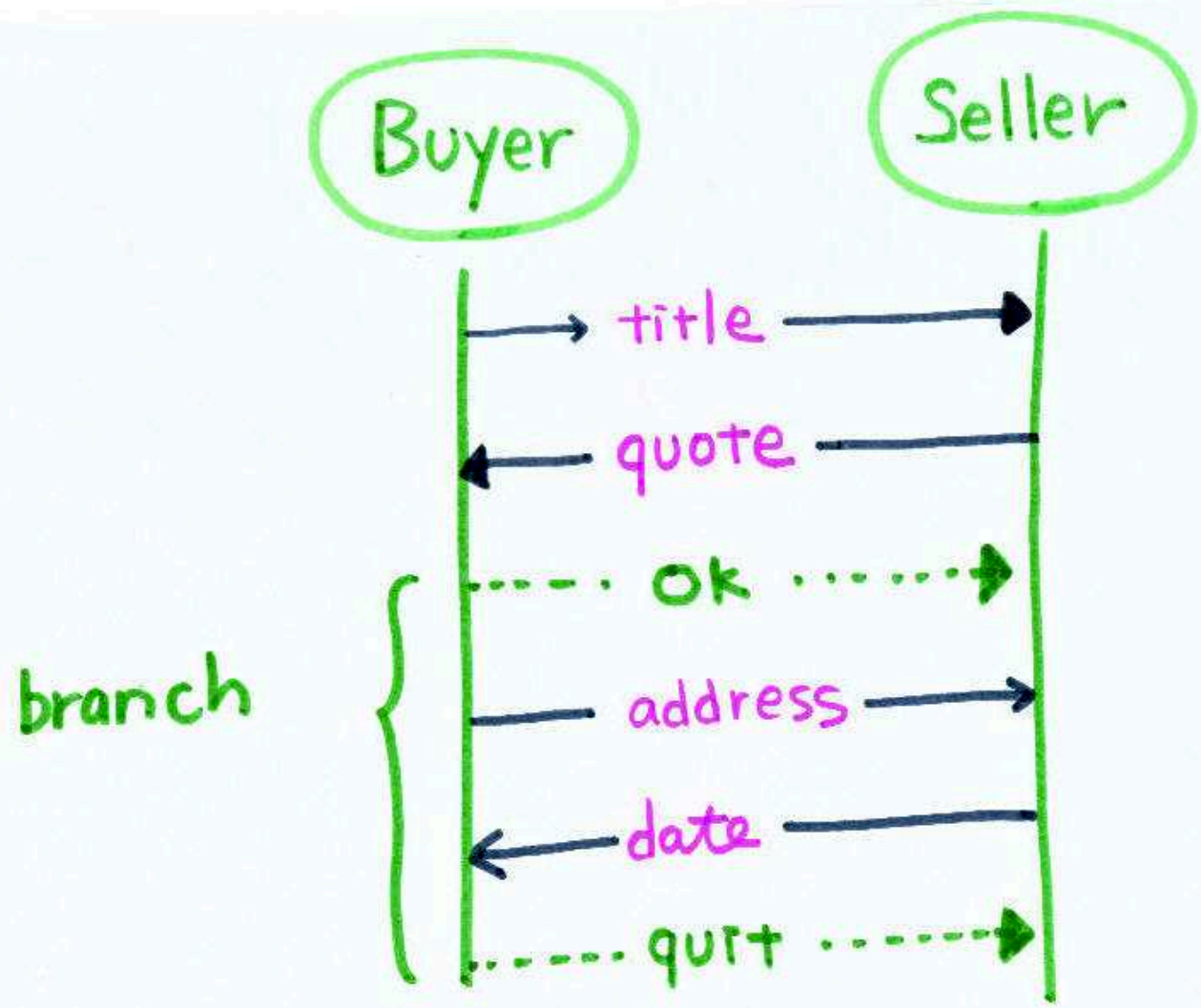


Multiparty Session Types [POPL'08]

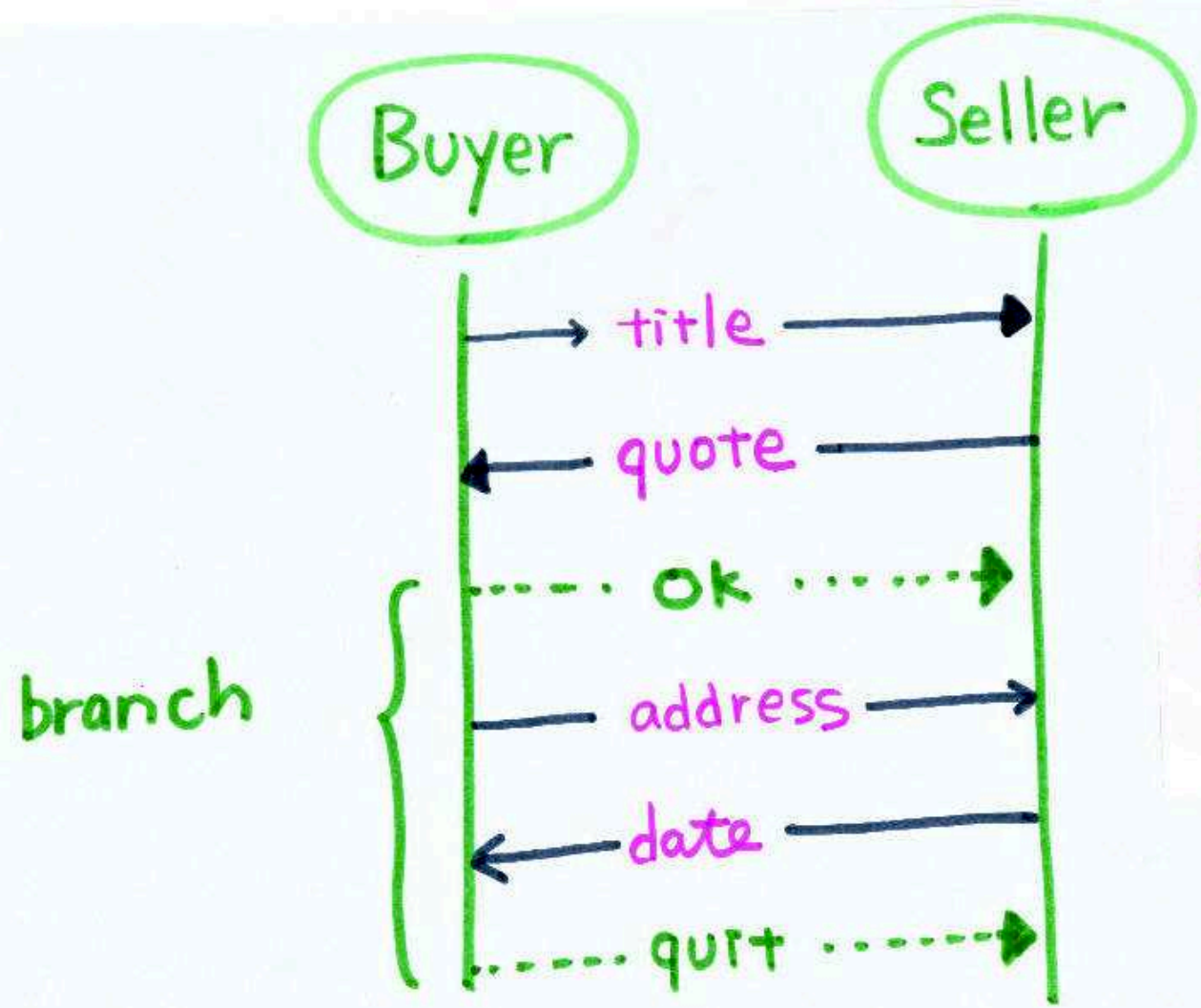


Binary Session Types : Buyer-Seller Protocol





! String ; ? Int ; ⊕ { ok : !String ; ? Date ; end , quit : end }



P has T
 Q has T dual
 P | Q typable

! String ; ? Int ; ⊕ { ok : !String ; ? Date ; end , quit : end }

dual ? String ; ! Int ; & { ok : ?String ; ! Date ; end , quit : end }

Binary Session Types

!String ; ?Int ; \oplus { ok : !String ; ?Date ; end, quit : end }

$\bar{a}(x)$. $x!; $x?(y)$; if $y < 50$ then
 $x \triangleleft \text{ok}$; $x!\langle "London" \rangle$; $x?(z)$; 0
else
 $x \triangleleft \text{quit}$$

Binary Session Types

!String ; ?Int ; \oplus { ok : !String ; ?Date ; end, quit : end }

dual ?String ; !Int ; \otimes { ok : ?String ; !Date ; end, quit : end }

$\bar{a}(x).$ x! <"Title"> ; x?(y); if y < 50 then
x < ok ; x! <"London"> ; x?(z); 0
else
x < quit

a(x). x?(y); x! <30>; x { ok \triangleright x?(z). x! <01/01>; 0
quit \triangleright 0 }

if T and \bar{T} then P | Q typable

Properties of Session Types

Intuitive Syntax, Light Weight Type Checking

① Communication Error-Freedom

No Communication Mismatch

② Session Fidelity

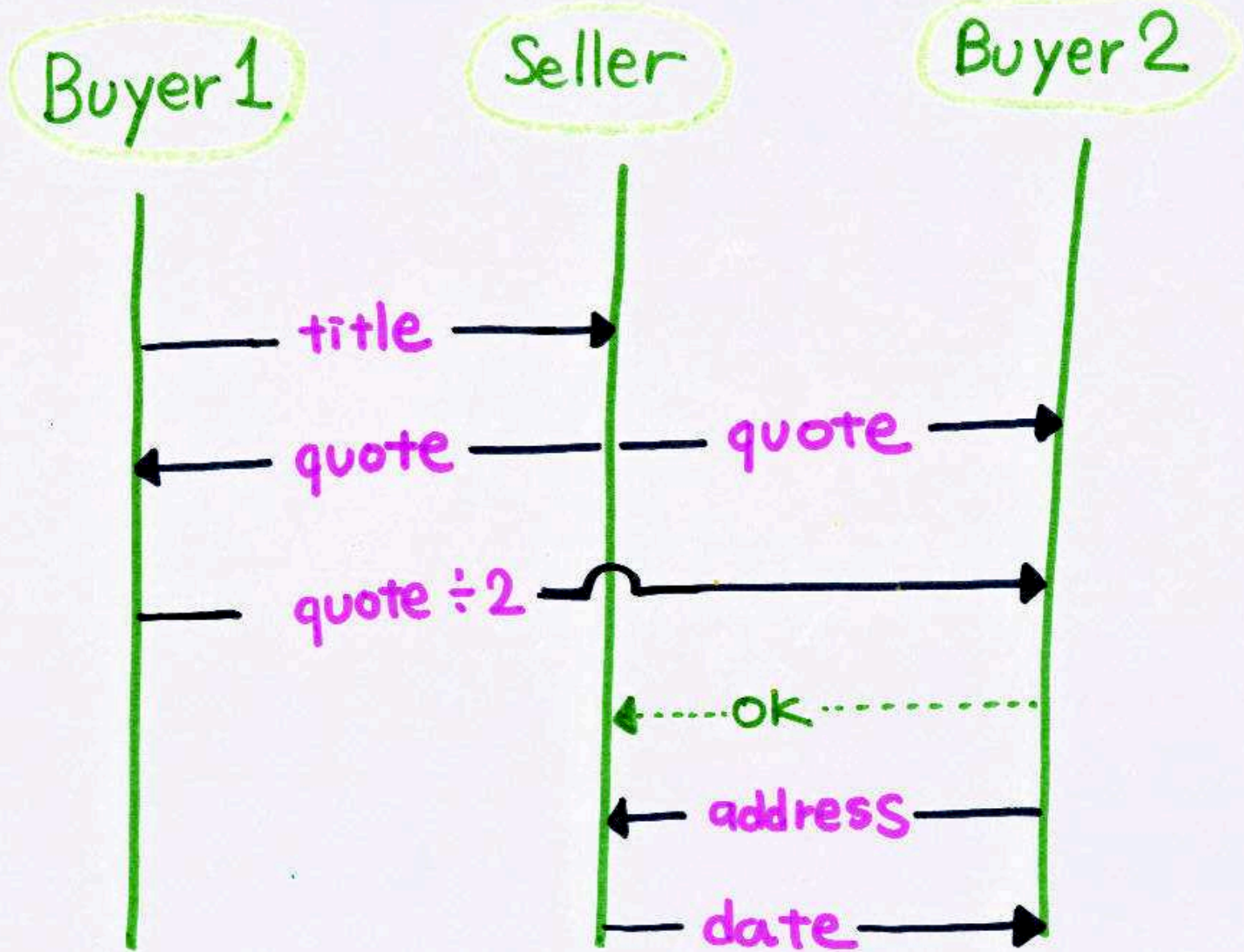
The Communication Sequence in a session follows the scenario declared in the types

