

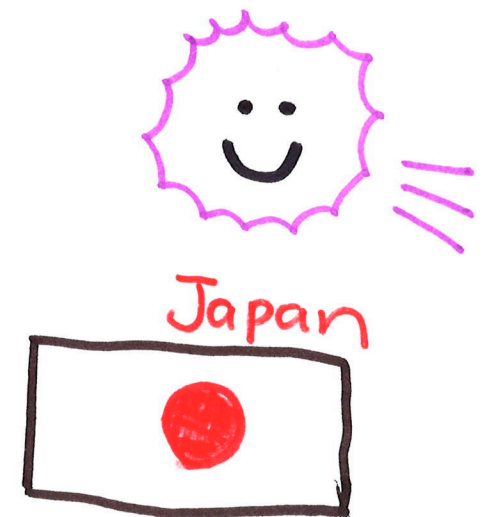
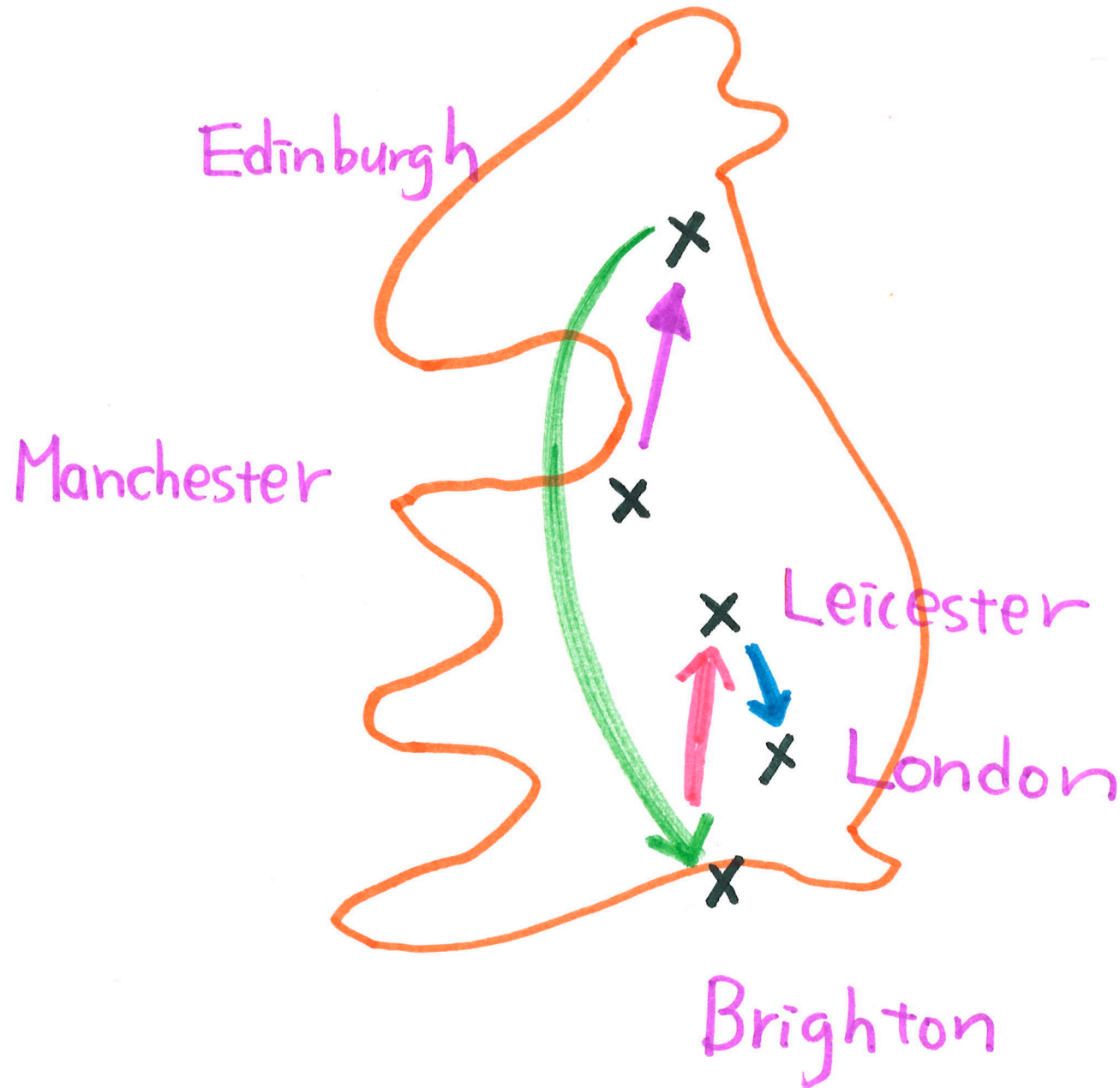
# Scribble, Runtime Verification and Multiparty Session Types



