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.

Origin of Multiparty Session Types

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

$\Downarrow$

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

$\Downarrow$

Formalisation of W3C WS-CDL [ESOP'07]

$\Downarrow$

Scribble at $\pi 4$ 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]

$\Downarrow$

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

$\Downarrow$

Formalisation of W3C WS-CDL [ESOP'07]

$\Downarrow$

Scribble at $\pi 4$ Technology

$\Downarrow$

Multiparty Session Types [POPL'08]

$\Downarrow$

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]

⇓

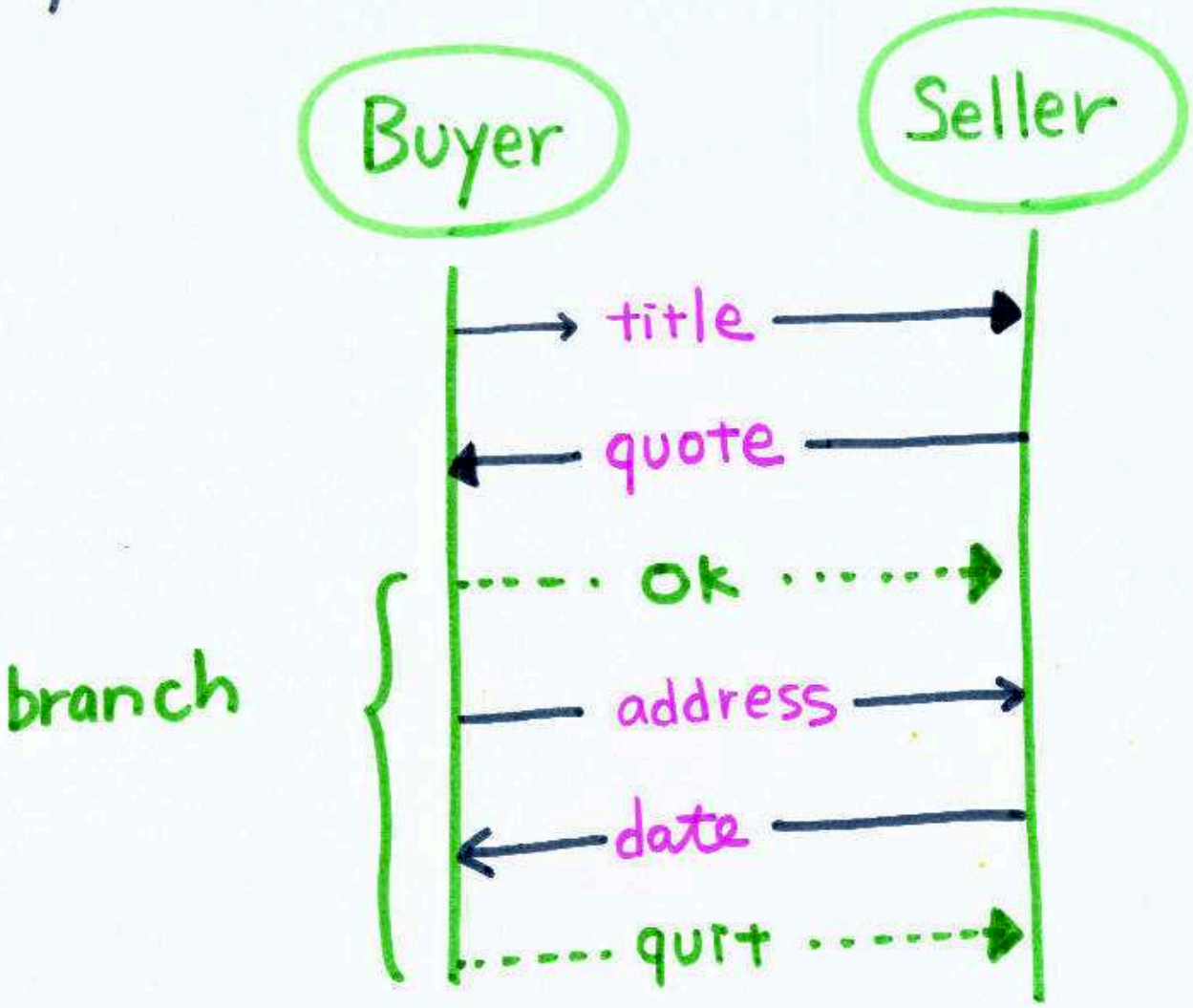
Scribble at π4 Technology

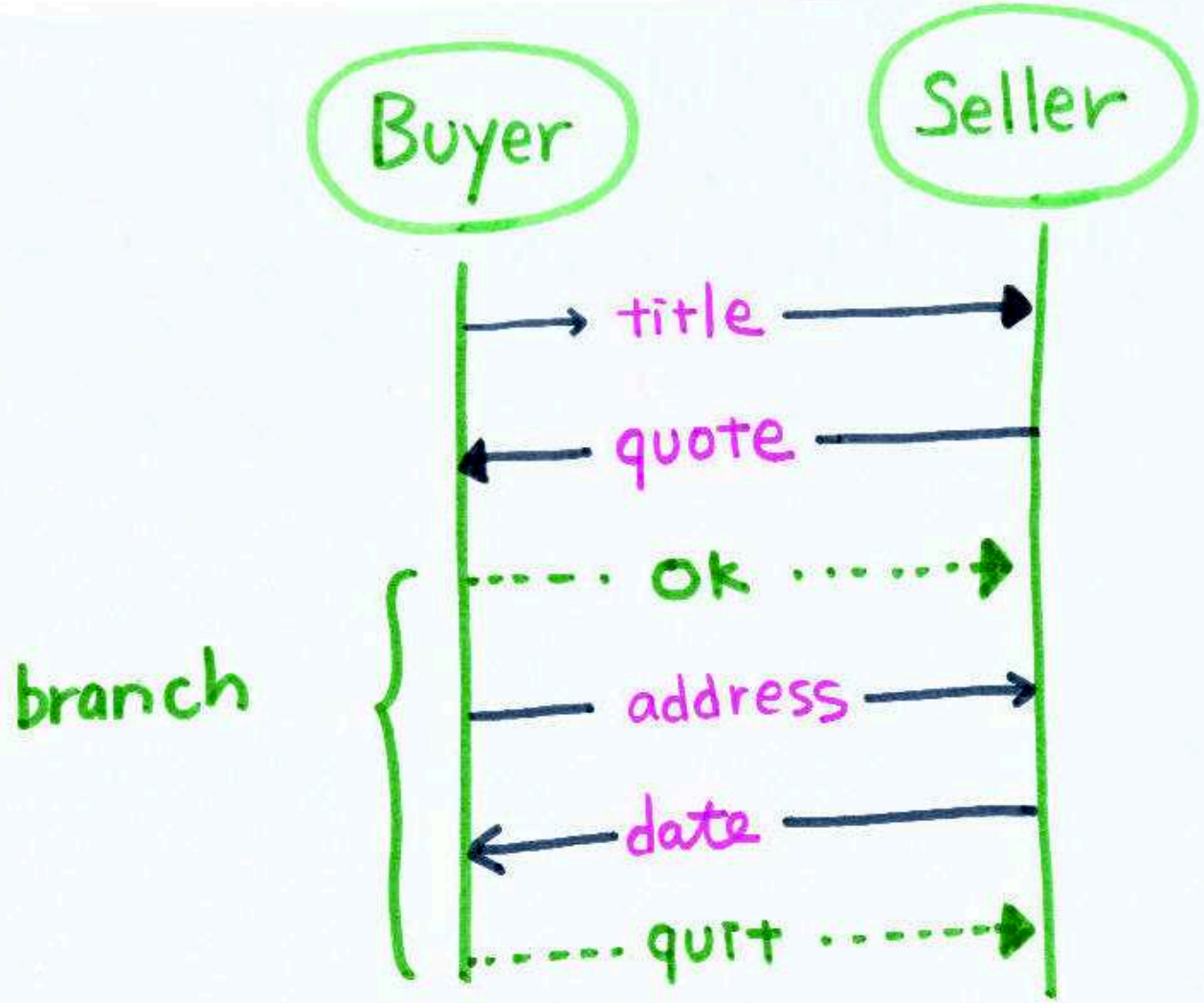
⇓

Multiparty Session Types [POPL'08]

