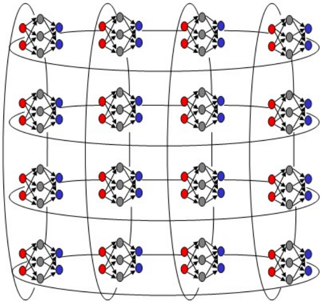
Parameterised Multiparty Session Types

Nobuko Yoshida Andi Bejleri Raymond Hu
Pierre-Malo Deniélou

Imperial College, London

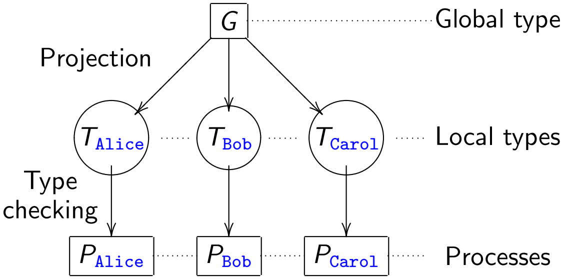


Multiparty Sessions for protocol specification

- Today's distributed applications involve more and more agents that interact through complex communication patterns.
- Multiparty sessions types can describe these interactions and statically ensure type and communication safety and fidelity to a stipulated protocol.

Multiparty Sessions for protocol specification

- Today's distributed applications involve more and more agents that interact through complex communication patterns.
- Multiparty sessions types can describe these interactions and statically ensure type and communication safety and fidelity to a stipulated protocol.



$$G = \texttt{Alice} \to \texttt{Bob} : \langle \mathsf{nat} \rangle . \texttt{Bob} \to \texttt{Carol} : \langle \mathsf{nat} \rangle . \mathsf{end}$$

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

$$y?\langle \texttt{Alice}, x \rangle ; \; y!\langle \texttt{Carol}, x + 1 \rangle ;$$

Beyond Multiparty Session Types

Many protocols are described as *parametric*

- the number of participants,
- the number of messages, or even
- part of the interaction structure

are not fixed at design time.

We need flexibility and modularity.

We want to be able to combine, associate, iterate protocol elements.

Families of global specifications

$$G \quad = \quad \texttt{Alice} \to \texttt{Bob} : \langle \text{nat} \rangle . \texttt{Bob} \to \texttt{Carol} : \langle \text{nat} \rangle . \text{end}$$

- Composition

$$G' = G; G = \quad \texttt{Alice} \to \texttt{Bob} : \langle \text{nat} \rangle . \texttt{Bob} \to \texttt{Carol} : \langle \text{nat} \rangle . \\ \texttt{Alice} \to \texttt{Bob} : \langle \text{nat} \rangle . \texttt{Bob} \to \texttt{Carol} : \langle \text{nat} \rangle . \text{end}$$

- k iterations of G

$$G; G; G; \ldots ; G \quad = \quad \texttt{foreach}(i < k)\{G\}$$

- Dependent product: family of global types

$$\Pi n . \texttt{foreach}(i < n)\{G\}$$

A protocol involving n parties?

We can parameterise *participant identities* ...

Parameterised participants

We want *indexed* participants (e.g. $\mathtt{W}[i]$ denotes the i-th worker).

- Ex: Sequence

$$\mathtt{W}[\mathrm{n}] \to \mathtt{W}[\mathrm{n}-1] : \langle \mathsf{nat} \rangle.$$
$$\ldots$$
$$\mathtt{W}[2] \to \mathtt{W}[1] : \langle \mathsf{nat} \rangle.$$
$$\mathtt{W}[1] \to \mathtt{W}[0] : \langle \mathsf{nat} \rangle.\mathsf{end}$$

Parameterised participants

We want *indexed* participants (e.g. $\mathtt{W}[i]$ denotes the i-th worker).

- Ex: Sequence

$$\mathtt{W}[\mathrm{n}] \to \mathtt{W}[\mathrm{n}-1] : \langle \mathsf{nat} \rangle .$$
$$\ldots$$
$$\mathtt{W}[2] \to \mathtt{W}[1] : \langle \mathsf{nat} \rangle .$$
$$\mathtt{W}[1] \to \mathtt{W}[0] : \langle \mathsf{nat} \rangle . \mathsf{end}$$

can be written:

$$\mathtt{foreach}(i < \mathrm{n})\{\mathtt{W}[i+1] \to \mathtt{W}[i] : \langle \mathsf{nat} \rangle\}$$

Contributions

1. A new expressive framework to globally specify and program a wide range of parametric communication protocols.
2. Decidable and flexible projection methods for extended typability.
3. A dependent typing system that allows decidable type-checking and guarantees type-safety and deadlock-freedom for well-typed processes.
4. Examples of parallel algorithms (e.g. FFT) and web services (e.g. WS-CDL use case).

