Multiparty session types + DbC

Hernán Melgratti

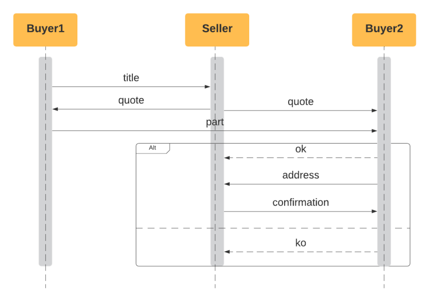
Multiparty session types [1]

- Extension of binary session types to multiparty sessions
- Asynchronous communications
- Interactions are abstracted as a global scenario, namely, Global types
  - specify dependencies and causal chains of multiparty asynchronous interactions

[1] Honda, K., Yoshida, N., & Carbone, M. Multiparty asynchronous session types. POPL 2008

Global Graph (Choreography)

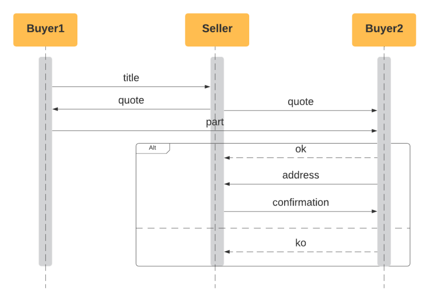
Two Buyers Protocol



$$
\mathsf{TBProt} = \begin{array}{l} \mathtt{b1}{\to}\mathtt{s} : \langle \mathtt{string} \rangle . \\ \mathtt{s}{\to}\mathtt{b1} : \langle \mathtt{float} \rangle . \\ \mathtt{s}{\to}\mathtt{b2} : \langle \mathtt{float} \rangle . \\ \mathtt{b1}{\to}\mathtt{b2} : \langle \mathtt{float} \rangle . \\ \mathtt{b2}{\to}\mathtt{s} : \left\{ \begin{array}{l} \mathit{ok} : \mathtt{b2}{\to}\mathtt{s} : \langle \mathtt{string} \rangle . \mathtt{s}{\to}\mathtt{b2} : \langle \mathtt{string} \rangle . \mathsf{end}, \\ \mathit{ko} : \mathsf{end} \end{array} \right\} \end{array}
$$

Local Types

Two Buyers Protocol



$$\texttt{Buyer1} \; = \; \texttt{!string.?float.!float.end}$$

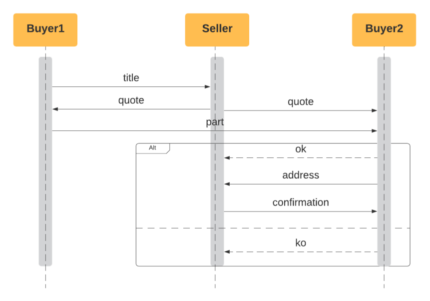
- The first message (`string`) is for the Seller and the second one (`float`) is for Buyer2. Information absent in the local type
- There are alternatives:
  - *a là* communicating machines: one channel for each pair in each direction

    $$\texttt{Buyer1} = b_1 s\texttt{!}\langle\texttt{string}\rangle\texttt{.}sb_1\texttt{?}\langle\texttt{float}\rangle\texttt{.}b_1 b_2\texttt{!}\langle\texttt{float}\rangle\texttt{.end}$$

  - Decorated global graphs

Local Types

Two Buyers Protocol



$$
\begin{aligned}
\texttt{TBProt} \;=\;& \mathtt{b1}{\to}\mathtt{s} : x\,\langle\mathtt{string}\rangle.\\
& \mathtt{s}{\to}\mathtt{b1} : y\,\langle\mathtt{float}\rangle.\\
& \mathtt{s}{\to}\mathtt{b2} : z_1\,\langle\mathtt{float}\rangle.\\
& \mathtt{b1}{\to}\mathtt{b2} : z_2\,\langle\mathtt{float}\rangle.\\
& \mathtt{b2}{\to}\mathtt{s} : x \left\{ \begin{array}{l} ok : \mathtt{b2}{\to}\mathtt{s} : x\,\langle\mathtt{string}\rangle.\mathtt{s}{\to}\mathtt{b2} : z_1\,\langle\mathtt{string}\rangle.\mathtt{end}, \\ ko : \mathtt{end} \end{array} \right\}
\end{aligned}
$$

$$
\texttt{Buyer1} \;=\; x!\langle\mathtt{string}\rangle.y?\langle\mathtt{float}\rangle.z_2!\langle\mathtt{float}\rangle.\mathtt{end}
$$

First-order, finite MST

Syntax

$\eta ::=$	$\mathsf{p} \to \mathsf{q} : x$	action
$\mathsf{G} ::=$	$\eta\langle\tilde{\mathsf{S}}\rangle.\mathsf{G}$	interaction
$\mid$	$\eta\{l_j : \mathsf{G}_j\}_{j \in J}$	branch
$\mid$	$\mathsf{G} \mid \mathsf{G}$	parallel
$\mid$	end	termination
$\mathsf{S} ::=$	$\mathtt{int} \mid \mathtt{unit} \mid \mathtt{bool} \mid \ldots$	basic sorts

- p, r, ... : participants (also roles)
- x, y, ...: communication channels
- l, ... : labels
- $\tilde{\_}$: tuples