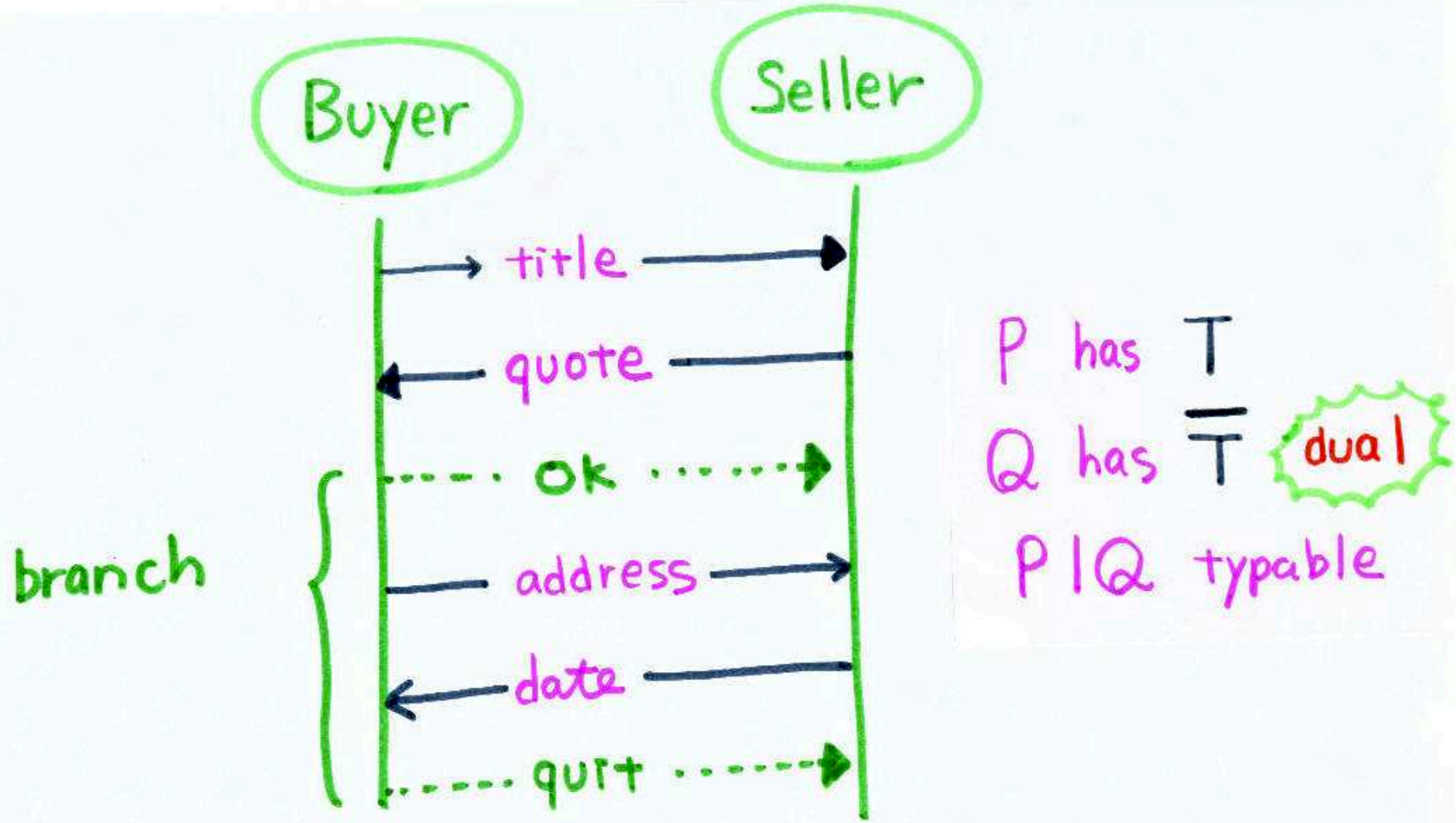
⇓

Binary Session Types : Buyer - Seller Protocol





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



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

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

Binary Session Types

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

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

Binary Session Types

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

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

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

$a(x).\ x?(y);x!\langle 30\rangle;x\{\mathrm{ok}\triangleright x?(z).\ x!\langle 01/01\rangle;0$
$\mathrm{quit}\triangleright 0\}$

if T and $\overline{T}$ then $P\mid Q$ typable

Properties of Session Types

Intuitive Syntax, Light Weight Type Checking

[1] Communication Error-Freedom

No Communication Mismatch

[2] Session Fidelity

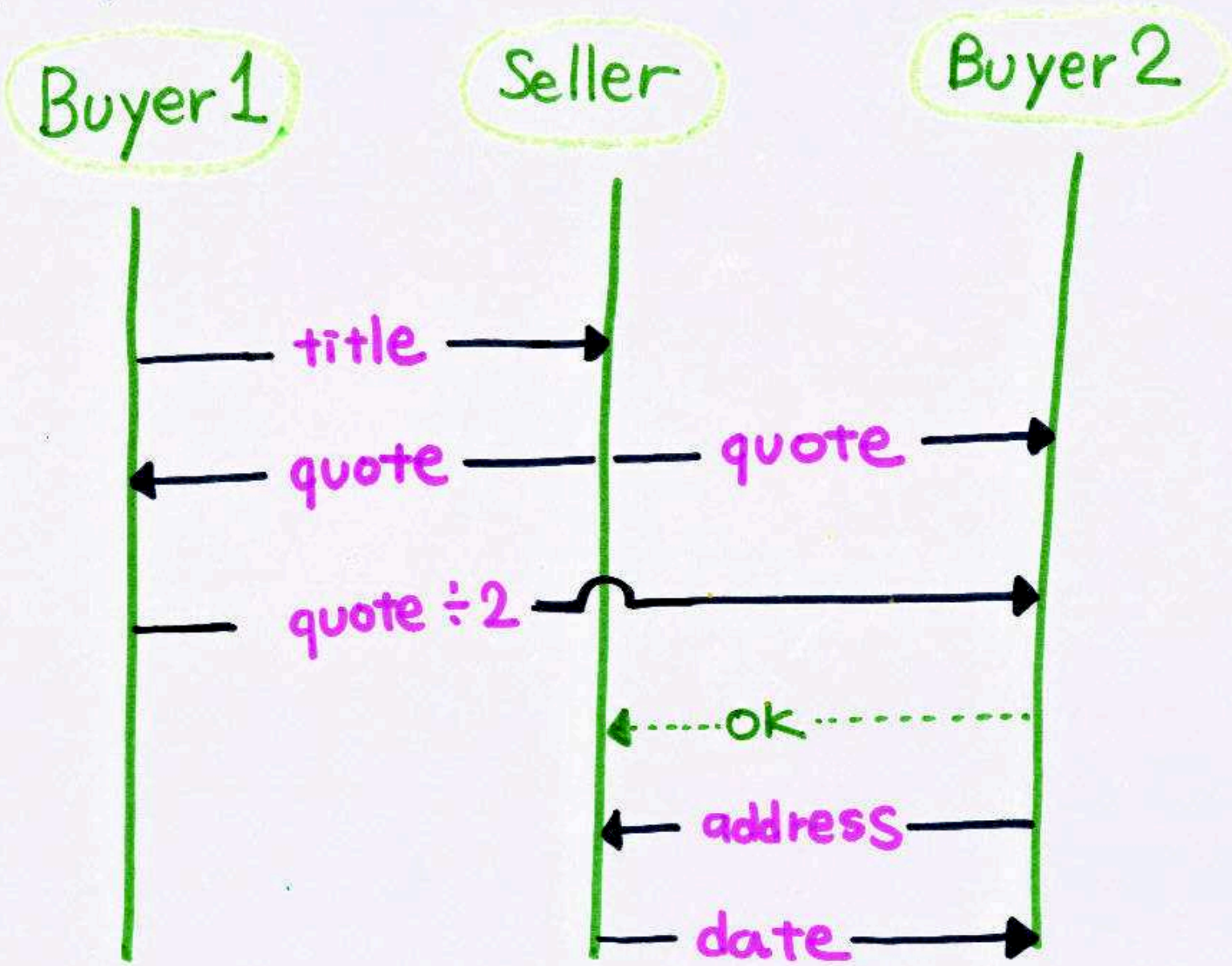
The Communication Sequence in a session follows the senario declared in the types