Parameterised Global types

Our types combine

- Multiparty Session Types [POPL'08]

$$G \quad ::= \quad \mathtt{p} \to \mathtt{p}' : \langle U \rangle . G \quad | \quad \mathtt{p} \to \mathtt{p}' : \{ l_k : G_k \}_{k \in K} \quad | \quad \ldots$$

Parameterised Global types

Our types combine

- Multiparty Session Types [POPL'08]

$$G \quad ::= \quad \mathtt{p} \rightarrow \mathtt{p}' : \langle U \rangle . G \quad | \quad \mathtt{p} \rightarrow \mathtt{p}' : \{ l_k : G_k \}_{k \in K} \quad | \quad \ldots$$

- Dependent Types with Primitive Recursion [Nelson, MFPS'91]:
  The recursor **R** comes from Gödel's System $\mathcal{T}$.

$$\begin{array}{llll} \mathbf{R}\ G\ \lambda i.\lambda \mathbf{x}.G' & 0 & \longrightarrow & G \\ \mathbf{R}\ G\ \lambda i.\lambda \mathbf{x}.G' & (\mathrm{n}{+}1) & \longrightarrow & G'\{\mathrm{n}/i\}\{\mathbf{R}\ G\ \lambda i.\lambda \mathbf{x}.G'\ \mathrm{n}/\mathbf{x}\} \end{array}$$

Parameterised Global types

Our types combine

- Multiparty Session Types [POPL'08]

$$G \quad ::= \quad \mathtt{p} \rightarrow \mathtt{p}' : \langle U \rangle . G \quad | \quad \mathtt{p} \rightarrow \mathtt{p}' : \{l_k : G_k\}_{k \in K} \quad | \quad \ldots$$

- Dependent Types with Primitive Recursion [Nelson, MFPS'91]: The recursor **R** comes from Gödel's System $\mathcal{T}$.

$$\begin{array}{lll} \mathbf{R}\ G\ \lambda i.\lambda \mathbf{x}.G' \quad 0 & \longrightarrow & G \\ \mathbf{R}\ G\ \lambda i.\lambda \mathbf{x}.G' \quad (\mathrm{n}+1) & \longrightarrow & G'\{\mathrm{n}/i\}\{\mathbf{R}\ G\ \lambda i.\lambda \mathbf{x}.G'\ \mathrm{n}/\mathbf{x}\} \end{array}$$

Product, composition, loop and test operators

$$\begin{array}{lll} \mathtt{foreach}(i<j)\{G\} & = & \mathbf{R}\ \mathsf{end}\ \lambda i.\lambda \mathbf{x}.G\{\mathbf{x}/\mathsf{end}\}\ j \\ G_1; G_2 & = & \mathbf{R}\ G_2\ \lambda i.\lambda \mathbf{x}.G_1\{\mathbf{x}/\mathsf{end}\}\ 1 \\ \Pi i.G & = & \mathbf{R}\ \mathsf{end}\ \lambda i.\lambda \mathbf{x}.G\{i+1/i\} \\ \mathsf{if}\ j\ \mathsf{then}\ G_1\ \mathsf{else}\ G_2 & = & \mathbf{R}\ G_2\ \lambda i.\lambda \mathbf{x}.G_1\ j \end{array}$$

Processes, by semantics

Primitive Recursion

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ 0 \longrightarrow P$$

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}+1 \longrightarrow P\{\mathrm{n}/i\}\{\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}/X\}$$

Processes, by semantics

Primitive Recursion

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ 0 \longrightarrow P$$

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}+1 \longrightarrow P\{\mathrm{n}/i\}\{\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}/X\}$$

Initialisation

$$\bar{a}[\mathrm{p}_0, .., \mathrm{p}_\mathrm{n}](y).P \longrightarrow (\nu s)(s : \epsilon \mid \bar{a}[\mathrm{p}_0] : s \mid P \mid ... \mid \bar{a}[\mathrm{p}_\mathrm{n}] : s)$$

$$\bar{a}[\mathrm{p}_k] : s \mid a[\mathrm{p}_k](y_k).P_k \longrightarrow P_k\{s[\mathrm{p}_k]/y_k\}$$