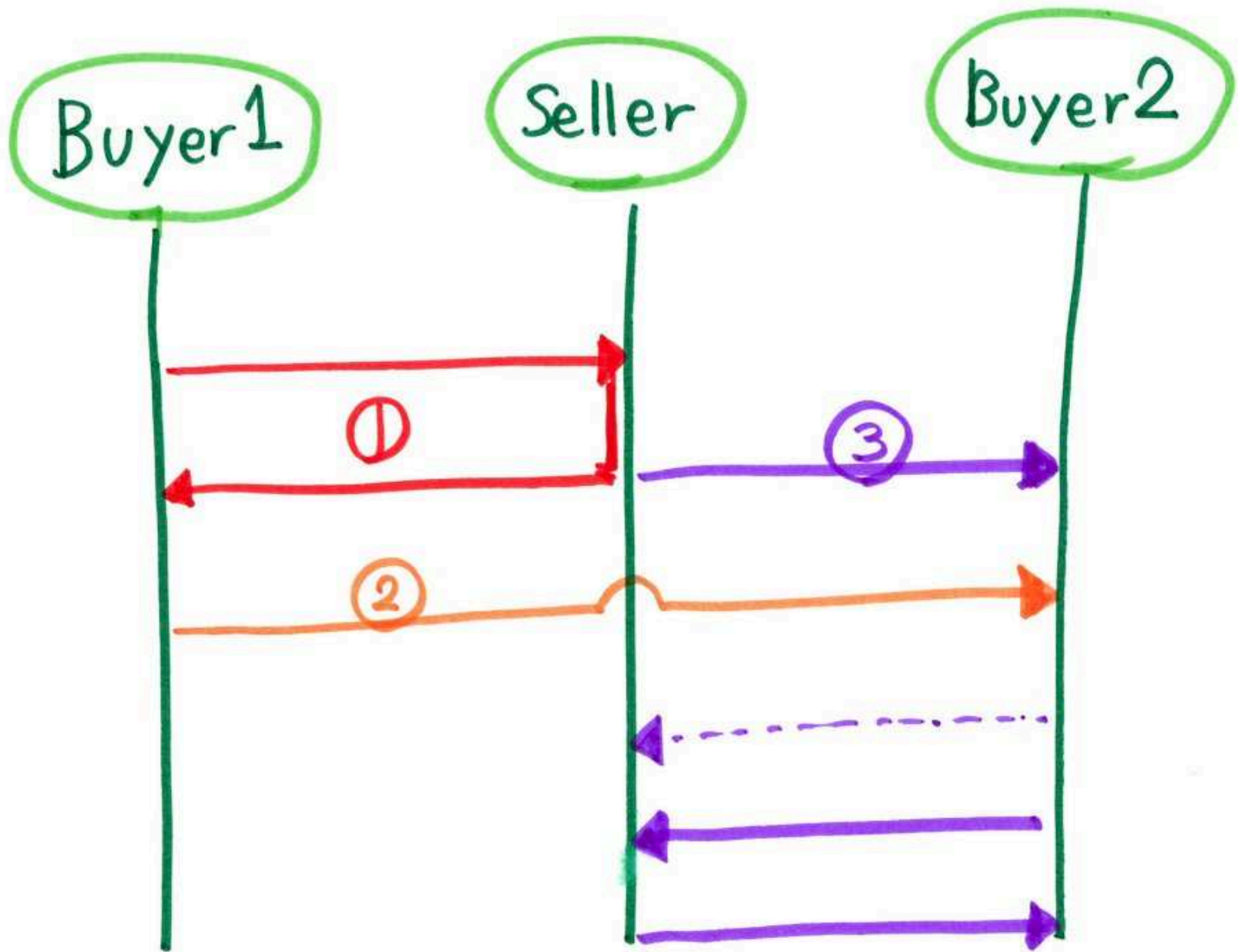
③ Progress

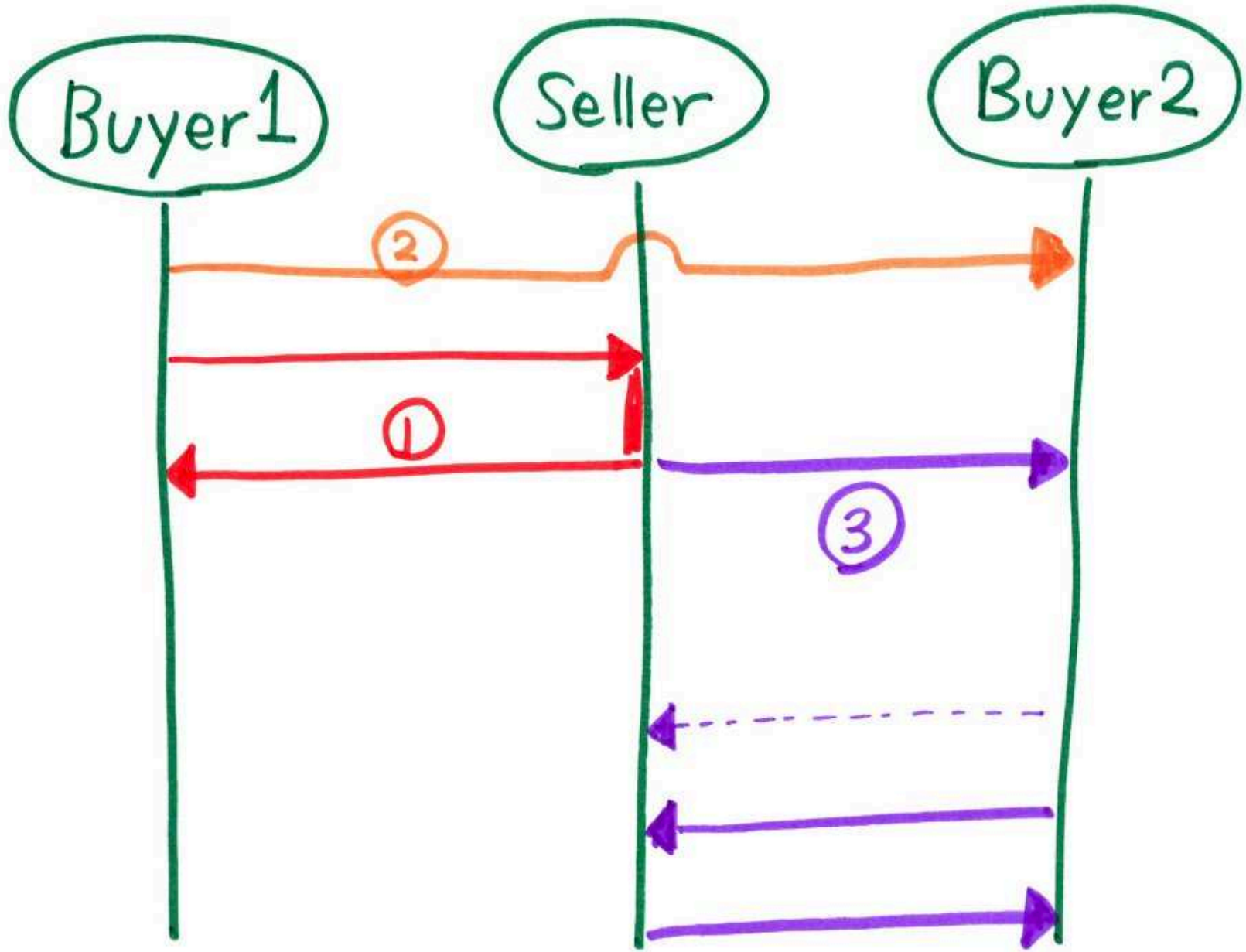
Single

No Deadlock / Stuck in a session

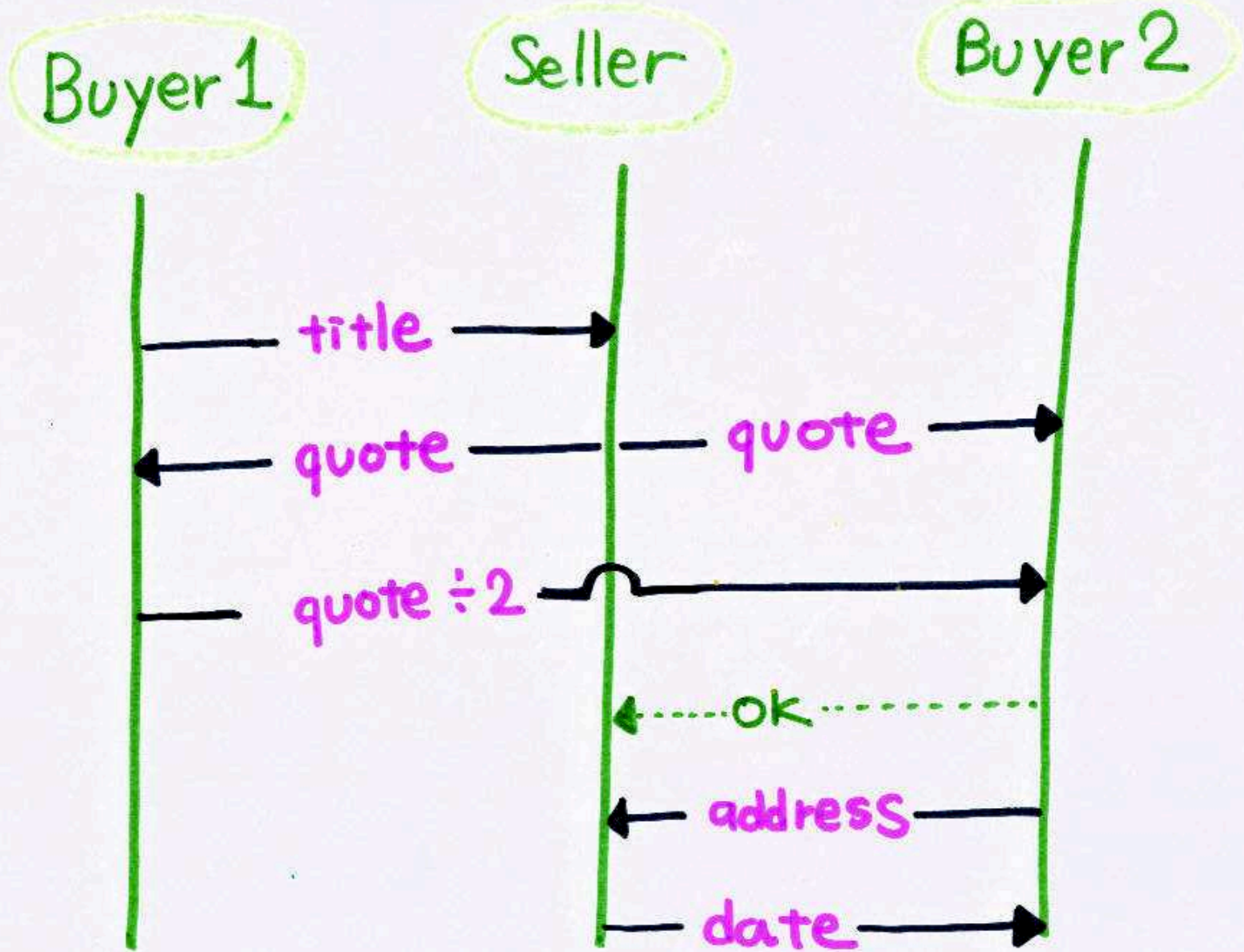
Multiparty Session Types





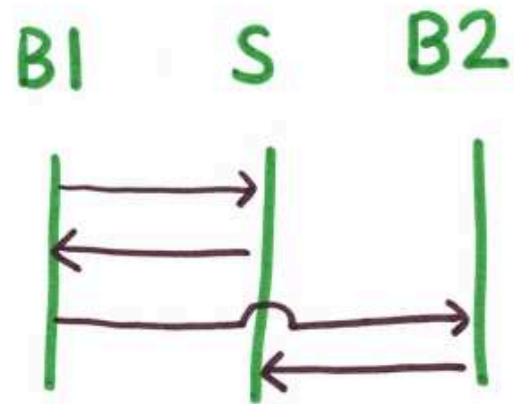


Multiparty Session Types



Multiparty Session Types

cf. Abstract Choreography
[WS-CDL]