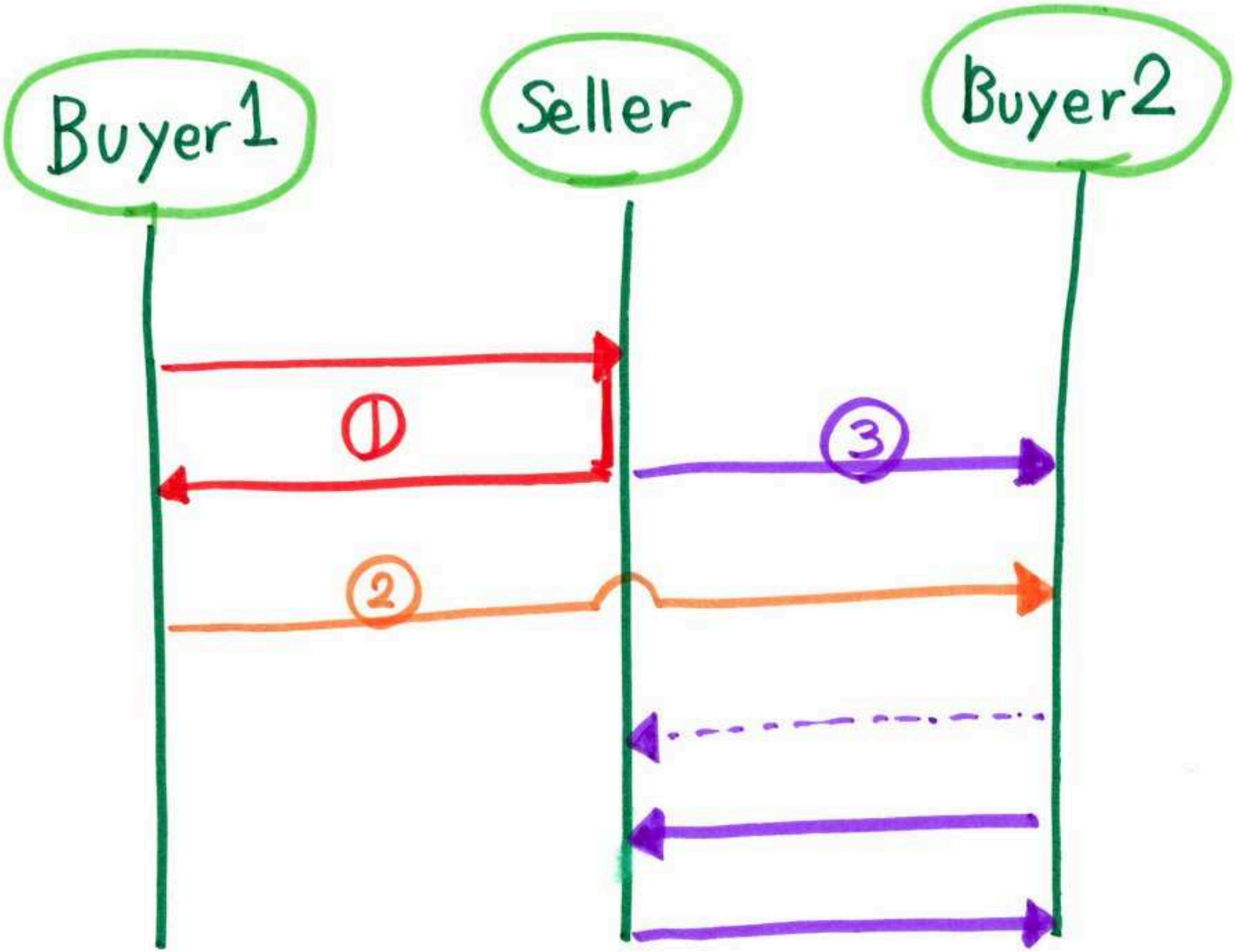
[3] Progress

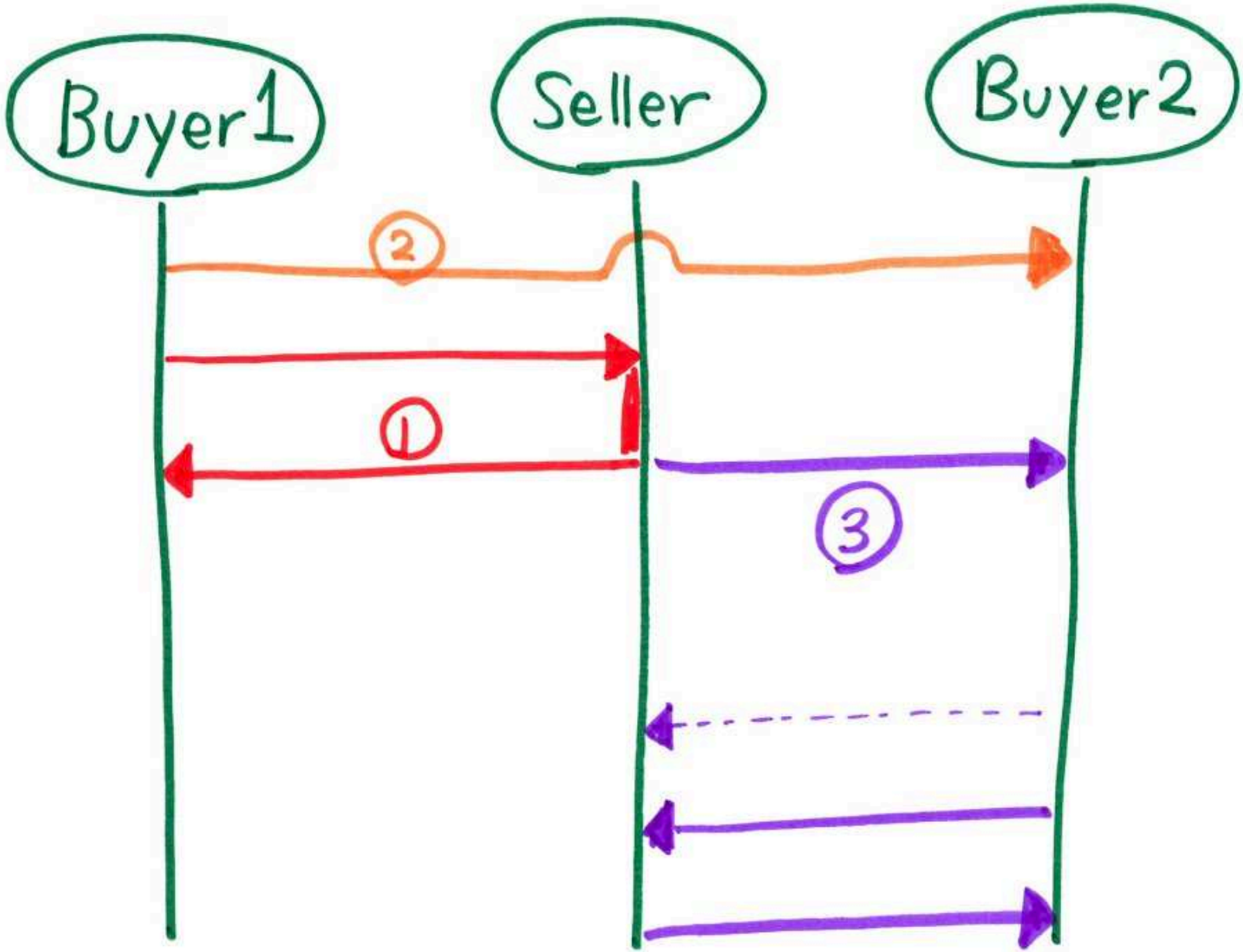
Single

No Deadlock / Stuck in a session

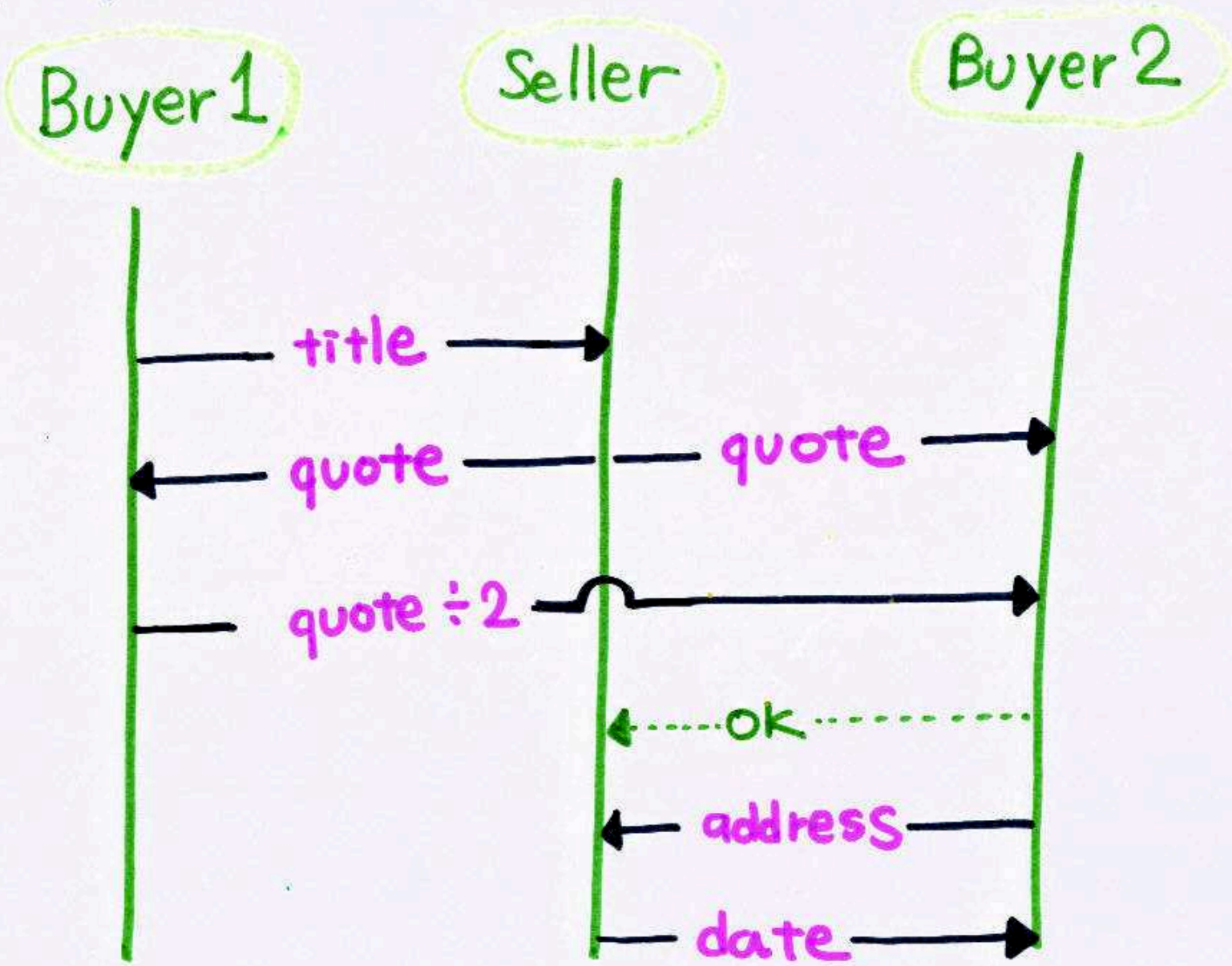
Multiparty Session Types




Buyer1
Seller
Buyer2
1
3
2

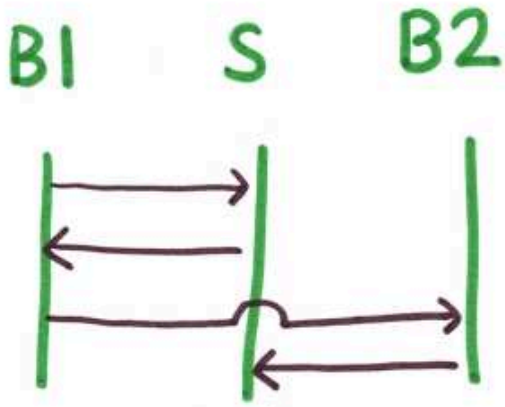

Buyer1
Seller
Buyer2
2
1
3

Multiparty Session Types