<http://mrg.doc.ic.ac.uk/>

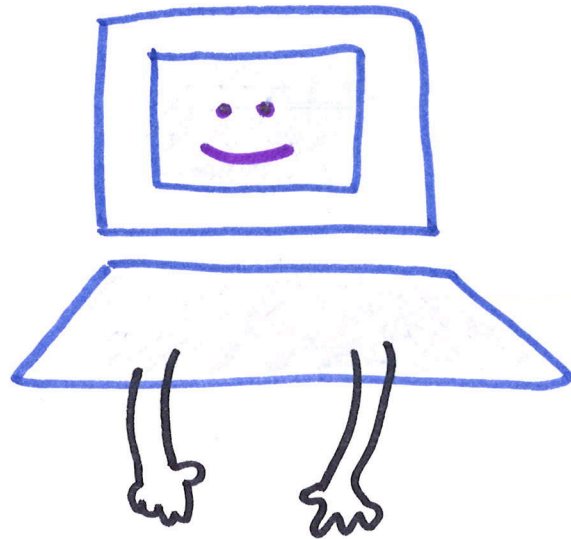
Nobuko Yoshida  
Imperial College London

# My Mobile Research Life in UK

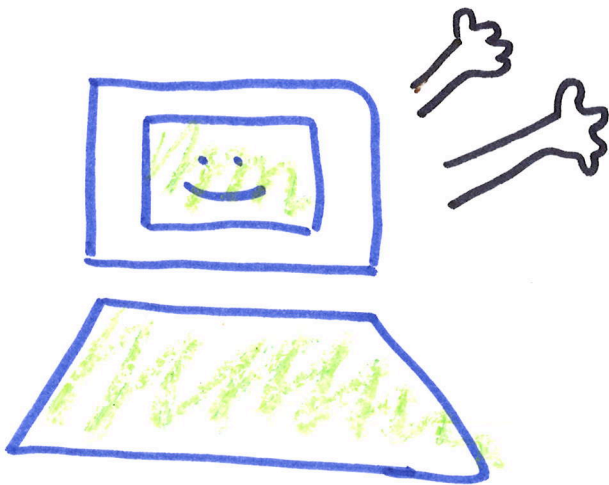
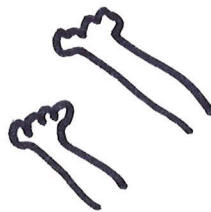