Local types

Syntax

$\mathsf{T} ::=$		$x?\langle\tilde{\mathsf{S}}\rangle.\mathsf{T}$	receive
	$\mid$	$x!\langle\tilde{\mathsf{S}}\rangle.\mathsf{T}$	send
	$\mid$	$x \oplus \{l_i : \mathsf{T}_i\}_{i \in I}$	select
	$\mid$	$x\&\{l_i : \mathsf{T}_i\}_{i \in I}$	branch
	$\mid$	$\mathtt{end}$	termination
$\mathsf{S} ::=$		$\mathtt{int} \mid \mathtt{unit} \mid \mathtt{bool} \mid \ldots$	basic sorts

Projection

An operation (algorithm) for obtaining local types from global types

Definition

$$
\mathsf{G}\!\upharpoonright\!\mathsf{p} = \begin{cases}
x\,!\langle\tilde{\mathsf{S}}\rangle.\mathsf{G}'\!\upharpoonright\!\mathsf{p} & \text{if } \mathsf{G} = \mathsf{p}\to\mathsf{q} : x\,\langle\tilde{\mathsf{S}}\rangle.\mathsf{G}' \textit{ and } \mathsf{p} \neq \mathsf{q} \\
x\,?\langle\tilde{\mathsf{S}}\rangle.\mathsf{G}'\!\upharpoonright\!\mathsf{p} & \text{if } \mathsf{G} = \mathsf{q}\to\mathsf{p} : x\,\langle\tilde{\mathsf{S}}\rangle.\mathsf{G}' \textit{ and } p \neq \mathsf{q} \\
\mathsf{G}'\!\upharpoonright\!\mathsf{p} & \text{if } \mathsf{G} = \mathsf{q}\to\mathsf{r} : x\,\langle\tilde{\mathsf{S}}\rangle.\mathsf{G}' \textit{ and } \mathsf{p} \neq \mathsf{q} \neq \mathsf{r} \\
x \oplus \{l_i : \mathsf{G}_i\!\upharpoonright\!\mathsf{p}\}_{i\in I} & \text{if } \mathsf{G} = \mathsf{p}\to\mathsf{q} : x\{l_i : \mathsf{G}_i\}_{i\in I} \textit{ and } \mathsf{p} \neq \mathsf{q} \\
x\&\{l_i : \mathsf{G}_i\!\upharpoonright\!\mathsf{p}\}_{i\in I} & \text{if } \mathsf{G} = \mathsf{q}\to\mathsf{p} : x\{l_i : \mathsf{G}_i\}_{i\in I} \textit{ and } \mathsf{p} \neq \mathsf{q} \\
\mathsf{G}_1\!\upharpoonright\!\mathsf{p} & \text{if } \mathsf{G} = \mathsf{q}\to\mathsf{r} : x\{l_i : \mathsf{G}_i\}_{i\in I} \textit{ and } \mathsf{p} \neq \mathsf{q} \neq \mathsf{r} \textit{ and} \\
& \qquad\qquad\qquad \forall i,j.\mathsf{G}_i\!\upharpoonright\!\mathsf{p} = \mathsf{G}_j\!\upharpoonright\!\mathsf{p} \\
\mathsf{G}_i\!\upharpoonright\!\mathsf{p} & \text{if } \mathsf{G} = \mathsf{G}_1 \mid \mathsf{G}_2 \textit{ and } \mathsf{p} \in \mathsf{G}_i \textit{ and } \mathsf{p} \notin \mathsf{G}_j \textit{ and} \\
& \qquad\qquad\qquad i \neq j \in \{1,2\} \\
\mathsf{end} & \text{if } \mathsf{G} = \mathsf{G}_1 \mid \mathsf{G}_2 \textit{ and } \mathsf{p} \notin \mathsf{G}_1 \textit{ and } \mathsf{p} \notin \mathsf{G}_2 \\
\mathsf{end} & \text{if } \mathsf{G} = \mathsf{end}
\end{cases}
$$

Projection

Two Buyers Protocol

$$
\begin{aligned}
\mathtt{TBProt} \;=\;\; & \mathsf{b1}{\to}\mathsf{s} : x\,\langle \mathtt{string}\rangle.\\
& \mathsf{s}{\to}\mathsf{b1} : y\,\langle \mathtt{float}\rangle.\\
& \mathsf{s}{\to}\mathsf{b2} : z_1\,\langle \mathtt{float}\rangle.\\
& \mathsf{b1}{\to}\mathsf{b2} : z_2\,\langle \mathtt{float}\rangle.\\
& \mathsf{b2}{\to}\mathsf{s} : x \left\{ \begin{array}{l} ok : \mathsf{b2}{\to}\mathsf{s} : x\,\langle \mathtt{string}\rangle.\mathsf{s}{\to}\mathsf{b2} : z_1\,\langle \mathtt{string}\rangle.\mathsf{end}, \\ ko : \mathsf{end} \end{array} \right\}
\end{aligned}
$$

$$
\begin{aligned}
\mathtt{TBProt}\restriction\mathsf{b1} &= x\,!\langle \mathtt{string}\rangle.y\,?\langle \mathtt{float}\rangle.z_2\,!\langle \mathtt{float}\rangle.\mathtt{end}\\
\mathtt{TBProt}\restriction\mathsf{s} &= x\,?\langle \mathtt{string}\rangle y\,!\langle \mathtt{float}\rangle.z_1\,!\langle \mathtt{float}\rangle.\\
&\qquad x \& \{ok : x\,?\langle \mathtt{string}\rangle.z_1\,!\langle \mathtt{string}\rangle.\mathtt{end}, ko : \mathtt{end}\}\\
\mathtt{TBProt}\restriction\mathsf{b2} &= z_1\,?\langle \mathtt{float}\rangle.z_2\,?\langle \mathtt{float}\rangle.\\
&\qquad x \oplus \{ok : x\,!\langle \mathtt{string}\rangle.z_1\,?\langle \mathtt{string}\rangle.\mathtt{end}, ko : \mathtt{end}\}
\end{aligned}
$$