Multiparty Session Types

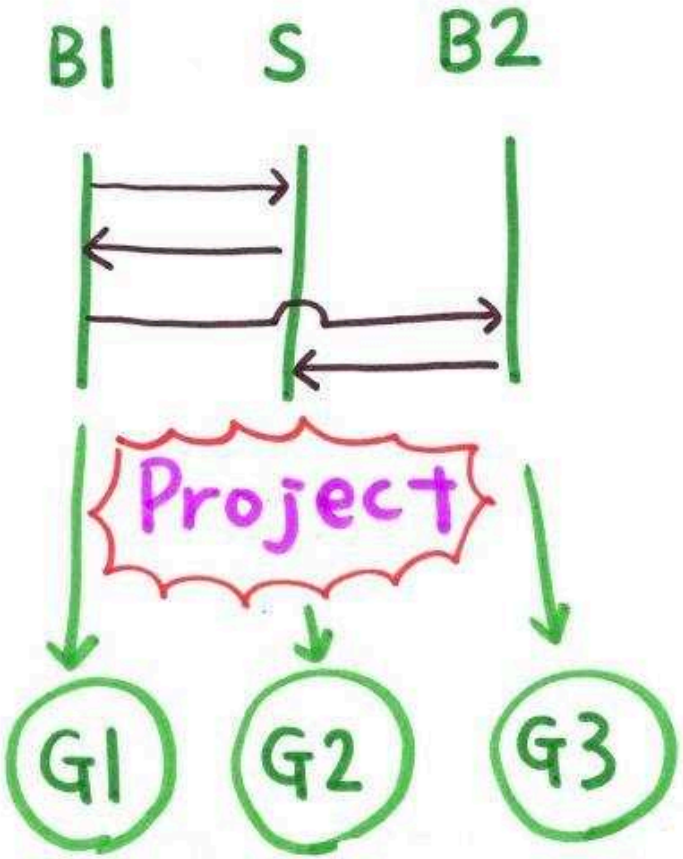
cf. Abstract Choreography [WS-CDL]



Step 1

Write Global Types

Multiparty Session Types



cf. Abstract Choreography [WS-CDL]

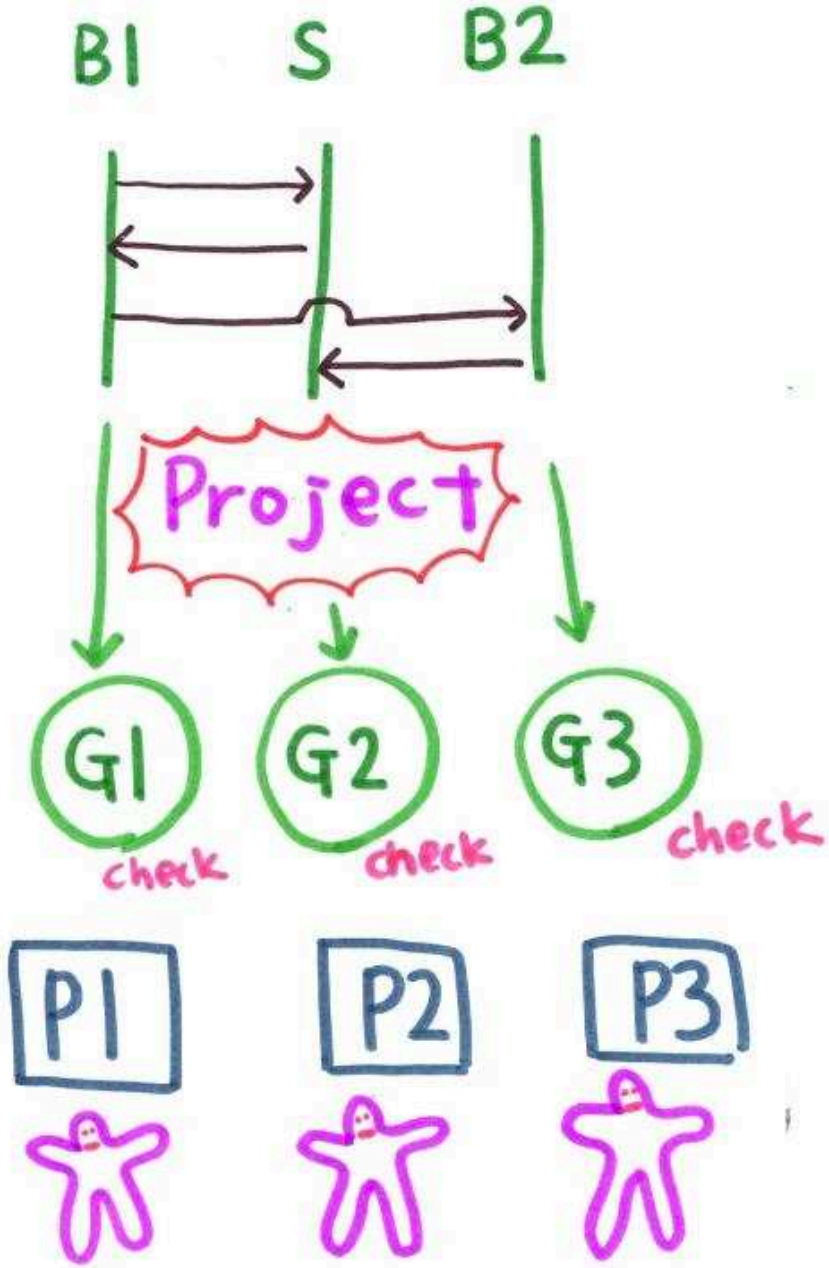
Step 1

Write Global Types

Step 2

Project Local Types

Multiparty Session Types



cf. Abstract Choreography [WS-CDL]