Interactive  
Computing is

Ubiquitous

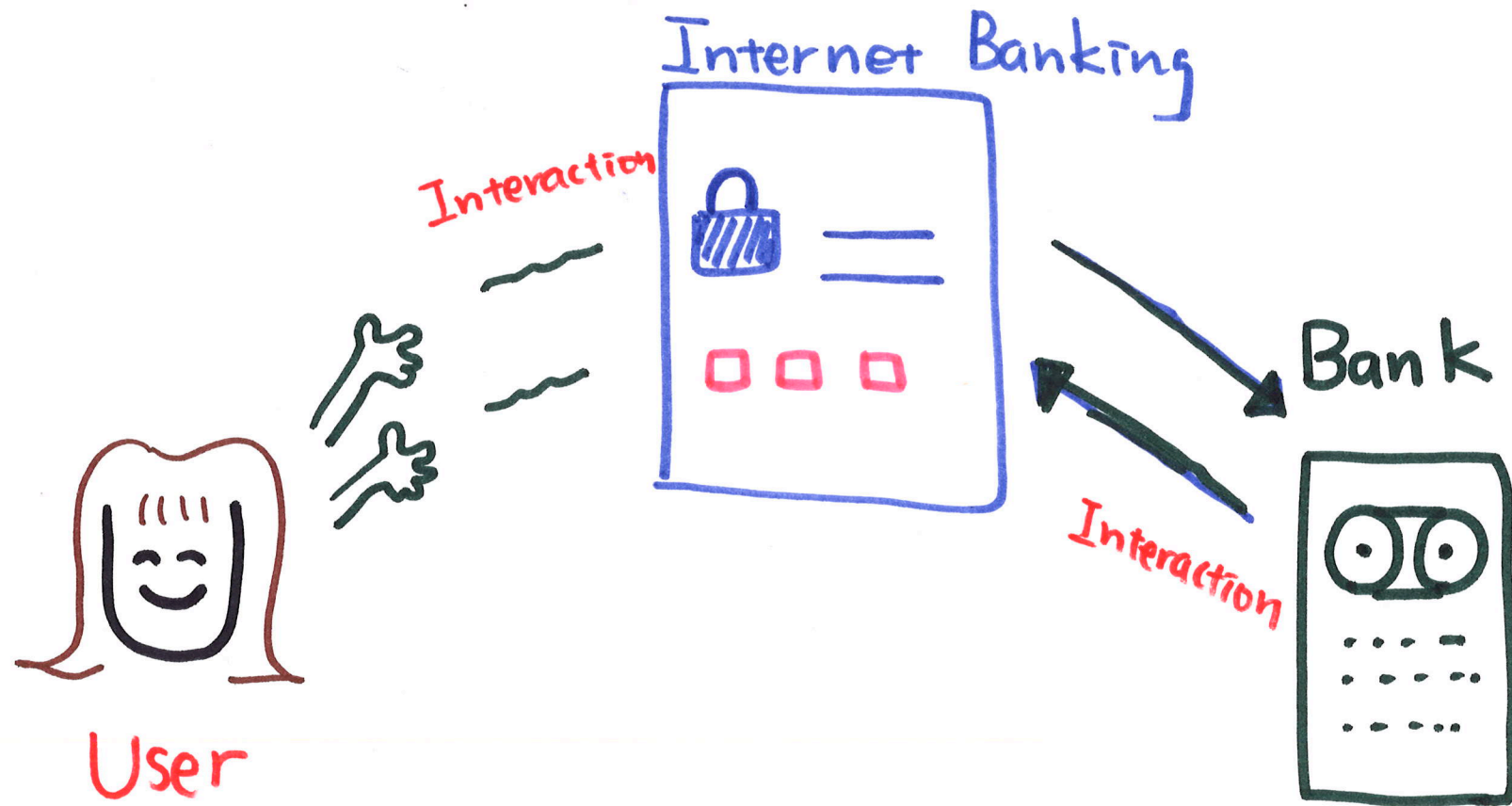


Task

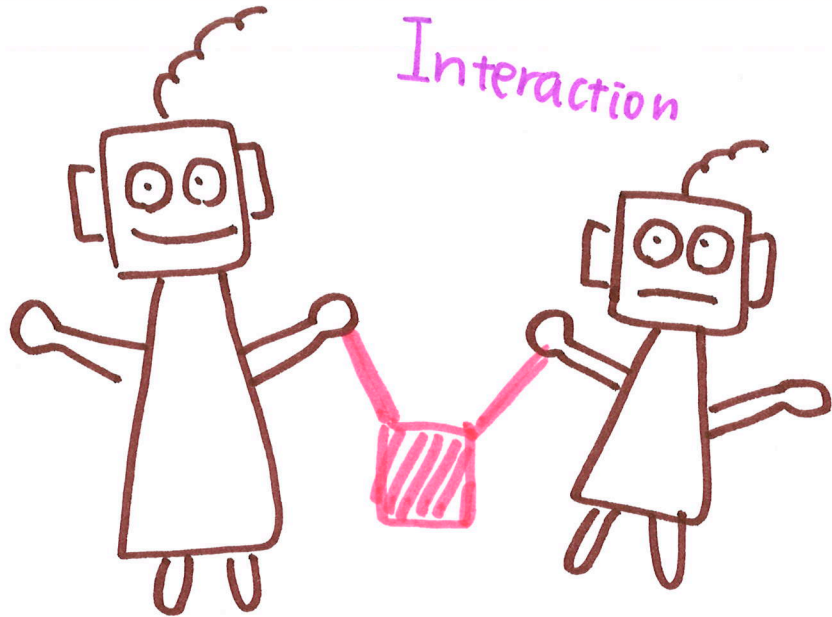


Computing Machines in Network

# Global Banking System

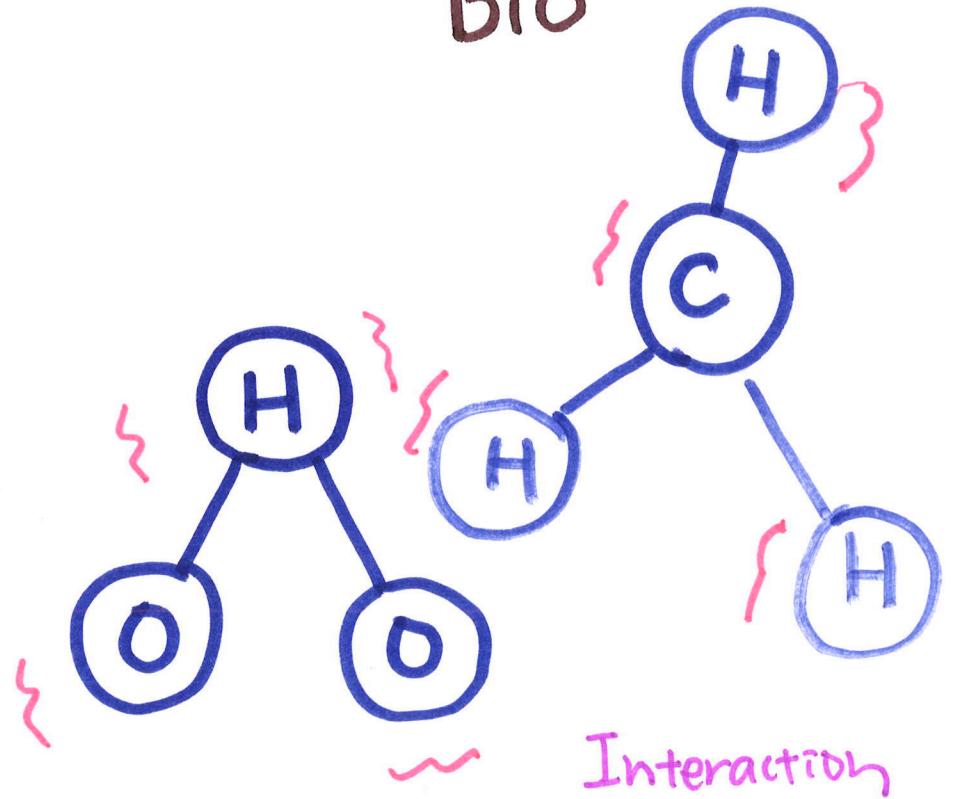


Interaction



Mobile Robots

Bio



Interaction

# Interactive Computing is Ubiquitous

- Computing Machines, Network Infrastructure, Global Banking System, Applications in Internet, Mobile Robots, Man-machine Interface, ...

- **Basic Questions**

- How to understand interactive behaviour **systematically**?
- What is this **abstract (mathematical) entity** called interactive behaviour?
- Given some system/software, how can we **formally** specify/describe/control its behaviour?



