Synchronous Multiparty Session Types

Andi Bejleri Nobuko Yoshida
Imperial College London
Aim of this work

The aim of this work is to study and implement a formal model where communication protocols are abstracted at type-level, and a static type system that checks if programs implement a coherent and loyal protocol with respect to the abstracted one.

The study of the formal model will be based on a variant of the π-Calculus and its implementation will be based on an extension of Java.
Communication between processes

- A feature that is widely used in: Web Services, Distributed Algorithms, Business Processes and Multi-core Programming.

- Programs of these systems implement a communication protocol: order of sending and receiving messages.
Binary session types (PARLE 94)

- A **session** is the interactive structure formed by the sending-receiving primitives and the conditional-iteration/recursion constructs.

- An abstraction at type-level of the session between **two** processes.

- Type system checks if the inferred binary session type are **reciprocal** with each other.
An arithmetic problem: Addition of two naturals

- In a context where addition, successor and predecessor are defined as processes; arguments and results are passed via communications.

- Client
  - Addition
    - arg1
    - arg2
    - true
    - result
  - Successor
    - arg1
    - arg1 + 1
  - Predecessor
    - arg2
    - arg2 - 1

branch
recursive
Representing the solution in binary session types

- **Client-Addition**: sending of the two arguments and result
  \[
  !\text{int};!\text{int};?\text{int};\text{end} \quad ?\text{int};?\text{int};!\text{int};\text{end}
  \]

- **Addition-Successor**: successor of the first argument
  \[
  \mu t.\{\text{true: end, false: } !\text{int};?\text{int};t\}\]

  \[
  \mu t.\&\{\text{true: end, false: } ?\text{int};!\text{int};t\}\]

- **Addition-Predecessor**: predecessor of the second argument
  \[
  \mu t.\{\text{true: end, false: } !\text{int};?\text{int};t\}\]

  \[
  \mu t.\&\{\text{true: end, false: } ?\text{int};!\text{int};t\}\]
Programming problem: Order of communications

- The three binary session types are separate between each other.

- Order of communications are not captured by binary session types.
  - The behavior of Client-Addition depends on the behavior of Successor-Addition and Predecessor-Addition.
  - Evident when the second argument is not 0
  - Addition has to perform addition in Successor-Addition and Predecessor-Addition before returning the result to Client.
Solution: Global types

- A syntax of arrows, names and types
- A global perspective of the protocol

\[ G = \text{Client} \longrightarrow \text{Addition} : \langle \text{int} \rangle. \]
\[ \text{Client} \longrightarrow \text{Addition} : \langle \text{int} \rangle. \]
\[ \mu t. \text{Addition} \longrightarrow \text{Successor, Predecessor} : \{ \]
\[ \text{true}: \text{Addition} \longrightarrow \text{Client} : \langle \text{int} \rangle. \text{end}, \]
\[ \text{false}: \text{Addition} \longrightarrow \text{Successor} : \langle \text{int} \rangle. \]
\[ \text{Successor} \longrightarrow \text{Addition} : \langle \text{int} \rangle. \]
\[ \text{Addition} \longrightarrow \text{Predecessor} : \langle \text{int} \rangle. \]
\[ \text{Predecessor} \longrightarrow \text{Addition} : \langle \text{int} \rangle.t \}\]

Channels are omitted for simplicity: Client-Add : r, Add-Succ : s, Add-Pred : t
Not only as a blueprint but also …

As a typing discipline.