Realizations (Implementations)

Implementation: a set of processes (programs), whose semantics is analogous to the binary case.

Two Buyer Protocol

$$
\begin{aligned}
P_{\texttt{Buyer}_1} &= \overline{a}_{[2..3]}(b_1, b_2, b_2', s).P_1\\
P_1 &= s\,\texttt{!}\,\text{"My Book"}\,\texttt{.}\,b_1\texttt{?}(quote)\,\texttt{.}\,b_2\,\texttt{!}(quote\ /\ 2)\,\texttt{.}\,\mathbf{0}\\
P_{\texttt{Buyer}_2} &= a_{[2]}(b_1, b_2, b_2', s).P_2\\
P_2 &= b_2\texttt{?}(quote)\,\texttt{.}\,b_2'\texttt{?}(contrib)\,\texttt{.}\\
&\qquad\quad \texttt{if}\ (contrib > quote/2)\\
&\qquad\qquad\quad \texttt{then}\ s \triangleright ok\,\texttt{.}\,s\,\texttt{!}''via...''\,\texttt{.}\,b_2\texttt{?}(x)\,\texttt{.}\,\mathbf{0}\\
&\qquad\qquad\quad \texttt{else}\ s \triangleright ko\,\texttt{.}\,\mathbf{0}\\
P_{\texttt{Seller}} &= a_{[3]}(b_1, b_2, b_2', s).Q\\
Q &= s\texttt{?}(title)\,\texttt{.}\,b_1\,\texttt{!}\,100\,\texttt{.}\,b_2\,\texttt{!}\,100\,\texttt{.}\\
&\qquad\quad s \triangleleft \{ok : s\texttt{?}(x)\,\texttt{.}\,b_2\,\texttt{!}\ldots\texttt{.}\,\mathbf{0},\ ko : \mathbf{0}\}
\end{aligned}
$$

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{int} \rangle.\mathtt{p}{\to}\mathtt{r} : y\,\langle \mathtt{bool} \rangle.\mathtt{end} \\
P_p &= x!1.y!\mathtt{true}.\mathbf{0} \\
P_q &= x?(i).\mathbf{0} \\
P_r &= y?(j).\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{int} \rangle.\mathtt{p}{\to}\mathtt{r} : x\,\langle \mathtt{bool} \rangle.\mathtt{end} \\
P_p &= x\,!\,1.x\,!\,\mathtt{true}.\mathbf{0} \\
P_q &= x\,?(i).\mathbf{0} \\
P_r &= x\,?(j).\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned} \mathtt{G} &= \mathtt{p}{\rightarrow}\mathtt{q} : x\,\langle \mathtt{int}\rangle.\mathtt{p}{\rightarrow}\mathtt{r} : x\,\langle \mathtt{bool}\rangle.\mathtt{end} \\ P_p &= x!1.x!\mathtt{true}.\mathbf{0} \\ P_q &= x?(i).\mathbf{0} \\ P_r &= x?(j).\mathbf{0} \end{aligned}$$

We cannot ensure that P_q gets 1 and P_r gets `true` (race on x)
Hence, `G` is bad (Output-to-Output bad)

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{r}{\to}\mathtt{q} : y\,\langle\mathtt{bool}\rangle.\mathtt{end}\\
P_p &= x\,!\,1.\mathbf{0}\\
P_q &= x\,?(i).y\,?(j).\mathbf{0}\\
P_r &= y\,!\,\mathtt{true}.\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned} \mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x \langle \mathtt{int} \rangle . \mathtt{r}{\to}\mathtt{q} : y \langle \mathtt{bool} \rangle . \mathtt{end} \\ P_p &= x!1.\mathbf{0} \\ P_q &= x?(i).y?(j).\mathbf{0} \\ P_r &= y!\mathtt{true}.\mathbf{0} \end{aligned}$$

No races on channels x and y
Hence, $\mathtt{G}$ is good (Input-to-Input good)

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{r}{\to}\mathtt{q} : x\,\langle\mathtt{bool}\rangle.\mathtt{end} \\
P_p &= x\,\mathtt{!}1.\mathbf{0} \\
P_q &= x\,\mathtt{?}(i).x\,\mathtt{?}(j).\mathbf{0} \\
P_r &= x\,\mathtt{!}\mathtt{true}.\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{int} \rangle.\mathtt{r}{\to}\mathtt{q} : x\,\langle \mathtt{bool} \rangle.\mathtt{end} \\
P_p &= x\mathtt{!}1.\mathbf{0} \\
P_q &= x\mathtt{?}(i).x\mathtt{?}(j).\mathbf{0} \\
P_r &= x\mathtt{!}\mathtt{true}.\mathbf{0}
\end{aligned}
$$