# Interactive Computing is Ubiquitous

- Computing Machines, Network Infrastructure, Global Banking System, Applications in Internet, Mobile Robots, Man-machine Interface, ...

➤ **Basic Questions**  $\implies$  **Model/Calculus**

➤ How to understand interactive behaviour **systematically**?

➤ What is this **abstract (mathematical) entity** called interactive behaviour?

➤ Given some system/software, how can we **formally** specify/describe/control its behaviour?



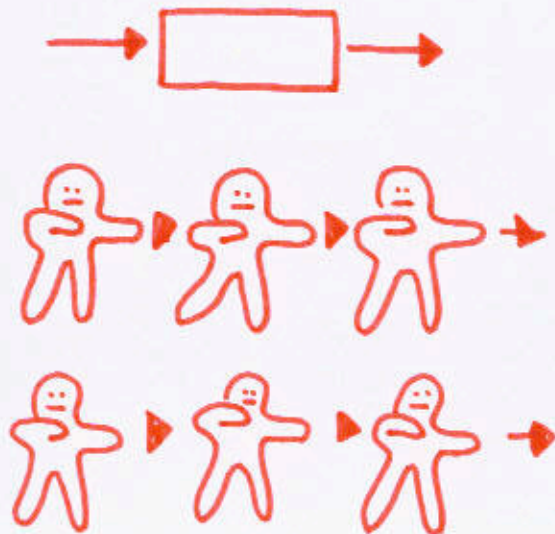
# Sequential vs Interactive Computation

## $\lambda$ -calculus

$M ::= x \mid \lambda x.M \mid MN$

Logic/Category Theory

Lisp, ML, Haskell,...



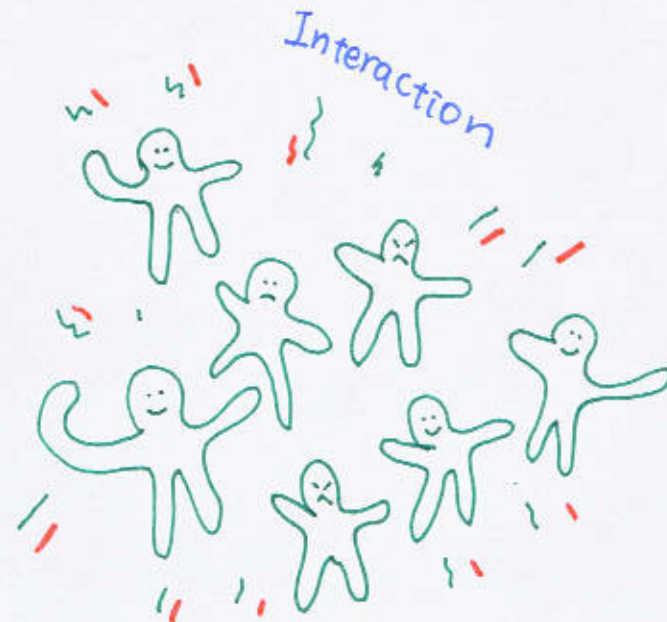
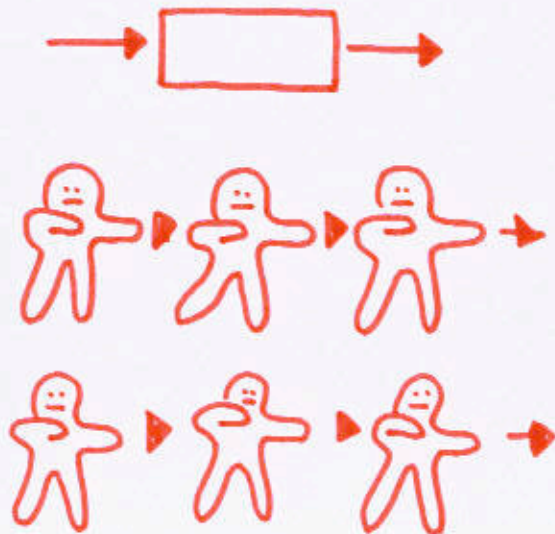
# Sequential vs Interactive Computation