(Honda et al, POPL 08)
Contributions

- **Synchronous Communications** model control for timing events and strong sequentiality order of messages.
- A simpler calculus
- Higher-order communication as $k!(k')!k?(x)$
- Multicasting (Addition $\rightarrow$ Successor, Predecessor)
- Global Type + Linearity
- Type Preservation
Linearity checking for global types

- In a context where channels are shared among different communications
- E.g. part of local memory
- The typing discipline (global types + projection) does not guarantee that the communications at run-time are the same as those specified in the global behavior
  - Ambiguity: \( k!^{<5>}; k!^{<4>} | k?(x) | k?(y) \)
  - Errors: Broken Invariants
Linearity on synchronous communications 1/3

\[
\begin{array}{|c|c|c|c|c|}
\hline
P & Q & \ldots & R & S \\
\hline
k!\langle 5 \rangle & k?\langle x \rangle & \ldots & k!\langle 7 \rangle & k?\langle x \rangle \\
\hline
\end{array}
\]
Linearity on synchronous communications 2/3

\[
\begin{array}{cccc}
P & Q & \ldots & R \\
k!<&5> & k?(x) & & \\
\end{array}
\]

\[
\begin{array}{cccc}
P & Q : k & Q & R : k \\
R & Q : l & Q & P : l \\
\end{array}
\]

\[
\begin{array}{cccc}
P & Q : k & Q & P : k \\
R & Q : l & R & Q : l \\
\end{array}
\]

\[
\begin{array}{cccc}
P & Q : k & Q & P : k \\
R & Q : l & Q & P : l \\
\end{array}
\]

\[
\begin{array}{cccc}
P & Q : k & (OO,II) & Q & P : k & (IO,OI) \\
\end{array}
\]
Linearity on synchronous communications 3/3

- G is *linear* if, whenever $n_1 = p_1 \rightarrow p_1':m$ and $n_2 = p_2 \rightarrow p_2':m$ are in G for some $m$ and do not occur in different branches of a branching, then a chain $(\varphi_1, ..., \varphi_n)$ such that $n_1 \preceq \varphi_1 ... \preceq \varphi_n n_2$ exists such that the two conditions are satisfied:
  
  1) $p_j \rightarrow p_2: m'$ or $p_2 \rightarrow p_j: m'$ is a node of the chain
  
  2) $p_k \rightarrow p_2':m''$ or $p_2' \rightarrow p_k:m''$ is a node of the chain.

In case of multicasting (Addition $\rightarrow$ Successor, Predecessor $:m$), all the chains obtained by distributing each prefix of multicasting on the rest of G have to be checked if they satisfy the above conditions. If G carries other global types, we inductively demand the same.
Global types syntax

G ::= p →p1, ..., pj: m1, ..., mj<Sp>, G' for all i, k \in \{1, ..., j\}.mi ≠ mk values
p →p''m<T@q>.G'

G, G'
μt.G

U ::= S1, ..., Sn | T@p Value
S ::= bool | nat | ... | <G> Sort
m ::= 1 | 2 | ...
Invisibility of delegation at global types

- Global Types should define only the interactions of that session.

\[ G = \text{Client} \rightarrow \text{Addition: } r\langle\text{int}\rangle. \quad G' = \text{Client} \rightarrow \text{P: } d<r?\langle\text{int}\rangle@\text{Client}> \]

\[ \mu t. \text{Addition} \rightarrow \text{Successor, Predecessor : } s, t\{\]
\[ \text{true: Addition} \rightarrow \text{Client : } r\langle\text{int}\rangle. \text{end}, \]
\[ \text{false: Addition} \rightarrow \text{Successor : } s\langle\text{int}\rangle. \]
\[ \text{Successor} \rightarrow \text{Addition : } s\langle\text{int}\rangle. \]
\[ \text{Addition} \rightarrow \text{Predecessor : } t\langle\text{int}\rangle. \]
\[ \text{Predecessor} \rightarrow \text{Addition : } t\langle\text{int}\rangle. t\} \]
Conclusion

- Multiparty Session Types: Global Types + Local Types (Types obtained by projection)

- Global Types: Intuitive syntax + global perspective = Clear view of the protocol

- Typing discipline: global type + linearity + projection + type-checking = Duality (reciprocity) + Fidelity (loyalty)
Related work

- Asynchronous Multiparty Session Types: Honda et al. (POPL 2008)
- Scribble at http://pi4scribble.wiki.sourceforge.net/