Processes, by semantics

Primitive Recursion

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ 0 \longrightarrow P$$

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}+1 \longrightarrow P\{\mathrm{n}/i\}\{\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}/X\}$$

Initialisation

$$\bar{a}[\mathrm{p}_0,..,\mathrm{p}_\mathrm{n}](y).P \longrightarrow (\nu s)(s:\epsilon \mid \bar{a}[\mathrm{p}_0]:s \mid P \mid \ldots \mid \bar{a}[\mathrm{p}_\mathrm{n}]:s)$$

$$\bar{a}[\mathrm{p}_k]:s \mid a[\mathrm{p}_k](y_k).P_k \longrightarrow P_k\{s[\mathrm{p}_k]/y_k\}$$

Communication

$$s[\mathrm{p}]!\langle \mathrm{q}, v\rangle; P \mid s:h \longrightarrow P \mid s:h\cdot(\mathrm{p},\mathrm{q},v)$$

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

Processes, by semantics

Primitive Recursion

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ 0 \longrightarrow P$$

$$\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}+1 \longrightarrow P\{\mathrm{n}/i\}\{\mathbf{R}\ P\ \lambda i.\lambda X.Q\ \mathrm{n}/X\}$$

Initialisation

$$\bar{a}[\mathrm{p}_0, .., \mathrm{p}_\mathrm{n}](y).P \longrightarrow (\nu s)(s : \epsilon \mid \bar{a}[\mathrm{p}_0] : s \mid P \mid ... \mid \bar{a}[\mathrm{p}_\mathrm{n}] : s)$$

$$\bar{a}[\mathrm{p}_k] : s \mid a[\mathrm{p}_k](y_k).P_k \longrightarrow P_k\{s[\mathrm{p}_k]/y_k\}$$

Communication

$$s[\mathrm{p}]!\langle \mathrm{q}, v\rangle; P \mid s : h \longrightarrow P \mid s : h \cdot (\mathrm{p}, \mathrm{q}, v)$$

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

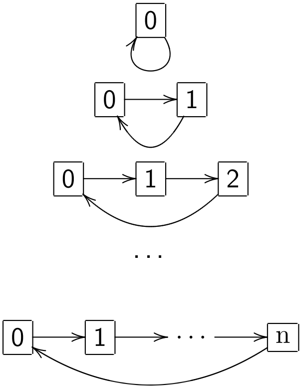
Branching and Selection

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

$$s[\mathrm{p}]\&(\mathrm{q}, \{l_k : P_k\}_{k\in K}) \ \mid s : (\mathrm{q}, \mathrm{p}, l_{k_0}) \cdot h \longrightarrow P_{i_0} \mid s : h \quad (i_0 \in I)$$

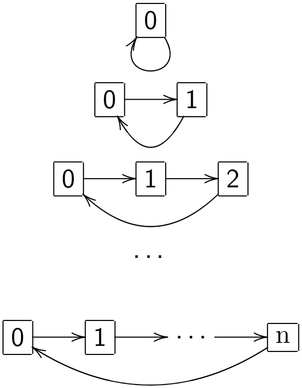
Ring family example

Ring pattern



Ring family example

Ring pattern



Parameterised Session Type

$$\Pi n : I.$$
$$(\mathbf{R}\ \mathtt{W}[n] \to \mathtt{W}[0] : \langle U \rangle.\mathsf{end}$$
$$\lambda i.\lambda \mathbf{x}.\mathtt{W}[n-i-1] \to \mathtt{W}[n-i] : \langle U \rangle.\mathbf{x}$$
$$n)$$

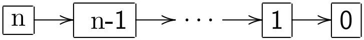
With syntactic sugar

$$\Pi n : I.$$
$$\mathtt{foreach}(i < n)\{\mathtt{W}[n-i-1] \to \mathtt{W}[n-i] : \langle U \rangle\};$$
$$\mathtt{W}[n] \to \mathtt{W}[0] : \langle U \rangle.\mathsf{end}$$

Using asynchrony

- Remember the sequence:

$$\texttt{foreach}(i < \mathrm{n})\{\texttt{W}[i+1] \to \texttt{W}[i] : \langle \mathsf{nat} \rangle\}$$



$\texttt{W}[n] \to \texttt{W}[n-1] : \langle U \rangle.$

$\ldots$

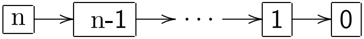
$\texttt{W}[2] \to \texttt{W}[1] : \langle U \rangle.$