Race on x
Hence, $\mathtt{G}$ is bad (Input-to-Input bad)

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{int} \rangle\mathtt{.}\mathtt{q}{\to}\mathtt{r} : x\,\langle \mathtt{int} \rangle\mathtt{.end} \\
P_p &= x\mathtt{!}1\mathtt{.}\mathbf{0} \\
P_q &= x\mathtt{?}(i)\mathtt{.}x\mathtt{!}i\mathtt{.}\mathbf{0} \\
P_r &= x\mathtt{?}(i)\mathtt{.}\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned} \mathtt{G} &= \mathtt{p}{\rightarrow}\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{q}{\rightarrow}\mathtt{r} : x\,\langle\mathtt{int}\rangle.\mathtt{end} \\ P_p &= x\,\mathtt{!}\,1\,\mathtt{.}\,\mathbf{0} \\ P_q &= x\,\mathtt{?}(i)\mathtt{.}\,x\,\mathtt{!}\,i\,\mathtt{.}\,\mathbf{0} \\ P_r &= x\,\mathtt{?}(i)\mathtt{.}\,\mathbf{0} \end{aligned}$$

Race on x

Hence, G is bad (Input-to-Output bad)

Realizations

Example

$$
\begin{aligned}
\mathtt{G} \;&= \mathtt{p}\!\rightarrow\!\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{q}\!\rightarrow\!\mathtt{r} : y\,\langle\mathtt{int}\rangle.\mathtt{end} \\
P_p &= x\,\mathtt{!}\,1\,.\,\mathbf{0} \\
P_q &= x\,\mathtt{?}(i)\,.\,y\,\mathtt{!}\,i\,.\,\mathbf{0} \\
P_r &= y\,\mathtt{?}(i)\,.\,\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned} \mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{q}{\to}\mathtt{r} : y\,\langle\mathtt{int}\rangle.\mathtt{end} \\ P_p &= x!1.\mathbf{0} \\ P_q &= x?(i).y!i.\mathbf{0} \\ P_r &= y?(i).\mathbf{0} \end{aligned}$$

No Races on x and y
Hence, G is good (Input-to-Output good)

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{p}{\to}\mathtt{q} : x\,\langle\mathtt{bool}\rangle.\mathtt{end}\\
P_p &= x\,\mathtt{!}\,\mathtt{1}.x\,\mathtt{!}\,\mathtt{true}.\mathbf{0}\\
P_q &= x\,\mathtt{?}(i).x\,\mathtt{?}(j).\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned} \mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{int}\rangle \mathtt{.}\, \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{bool}\rangle \mathtt{.end} \\ P_p &= x\,\mathtt{!}\,1\,\mathtt{.}\,x\,\mathtt{!}\,\mathtt{true}\,\mathtt{.}\mathbf{0} \\ P_q &= x\,\mathtt{?}(i)\,\mathtt{.}\,y\,\mathtt{?}(j)\,\mathtt{.}\mathbf{0} \end{aligned}$$

No Races on x

Hence, $\mathtt{G}$ is good (Input-to-Input, Output-to-Output good)

Realizations

Example

$$
\begin{aligned}
\mathtt{G} \;\; &= \mathtt{p}{\rightarrow}\mathtt{q} : x\,\langle \mathtt{int} \rangle.\mathtt{s}{\rightarrow}\mathtt{r} : y\,\langle \mathtt{bool} \rangle.\mathtt{p}{\rightarrow}\mathtt{r} : x\,\langle \mathtt{bool} \rangle.\mathtt{end} \\
P_p &= x!1.x!\mathtt{true}.\mathbf{0} \\
P_q &= x?(i).\mathbf{0} \\
P_r &= y?(i).x?(j).\mathbf{0} \\
P_s &= y!\mathtt{true}.\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{s}{\to}\mathtt{r} : y\,\langle\mathtt{bool}\rangle.\mathtt{p}{\to}\mathtt{r} : x\,\langle\mathtt{bool}\rangle.\mathtt{end}\\
P_p &= x!1.x!\mathtt{true}.\mathbf{0}\\
P_q &= x?(i).\mathbf{0}\\
P_r &= y?(i).x?(j).\mathbf{0}\\
P_s &= y!\mathtt{true}.\mathbf{0}
\end{aligned}$$

Races on x

Hence, G is bad

Realizations

Example

$$
\begin{aligned}
\mathtt{G} &= \mathtt{p}{\to}\mathtt{q} : x\,\langle \mathtt{int}\rangle.\mathtt{q}{\to}\mathtt{r} : y\,\langle \mathtt{bool}\rangle.\mathtt{p}{\to}\mathtt{r} : x\,\langle \mathtt{bool}\rangle.\mathtt{end} \\
P_p &= x!1.x!\mathtt{true}.\mathbf{0} \\
P_q &= x?(i).y!\mathtt{true}.\mathbf{0} \\
P_r &= y?(i).x?(j).\mathbf{0}
\end{aligned}
$$