Step 1

Write Global Types

Step 2

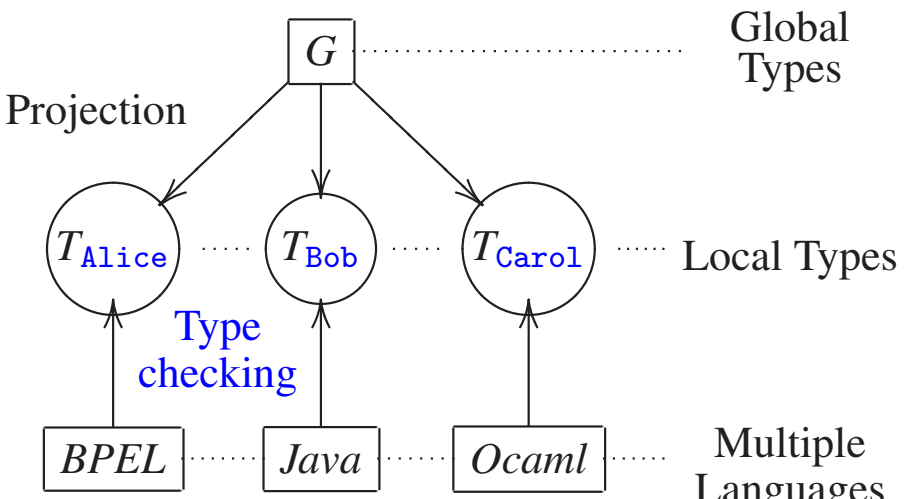
Project

Step 3

Write Local Programs

Type and Check Locally

Multiparty Session Types

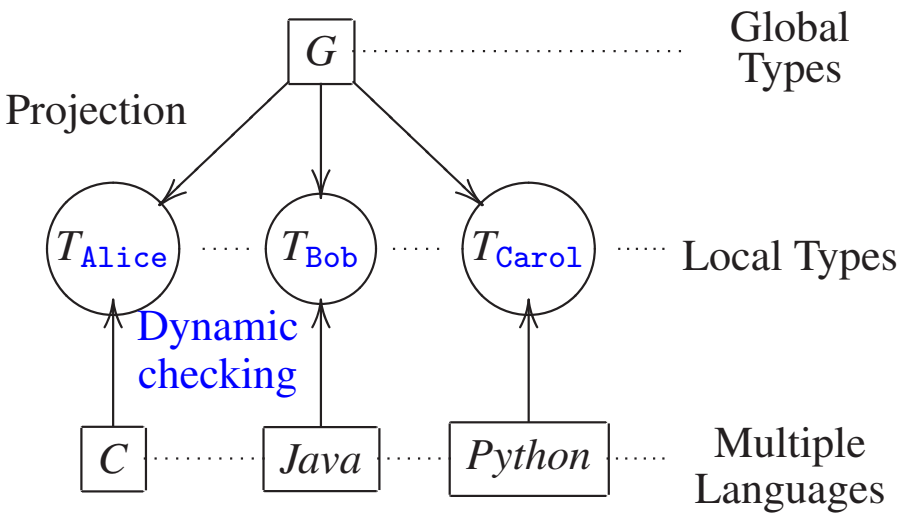


$\texttt{Alice} \to \texttt{Bob}\colon \langle \mathsf{Nat} \rangle.$
$\texttt{Bob} \to \texttt{Carol}\colon \langle \mathsf{Nat} \rangle.\mathsf{end}$

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

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

Multiparty Session Types

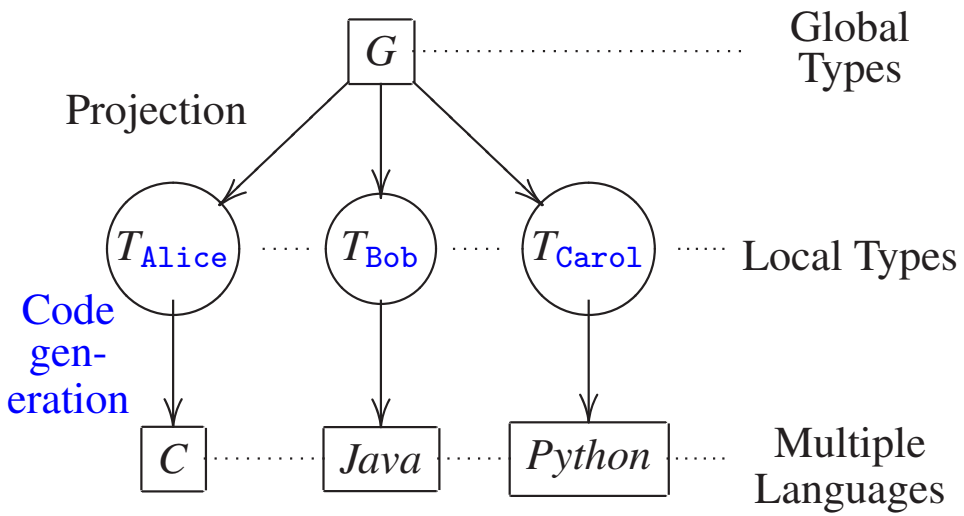


$\texttt{Alice} \to \texttt{Bob}\colon \langle \mathsf{Nat} \rangle.$
$\texttt{Bob} \to \texttt{Carol}\colon \langle \mathsf{Nat} \rangle.\mathsf{end}$

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

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

Multiparty Session Types



$\texttt{Alice} \to \texttt{Bob}\colon \langle \mathsf{Nat} \rangle.$
$\texttt{Bob} \to \texttt{Carol}\colon \langle \mathsf{Nat} \rangle.\mathsf{end}$

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

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

Global Types

global

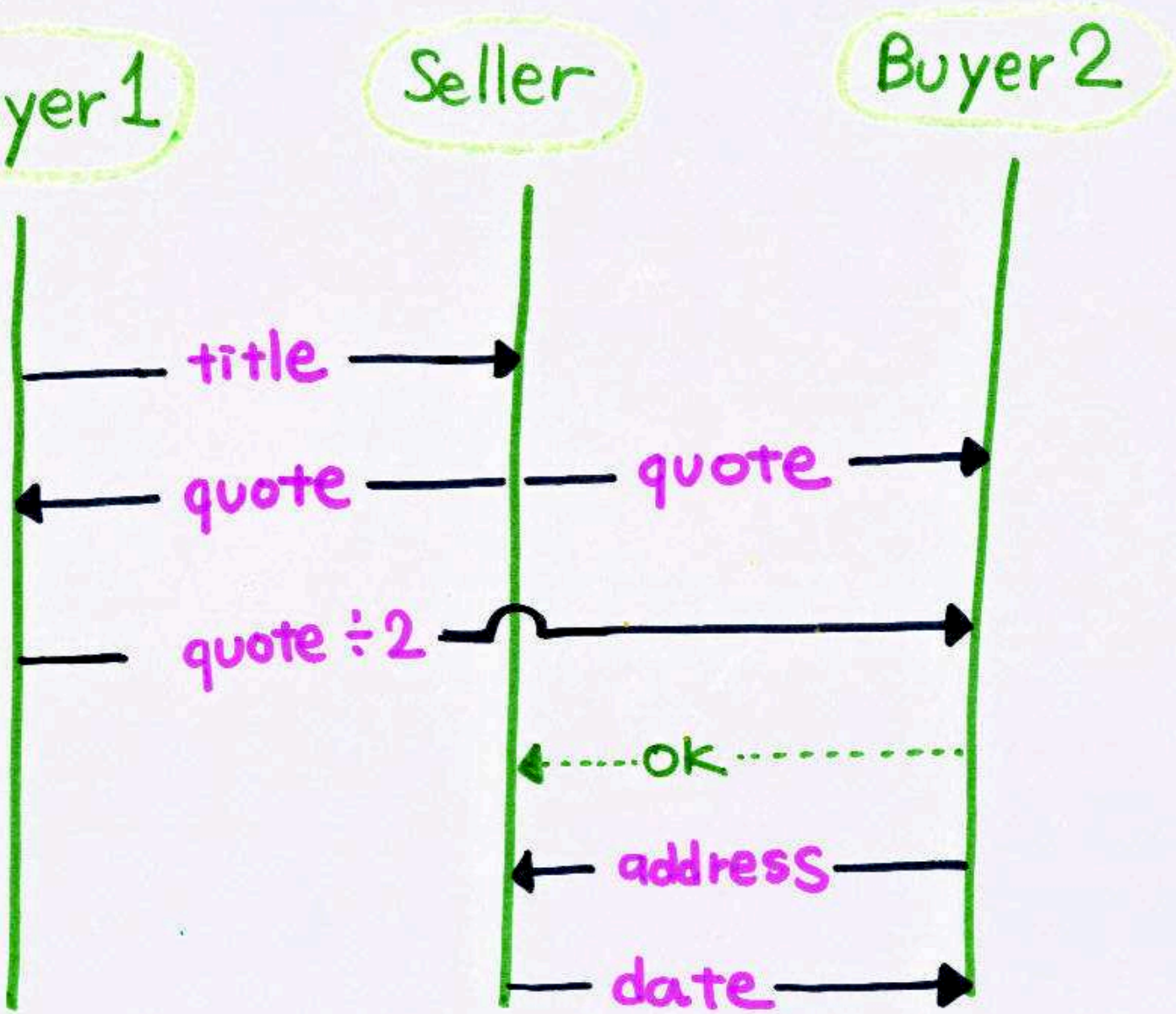
$$G ::= p \to p_1, \ldots, p_n : \langle U \rangle . G'$$
$$\mid p \to p_1, \ldots, p_n : \{ l_j : G_j \}_{j \in J}$$
$$\mid \mu t . G$$
$$\mid \text{end} \mid t$$

value

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

Session

Multiparty Session Types



Three Buyers - Seller Example

G1

Alice = Buyer 1

Bob = Buyer 2

$A \to S : \langle String \rangle.$

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

$A \to B : \langle Int \rangle.$

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

$S \to B : \langle Date \rangle.$

$quit : \}$

Projections of the Global type G_1

$$
\begin{aligned}
\mathsf{G}_1 \upharpoonright S &= ?(A, \mathsf{string}).!\langle \{A, B\}, \mathsf{int} \rangle. \\
&\quad \&(B, \{\mathsf{ok} : ?(B, \mathsf{string}).!\langle B, \mathsf{date} \rangle.\mathsf{end}, \\
&\qquad \mathsf{quit} : \mathsf{end}\}) \\
\\
\mathsf{G}_1 \upharpoonright A &= !\langle S, \mathsf{string} \rangle.?(S, \mathsf{int}).!\langle B, \mathsf{int} \rangle.\&(B, \{\mathsf{ok} : \mathsf{end}, \mathsf{quit} : \mathsf{end}\}) \\
\\
\mathsf{G}_1 \upharpoonright B &= ?(S, \mathsf{int}).?(A, \mathsf{int}). \\
&\quad \oplus\langle \{S, A\}, \{\mathsf{ok} : !\langle S, \mathsf{string} \rangle.?(S, \mathsf{date}).\mathsf{end}, \\
&\qquad \mathsf{quit} : \mathsf{end}\}) \rangle
\end{aligned}
$$