$\texttt{W}[1] \to \texttt{W}[0] : \langle U \rangle.\mathsf{end}$

Using asynchrony

- Remember the sequence:

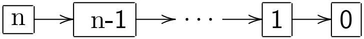
$$\texttt{foreach}(i < \mathrm{n})\{\texttt{W}[i+1] \to \texttt{W}[i] : \langle \mathsf{nat} \rangle\}$$



$\texttt{W}[n] \to \texttt{W}[n-1] : \langle U \rangle.$
$\ldots$
$\texttt{W}[2] \to \texttt{W}[1] : \langle U \rangle.$
$\texttt{W}[1] \to \texttt{W}[0] : \langle U \rangle.\mathsf{end}$

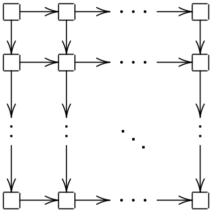
- We change the order: parallel steps within a ring.

$$\texttt{foreach}(i < \mathrm{n})\{\texttt{W}[\mathrm{n} - i] \to \texttt{W}[\mathrm{n} - i - 1] : \langle \mathsf{nat} \rangle\}$$



$\texttt{W}[1] \to \texttt{W}[0] : \langle U \rangle.$
$\texttt{W}[2] \to \texttt{W}[1] : \langle U \rangle.$
$\ldots$
$\texttt{W}[n] \to \texttt{W}[n-1] : \langle U \rangle.\mathsf{end}$

Mesh example



$\Pi n.\Pi m.$

$\texttt{foreach}(i < n)\{$

$\quad \texttt{foreach}(j < m)\{$

$\quad\quad \texttt{W}[i+1][j+1] \to \texttt{W}[i][j+1] : \langle \mathsf{nat} \rangle.$

$\quad\quad \texttt{W}[i+1][j+1] \to \texttt{W}[i+1][j] : \langle \mathsf{nat} \rangle\};$

$\quad \texttt{W}[i+1][0] \to \texttt{W}[i][0] : \langle \mathsf{nat} \rangle\};$

$\texttt{foreach}(k < m)\{\texttt{W}[0][k+1] \to \texttt{W}[0][k] : \langle \mathsf{nat} \rangle\}$

Fast Fourier Transform

Fast Fourier Transform

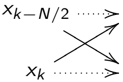
(a) Butterfly pattern

$x_{k-N/2}$ $\quad X_{k-N/2} = x_{k-N/2} + x_k * \omega_N^{k-N/2}$

x_k $\quad X_k = x_{k-N/2} + x_k * \omega_N^k$

Fast Fourier Transform

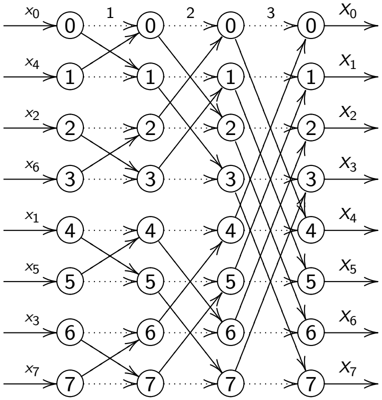
(a) Butterfly pattern



$$X_{k-N/2} = x_{k-N/2} + x_k * \omega_N^{k-N/2}$$

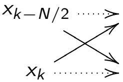
$$X_k = x_{k-N/2} + x_k * \omega_N^{k}$$