Realizations

Example

$$\begin{aligned}
\mathtt{G} &= \mathtt{p}\to\mathtt{q} : x\,\langle\mathtt{int}\rangle.\mathtt{q}\to\mathtt{r} : y\,\langle\mathtt{bool}\rangle.\mathtt{p}\to\mathtt{r} : x\,\langle\mathtt{bool}\rangle.\mathtt{end}\\
P_p &= x\,!\,1.x\,!\,\mathtt{true}.\mathbf{0}\\
P_q &= x\,?(i).y\,!\,\mathtt{true}.\mathbf{0}\\
P_r &= y\,?(i).x\,?(j).\mathbf{0}
\end{aligned}$$

No races on x and y
Hence, $\mathtt{G}$ is good

This notion is formalised as LINEARITY

Coherence (a.k.a well-formedness)

Coherence

$\mathtt{G}$ is coherent if it is linear and $\mathtt{G}\restriction \mathtt{p}$ is well-defined for each $\mathtt{p}$

Typing

Context for linear resources

$$\begin{array}{rcll} \Delta & ::= & \emptyset & \textit{empty context} \\ & | & \tilde{s} : \mathsf{T} \,@\, p & \textit{session channels are of local session types} \end{array}$$

Typing of Processes $\Gamma \;\vdash\; P \;\triangleright\; \Delta$

Properties

Subject reduction (approx)

- $\Gamma \vdash P \triangleright \Delta$ such that Δ is coherent[a] and $P \xrightarrow{\alpha} P'$ imply $\Gamma \vdash P' \triangleright \Delta'$ where $\Delta = \Delta'$ or $\Delta \to \Delta'$

[a] linear, and all projections well-defined

Properties

Session fidelity (approx)

- $\Gamma \vdash P \triangleright \Delta$ such that Δ is coherent and $\Delta(\tilde{s}) = \mathtt{G}\!\upharpoonright\!\mathsf{p}_1 \,@\, p_n, \ldots, \mathtt{G}\!\upharpoonright\!\mathsf{p}_n \,@\, p_n$. If $P \xrightarrow{s_k} P'$ then $\mathtt{G} \rightarrow \mathtt{G}'$ and $\Gamma \vdash P' \triangleright \Delta'$ and $\Delta'(\tilde{s}) = \mathtt{G}'\!\upharpoonright\!\mathsf{p}_1 \,@\, p_1, \ldots, \mathtt{G}'\!\upharpoonright\!\mathsf{p}_n \,@\, p_n$.

Properties

Progress (approx)

$\Gamma \vdash P \triangleright \Delta$ such that Δ is coherent, P simple, well-linked and queue-full. Then,

- If $P \not\equiv \mathbf{0}$ then $P \xrightarrow{\alpha} P'$ for some P',
- If $\Delta(\tilde{s}) = \mathtt{G}\!\restriction \mathtt{p}_1 \,@\, p_n, \ldots, \mathtt{G}\!\restriction \mathtt{p}_n \,@\, p_n$, and $\mathtt{G} \xrightarrow{\ell} \mathtt{G}'$ then $P \xrightarrow{s} P'$ and $ch(\ell) = s$.

DbC + Multiparty session types

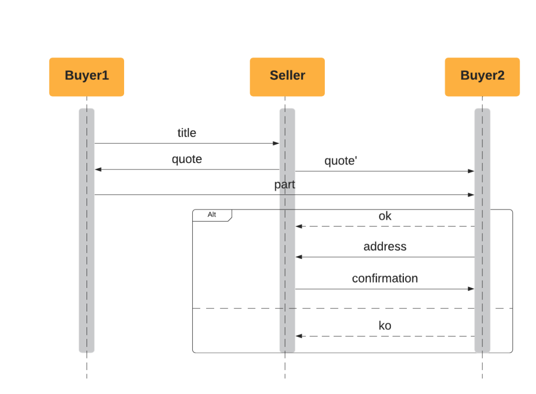
DbC + Multiparty session types[1]

- Extension of multiparty session types with assertions about communicated values

[1] Laura Bocchi, Kohei Honda, Emilio Tuosto, Nobuko Yoshida: A Theory of Design-by-Contract for Distributed Multiparty Interactions. CONCUR 2010

Global Graph (Choreography)

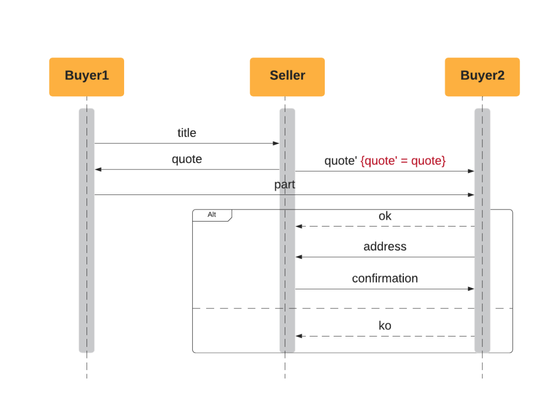
Two Buyers Protocol



$$\begin{aligned}\texttt{TBProt} = {} & \texttt{b1}\to\texttt{s} : x\,\langle\texttt{string}\rangle.\\ & \texttt{s}\to\texttt{b1} : y\,\langle\texttt{float}\rangle.\\ & \texttt{s}\to\texttt{b2} : z_1\,\langle\texttt{float}\rangle.\\ & \texttt{b1}\to\texttt{b2} : z_2\,\langle\texttt{float}\rangle.\ldots\end{aligned}$$