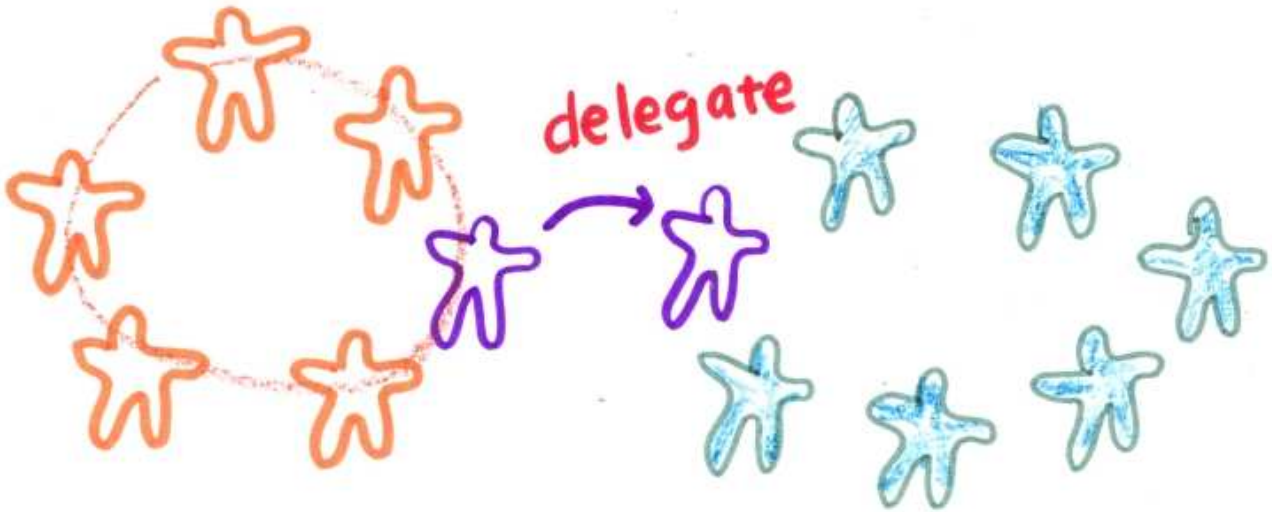
The projection of a global type G onto a participant q

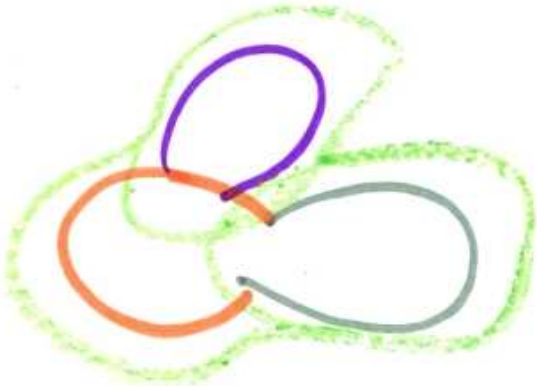
$$(\mathtt{p} \to \mathtt{p}' : \langle U \rangle . G') \upharpoonright \mathtt{q} = \begin{cases} !\langle \mathtt{p}', U \rangle . (G' \upharpoonright \mathtt{q}) & \text{if } \mathtt{q} = \mathtt{p}, \\ ?(\mathtt{p}, U) . (G' \upharpoonright \mathtt{q}) & \text{if } \mathtt{q} = \mathtt{p}', \\ G'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

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

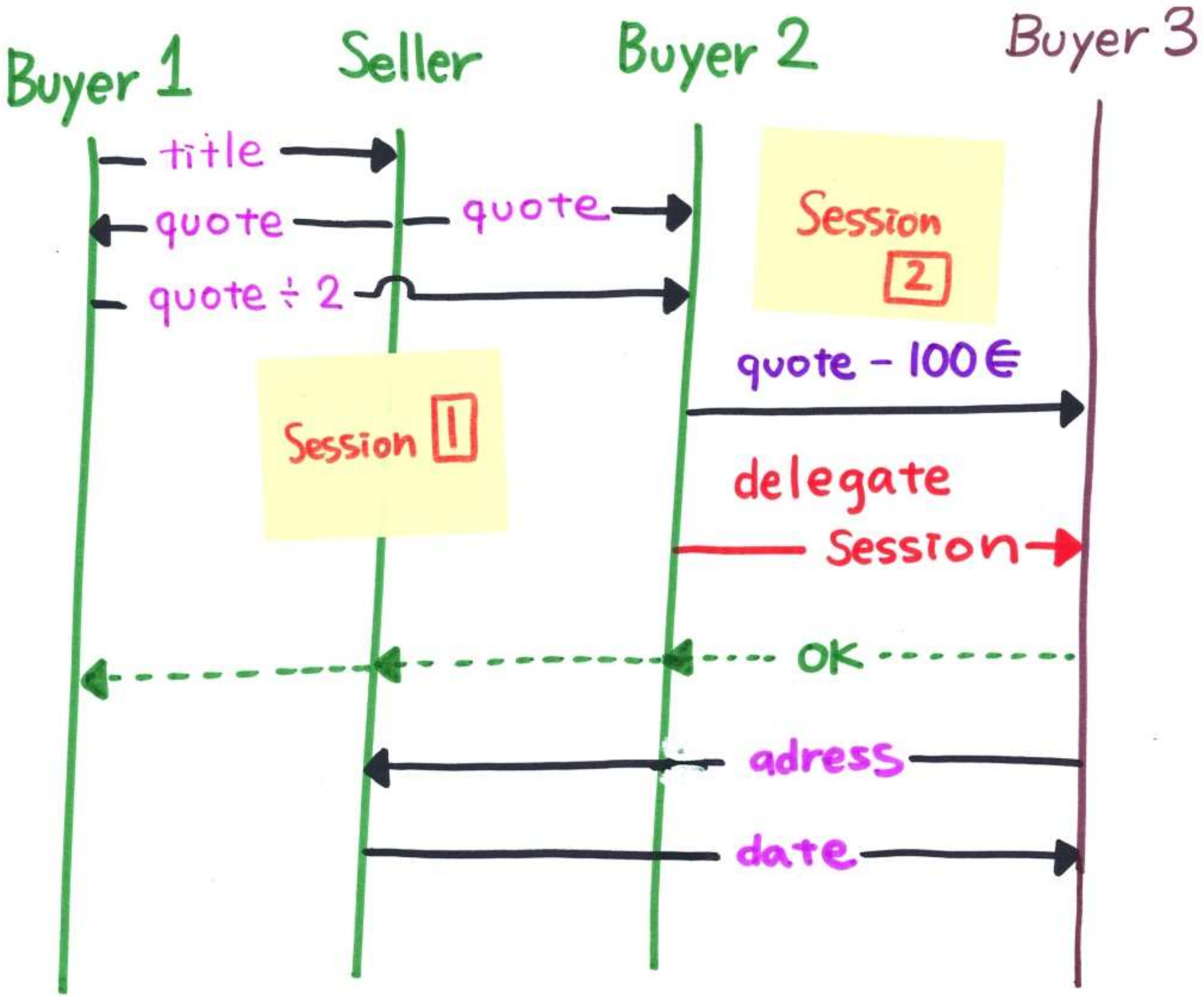
$$(\mu \mathbf{t} . G) \upharpoonright \mathtt{q} = \begin{cases} \mu \mathbf{t} . (G \upharpoonright \mathtt{q}) & \text{if } G \upharpoonright \mathtt{q} \neq \mathbf{t}, \\ \mathsf{end} & \text{otherwise.} \end{cases} \qquad \mathbf{t} \upharpoonright \mathtt{q} = \mathbf{t} \quad \mathsf{end} \upharpoonright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

Multiparty Sessions + Dynamic Mobility of Sessions Delegation = Interleaved Multiparty Sessions





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


Buyer 1
Seller
Buyer 2
Buyer 3
title
quote
quote
quote ÷ 2
Session 2
quote - 100€
Session 1
delegate
Session
OK
adress
date

Three Buyers- Seller Example

G1

$A \to S : \langle String \rangle.$

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

$A \to B : \langle Int \rangle.$

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

$S \to B : \langle Date \rangle.$

$quit: \}$

G2

$B \to C : \langle Int \rangle.$

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

$C \to B : \{ ok: \ldots \ quit: \ldots \}$

T =
+({S,A},
ok:!<S,string>;
?<S,date>,
quit:)

Global types for the three buyer protocol

$$
\begin{array}{lrcl}
\mathsf{G}_a = & 2 & \longrightarrow 3 & : \langle \mathsf{string} \rangle . \\
& 3 & \longrightarrow \{1,2\} & : \langle \mathsf{int} \rangle . \\
& 2 & \longrightarrow 1 & : \langle \mathsf{int} \rangle . \\
& 1 & \longrightarrow \{2,3\} & : \{\mathsf{ok} : 1 \longrightarrow 3 : \langle \mathsf{string} \rangle . \\
& & & \qquad 3 \longrightarrow 1 : \langle \mathsf{date} \rangle . \mathsf{end}, \\
& & & \quad \mathsf{quit} : \mathsf{end}\}
\end{array}
$$

$$
\begin{array}{lrcl}
\mathsf{G}_b = & 2 & \longrightarrow 1 & : \langle \mathsf{int} \rangle . \\
& 2 & \longrightarrow 1 & : \langle T \rangle . \\
& 1 & \longrightarrow 2 & : \{\mathsf{ok} : \mathsf{end}, \mathsf{quit} : \mathsf{end}\}
\end{array}
$$