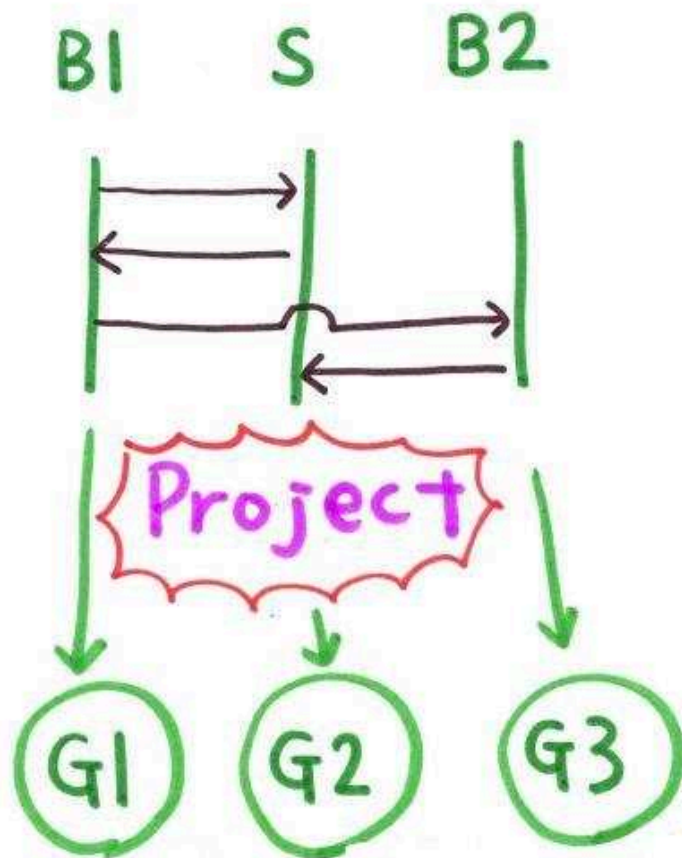


Step 1

Write Global Types

Multiparty Session Types

cf. Abstract Choreography
[WS-CDL]



Step 1

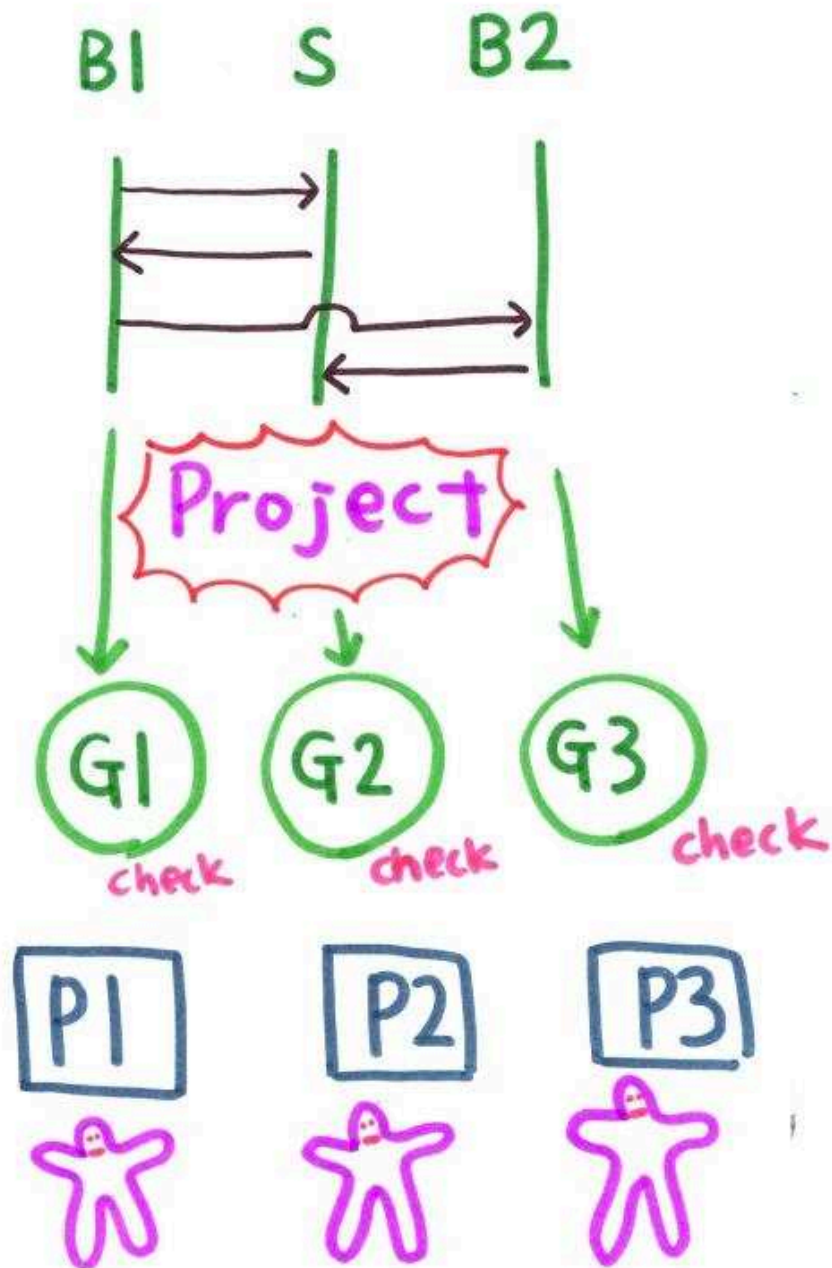
Write Global Types

Step 2

Project Local Types

Multiparty Session Types

cf. Abstract Choreography
[WS-CDL]



Step 1

Write Global Types

Step 2

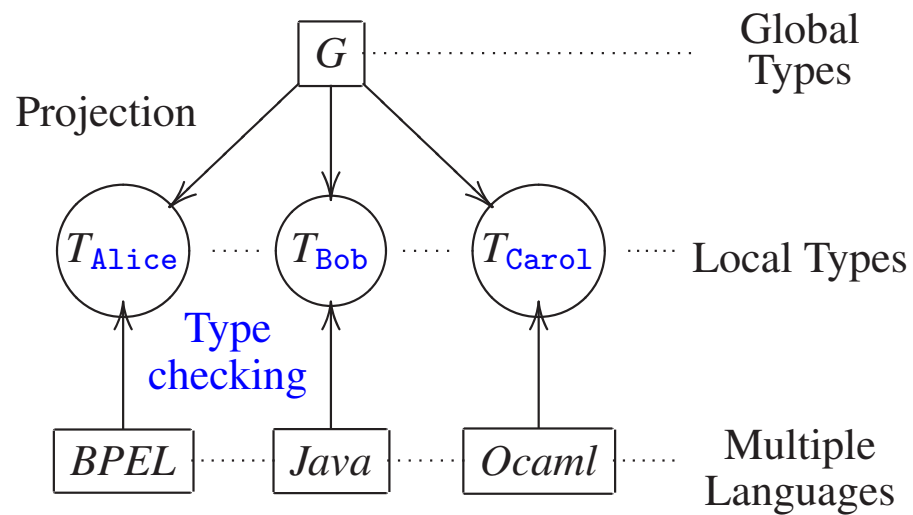
Project

Step 3

Write Local Programs

and ^{Type} Check Locally

Multiparty Session Types

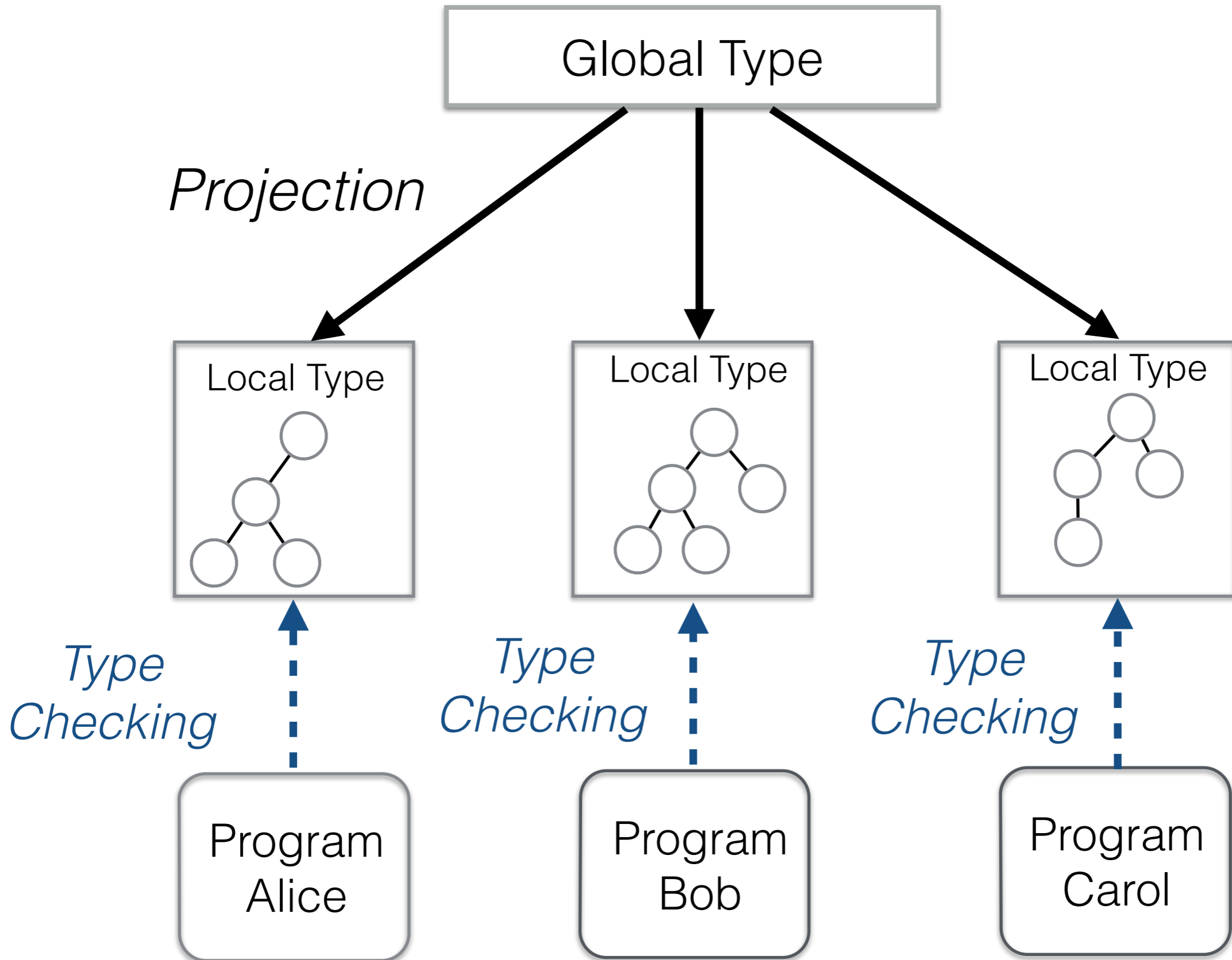


$Alice \rightarrow Bob: \langle Nat \rangle.$
 $Bob \rightarrow Carol: \langle Nat \rangle.end$