Global Graph (Choreography) + Assertions

Two Buyers Protocol



$$
\begin{aligned}
\mathtt{TBProt} \;=\; & \mathtt{b1}{\rightarrow}\mathtt{s} : x\,\langle \mathtt{string}\rangle. \\
& \mathtt{s}{\rightarrow}\mathtt{b1} : y\,\langle \mathit{quote} : \mathtt{float}\rangle. \\
& \mathtt{s}{\rightarrow}\mathtt{b2} : z_1\,\langle \mathit{quote}' : \mathtt{float}\rangle\{\mathit{quote} = \mathit{quote}'\}. \\
& \mathtt{b1}{\rightarrow}\mathtt{b2} : z_2\,\langle \mathtt{float}\rangle.\ldots
\end{aligned}
$$

Interactions with assertions

Assertions

$$\mathtt{s}\to\mathtt{b2} : z_1 \, \langle quote' : \mathtt{float}\rangle\{quote = quote'\}.\mathtt{G}$$

- $quote'$ is a logical variable
  - is called an *interaction variable*
  - denotes a potential message value
  - bind its occurrences in $\{quote = quote'\}$ and $\mathtt{G}$
- $\mathtt{s}$ guarantees the interaction predicate $\{quote = quote'\}$ (which constrains $quote'$)
- $\mathtt{b2}$ relies on $\{quote = quote'\}$
- $\{quote = quote'\}$
  - is a *precondition* for $\mathtt{s}$ (since it is what $\mathtt{s}$ should ensure it)
  - is a *post-condition* for $\mathtt{b2}$ (since it is guaranteed to the receiver).

(Finite) Global Assertions = (Finite) MST + Assertions

Syntax of Global Assertions

η ::=		$\mathsf{p}\to\mathsf{q} : x$	action
G ::=		$\eta\langle\tilde{v} : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{G}$	interaction
	$\mid$	$\eta\{\{A_j\}l_j : \mathsf{G}_j\}_{j\in J}$	branch
	$\mid$	$\mathsf{G} \mid \mathsf{G}$	parallel
	$\mid$	end	termination
S ::=		int $\mid$ unit $\mid$ bool $\mid$ ...	basic sorts

- p, r, ... : participants (also roles)
- x, y, ...: communication channels
- v, w, ...: logical variables
- l, ... : labels
- $\tilde{\_}$: tuples
- A: Assertion on values

Assertions in branches

Assertions

$$\mathsf{p}{\rightarrow}\mathsf{q} : x\{\{A_j\}l_j : \mathtt{G}_j\}_{j \in J}$$

- participant p sends one of the labels l_j to channel x
- Each branch j takes place place on the condition $\{A_j\}$
  - this is guaranteed by the sender
  - and relied upon by the receiver

Coherence (a.k.a well-formedness)

Erase : Global Assertion $\rightarrow$ Global Type

- Remove interaction variables and predicates from global assertions

$$
\begin{aligned}
\mathtt{erase}(\mathtt{p}\to\mathtt{q} : x\langle \tilde{v} : \tilde{\mathtt{S}}\rangle\{A\}.\mathtt{G}) &= \mathtt{p}\to\mathtt{q} : x\langle \tilde{\mathtt{S}}\rangle.\mathtt{erase}(\mathtt{G}) \\
\mathtt{erase}(\mathtt{p}\to\mathtt{q} : x\{\{A_j\}l_j : \mathtt{G}_j\}_{j\in J}) &= \mathtt{p}\to\mathtt{q} : x\{l_j : \mathtt{erase}(\mathtt{G}_j)\}_{j\in J} \\
\ldots
\end{aligned}
$$

- A global assertion is *coherent* if $\mathtt{erase}(\mathtt{G})$ is *coherent* (linear and projectable)

Logical language

Syntax

$$A ::= e_1 = e_2 \mid e_1 > e_2 \mid \phi(e_1, \ldots, e_n) \mid A_1 \wedge A_2 \mid \neg A \mid \exists v(A)$$

- $e_1, e_2, \ldots$ expressions
- ϕ predicates with fixed arity

Convention

- validity of each closed atomic formula including equality and inequality is polynomially decidable.
- validity of closed formulae is decidable
- e.g., Presburger arithmetic.

Consistency

Example

$$\mathtt{p}\to\mathtt{q} : x(v : \mathtt{int})\{v > 10\}.\mathtt{r}\to\mathtt{q} : x(w : \mathtt{int})\{w > v\}.\mathtt{end}$$

Example

$$\mathtt{p}\to\mathtt{q} : x(v : \mathtt{int})\{v > 10\}.\mathtt{r}\to\mathtt{q} : x(w : \mathtt{int})\{w > v\}.\mathtt{end}$$

- $\mathtt{r}$ does not know the value v sent by $\mathtt{p}$ on the first interaction
- $\mathtt{r}$ is unable to guarantee the assertion $\{w > v\}$

history-sensitivity

An interaction predicate guaranteed by a participant is defined only on interaction variables introduced in the preceding interactions in which the participant is involved

Consistency

Example

$$\mathsf{p}{\to}\mathsf{q} : x(v : \mathtt{int})\{v > 10\}.\mathsf{q}{\to}\mathsf{r} : x(w : \mathtt{int})\{v < 5\}.\mathsf{end}$$