(b) FFT diagram



Fast Fourier Transform

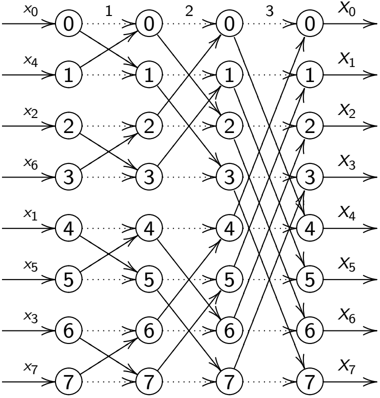
(a) Butterfly pattern



$$X_{k-N/2} = x_{k-N/2} + x_k * \omega_N^{k-N/2}$$

$$X_k = x_{k-N/2} + x_k * \omega_N^k$$

(b) FFT diagram

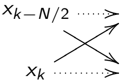


(c) Global type $G =$

$\Pi n.$
$\texttt{foreach}(i < 2^n)\{i \to i : \langle \mathsf{nat} \rangle\};$
$\texttt{foreach}(l < n)\{$
$\quad \texttt{foreach}(i < 2^l)\{$
$\quad\quad \texttt{foreach}(j < 2^{n-l-1})\{$
$\quad\quad\quad \texttt{foreach}(k < 2)\{$
$\quad\quad\quad\quad \texttt{foreach}(k' < 2)\{$
$\quad\quad\quad\quad\quad i * 2^{n-l} + k * 2^{n-l-1} + j$
$\quad\quad\quad\quad\quad \to i * 2^{n-l} + k' * 2^{n-l-1} + j : \langle \mathsf{nat} \rangle\}\}\}\}\}$

Fast Fourier Transform

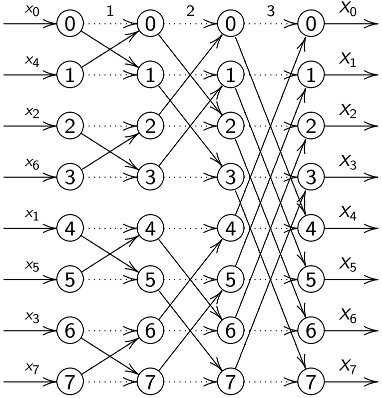
(a) Butterfly pattern



$X_{k-N/2} = x_{k-N/2} + x_k * \omega_N^{k-N/2}$

$X_k = x_{k-N/2} + x_k * \omega_N^k$

(b) FFT diagram



(c) Global type $G =$

$\Pi n.$
foreach$(i < 2^n)\{i \to i : \langle \mathsf{nat} \rangle\};$
foreach$(l < n)\{$
  foreach$(i < 2^l)\{$
    foreach$(j < 2^{n-l-1})\{$
      foreach$(k < 2)\{$
        foreach$(k' < 2)\{$
          $i * 2^{n-l} + k * 2^{n-l-1} + j$
          $\to i * 2^{n-l} + k' * 2^{n-l-1} + j : \langle \mathsf{nat} \rangle\}\}\}\}\}$

(d) Processes $P(n, \mathrm{p}, x_{\bar{\mathrm{p}}}, y, r_{\mathrm{p}}) =$

$y!\langle \mathrm{p}, x_{\bar{\mathrm{p}}} \rangle;$
foreach$(l < n)\{$
  if $\mathrm{bit}_{n-l}(\mathrm{p}) = 0$
  then $y?\langle \mathrm{p}, x \rangle; y!\langle \mathrm{p} + 2^{n-l-1}, x \rangle;$
    $y?\langle \mathrm{p} + 2^{n-l-1}, z \rangle; y!\langle \mathrm{p}, x + z\, \omega_N^{g(l,\mathrm{p})} \rangle;$
  else $y?\langle \mathrm{p}, x \rangle; y!\langle \mathrm{p} - 2^{n-l-1}, x \rangle;$
    $y?\langle \mathrm{p} - 2^{n-l-1}, z \rangle; y!\langle \mathrm{p}, z + x\, \omega_N^{g(l,\mathrm{p})} \rangle; \};$
$y?\langle \mathrm{p}, x \rangle; r_{\mathrm{p}}!\langle 0, x \rangle;$

Generic Projection

Generic Projection

$$
\mathrm{p} \to \mathrm{p}' : \langle U \rangle . G \upharpoonright \mathrm{q} \;=\; \begin{array}{l} \text{if q=p then } !\langle \mathrm{p}', U \rangle ; G \upharpoonright \mathrm{q} \\ \text{else if q=p' then } ?\langle \mathrm{p}, U \rangle ; G \upharpoonright \mathrm{q} \\ \text{else } G \upharpoonright \mathrm{q} \end{array}
$$