$T_{Bob} = ?\langle Alice, Nat \rangle;$
 $!\langle Carol, Nat \rangle; end$

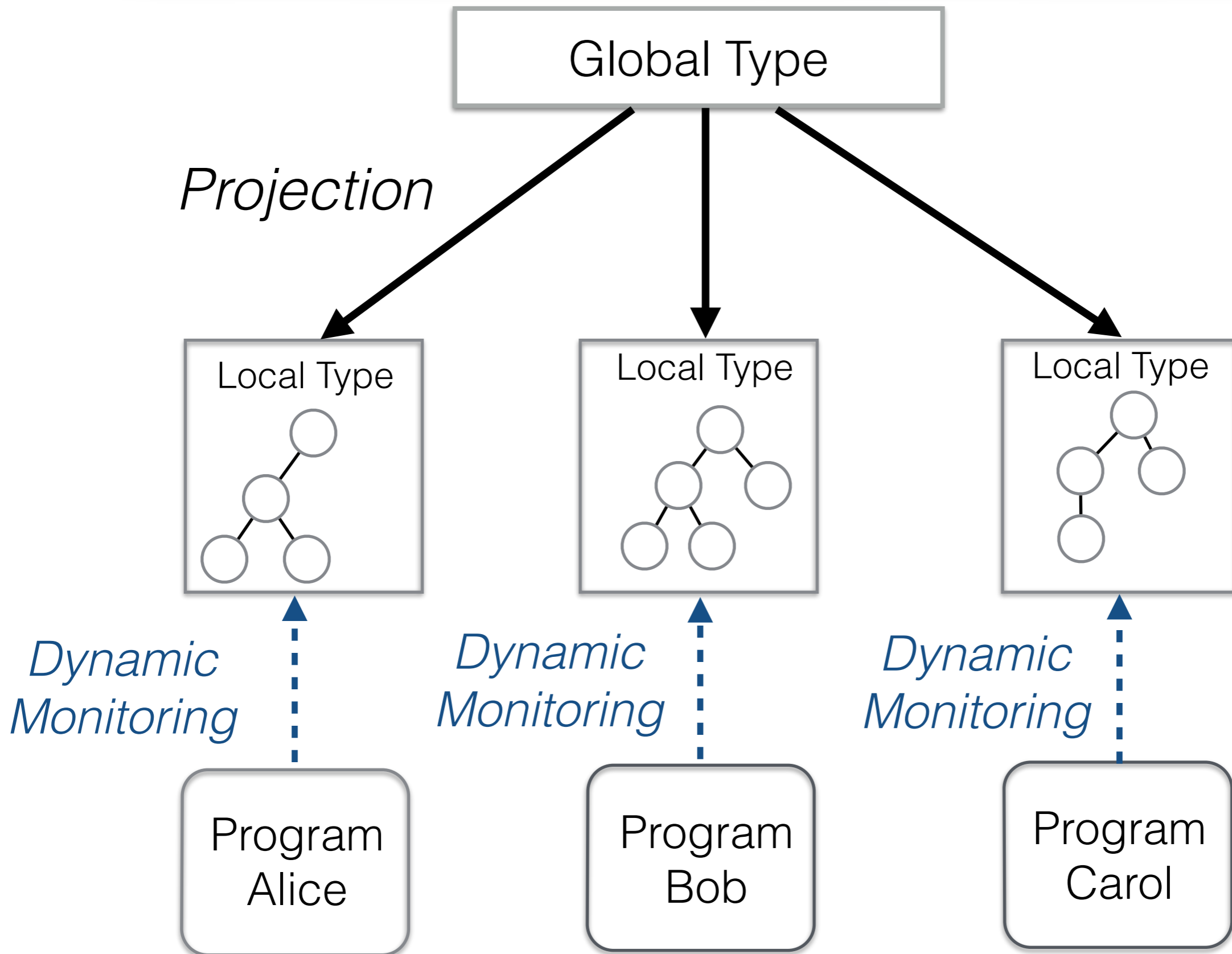
$P_{Bob} = s?(Alice, x);$
 $s!\langle Carol, x \rangle; 0$

Type Checking [OOPSLA'15, POPL'16]

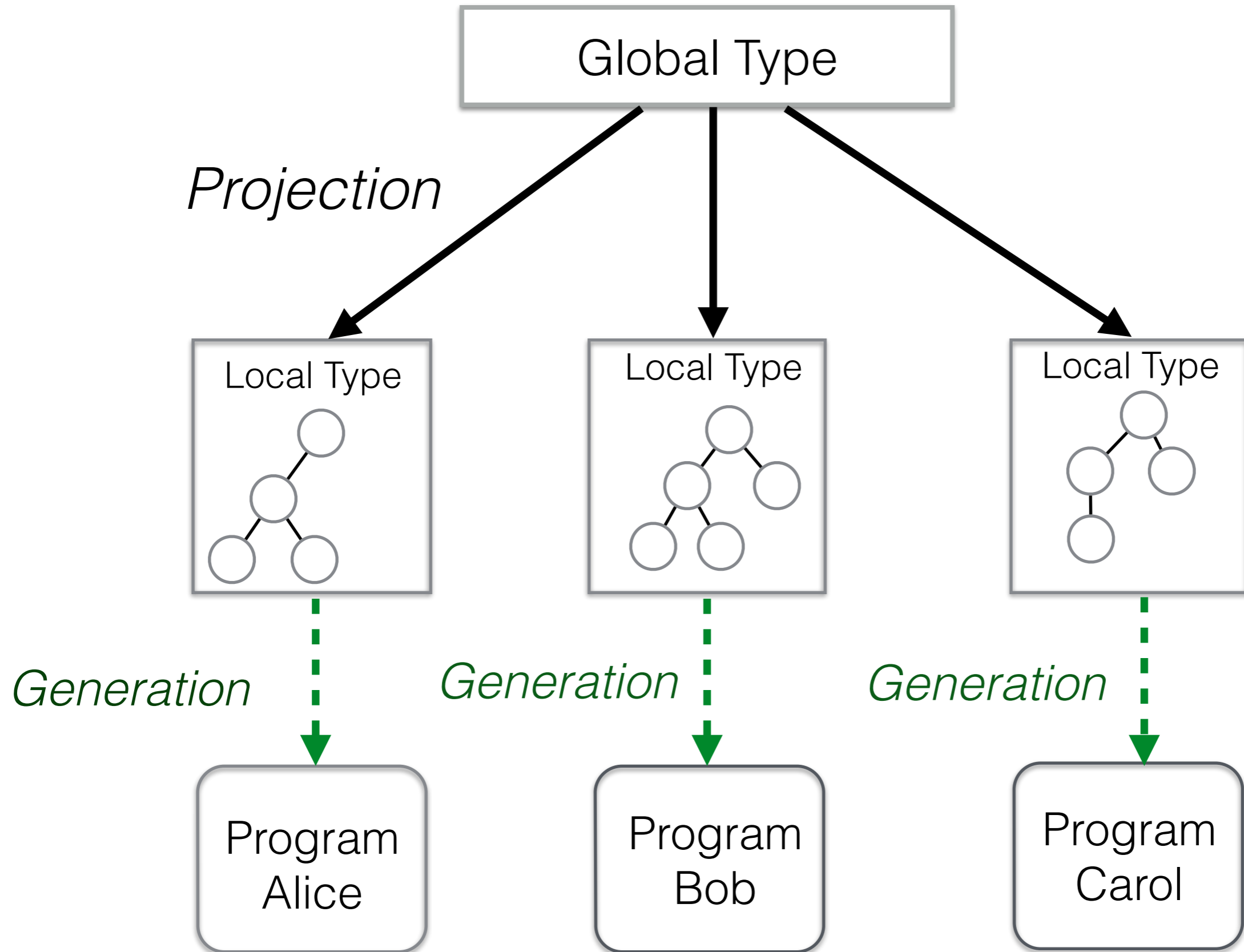


Dynamic Monitoring

[RV'13, COORDINATION'14, FMSSD'15]

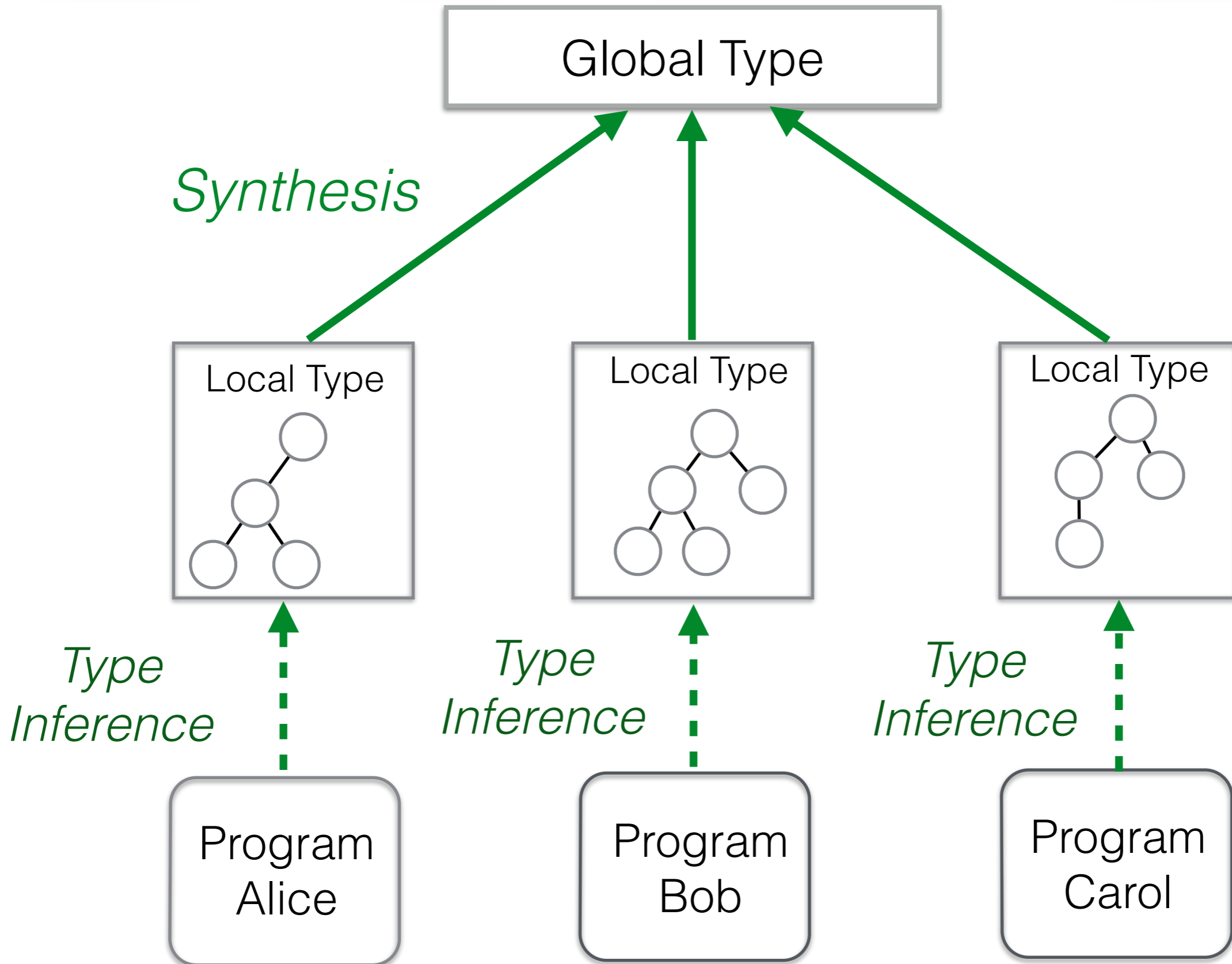


Code Generation [CC'15, FASE'16]



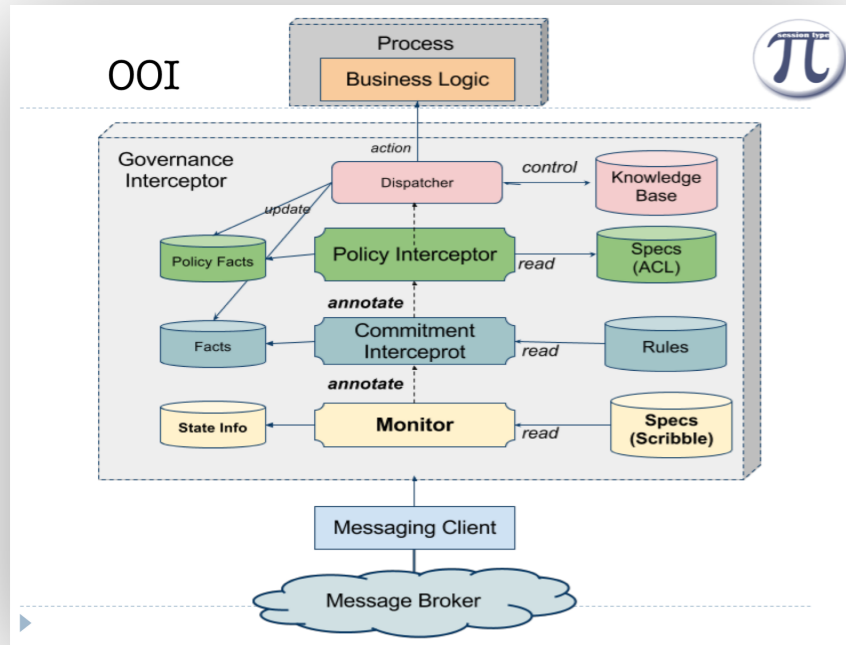
Synthesis

[ICALP'13, POPL'15, CONCUR'15, TACAS'16]