Example

$$\mathsf{p}{\to}\mathsf{q} : x(v : \mathtt{int})\{v > 10\}.\mathsf{q}{\to}\mathsf{p} : x(w : \mathtt{int})\{v < 5\}.\mathsf{end}$$

- there is no way for q to ensure that the value v sent by p satisfies $\{v < 5\}$
- moreover, there is no way to satisfy both assertions $\{v > 10\}$ and $\{v < 5\}$

Locality

An interaction formula should only add constraints to the variables it introduces

Consistency

Example

$$\mathtt{p}\to\mathtt{q} : x(v : \mathtt{int})\{v > 10\}.\mathtt{q}\to\mathtt{p} : x(w : \mathtt{int})\{v = w \wedge w < 12\}.\mathtt{end}$$

Example

$$\mathtt{p}\to\mathtt{q} : x(v : \mathtt{int})\{v > 10\}.\mathtt{q}\to\mathtt{p} : x(w : \mathtt{int})\{v = w \wedge w < 12\}.\mathtt{end}$$

- if p sends some value v greater than 11, then q is unable to guarantee the assertion $\{v = w \wedge w < 12\}$

Temporal satisfiability

For each possible value that satisfies a predicate A, it is possible, for each interaction predicate A' that appear after A, to find values satisfying A'.

Checking consistency

- History-sensitivity: via a typing system that keeps track of the variables that can appear in an interaction.
- Locality and Temporal satisfiability: via evaluating a boolean formula $\mathscr{G}_{sat}(\mathtt{G}, \mathtt{true}) = \mathtt{true}$
- $\mathscr{G}_{sat}(\mathtt{G}, A)$ is defined recursively on $\mathtt{G}$

$$\mathscr{G}_{sat}(\mathsf{p}\to\mathsf{q} : x\langle \tilde{v} : \tilde{\mathtt{S}}\rangle\{A'\}.\mathtt{G}', A) = \begin{array}{l}\textbf{if } A \implies \exists \tilde{v}(A') \textbf{ then } \mathscr{G}_{sat}(\mathtt{G}', A \wedge A') \\ \qquad\qquad\qquad\qquad\quad \textbf{else } \mathtt{false}\end{array}$$

$$\dots$$

$$\mathscr{G}_{sat}(\mathsf{end}, A) = \mathtt{true}$$

Local types + Assertions

Syntax

$$
\begin{array}{rcll}
\mathsf{T} ::= & & x?\langle \tilde{v} : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T} & \text{receive} \\
& \mid & x!\langle \tilde{v} : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T} & \text{send} \\
& \mid & x \oplus \{\{A_i\}l_i : \mathsf{T}_i\}_{i\in I} & \text{select} \\
& \mid & x\&\{\{A_i\}l_i : \mathsf{T}_i\}_{i\in I} & \text{branch} \\
& \mid & \mathbf{end} & \text{termination} \\
\mathsf{S} ::= & & \mathtt{int} \mid \mathtt{unit} \mid \mathtt{bool} \mid \ldots & \text{basic sorts}
\end{array}
$$

- $x!\langle \tilde{v} : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T}$ the sender should *guarantee* that the sent values $\tilde{v}$ satisfy A
- $x?\langle \tilde{v} : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T}$ the receiver can *rely* on the arriving values $\tilde{v}$ satisfy A

Projection + Causal dependency on assertions

Example

$$\begin{aligned}\mathtt{G} = \quad & \mathit{User} \to \mathit{Agent} : x(c : \mathit{Command})\{c \neq \mathtt{switch-off}\}. \\ & \mathit{Agent} \to \mathit{Device} : y(c' : \mathit{Command})\{c = c'\}. \ldots\end{aligned}$$

$$\begin{aligned}\mathtt{G}\restriction \mathit{User} &= x!\langle c : \mathit{Command}\rangle\{c \neq \mathtt{switch-off}\}. \ldots \\ \mathtt{G}\restriction \mathit{Agent} &= x?\langle c : \mathit{Command}\rangle\{c \neq \mathtt{switch-off}\}. \ldots \\ \mathtt{G}\restriction \mathit{Device} &= y?\langle c' : \mathit{Command}\rangle\{\exists c(c \neq \mathtt{switch-off} \wedge c = c'\}. \ldots\end{aligned}$$

Projection $\mathit{Proj}(\mathtt{G}, A, \mathsf{p})$

$$\mathit{Proj}(\mathsf{p} \to \mathsf{q} : x(\tilde{v} : \tilde{S})\{A'\}.\mathtt{G}, A, \mathsf{r}) = \begin{cases} x!\langle \tilde{v} : \tilde{\mathtt{S}}\rangle\{A'\}.\mathtt{T} & \mathit{if}\ \mathsf{r} = \mathsf{p} \\ x?\langle \tilde{v} : \tilde{\mathtt{S}}\rangle\{B\}.\mathtt{T} & \mathit{if}\ \mathsf{r} = \mathsf{q} \\ \mathtt{T} & \mathit{otw}. \end{cases}$$

where
$\mathtt{T} = \mathit{Proj}(\mathtt{G}, A \wedge A', \mathsf{r})$
$B = \exists V_q(A \wedge A')$ and V_q are the variables in A not known to q