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

Projections of the three buyer protocol

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

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

Implementation of the three buyer protocol

$$\begin{aligned}\texttt{Seller} = {} & \overline{a}[3](y).y?(2,\mathit{title}).y!\langle\{1,2\},\mathsf{quote}\rangle. \\ & y\&(1,\{\mathsf{ok} : y?(1,\mathit{address}). \\ & y!\langle 1,\mathsf{date}\rangle.\mathbf{0},\mathsf{quit} : \mathbf{0}\})\end{aligned}$$

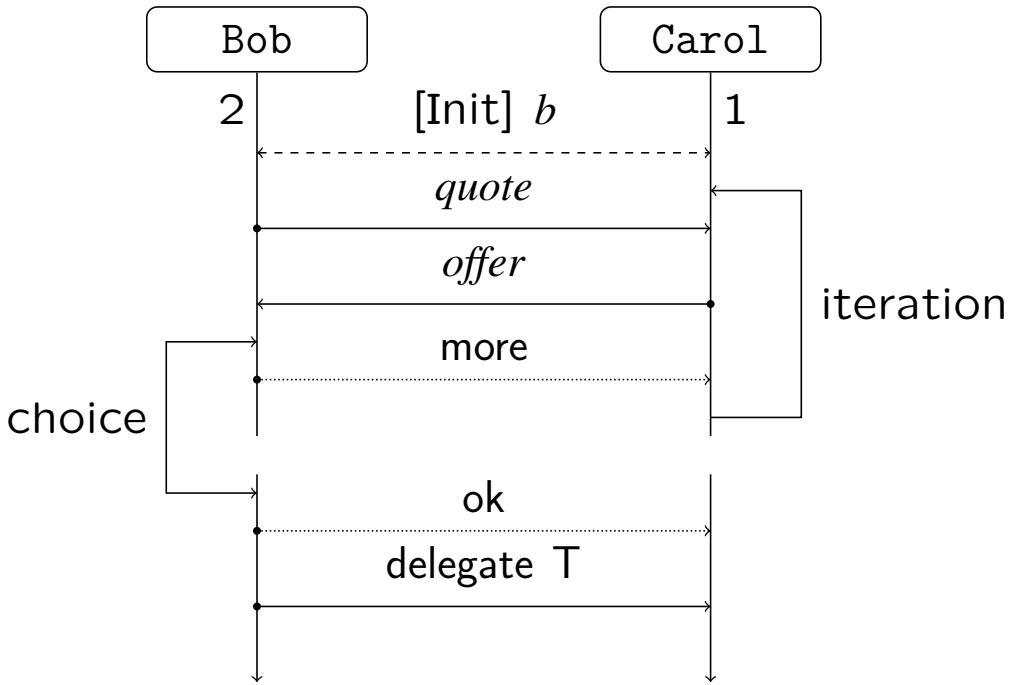
$$\begin{aligned}\texttt{Alice} = {} & a[2](y).y!\langle 3,\text{"}\texttt{Title}\text{"}\rangle.y?(3,\mathit{quote}). \\ & y!\langle 1,\mathit{quote}\,\mathsf{div}\,2\rangle. \\ & y\&(1,\{\mathsf{ok} : \mathbf{0},\ \ \mathsf{quit} : \mathbf{0}\})\end{aligned}$$

Implementation of the three buyer protocol

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

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

The three buyer protocol with recursion



Global type for recursive negotiation

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

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

The three buyer example with recursion

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

The three buyer example with recursion

$$
\begin{aligned}
\texttt{Carol} = {} & b[1](z).\mathsf{def}\ Y(z') = \\
& z'?(2,x).z'!\langle 2, \mathit{offer}(x)\rangle. \\
& z'\&(2,\{\mathsf{ok} : z'?((2,t)).t \oplus \langle\{2,3\},\mathsf{ok}\rangle. \\
& \qquad\qquad t!\langle 3,\text{"}\texttt{Address}\text{"}\rangle.t?(3,\mathit{date}).\mathbf{0} \\
& \qquad\quad \mathsf{more} : Y\langle z'\rangle, \\
& \qquad\quad \mathsf{quit} : \mathbf{0}\}) \\
& \mathsf{in}\ Y\langle z'\rangle
\end{aligned}
$$

Dynamic Semantics

Multicast

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

$$\rightarrow (\nu\, s)\,(\, P_1[s[1]/y] \mid P_2[s[2]/y] \mid \cdots \mid P_n[s[n]/y] \mid s : \emptyset \,)$$

queue

a service channel .... shared / interfered

s session channel .... linearised

Send

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

Recv

$$s[p]?(q, x); P \mid s : (q, p, V) \cdot h$$

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

Selection

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

Branching

$$s[p] \& (q, \{ \ell_i : P_i \}_{i \in I}) \mid s : (q, p, \ell_j) \cdot h \rightarrow P_j \mid s : h$$

Asynchronous Sessions

- Sender Non-Blocking
- Message Order Preserving per Session

Reduction rule for the Process Call

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

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

Communication Typing

Local Session Types

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

INIT

$$\frac{\Gamma \vdash u : \langle G \rangle \qquad \Gamma \vdash P \triangleright \Delta, y : G \lceil p}{\Gamma \vdash u[p](y).\, P \triangleright \Delta}$$

Send

$$\frac{\Gamma \vdash e : S \qquad \Gamma \vdash P \triangleright \Delta, c : T}{\Gamma \vdash c!\langle \tilde{p}, e \rangle ; P \triangleright c!\langle \tilde{p}, S \rangle ; T}$$

Par

$$\frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash Q \triangleright \Delta' \qquad \Delta, \Delta' \text{ disjoint}}{\Gamma \vdash P \mid Q \triangleright \Delta, \Delta'}$$

Typing rules for delegation and session receiving

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

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

Delegated channel type T derivation example

$$\dfrac{\dfrac{\vdash \text{"}\texttt{Address}\text{"} : \mathsf{string} \qquad \dfrac{date : \mathsf{date} \vdash 0 \triangleright y : \mathsf{end}}{\vdash y?(3, date).0 \triangleright y :?(3, \mathsf{date}).\mathsf{end}}\,(\mathsf{Rcv})}{\vdash y!\langle 3, \text{"}\texttt{Address}\text{"}\rangle.y?(3, date).0 \triangleright y :!\langle 3, \mathsf{string}\rangle.?(3, \mathsf{date}).\mathsf{end}}\,(\mathsf{Send})}{\vdash y \oplus \langle \{2,3\}, \mathtt{ok}\rangle.y!\langle 3, \text{"}\texttt{Address}\text{"}\rangle.y?(3, date).0 \triangleright y : T}\,(\mathsf{Select})$$

where

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

Example of channel delegation type derivation

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

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

$$\dfrac{\dfrac{\dfrac{\{c : \mathsf{end}\}\ \mathsf{end}\ \mathsf{only}}{z : \mathsf{Bool} \vdash 0 \triangleright c : \mathsf{end}}\ (\mathsf{Inact})}{\vdash c?(5, z).0 \triangleright c :?(5, \mathsf{Bool}).\mathsf{end}}\ (\mathsf{Rcv})}{\vdash c!\langle\langle 5, c'\rangle\rangle.c?(5, z).0 \triangleright c :!\langle 5, T\rangle.?(5, \mathsf{Bool}).\mathsf{end}, c' : T}\ (\mathsf{Deleg})$$

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 $X\langle \mathsf{true}, c\rangle$

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

$$\dfrac{\Theta \qquad \dfrac{\vdash \mathsf{true} : \mathsf{Bool}}{X : \mathsf{Bool}\ \mu\mathbf{t}.T \vdash X\langle \mathsf{true}, c\rangle \triangleright c : \mu\mathbf{t}.T}\ (\mathsf{Var})}{\vdash \mathsf{def}\ X(y,x) = x!\langle 1, y\rangle.X\langle y,x\rangle\ \mathsf{in}\ X\langle \mathsf{true}, c\rangle \triangleright c : \mu\mathbf{t}.T}\ (\mathsf{Def})$$

where Θ is the derivation :

$$\dfrac{y : \mathsf{Bool} \vdash y : \mathsf{Bool} \qquad \dfrac{y : \mathsf{Bool} \vdash y : \mathsf{Bool}}{X : \mathsf{Bool}\ \mathbf{t}, y : \mathsf{Bool} \vdash X\langle y, x\rangle \triangleright x : \mathbf{t}}\ (\mathsf{Var})}{X : \mathsf{Bool}\ \mathbf{t}, y : \mathsf{Bool} \vdash x!\langle 1, y\rangle.X\langle y, x\rangle \triangleright x :!\langle 1, \mathsf{Bool}\rangle.\mathbf{t}}\ (\mathsf{Send})$$