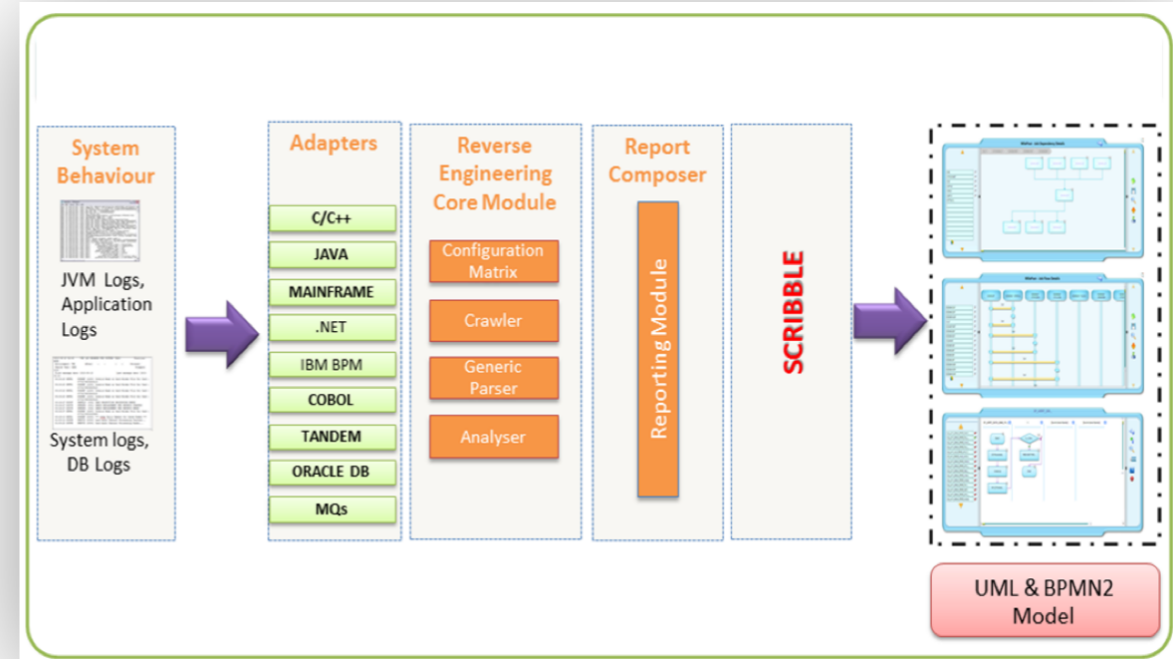


Applications

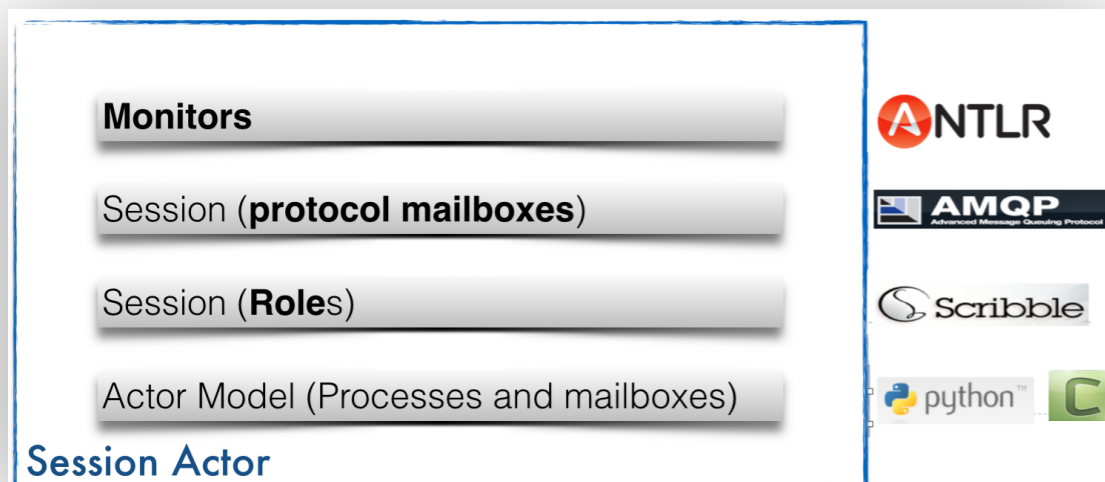
OOI Governance



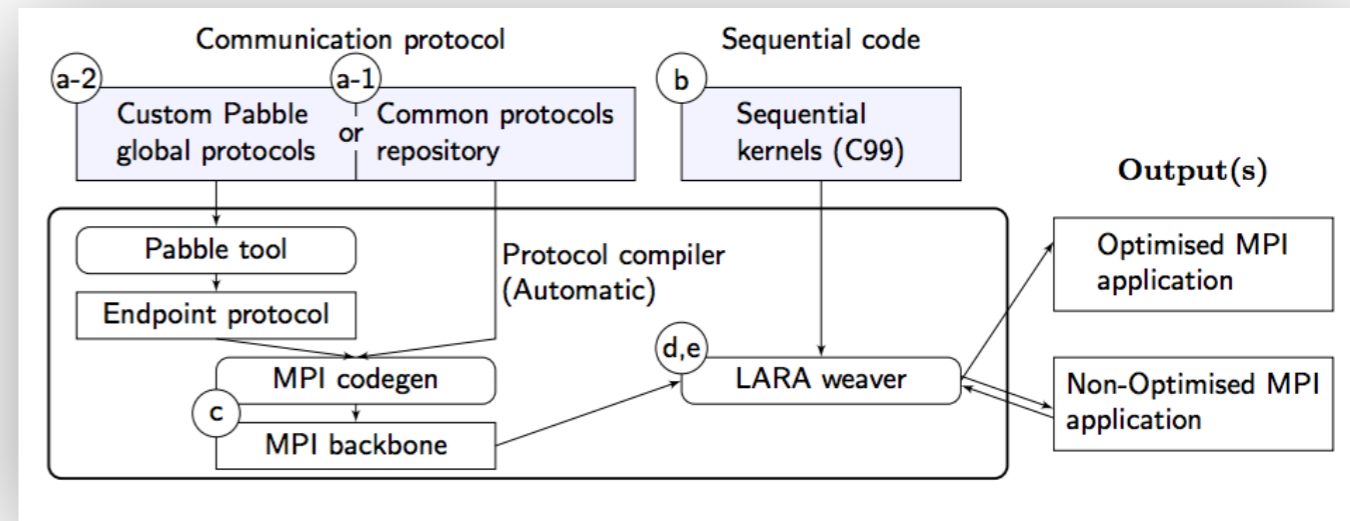
ZDLC: Process Modeling



Protocol Verification



MPI code generations



Global Types

global

$G ::= P \rightarrow P_1, \dots, P_m : \langle U \rangle. G'$

| $P \rightarrow P_1, \dots, P_m : \{ l_j : G_j \}_{j \in J}$

| $\text{mt}. G$

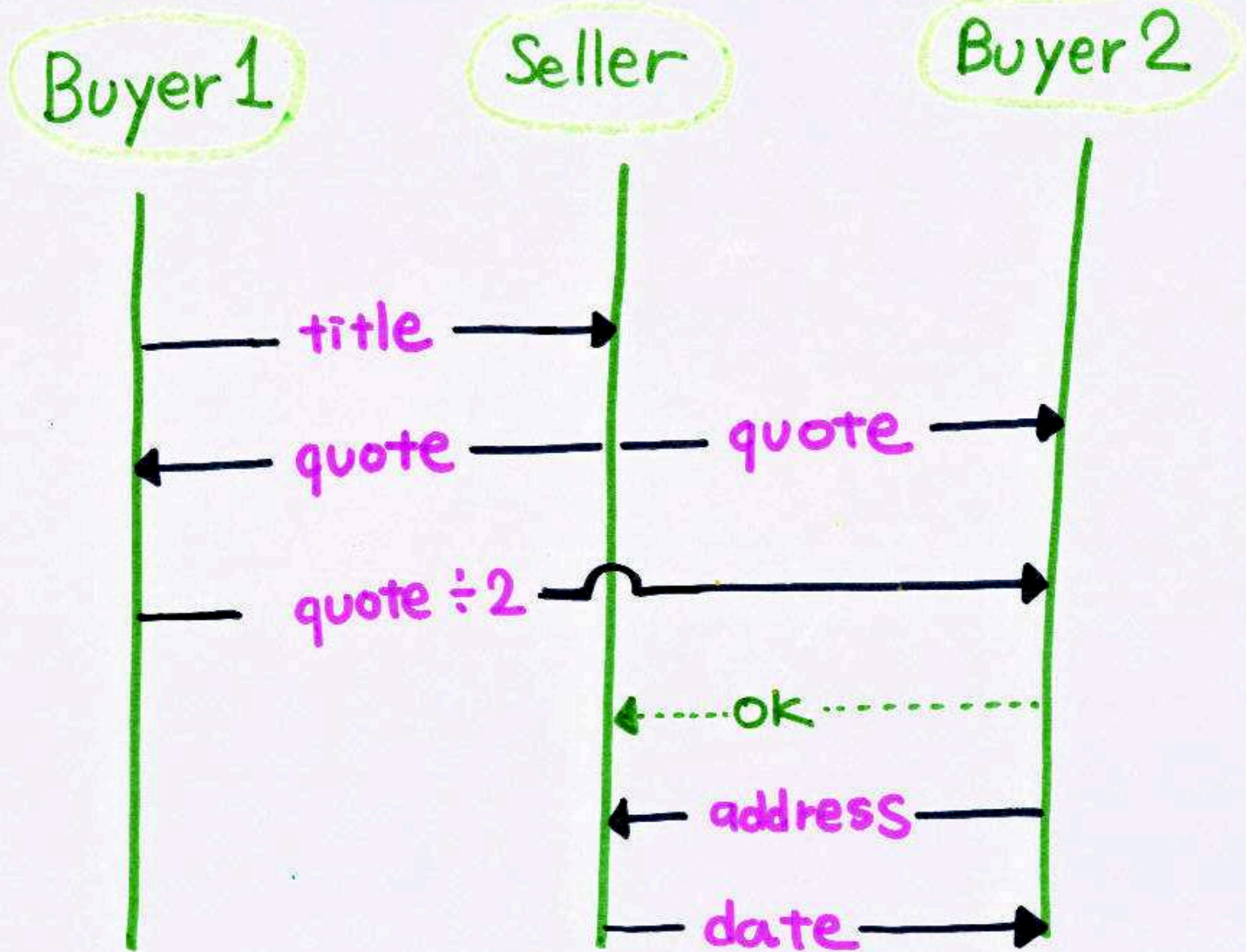
| $\text{end} \quad | \quad \text{t}$

value

$U ::= \text{nat} \mid \text{bool} \mid \dots \mid G \mid T$

Session

Multiparty Session Types



Three Buyers - Seller Example

G1

$A \rightarrow S : \langle \text{String} \rangle.$

$S \rightarrow \{A, B\} : \langle \text{Int} \rangle.$

$A \rightarrow B : \langle \text{Int} \rangle.$

$B \rightarrow \{S, A\} : \{ \text{OK} : B \rightarrow S : \langle \text{String} \rangle.$

$S \rightarrow B : \langle \text{Date} \rangle.$

quit: }

Alice = Buyer 1

Bob = Buyer 2

Projections of the Global type G_1

$$G_1 \upharpoonright S = ?(A, \text{string}).!\langle\{A, B\}, \text{int}\rangle. \\ \&(B, \{\text{ok} : ?(B, \text{string}).!\langle B, \text{date}\rangle.\text{end}, \\ \text{quit} : \text{end}\})$$

$$G_1 \upharpoonright A = !\langle S, \text{string}\rangle.?(S, \text{int}).!\langle B, \text{int}\rangle.\&(B, \{\text{ok} : \text{end}, \text{quit} : \text{end}\})$$