Consistency of global types, ensures consistency on local types

Processes

Syntax

$$
\begin{array}{rcll}
P & ::= & s\,!\,\tilde{e}(\tilde{v})\{A\}.P & \text{send} \\
 & \mid & s?(\tilde{v})\{A\}.P & \text{receive} \\
 & \mid & s \triangleright l\{A\}.P & \text{selection} \\
 & \mid & s \triangleleft \{\{A_i\}l_i : P_i\}_{i \in I} & \text{branch} \\
 & \mid & \mathbf{0} & \text{ended} \\
 & \mid & P \mid P & \text{parallel} \\
 & \mid & \mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ P & \text{conditional} \\
 & \mid & a_{[i]}(\tilde{s}).P & \text{session acceptance} \\
 & \mid & \mathtt{errH} \mid \mathtt{errT} & \text{Contract violations} \\
 & \mid & \overline{a}_{[2..n]}(\tilde{s}).P & \text{session request} \\
 & \mid & (\nu w)P & \text{hiding} \\
 & \mid & s :: \tilde{h} & \text{message queue} \\
e & ::= & v \mid e\ or\ e \mid \ldots & \text{expressions} \\
v & ::= & \mathtt{true} \mid \mathtt{false} & \text{values} \\
w & ::= & a \mid \tilde{s} & \\
h & ::= & \tilde{v} \mid l & \text{message in transit}
\end{array}
$$

Processes

Semantics (few rules)

$$\frac{\tilde{e} \downarrow \tilde{n} \qquad \{A\}\{\tilde{n}/\tilde{v}\} \downarrow \mathtt{true}}{s!\tilde{e}(\tilde{v})\{A\}.P \mid s :: h \xrightarrow{s} P \mid s :: h \cdot \tilde{n}}\ \text{SSend}$$

$$\frac{\{A\}\{\tilde{n}/\tilde{v}\} \downarrow \mathtt{true}}{s?(\tilde{v})\{A\}.P \mid s :: \tilde{n} \cdot h \xrightarrow{s} P\{\tilde{n}/\tilde{x}\} \mid s :: h}\ \text{SRec}$$

$$\frac{\tilde{e} \downarrow \tilde{n} \qquad \{A\}\{\tilde{n}/\tilde{v}\} \downarrow \mathtt{false}}{s!\tilde{e}(\tilde{v})\{A\}.P \xrightarrow{s} ErrH}\ \text{SSendErr}$$

$$\frac{\{A\}\{\tilde{n}/\tilde{v}\} \downarrow \mathtt{true}}{s?(\tilde{v})\{A\}.P \mid s :: \tilde{n} \cdot h \xrightarrow{s} ErrT \mid s :: h}\ \text{SRecErr}$$

Typing (few rules)

Processes $\kappa; \Gamma \vdash P \triangleright \Delta$ where κ is a constraint

$$\frac{\kappa \wedge A; \Gamma, \tilde{v} : \tilde{S} \vdash P \triangleright \Delta, \tilde{s} : T \,@\, p}{\kappa; \Gamma \vdash s_k?(v)\{A\}.P \triangleright \Delta, \tilde{s} : s_k?\langle \tilde{v} : \tilde{S} \rangle \{A\}.T \,@\, p} \text{ Rec}$$

$$\frac{\kappa \models A\{\tilde{e}/\tilde{v}\} \qquad \Gamma \vdash \tilde{e} \triangleright \tilde{S} \qquad \kappa; \Gamma \vdash P\{\tilde{e}/\tilde{v}\} \triangleright \Delta, \tilde{s} : T\{\tilde{e}/\tilde{v}\} \,@\, p}{\kappa; \Gamma \vdash s_k!\tilde{e}(\tilde{v})\{A\}.P \triangleright \Delta, \tilde{s} : s_k!\langle \tilde{v} : \tilde{S} \rangle \{A\}.T \,@\, p} \text{ Send}$$

Property

Typing ensures that well-typed processes never violate assertions

Final words

- This is just the starting point in a very active research area!
- Several works about
  - expressiveness
    - less restrictions on communication patterns (context-free, flexible merge, relaxed well-formed conditions, global graphs)
    - relaxing linearity (allowing races), shared resources
    - alternative communication models (broadcast, publish/subscribe), event notification, weak consistent logs
    - types with parameterised parties
    - composition (open choreographies)
  - Interaction with other aspects of a language
    - Exceptions
    - Quantitative properties to reason about resource usages and complexity
    - Temporal properties
    - Probabilistic reasoning
    - Adaptability
    - Reversibility
  - Foundational aspects
    - relation with other well-known notions of programming languages (linearity, dependent types, effects)
    - Logical characterisation
    - Decomposition of Multiparty into Binary sessions
    - Synthesis (inference) of global types
    - Decidability aspects of typing/subtyping
    - Graduality
    - Monitoring

Final words

- Ensured properties
  - Type safety, Fidelity, Progress, Deadlock freedom, Lock-freedom.
  - Complete vs partial realizations
  - Security properties (e.g., information flow)
- Implementation in programming languages
  - `http://groups.inf.ed.ac.uk/abcd/session-implementations.html` (not up-to-date).
  - Typestates in Java and Join, Dependent types in Dotty (to name a few)
- New domains
  - Smart contracts