Multiparty Session Types and their Applications to Large Distributed Systems

Nobuko Yoshida and Raymond Hu
Imperial College London
Session Type Projects

- **COST Action**  *Behavioural Types for Reliable Large-Scale Software Systems*, over 60 academic members in 17 countries

- **SADEA**  EPSRC *Exploiting Parallelism through Type Transformations for Hybrid Manycore Systems*, with Vanderbauwhede (GL), Scholz (HW) (1.53M)

- **Programme Grant**  EPSRC *From Data Types to Session Types: A Basis for Concurrency and Distribution*, with Wadler (ED) and Gay (GL) (3.9M)

- EPSRC *Conversation-Based Governance for Distributed Systems by Multiparty Session Types* (1.5M)

- **EU FP7 FETOpenX**  UpScale with de Boer (CWI), Clark, Wrigstad (Uppsala) Johnsen (Oslo) and Drossopoulou

- **VMware**  Dynamic Assurance based on Multiparty Session Types

- **Cognizant**  EPSRC Knowledge Transfer Secondments

- EPSRC Imperial Doctoral Prize Fellowship
In collaboration with:

Matthew Arrott (OOI)
Gary Brown (Red Hat)
Stephen Henrie (OOI)
Bippin Makoond (Cognizant)
Michael Meisinger (OOI)
Matthew Rawlings (ISO TC68 WG4/5)
Alexis Richardson (RabbitMQ/VMware)
Steve Ross-Talbot (Cognizant)

and all our academic colleagues

Laura Bocchi, Tzu-Chun Chen, Romain Demangeon, Pierre-Malo Denielóu, Luca Fossati, Dimitrios Kouzapas, Rumyana Neykova,
Nicholas Ng, Weizhen Yang
Communication is Ubiquitous

➤ Internet, the WWW, Cloud Computing, the next-generation manycore chips, message-passing parallel computations, large-scale cyberinfrastructure for e-Science.

➤ The way to organise software is increasingly based on communications.

➤ Applications need *structured* series of communications.

➤ Question

➤ How to formally abstract/specify/implement/control communications?
Communication is Ubiquitous

➤ Internet, the WWW, Cloud Computing, the next-generation manycore chips, message-passing parallel computations, large-scale cyberinfrastructure for e-Science.

➤ The way to organise software is increasingly based on communications.

➤ Applications need *structured* series of communications.

➤ **Question**

➤ How to *formally* abstract/specify/implement/control communications?
Communication is Ubiquitous

➤ Internet, the WWW, Cloud Computing, the next-generation manycore chips, message-passing parallel computations, large-scale cyberinfrastructure for e-Science.

➤ The way to organise software is increasingly based on communications.

➤ Applications need *structured* series of communications.

Question $\Rightarrow$ Multiparty session type theory

➤ How to formally abstract/specify/implement/control communications?
Ocean Observatories Initiative

➤ A NSF project (400M$, 5 Years) to build a cyberinfrastructure for observing oceans around US and beyond.

➤ Real-time sensor data constantly coming from both off-shore and on-shore (e.g. buoys, submarines, under-water cameras, satellites), transmitted via high-speed networks.
Ocean Observatories Initiative
Ocean Observatories Initiative
Challenges

➤ The need to specify, catalogue, program, implement and manage *multiparty message passing protocols*.

➤ Communication assurance
  ➢ Correct message ordering and synchronisation
  ➢ Deadlock-freedom, progress and liveness
  ➢ Dynamic message monitoring and recovery
  ➢ Logical constraints on message values

➤ Shared and used over a long-term period (e.g. 30 years in OOI).
Why Multiparty Session Types?

➤ Robin Milner (2002): *Types are the leaven of computer programming; they make it digestible.*
  ➞ Can describe communication protocols as *types*
  ➞ Can be materialised as *new communications programming languages* and *tool chains*.

➤ *Scalable* automatic verifications (deadlock-freedom, safety and liveness) without *state-space explosion problems* (*polynomial time complexity*).