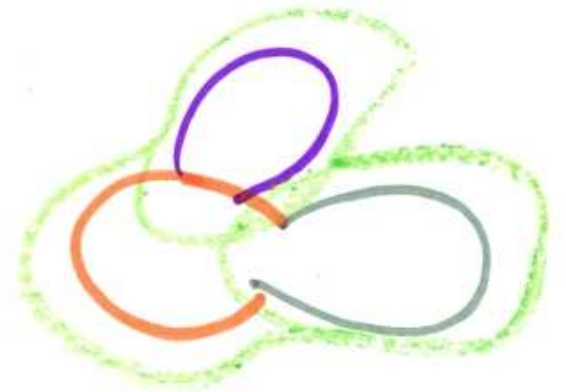
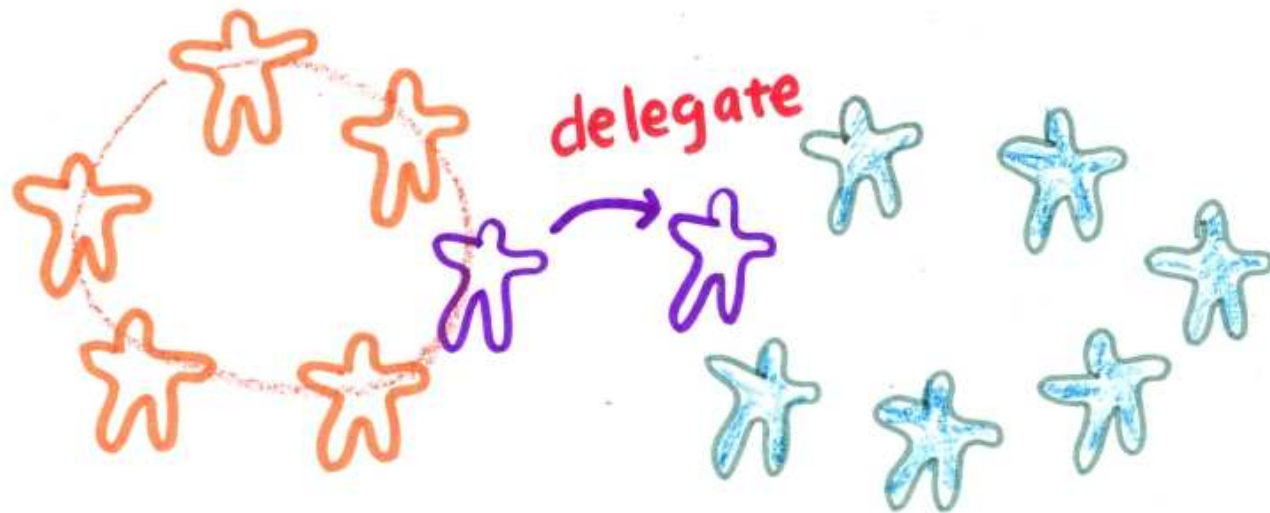
$$G_1 \upharpoonright B = ?(S, \text{int}).?(A, \text{int}). \\ \oplus\langle\{S, A\}, \{\text{ok} : !\langle S, \text{string}\rangle.?(S, \text{date}).\text{end}, \\ \text{quit} : \text{end}\}\rangle$$

The projection of a global type G onto a participant q

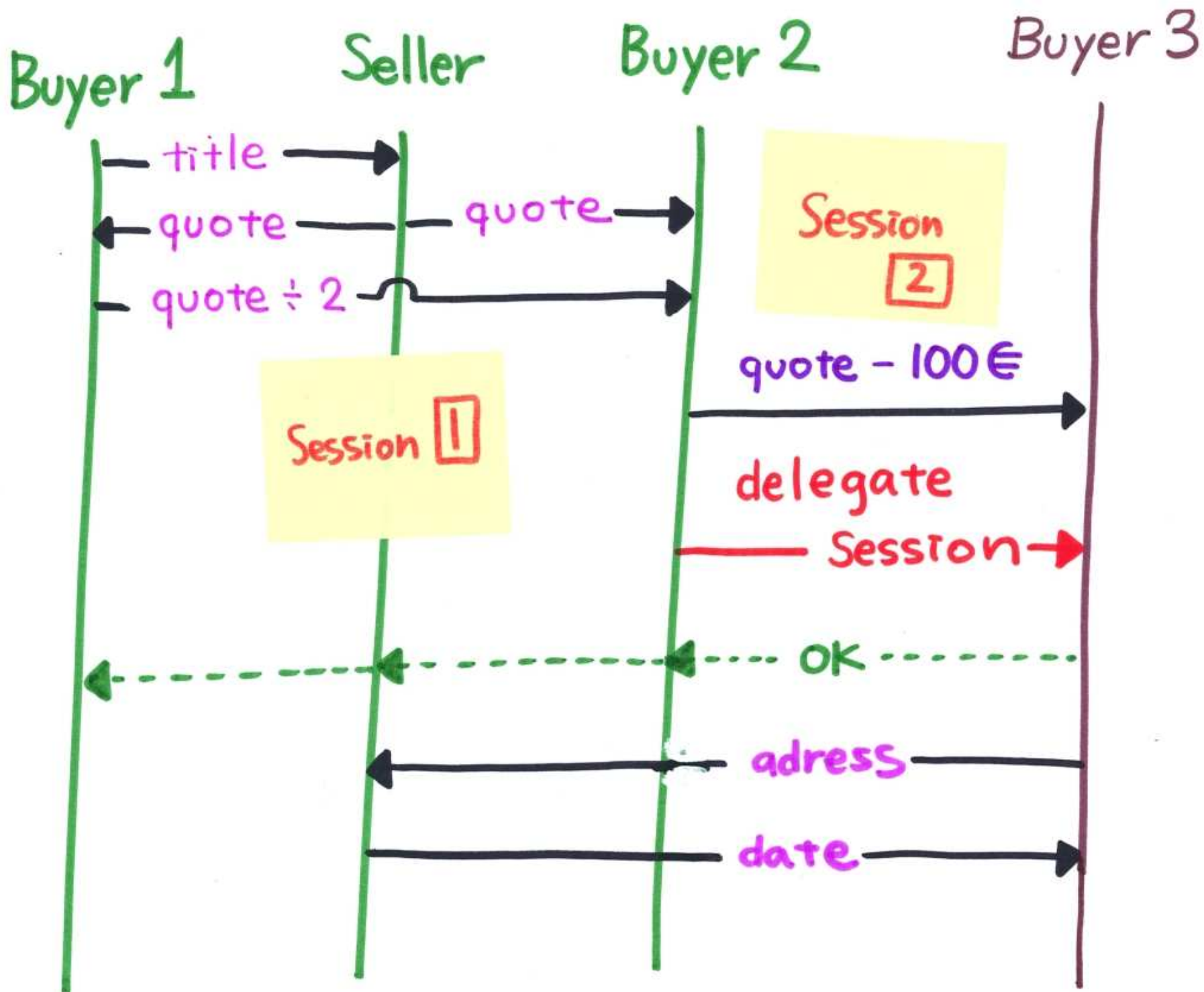
$$(\mathbf{p} \rightarrow \mathbf{p}' : \langle U \rangle . G') \upharpoonright q = \begin{cases} !\langle \mathbf{p}', U \rangle . (G' \upharpoonright q) & \text{if } q = \mathbf{p}, \\ ?(\mathbf{p}, U) . (G' \upharpoonright q) & \text{if } q = \mathbf{p}', \\ G' \upharpoonright q & \text{otherwise.}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

$$(\mathbf{p} \rightarrow \mathbf{p}' : \{l_i : G_i\}_{i \in I}) \upharpoonright q = \begin{cases} \oplus \langle \mathbf{p}', \{l_i : T_i\}_{i \in I} \rangle & \text{if } q = \mathbf{p} \\ \&(\mathbf{p}, \{l_i : G_i \upharpoonright q\}_{i \in I}) & \text{if } q = \mathbf{p}' \\ G_{i_0} \upharpoonright q \text{ where } i_0 \in I & \text{if } q \neq \mathbf{p}, q \neq \mathbf{p}' \\ & \text{and } G_i \upharpoonright q = G_j \upharpoonright q \text{ for all } i, j \in I. \end{cases}$$

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



rely on other parties to complete tasks transparently type safe manner in a



Three Buyers - Seller Example

G1

$A \rightarrow S : \langle \text{String} \rangle.$

$S \rightarrow \{A, B\} : \langle \text{Int} \rangle.$

$A \rightarrow B : \langle \text{Int} \rangle.$

$B \rightarrow \{S, A\} : \{ \text{ok} : B \rightarrow S : \langle \text{String} \rangle.$

$S \rightarrow B : \langle \text{Date} \rangle.$

$\text{quit} : \}$

G2

$B \rightarrow C : \langle \text{Int} \rangle.$

$B \rightarrow C : \langle T \rangle.$

$C \rightarrow B : \{ \text{ok} : .. \text{quit} : ... \}$

$T =$
 $+(\{S, A\},$
 $\text{ok} : !\langle S, \text{string} \rangle,$
 $\quad ?\langle S, \text{date} \rangle,$
 $\text{quit} :)$

Global types for the three buyer protocol

$$\begin{aligned} G_a = & \quad 2 \longrightarrow 3 \quad : \langle \text{string} \rangle. \\ & \quad 3 \longrightarrow \{1, 2\} \quad : \langle \text{int} \rangle. \\ & \quad 2 \longrightarrow 1 \quad : \langle \text{int} \rangle. \\ & \quad 1 \longrightarrow \{2, 3\} \quad : \{ \text{ok} : 1 \longrightarrow 3 : \langle \text{string} \rangle. \\ & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad 3 \longrightarrow 1 : \langle \text{date} \rangle. \text{end}, \\ & \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{quit} : \text{end} \} \end{aligned}$$

$$\begin{aligned} G_b = & \quad 2 \longrightarrow 1 : \langle \text{int} \rangle. \\ & \quad 2 \longrightarrow 1 : \langle T \rangle. \\ & \quad 1 \longrightarrow 2 : \{ \text{ok} : \text{end}, \text{quit} : \text{end} \} \end{aligned}$$

$$T = \oplus \langle \{2, 3\}, \{ \text{ok} : ! \langle 3, \text{string} \rangle. ? (3, \text{date}). \text{end}, \text{quit} : \text{end} \} \rangle$$

Projections of the three buyer protocol

$$G_a \upharpoonright 3 = ?(2, \text{string}).!\langle\{1, 2\}, \text{int}\rangle; \&(1, \{\text{ok} : ?(1, \text{string}).$$
$$!\langle 1, \text{date}\rangle.\text{end}, \text{quit} : \text{end}\})$$

$$G_b \upharpoonright 1 = ?(2, \text{int}).?(2, \text{T}).\oplus \langle 2, \{\text{ok} : \text{end}, \text{quit} : \text{end}\}\rangle$$

Implementation of the three buyer protocol

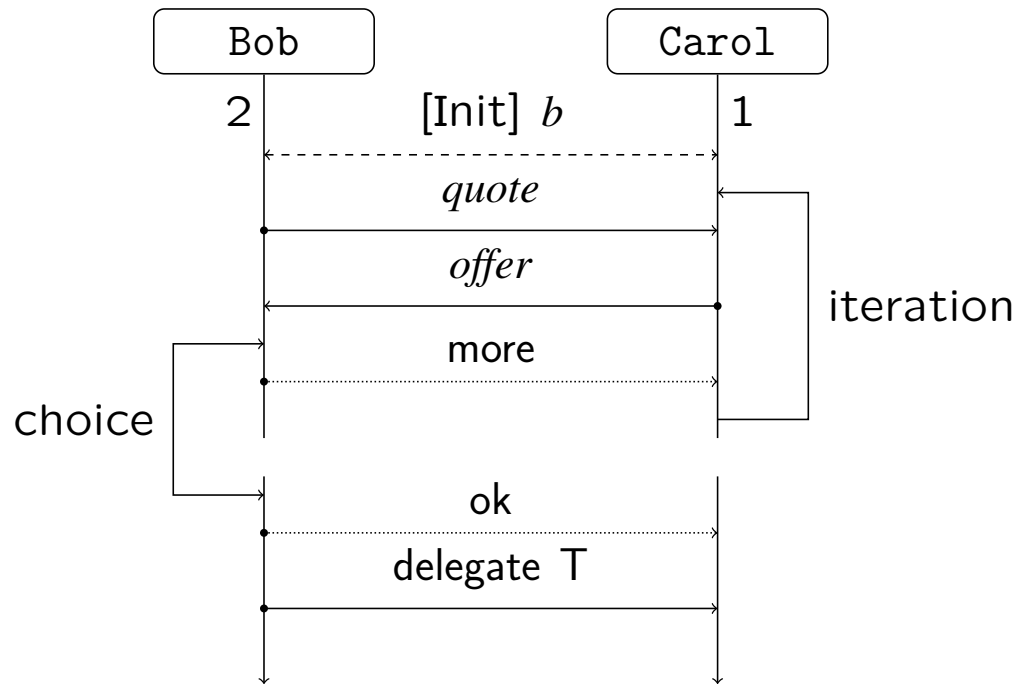
$$\begin{aligned} \text{Seller} = & \bar{a}[3](y).y?(2, \textit{title}).y!\langle\{1, 2\}, \textit{quote}\rangle. \\ & y\&(1, \{\textit{ok} : y?(1, \textit{address}). \\ & y!\langle 1, \textit{date}\rangle.\mathbf{0}, \textit{quit} : \mathbf{0}\}) \end{aligned}$$
$$\begin{aligned} \text{Alice} = & a[2](y).y!\langle 3, \textit{"Title"}\rangle.y?(3, \textit{quote}). \\ & y!\langle 1, \textit{quote div 2}\rangle. \\ & y\&(1, \{\textit{ok} : \mathbf{0}, \textit{quit} : \mathbf{0}\}) \end{aligned}$$