## $\lambda$ -calculus

$$M ::= x \mid \lambda x.M \mid MN$$

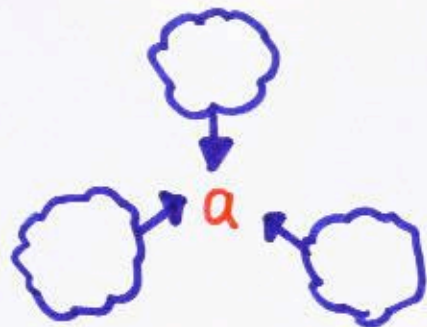
Logic/Category Theory

Lisp, ML, Haskell,...

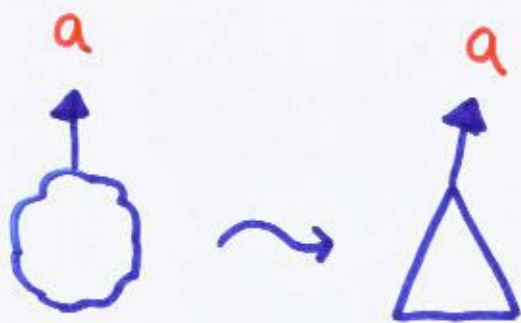


# The $\pi$ -Calculus [1989–]

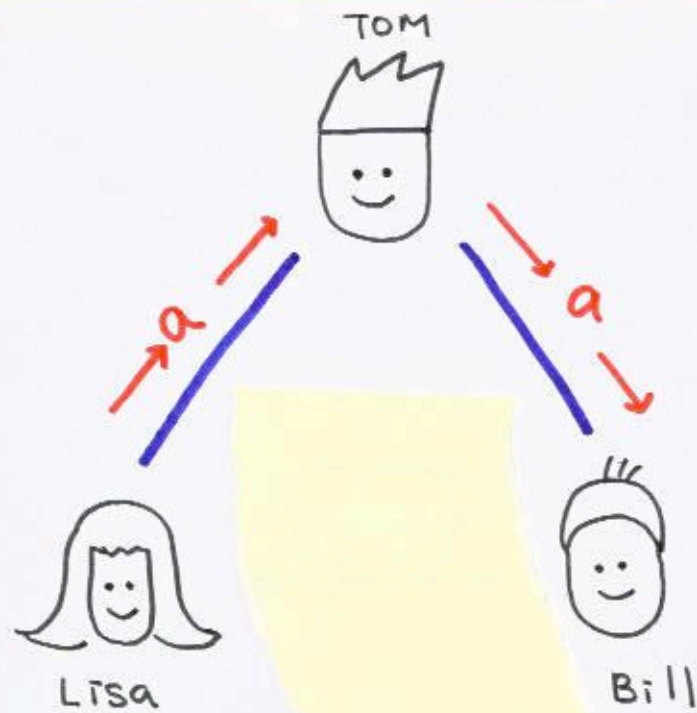
Shared Point



Identity of Agent

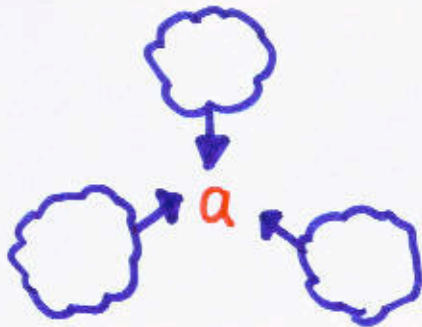


Name Passing as Pointer Passing

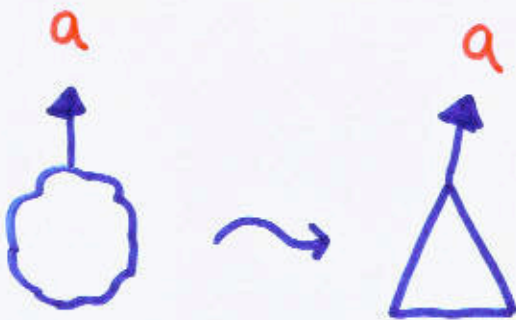