➤ Extendable to *logical verifications* and flexible *dynamic monitoring*. 
Dialogue between Industry and Academia

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 $\pi^4$ Technology
CDL Equivalent

• Basic example:

```java
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;
        }
    }
}
```

Dr Gary Brown (Pi4 Tech) in 2007
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:

```haskell
protocol HelloWorld {
    role You, World;
    Hello from You to World;
}
```
Dialogue between Industry and Academia

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 \( \pi^4 \) Technology

⇓

Multiparty Session Types [POPL’08]
Dialogue between Industry and Academia

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

- Buyer
  - title
  - quote
  - ok
  - address
  - date
  - quit

- Seller

branch
! String ; ? Int ; ⊕ { OK : ! String ; ? Date ; end , quit : end }
! String ; ? Int ; ⊕ { ok: ! String ; ? Date ; end , quit : end }
Multiparty Session Types

Buyer 1

- title
- quote
- quote ÷ 2

Seller

- quote
- ok
- address
- date

Buyer 2
Multiparty Session Types

Buyer 1 → Seller → Buyer 2

title
QUOTE
QUOTE ÷ 2
QUOTE

address
ok
date
Multiparty Session Types

Global Types

Projection

$T_{Alice} \quad T_{Bob} \quad T_{Carol}$

Local Types

Multiple Languages

Type checking

Multiparty Session Types

$G$

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

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

$P_{Bob} = s?(\text{Alice}, x); s!(\text{Carol}, x); 0$
Binary session types correspond to two compatible, deterministic CFSMs with non-mixed states [Gouda et al 86]

⇒ Multiparty session types and CFSMs [ESOP’12,ICALP’13].
Others’ code may be unreliable, specifications can change.

Use CFSMs generated from local types as monitors, checking incoming and outgoing messages in linear time, managing global behaviour.

Theories of dynamic monitoring and logics

[CONCUR’10, TGC’11, TGC’12, CONCUR’12, FMOODS’13].
Use Case: Command Instrument

Note: The pictured instrument above is SBE49 FastCAT.
Distribute Application Facility

Service Resource
- Service
- Service Resource Control Protocol
- Protocol Factory
  - Magnet
  - txAMQP
  - Twisted Reactor

Physical Resource
- Managed Resource Agent
- Physical Resource Control Protocol

Managed Resource Agent
- FSM

Proxy Resource Control
- FSM

Message Service
- AMQP Broker

Ocean Observatories Initiative
**Command Instrument Specification**

![Diagram of Command Instrument Specification]

**Predicates**

- $A = (y \geq 0)$
- $A2 = (x_n > 0)$
- $A4 = (x_p = \text{high} \supset x_e \neq \text{busy})$
- $A5 = (y > 0 \land x_{com} \neq \text{switch-off})$
Use Case

Global View:

\[ G_{IC} = \text{User} \rightarrow \text{Register} : (x_{int} : \text{InterfaceId}) \]
\[ \text{Register} \rightarrow \text{User} : (x_n : \text{Int}) \]
\[ \text{User} \rightarrow \text{Agent} : (x_P : \text{Priority}) \]
\[ \text{Agent} \rightarrow \text{User} : \{\text{accept}().G_{acc}, \text{reject}(x_P : \text{ErrData})\} \]

\[ G_{acc} = \mu t(x_n)(y)\{y \geq 0\}. \]
\[ \text{User} \rightarrow \text{Agent} : \{\text{more}(x_{com} : \text{Command}).G_{com}, \text{quit}()\} \]

\[ G_{com} = \text{Agent} \rightarrow \text{Instrument} : (y_{com} : \text{Command}). \]
\[ \text{Instrument} \rightarrow \text{Agent} : (y_{r} : \text{Response}). \]
\[ \text{Agent} \rightarrow \text{User} : (x_{r} : \text{Response}). t(y - 1) \]