Implementation of the three buyer protocol

Bob = $a[1](y).y?(3, quote).y?(2, contrib).$
if ($quote - contrib < 100$)
then
 $y \oplus \langle \{2, 3\}, ok \rangle . y! \langle 3, "Address" \rangle . y?(3, date). \mathbf{0}$
else
 $\bar{b}[2](z).z! \langle 1, quote - contrib - 99 \rangle . z! \langle \langle 1, y \rangle \rangle .$
 $z \& (1, \{ok : \mathbf{0}, quit : \mathbf{0}\})$

Carol = $b[1](z).z?(2, x).z?(\langle 2, t \rangle).$
if ($x < 100$)
then
 $z \oplus \langle 2, ok \rangle . t \oplus \langle \{2, 3\}, ok \rangle . t! \langle 3, "Address" \rangle .$
 $t?(3, date). \mathbf{0}$
else
 $z \oplus \langle 2, quit \rangle . t \oplus \langle \{2, 3\}, quit \rangle . \mathbf{0}$

The three buyer protocol with recursion



Global type for recursive negotiation

$$\begin{aligned} G_b = & \quad \mu \mathbf{t}. 2 \longrightarrow 1 : \langle \text{int} \rangle. \\ & \quad 1 \longrightarrow 2 : \langle \text{int} \rangle. \\ & \quad 2 \longrightarrow 1 : \{ \text{ok} : 2 \longrightarrow 1 : \langle T \rangle. \text{end}, \\ & \quad \quad \text{more} : \mathbf{t}, \\ & \quad \quad \text{quit} : \text{end} \} \end{aligned}$$

$$T = \oplus \langle \{3, 2\}, \{ \text{ok} : ! \langle 3, \text{string} \rangle. ?(3, \text{date}). \text{end}, \text{quit} : \text{end} \} \rangle$$

The three buyer example with recursion

Bob = $a[1](y).y?(3, \text{quote}).y?(2, \text{contrib})$.
if $(\text{quote} - \text{contrib} < 100)$
then
 $y \oplus \langle \{2, 3\}, \text{ok} \rangle . y! \langle 3, \text{"Address"} \rangle . y?(3, \text{date}).\mathbf{0}$
else
 $\bar{b}[2](z)$.
def $X(x', z', y') = z'! \langle 1, x' \rangle . z?(1, w)$.
 if $\text{good}(w)$ then $z' \oplus \langle 1, \text{ok} \rangle . z'! \langle \langle 1, y' \rangle \rangle . \mathbf{0}$
 else if $\text{negotiable}(w)$ then $z' \oplus \langle 1, \text{more} \rangle$.
 $X \langle \text{newproposal}(w), z', y' \rangle$
 else $z' \oplus \langle 1, \text{quit} \rangle . y' \oplus \langle \{2, 3\}, \text{quit} \rangle . \mathbf{0}$
in $X \langle \text{firstproposal}(\text{quote}), z, y \rangle$

The three buyer example with recursion

```
Carol = b[1](z).def Y(z') =  
  z'?(2,x).z'!\langle 2, offer(x)\rangle.  
  z'&(2, {ok : z'?((2,t)).t ⊕ \langle {2,3}, ok\rangle.  
    t!\langle 3, "Address"\rangle.t?(3, date).0  
    more : Y\langle z'\rangle,  
    quit : 0})  
  in Y\langle z'\rangle
```

Dynamic Semantics

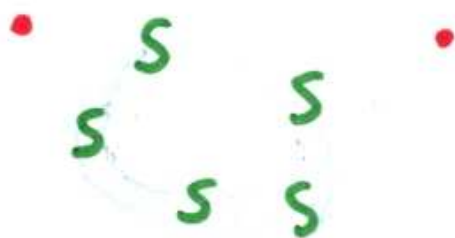
Multicast

$\bar{a}[n](y). P_n \mid a[1](y). P_1 \mid \dots \mid a[n-1](y). P_{n-1}$

$\rightarrow (v s) (P_1 [S[1]/y] \mid P_2 [S[2]/y] \mid \dots \mid P_n [S[n]/y])$

a service channel shared/interfered $S: \emptyset$
queue

S session channel linearised



Send



$S[P]! \langle \tilde{p}, V \rangle; P \mid s:h \rightarrow P \mid s:h \cdot (\underline{P}, \tilde{p}, V)$

Recv

$S[P]?(q, x); P \mid s:(\underline{q}, \underline{P}, V) \cdot h$



$\rightarrow P[V/x] \mid s:h$

Selection

$$\underline{s[P]} \triangleleft \langle \tilde{q}, l \rangle; P \mid s:h \rightarrow P \mid \underline{s:h} \cdot (P, \tilde{q}, l)$$

Branching

$$\underline{s[P]} \delta (q, \{l_i : P_i\}_{i \in I}) \mid s: (q, \underline{P}, l_j) \cdot h$$
$$\rightarrow P_j \mid s:h$$

Asynchronous Sessions

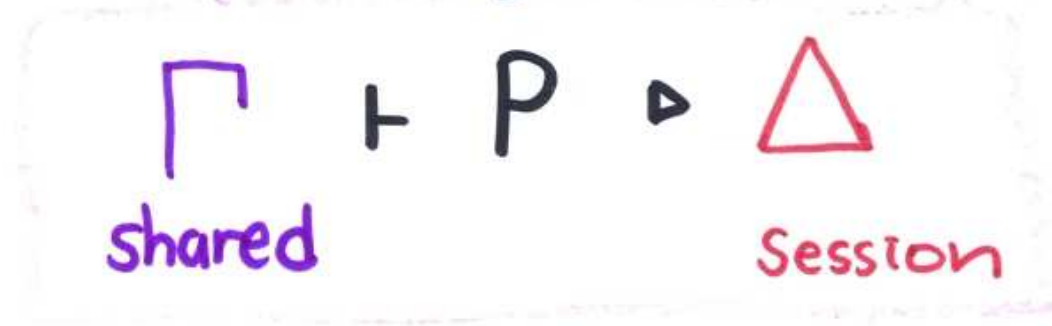
- Sender Non-Blocking
- Message Order Preserving per Session

Reduction rule for the Process Call

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

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

Communication Typing



Local Session Types

$$\begin{aligned} T ::= & ! \langle \tilde{P}, U \rangle ; T \mid ? \langle P, U \rangle ; T \\ & \mid \oplus \tilde{P}, \{ l_i : T_i \}_{i \in I} \mid \otimes P, \{ l_i : T_i \}_{i \in I} \\ & \mid \mu t. T \mid \text{end} \mid t \end{aligned}$$

INIT

$$\Gamma \vdash u : \langle G \rangle \quad \Gamma \vdash P \triangleright \Delta, y : G \mid P$$

$$\Gamma \vdash u[P](y). P \triangleright \Delta$$

Send

$$\Gamma \vdash e : S \quad \Gamma \vdash P \triangleright \Delta, c : T$$

$$\Gamma \vdash c! \langle \tilde{P}, e \rangle ; P \triangleright c! \langle \tilde{P}, S \rangle ; T$$

Par

$$\Gamma \vdash P \triangleright \Delta \quad \Gamma \vdash Q \triangleright \Delta'$$

Δ, Δ' disjoint

$$\Gamma \vdash P \mid Q \triangleright \Delta, \Delta'$$

Typing rules for delegation and session receiving

$$\frac{\Gamma \vdash P \triangleright \Delta, c : T}{\Gamma \vdash c! \langle \langle p, c' \rangle \rangle . P \triangleright \Delta, c : ! \langle p, T \rangle . T, c' : T} \text{ (Deleg)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta, c : T, y : T}{\Gamma \vdash c? \langle \langle q, y \rangle \rangle . P \triangleright \Delta, c : ? \langle q, T \rangle . T} \text{ (Srcv)}$$

Delegated channel type T derivation example

$$\begin{array}{c} \frac{\frac{\frac{}{\vdash \text{"Address"} : \text{string}}{\vdash y?(3, \text{date}).0 \triangleright y : ?(3, \text{date}).\text{end}} \text{(Rcv)}}{\vdash y!(3, \text{Address}).y?(3, \text{date}).0 \triangleright y : !(3, \text{string}).?(3, \text{date}).\text{end}} \text{(Send)}}{\vdash y \oplus \langle \{2, 3\}, \text{ok} \rangle . y!(3, \text{Address}).y?(3, \text{date}).0 \triangleright y : T} \text{(Select)} \end{array}$$

where

$$T \equiv \oplus \langle \{2, 3\}, \{ \text{ok} : !(3, \text{string}).?(3, \text{date}).\text{end}, \text{quit} : \text{end} \} \rangle.$$

Example of channel delegation type derivation

How to derive a type for a process P which inputs $z : \text{Bool}$ and delegates $c' : T$?

$$P = c! \langle \langle 5, c' \rangle \rangle . c?(5, z) . 0$$

$$\frac{\{c : \text{end}\} \text{ end only}}{\text{Inact}}$$

$$z : \text{Bool} \vdash 0 \triangleright c : \text{end}$$

$$\frac{\text{Inact} \quad z : \text{Bool} \vdash 0 \triangleright c : \text{end}}{\vdash c?(5, z) . 0 \triangleright c : ?(5, \text{Bool}) . \text{end}} \text{ (Rcv)}$$

$$\frac{\vdash c?(5, z) . 0 \triangleright c : ?(5, \text{Bool}) . \text{end}}{\vdash c! \langle \langle 5, c' \rangle \rangle . c?(5, z) . 0 \triangleright c : ! \langle 5, T \rangle . ?(5, \text{Bool}) . \text{end}, c' : T} \text{ (Deleg)}$$

Example of recursive type derivation

How do we derive a type of a recursive process with a boolean expression ?

def $X(y,x) = x!\langle 1,y \rangle.X\langle y,x \rangle$ in $X\langle \text{true},c \rangle$

Let $T = !\langle 1, \text{Bool} \rangle.t$

$$\frac{\frac{\vdash \text{true} : \text{Bool}}{\text{⊖} \quad X : \text{Bool} \ \mu t.T \vdash X\langle \text{true},c \rangle \triangleright c : \mu t.T} \text{(Var)}}{\vdash \text{def } X(y,x) = x!\langle 1,y \rangle.X\langle y,x \rangle \text{ in } X\langle \text{true},c \rangle \triangleright c : \mu t.T} \text{(Def)}$$

where Θ is the derivation :

$$\frac{\frac{y : \text{Bool} \vdash y : \text{Bool}}{\text{---}} \text{(Var)} \quad y : \text{Bool} \vdash y : \text{Bool} \quad X : \text{Bool} \ \mathbf{t}, y : \text{Bool} \vdash X \langle y, x \rangle \triangleright x : \mathbf{t}}{\text{---}} \text{(Send)} \\ X : \text{Bool} \ \mathbf{t}, y : \text{Bool} \vdash x! \langle 1, y \rangle . X \langle y, x \rangle \triangleright x :! \langle 1, \text{Bool} \rangle . \mathbf{t}$$

Three Buyers - Seller Example

G1

$A \rightarrow S : \langle \text{String} \rangle.$

$S \rightarrow \{A, B\} : \langle \text{Int} \rangle.$

$A \rightarrow B : \langle \text{Int} \rangle.$

$B \rightarrow \{S, A\} : \{ \text{ok} : B \rightarrow S : \langle \text{String} \rangle.$

$S \rightarrow B : \langle \text{Date} \rangle.$

$\text{quit} : \}$

G2

$B \rightarrow C : \langle \text{Int} \rangle.$

$B \rightarrow C : \langle T \rangle.$

$C \rightarrow B : \{ \text{ok} : .. \text{quit} : ... \}$

$T =$
 $+(\{S, A\},$
 $\text{ok} : !\langle S, \text{string} \rangle,$
 $\quad ?\langle S, \text{date} \rangle,$
 $\text{quit} :)$