Generic Projection

$$
\begin{array}{rcl}
\mathrm{p} \to \mathrm{p}' : \langle U \rangle . G \restriction \mathrm{q} & = & \text{if q=p then } !\langle \mathrm{p}', U \rangle; G \restriction \mathrm{q} \\
 & & \text{else if q=p' then } ?\langle \mathrm{p}, U \rangle; G \restriction \mathrm{q} \\
 & & \text{else } G \restriction \mathrm{q} \\
\mathrm{p} \to \mathrm{p}' : \{ l_k : G_k \}_{k \in K} \restriction \mathrm{q} & = & \text{if q=p' then } \& \langle \mathrm{p}, \{ l_k : G_k \restriction \mathrm{q} \}_{k \in K} \rangle \\
 & & \text{else } \sqcup_{k \in K} G_k \restriction \mathrm{q} \\
\mathbf{R}\ G\ \lambda i : I . \lambda \mathbf{x} . G' \restriction \mathrm{q} & = & \mathbf{R}\ G \restriction \mathrm{q}\ \lambda i : I . \lambda \mathbf{x} . G' \restriction \mathrm{q} \\
(\mu \mathbf{t} . G) \restriction \mathrm{p} & = & \mu \mathbf{t} . G \restriction \mathrm{p} \\
\mathbf{x} \restriction \mathrm{p} & = & \mathbf{x} \\
(G\ \mathtt{i}) \restriction \mathrm{p} & = & (G \restriction \mathrm{p})\ \mathtt{i} \\
\text{end} \restriction \mathrm{p} & = & \text{end}
\end{array}
$$

- $\sqcup_{k \in K}$ corresponds to the notion of mergeability (union of distinct branches).

How do we relate the generic projection to the processes?

Typing: $\Gamma \vdash P \rhd \tau$

$$\frac{\Gamma \vdash \mathsf{Env}}{\Gamma \vdash \mathrm{n} \rhd \mathsf{nat}}\ [\textsc{TNat}] \qquad \frac{\Gamma \vdash \kappa}{\Gamma \vdash \texttt{Alice} \rhd \kappa}\ [\textsc{TId}] \qquad \frac{\Gamma \vdash \mathtt{p} \rhd \Pi i : I.\kappa \quad \Gamma \models \mathtt{i} : I}{\Gamma \vdash \mathtt{p}[\mathtt{i}] \rhd \kappa\{\mathtt{i}/i\}}\ [\textsc{TP}]$$

$$\frac{\Gamma, i : I^{-}, X : \tau\{i/j\} \vdash Q \rhd \tau\{i+1/j\} \quad \Gamma \vdash P \rhd \tau\{0/j\} \quad \Gamma, j : I \vdash \tau \rhd \kappa}{\Gamma \vdash \mathbf{R}\ P\ \lambda i.\lambda X.Q \rhd \Pi j : I.\tau}\ [\textsc{TPRec}]$$

$$\frac{\Gamma \vdash \mathsf{whnf}(G_1) \equiv_{\mathsf{wf}} \mathsf{whnf}(G_2)}{\Gamma \vdash G_1 \equiv G_2}\ [\textsc{WF}] \qquad \frac{\Gamma \vdash P \rhd \tau \quad \Gamma \vdash \tau \equiv \tau'}{\Gamma \vdash P \rhd \tau'}\ [\textsc{TEq}] \qquad \frac{\Gamma \vdash P \rhd \tau \quad \Gamma \vdash \tau \leq \tau'}{\Gamma \vdash P \rhd \tau'}\ [\textsc{TSub}]$$

$$\frac{\Gamma, X : \tau \vdash P \rhd \tau}{\Gamma \vdash \mu X.P \rhd \tau}\ [\textsc{TRec}] \qquad \frac{\Gamma, X : \tau \vdash \mathsf{Env}}{\Gamma, X : \tau \vdash X \rhd \tau}\ [\textsc{TVar}] \qquad \frac{\Gamma \vdash P \rhd \Pi i : I.\tau \quad \Gamma \models \mathtt{i} \in I}{\Gamma \vdash P\ \mathtt{i} \rhd \tau\{\mathtt{i}/i\}}\ [\textsc{TApp}]$$

$$\frac{\begin{array}{c}\Gamma \vdash u : \langle G \rangle \quad \Gamma \vdash P \rhd \Delta, y : G \restriction \mathrm{p}_0 \\ \Gamma \vdash \mathrm{p}_i \rhd \mathsf{nat} \quad \Gamma \models \mathsf{pid}(G) = \{\mathrm{p}_0..\mathrm{p}_\mathrm{n}\}\end{array}}{\Gamma \vdash \bar{u}[\mathrm{p}_0, .., \mathrm{p}_\mathrm{n}](y).P \rhd \Delta}\ [\textsc{TInit}] \qquad \frac{\begin{array}{c}\Gamma \vdash u : \langle G \rangle \quad \Gamma \vdash P \rhd \Delta, y : G \restriction \mathrm{p} \\ \Gamma \vdash \mathrm{p} \rhd \mathsf{nat} \quad \Gamma \models \mathrm{p} \in \mathsf{pid}(G)\end{array}}{\Gamma \vdash u[\mathrm{p}](y).P \rhd \Delta}\ [\textsc{TAcc}]$$