Scribble:

session AccessConversation =
roles user, agent, register

global inv main = {
  InterfaceId(string) from user to register;
  MaxCommands(string) from register to user;
  Priority(string) from user to agent;
  choice from user to agent
  \{Accept.
    \{more_commands:
      choice from agent to user
      \{More.
        CCommand(string) from user to agent;
        choice from agent to instrument
        \{Command.
          CCommand(string) from agent to instrument;
          Response(string) from instrument to agent
          | Quit;
          Response(string) from agent to user
          | Quit;
          more_commands\}
        | Reject.(ErroData(string) from agent to user;)}\}
  }
Figure 5: A coordinated set of autonomous underwater vehicles
Figure 3: Observatory comprised of ships, aircraft and autonomous vehicles linked to assimilation modeling capabilities on shore
Welcome to Release 2 of the Ocean Observatories Initiative Observatory (OOI). You already have access to many OOI features and real-time data. Just click on something that looks interesting on this page to start using the OOI as our guest.

For personalized services, such as setting up notifications and preserving settings for your next visit, create a free account by clicking on "Create Account" at the top of the page.

National Science Foundation working with Consortium for Ocean Leadership

Funding for the Ocean Observatories Initiative is provided by the National Science Foundation through a Cooperative Agreement with the Consortium for Ocean Leadership. The OOI Program: Implementing Organizations are funded through sub-awards from the Consortium for Ocean Leadership.
Multiparty Session Type Theory

- Multiparty Asynchronous Session Types [POPL’08]
- Progress
  - 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 for Distributed Programs [CONCUR’12]
- Multiparty, Multi-session Logic [TGC’12]

Extensions of Multiparty Session Types

- Multiparty Symmetric Sum Types [Express’10]
- Parameterised Multiparty Session Types [FoSSaCs’10, LMCS]
- Dynamic Multirole Session Types [POPL’11]
- Nested Multiparty Sessions [CONCUR’12]
Dynamic Monitoring
- Asynchronous Distributed Monitoring for Multiparty Session Enforcement [TGC’11]
- Monitoring Networks through Multiparty Sessions [FORTE’13]

Automata Theories
- Multiparty Session Automata [ESOP’12]
- Synthesis in Communicating Automata [ICALP’13]

Typed Behavioural Theories
- On Asynchronous Eventful Session Semantics [FORTE’11]
- Governed Session Semantics [CONCUR’13]

Choreography Languages
- Compositional Choreographies [CONCUR’13]
Language and Implementations

- Carrying out large-scale experiences with OOI, VMWare, Red Hat, Congnizant, UNIFI, TrustCare

- JBoss S C R I B B L E [ICDCIT’10, COB’12, TGC’13] and S A V A R A projects

- High-performance computing
  - Session Java [ECOOP’08, ECOOP’10, Coordination’11] ➞ C and MPI [TOOLS’12][Hearts’12][EuroMPI’12][PDP’14]

- Multiparty session languages
  - Ocaml, Java, C, Python, Scala, Jolie
  - Trustworthy Pervasive Healthcare Services via Multiparty Session Types [FHIES’12]
  - SPY: Local Verification of Global Protocols [RV’13]
  - Practical interruptible conversations: Distributed dynamic verification with session types and Python [RV’13]
Session Type Reading List

[ESOP’98] Honda, Vasconcelos and Kubo, Language Primitives and Type Disciplines for Structured Communication-based Programming,

[SecRet’06] Yoshida and Vasconcelos, Language Primitives and Type Disciplines for Structured Communication-based Programming Revisited, ENTCS.

[ECOOP’08] Hu, Yoshida and Honda, Session-Based Distributed Programming in Java

[POPL’08] Carbone, Yoshida and Honda, Multiparty Asynchronous Session Types

[WS-FM’09] Dezani-Ciancaglini and de’Liguoro, Sessions and Session Types

[TOOLS’12] Ng, Yoshida and Honda, Multiparty Session C

[CONCUR’10] Caires and Pfenning, Session Types as Intuitionistic Linear Propositions; [ICFP’12] Walker, as Classical Linear Propositions.

[OOI] Video by John Orcutt, Professor of Geophysics, UCSD, Ocean Observing: Oceanography in the 21st Century