# The $\pi$ -Calculus [1989–]

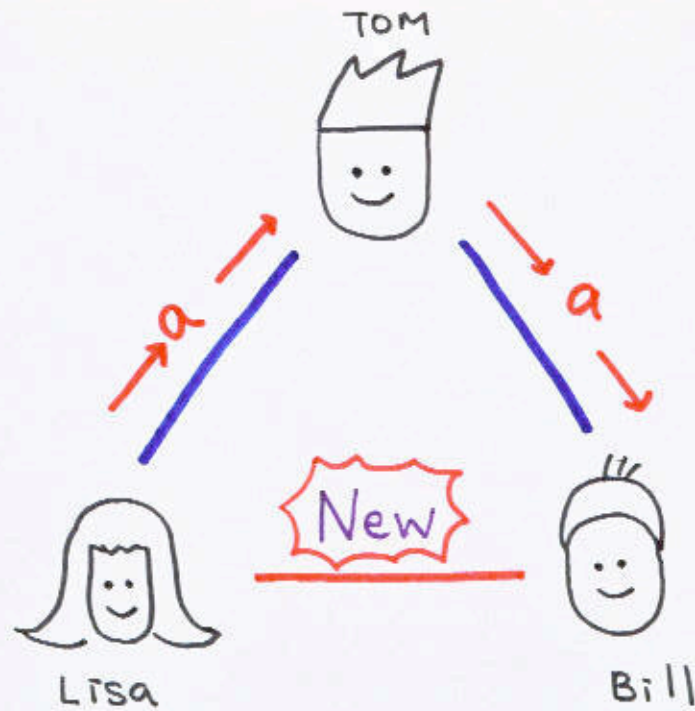
Shared Point



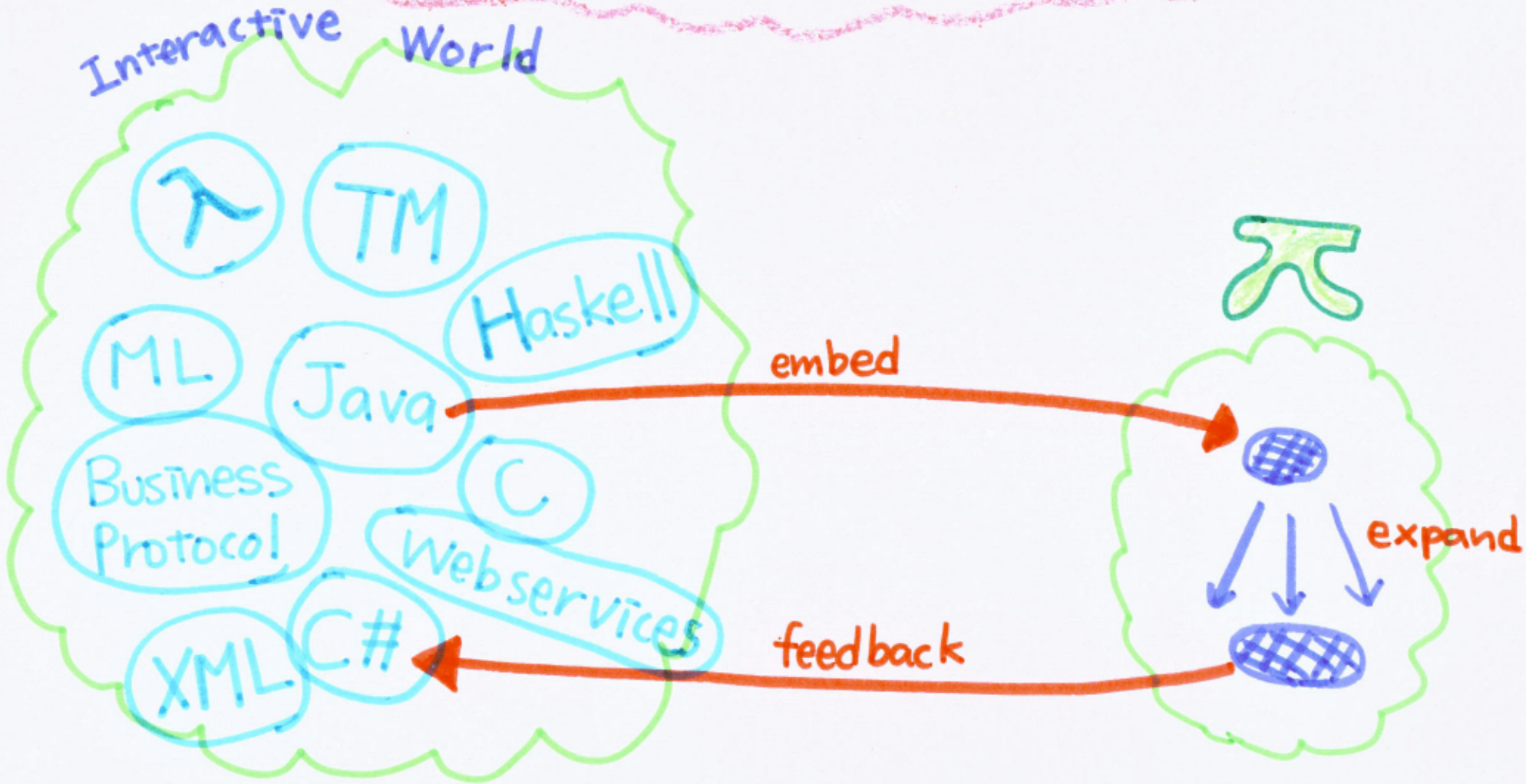
Identity of Agent



Name Passing as Pointer Passing



$$\underline{a}(x).P \mid \underline{\bar{a}}\langle v \rangle \rightarrow P[v/x]$$



$$P \approx Q$$

$\Leftrightarrow$   
Full Abstraction

$$\llbracket P \rrbracket_u \approx \llbracket Q \rrbracket_u$$

## Current: Communication is Ubiquitous

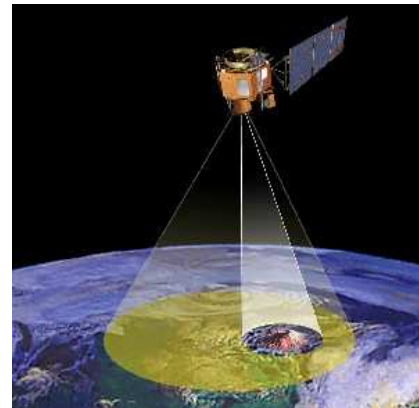