$$\frac{\Gamma \vdash a : \langle G \rangle \quad \Gamma \vdash \mathrm{p} \rhd \mathsf{nat} \quad \Gamma \models \mathrm{p} \in \mathsf{pid}(G)}{\Gamma \vdash \bar{a}[\mathrm{p}] : s \rhd s[\mathrm{p}] : G \restriction \mathrm{p}}\ [\textsc{TReq}] \qquad \frac{\Gamma \vdash e \rhd S \quad \Gamma \vdash P \rhd \Delta, c : T}{\Gamma \vdash c!\langle \mathrm{p}, e\rangle; P \rhd \Delta, c : !\langle \mathrm{p}, S\rangle; T}\ [\textsc{TOut}]$$

Typing (a selection)

$$\frac{\Gamma, i : I^-, X : \tau\{i/j\} \vdash Q \triangleright \tau\{i+1/j\} \quad \Gamma \vdash P \triangleright \tau\{0/j\} \quad \Gamma, j : I \vdash \tau \triangleright \kappa}{\Gamma \vdash \mathbf{R}\ P\ \lambda i.\lambda X.Q \triangleright \Pi j : I.\tau}\ [\text{TPRec}]$$

$$\frac{\Gamma \vdash \mathsf{whnf}(G_1) \equiv_{\mathsf{wf}} \mathsf{whnf}(G_2)}{\Gamma \vdash G_1 \equiv G_2}\ [\text{WF}] \qquad \frac{\Gamma \vdash P \triangleright \tau \quad \Gamma \vdash \tau \equiv \tau'}{\Gamma \vdash P \triangleright \tau'}\ [\text{TEq}]$$

$$\frac{\begin{cases} \Gamma \vdash G_1 \equiv G_2 \quad \Gamma \vdash G_1' \equiv G_2' \qquad \text{or} \\ \Gamma \vdash \mathbf{R}\ G_1\ \lambda i : I.\lambda \mathbf{x}.G_1'\ \mathrm{n} \equiv \mathbf{R}\ G_2\ \lambda i : I.\lambda \mathbf{x}.G_2'\ \mathrm{n} \text{ with } \Gamma \models I = [0..\mathrm{m}], 0 \leq \mathrm{n} \leq \mathrm{m} \end{cases}}{\Gamma \vdash \mathbf{R}\ G_1\ \lambda i : I.\lambda \mathbf{x}.G_1' \equiv_{\mathsf{wf}} \mathbf{R}\ G_2\ \lambda i : I.\lambda \mathbf{x}.G_2'}\ [\text{PRec}]$$

Properties of the type system

Termination

- Termination and confluence of $\longrightarrow$ on global and end-point types.
- Termination of type-equality checking
- Termination of type checking

provided that it is possible to decide the formulas $\Gamma \models \mathtt{P}$ appearing in judgements.

- The arithmetic fragment is kept as a parameter.

Theorems

Theorem (Subject reduction)

If $\Gamma \vdash P \triangleright \tau$ *and* $P \longrightarrow^* P'$, *then* $\Gamma \vdash P' \triangleright \tau'$ *for some* τ' *such that* $\tau \Rightarrow^* \tau'$.

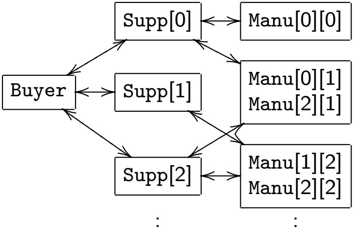
Theorem (Progress)

If P *is well-linked and does not contain the run-time syntax and* $\Gamma \vdash^{\star} P \triangleright \emptyset$. *Then for all* $P \longrightarrow^* Q$, *either* $Q \equiv \mathbf{0}$ *or* $Q \longrightarrow R$ *for some* R.

Type Safety and Deadlock-freedom of FFT

For all m, $\emptyset \vdash P_{\mathsf{fft}}\ \mathrm{m} \triangleright \emptyset$; and for all Q such that $P_{\mathsf{fft}}\ \mathrm{m} \longrightarrow^* Q$, Q is able to send the final values X_{p} on external channels r_{p}.