Three Buyers- Seller Example

G1

$A \to S : \langle String \rangle.$

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

$A \to B : \langle Int \rangle.$

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

$S \to B : \langle Date \rangle.$

$quit: \}$

G2

$B \to C : \langle Int \rangle.$

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

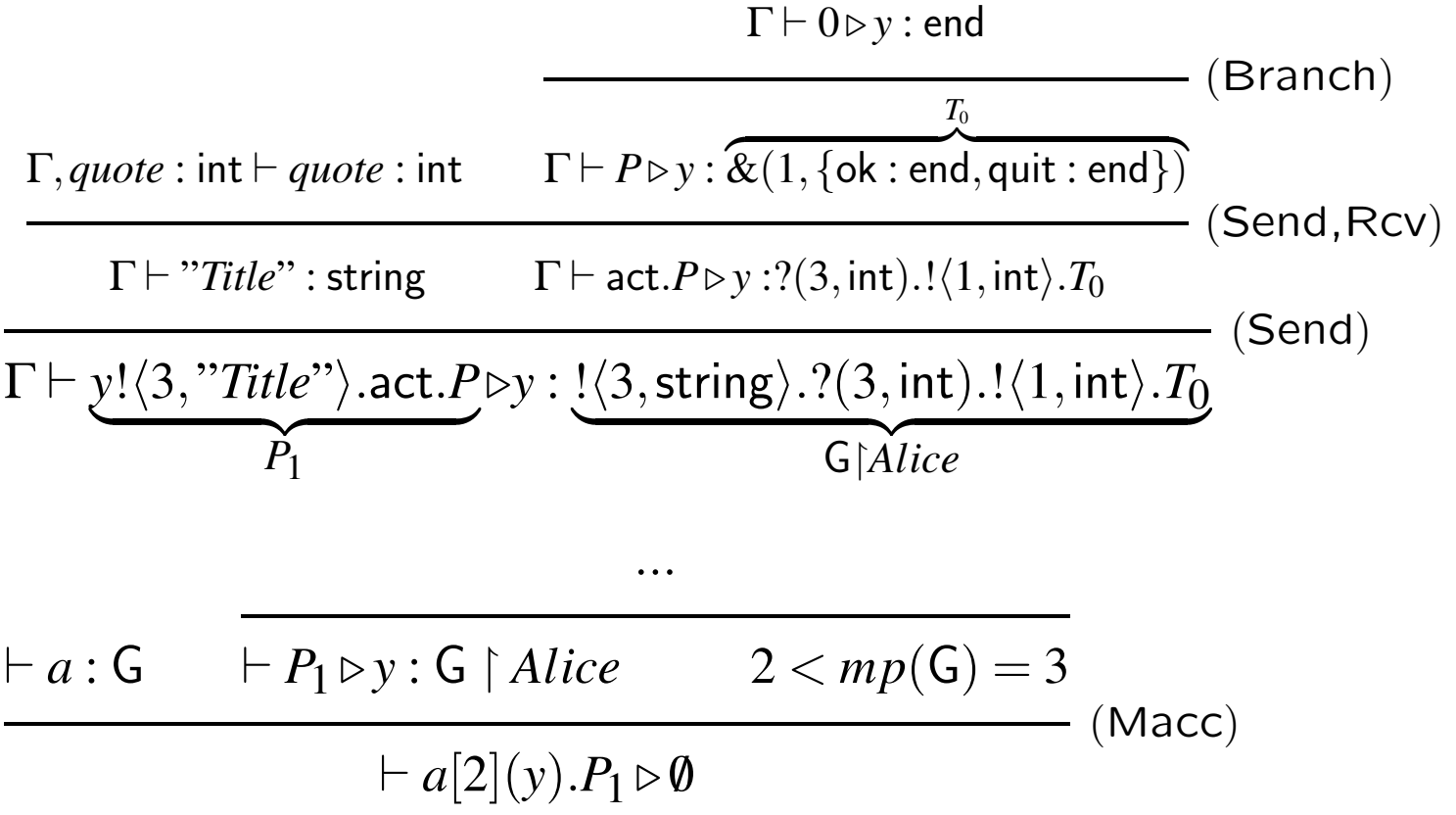
$C \to B : \{ ok: \ldots \ quit: \ldots \}$

T =
+({S,A},
ok:!<S,string>;
?<S,date>,
quit:)

Type derivation for the process Alice

$$\texttt{Alice} = a[2](y).y!(3,\text{"}Title\text{"}).\underbrace{y?(3,quote).y!\langle 1, quote\,\mathsf{div2}\rangle.}_{act}$$

$$\underbrace{y\&(1,\{ok:\mathbf{0}, quit:\mathbf{0}\})}_{P}$$


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

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 senario declared in the types

[3] Progress

Single

No Deadlock / Stuck in a session

Communication Error Freedom

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

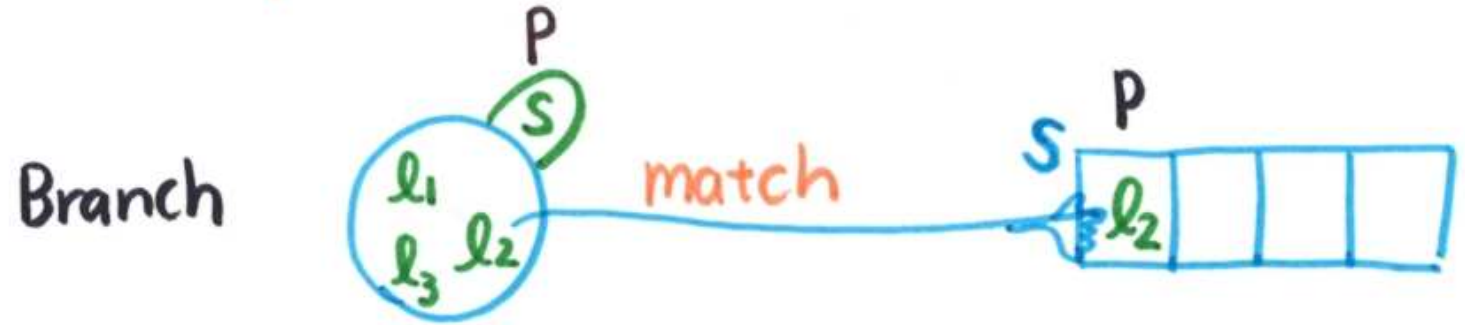
Value Error

✗ $s?[p](q, y); P \mid s?[p](q, y'); P'$

Linearity Error

✗ $s![p]\langle q, V \rangle \mid s![p]\langle q, W \rangle; P'$

Linearity Error



Session Fidelity

$\Gamma \vdash P \triangleright \Delta$

$P \xrightarrow[A \to B]{} P'$

$\Delta \xrightarrow[A \to B]{} \Delta'$

$s!\langle U \rangle;T \mid s?\langle U' \rangle;T' \xrightarrow[A \to B]{} T \mid T'$

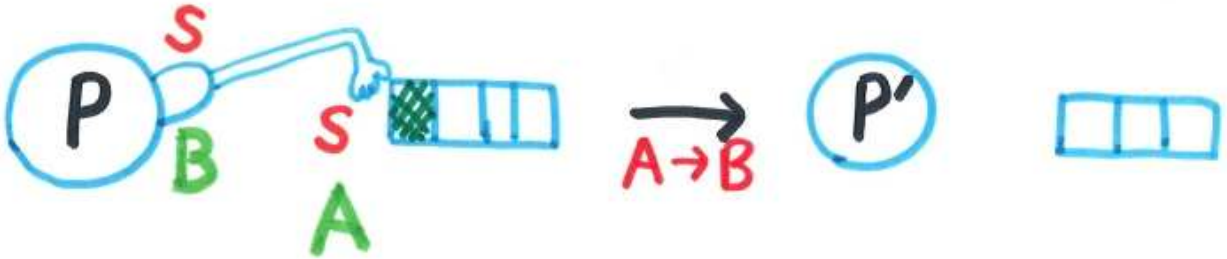
$G \xrightarrow[A \to B]{} G'$

$A \to B : \langle U \rangle.$

Progress

$\Gamma \vdash P \triangleright \Delta$ P is in a single session

$P \xrightarrow[A \to B]{} P'$



$\Delta \xrightarrow[A \to B]{} \Delta'$

$s\,!\langle U \rangle^{A}; T \mid s\,?\langle U' \rangle^{B}; T' \xrightarrow[A \to B]{} T \mid T'$

$G \xrightarrow[A \to B]{} G'$

$A \to B : \langle U \rangle.$

Subject Reduction Theorem

- Subject Congruence

  $\Gamma \vdash P \triangleright \Delta$ and $P \equiv Q \Longrightarrow \Gamma \vdash Q \triangleright \Delta$

- Subject Reduction

  $\Gamma \vdash P \triangleright \Delta$ and $P \rightarrow Q \Longrightarrow \Gamma \vdash Q \triangleright \Delta'$ with $\Delta \rightarrow \Delta'$

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

Single Multiparty Session Progress

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

  Suppose $P_1 \mid \cdots \mid P_n \rightarrow^+ Q$. 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 guranteed 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.