Type derivation for the process Alice

$$\text{Alice} = a[2](y).y!(3, \textit{Title}).\underbrace{y?(3, \textit{quote}).y!\langle 1, \textit{quotediv2}\rangle}_{act}.$$
$$\underbrace{y\&(1, \{ok : \mathbf{0}, quit : \mathbf{0}\})}_P$$

$$\begin{array}{c}
\frac{\Gamma \vdash 0 \triangleright y : \text{end}}{\text{(Branch)}} \\
\frac{\Gamma, \text{quote} : \text{int} \vdash \text{quote} : \text{int} \quad \Gamma \vdash P \triangleright y : \overbrace{\&(1, \{\text{ok} : \text{end}, \text{quit} : \text{end}\})}^{T_0}}{\text{(Send, Rcv)}} \\
\frac{\Gamma \vdash \text{"Title"} : \text{string} \quad \Gamma \vdash \text{act}.P \triangleright y : ?(3, \text{int}).!\langle 1, \text{int} \rangle.T_0}{\text{(Send)}} \\
\Gamma \vdash \underbrace{y!\langle 3, \text{"Title"} \rangle.\text{act}.P}_{P_1} \triangleright y : \underbrace{!\langle 3, \text{string} \rangle.?(3, \text{int}).!\langle 1, \text{int} \rangle.T_0}_{G \upharpoonright \text{Alice}} \\
\dots \\
\frac{\vdash a : G \quad \vdash P_1 \triangleright y : G \upharpoonright \text{Alice} \quad 2 < mp(G) = 3}{\vdash a[2](y).P_1 \triangleright \emptyset} \text{(Macc)}
\end{array}$$

Properties of Session Types

Intuitive Syntax, Light Weight Type Checking

1 Communication Error-Freedom

No Communication Mismatch

2 Session Fidelity

The Communication Sequence in a session follows the scenario declared in the types

3 Progress

Single

No Deadlock / Stuck in a session

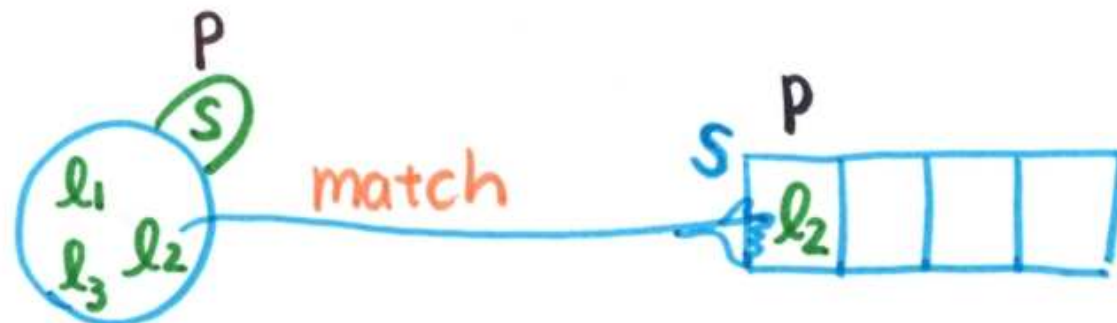
Communication Error Freedom

X $s?[P](q, x); s'! \langle x+1 \rangle \mid s[q]! \langle p, \text{"Apple"} \rangle$
Value Error

X $s?[P](q, y); P \mid s?[P](q, y'); P'$
Linearity Error

X $s![P] \langle q, V \rangle \mid s![P] \langle q, W \rangle; P'$
Linearity Error

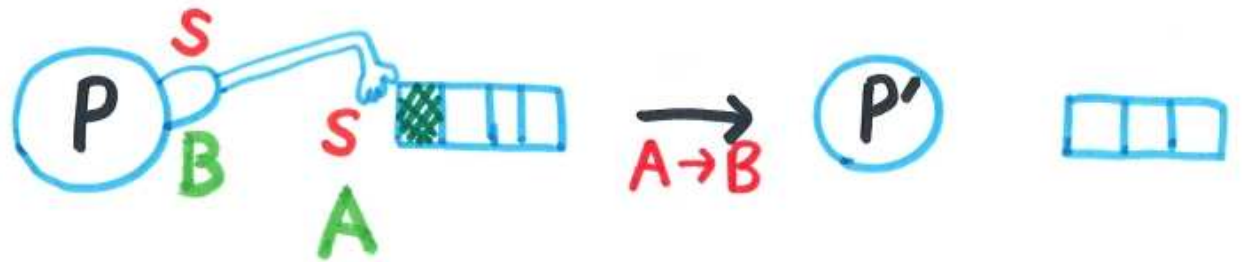
Branch



Session Fidelity

$$\Gamma \vdash P \triangleright \Delta$$

$$P \xrightarrow{A \rightarrow B} P'$$



$$\Delta \xrightarrow{A \rightarrow B} \Delta'$$

$$\underline{s! \langle U \rangle; T} \mid \underline{s? \langle U' \rangle; T'} \xrightarrow{A \rightarrow B} T \mid T'$$



$$\xrightarrow{A \rightarrow B} T \mid T'$$



$$G \xrightarrow{A \rightarrow B} G'$$

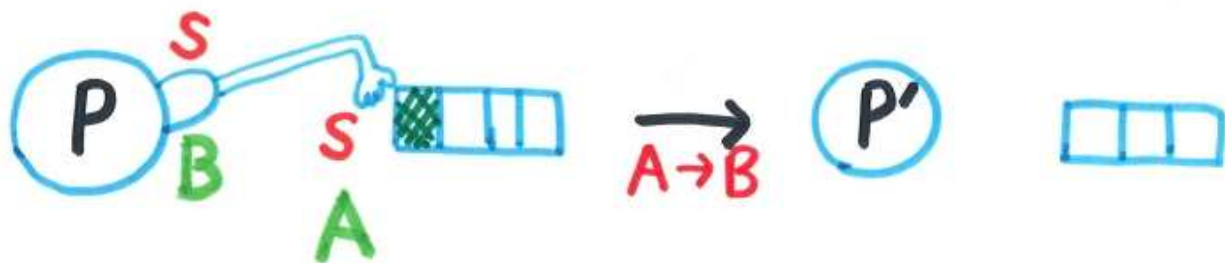
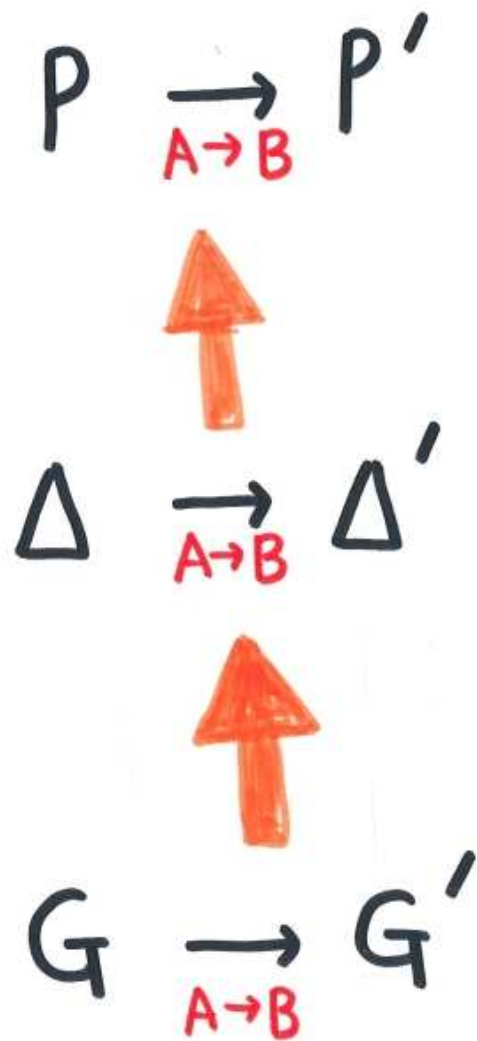
$$A \rightarrow B: \langle U \rangle$$

$$\xrightarrow{A \rightarrow B} G'$$

Progress

$\Gamma \vdash P \triangleright \Delta$

P is in
a single session



$\underline{S! \langle U \rangle}; T \mid \underline{S? \langle U' \rangle}; T' \xrightarrow{A \rightarrow B} T \mid T'$



$A \rightarrow B: \langle U \rangle$



Subject Reduction Theorem

- Subject Congruence

$$\Gamma \vdash P \triangleright \Delta \text{ and } P \equiv Q \implies \Gamma \vdash Q \triangleright \Delta$$

- Subject Reduction

$$\Gamma \vdash P \triangleright \Delta \text{ and } P \rightarrow Q \implies \Gamma \vdash Q \triangleright \Delta' \text{ with } \Delta \rightarrow \Delta'$$

- The typing system for runtime processes and the proofs can be found in the lecture notes.

Single Multiparty Session Progress

- Suppose $a : G \vdash P_1 \mid \dots \mid P_n \triangleright \emptyset$ where P_i does not contain any restriction and either an accept or a request.

Suppose $P_1 \mid \dots \mid P_n \rightarrow^+ Q$. Then $Q \equiv \mathbf{0}$ or $\exists R. Q \rightarrow R$.

- The proofs (for more general progress) can be found in [\[POPL'08\]](#).
- Progress for the interleaved sessions are guaranteed by the interaction typing system in [\[Coppo, Dezani, Padvani and NY'14\]](#).

Multiparty Session Type Theory

- ▶ Multiparty Asynchronous Session Types [POPL'08]
- ▶ Progress
 - Global Progress in Dynamically Interleaved Multiparty Sessions [CONCUR'08], [Math. Struct. Comp. Sci.]
 - Inference of Progress Typing [Coordination'13]
- ▶ Asynchronous Optimisations and Resource Analysis
 - Global Principal Typing in Partially Commutative Asynchronous Sessions [ESOP'09]
 - Higher-Order Pi-Calculus [TLCA'07, TLCA'09]
 - Buffered Communication Analysis in Distributed Multiparty Sessions [CONCUR'10]

➤ Logics

- Design-by-Contract for Distributed Multiparty Interactions [CONCUR'10]
- Specifying Stateful Asynchronous Properties [CONCUR'12]
- Multiparty, Multi-session Logic [TGC'12]

➤ Extensions of Multiparty Session Types

- Multiparty Symmetric Sum Types [Express'10]
- Trustworthy Pervasive Healthcare Services via Multi-party Session Types [FHIES'12]
- Parameterised Multiparty Session Types [FoSSaCs'10, LMCS]
- Global Escape in Multiparty Sessions [FSTTCS'10]
[Math. Struct. Comp. Sci.]
- Dynamic Multirole Session Types [POPL'11]
- Nested Multiparty Sessions [CONCUR'12]
- Timed Multiparty Session Types [CONCUR'14]

- Dynamic Monitoring
 - Monitoring Networks through Multiparty Sessions [TGC'11] [FORTE'13]
- Automata Theories
 - Multiparty Session Automata [ESOP'12]
 - Synthesis in Communicating Automata [ICALP'13]
 - From communicating machines to graphical choreographies [POPL'15]
- Petri Nets
 - Multiparty Session Nets [TGC'14]
- Typed Behavioural Theories
 - Governed Session Semantics [CONCUR'13]
- Choreography Languages
 - Compositional Choreographies [CONCUR'13]

Session Type Reading List

- Home Page <http://mrg.doc.ic.ac.uk/>
- [ESOP'98] Language Primitives and Type Disciplines for Structured Communication-based Programming, Honda, Vasconcelos and Kubo
- [SecRet'06] Language Primitives and Type Disciplines for Structured Communication-based Programming *Revisited*, Yoshida and Vasconcelos, ENTCS.
- [SFM'15] Gentle Introduction to Multiparty Asynchronous Session Types, Coppo et al.
- [POPL'15] From communicating machines to graphical choreographies, Lange, Tuosto and Yoshida.

- [COB'14,TGC'13] The Scribble Protocol Language, Honda et al.
- [ECOOP'08] Session-Based Distributed Programming in Java, Hu, Yoshida and Honda.
- [FMSD'15] Practical interruptible conversations: Distributed dynamic verification with multiparty session types and Python, Demangeon, Honda, Hu, Neykova and Yoshida.
- [CC'15] Protocols by Default: Safe MPI Code Generation based on Session Types, Ng, Coutinho and Yoshida.