Web service example



1. A buyer requests a quote from a set of suppliers.
   $G_1 = \texttt{foreach}(i < n)\{\texttt{Buyer} \to \texttt{Supp}[i] : \langle \text{Quote} \rangle\}$
2. All suppliers receive the request ask their respective manufacturers for a bill of material items. The suppliers interact with their manufacturers to build their quotes for the buyer.
   $G_2(i) = \texttt{foreach}(j : J_i)\{\ \texttt{Supp}[i] \to \texttt{Manu}[i][j] : \langle \text{Item} \rangle . \ \texttt{Manu}[i][j] \to \texttt{Supp}[i] : \langle \text{Quote} \rangle\}$
   The eventual quote is sent back to the buyer.
   $G_2 = \texttt{foreach}(i : I)\{G_2(i); \texttt{Supp}[i] \to \texttt{Buyer} : \langle \text{Quote} \rangle\}$
3. EITHER
   1. The buyer agrees with one or more of the quotes and places the order(s). OR
   2. The buyer responds to one or more of the quotes by modifying and sending them back to the relevant suppliers.
4. EITHER
   1. The suppliers respond to a modified quote by agreeing to it and sending a confirmation message back to the buyer. OR
   2. The supplier responds by modifying the quote and sending it back to the buyer and the buyer goes back to STEP 3. OR
   3. The supplier responds to the buyer rejecting the modified quote. OR
   4. The quotes from the manufacturers need to be renegotiated by the supplier. Go to STEP 2.

Web service use case (continued)

- Types:

$$G_3 = \mathbf{R}\ \ \mathbf{t}\ \ \lambda i.\lambda \mathbf{y}.\texttt{Buyer} \to \texttt{Supp}[i] : \{ \begin{array}{ll} \mathsf{ok} : & \mathsf{end} \\ \mathsf{modify} : & \texttt{Buyer} \to \texttt{Supp}[i] : \langle \mathsf{Quote} \rangle . \\ & \texttt{Supp}[i] \to \texttt{Buyer} : \{ \begin{array}{ll} \mathsf{ok} : & \mathsf{end} \\ \mathsf{retryStep3} : & \mathbf{y} \\ \mathsf{reject} : & \mathsf{end}\}\}\ i \end{array} \end{array}$$

- Projection:

$$G_3 \upharpoonright \texttt{Supp}[\mathrm{n}] = \&\langle \texttt{Buyer}, \{ \begin{array}{ll} \mathsf{ok} : & \mathsf{end} \\ \mathsf{modify} : & ?\langle \texttt{Buyer}, \mathsf{Quote} \rangle ; \oplus \langle \texttt{Buyer}, \{ \begin{array}{ll} \mathsf{ok} : & \mathsf{end} \\ \mathsf{retryStep3} : & \mathbf{y} \\ \mathsf{reject} : & \mathsf{end}\}\rangle\}\rangle \end{array} \end{array}$$

Conclusion

We improved multiparty session types' expressiveness towards a better modularity. We proved the safety of protocols that were beyond existing session type technology.

Future challenges

- Parameterised session types in programming languages (Session Java,. . .).
  - Link with existing frameworks (MPI)?
- Security
  - Cryptographic protocol synthesis from session specification [Bhargavan, Corin, Deniélou, Fournet, Leifer CSF'07, CSF'09].
  - Specification/verification of multiparty cryptographic protocols. Ex: Group key agreement protocol [CCS'98]

$$\Pi n : I.(\texttt{foreach}(i < n)\{\texttt{W}[n-i] \to \texttt{W}[n-i+1] : \langle \mathsf{nat} \rangle\};$$
$$\texttt{foreach}(i < n)\{\texttt{W}[n-i] \to \texttt{W}[n] : \langle \mathsf{nat} \rangle . \texttt{W}[n] \to \texttt{W}[n-i] : \langle \mathsf{nat} \rangle\})$$

Related Work

- Dependent types
  - Gödel's system T
  - Primitive Recursion [Nelson, MFPS'91]
  - Dependent ML [Xi, Pfenning, POPL'99]
- Session types
  - Multiparty Session Types [Carbone, Honda, Yoshida, POPL'08, CONCUR'08, ESOP'09]
  - Session Java [Hu, Yoshida, Honda, ECOOP'08, ECOOP'10]
- Other session type system
  - Conversation Types [Caires, Vieira, ESOP'09]
  - Contracts [Castagna, Padovani, CONCUR'09]
- Examples
  - FFT Algorithm [Cooley, Tuckey, Math. of Comp. '65]
  - Web Service Choreography Working Group

Thanks!