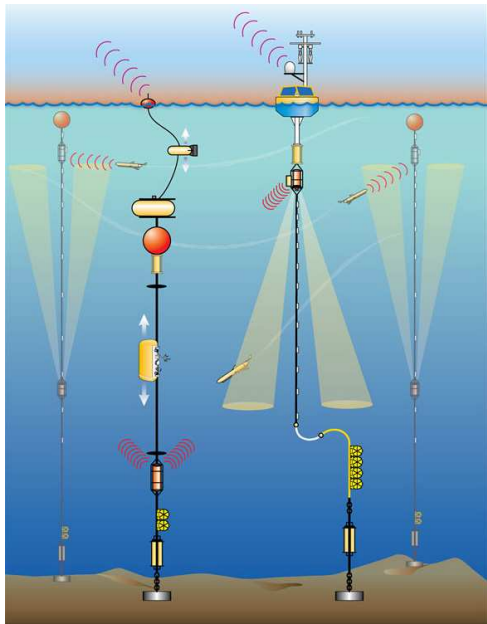
- The way to organise software is increasingly based on communications (Cloud Computing, many cores, message-passing parallel computation, ...)
- **Question**
  - How to **formally** abstract/specify/implement/control communications?
  - How to apply mobile processes and their type theories to real distributed applications and programming languages?

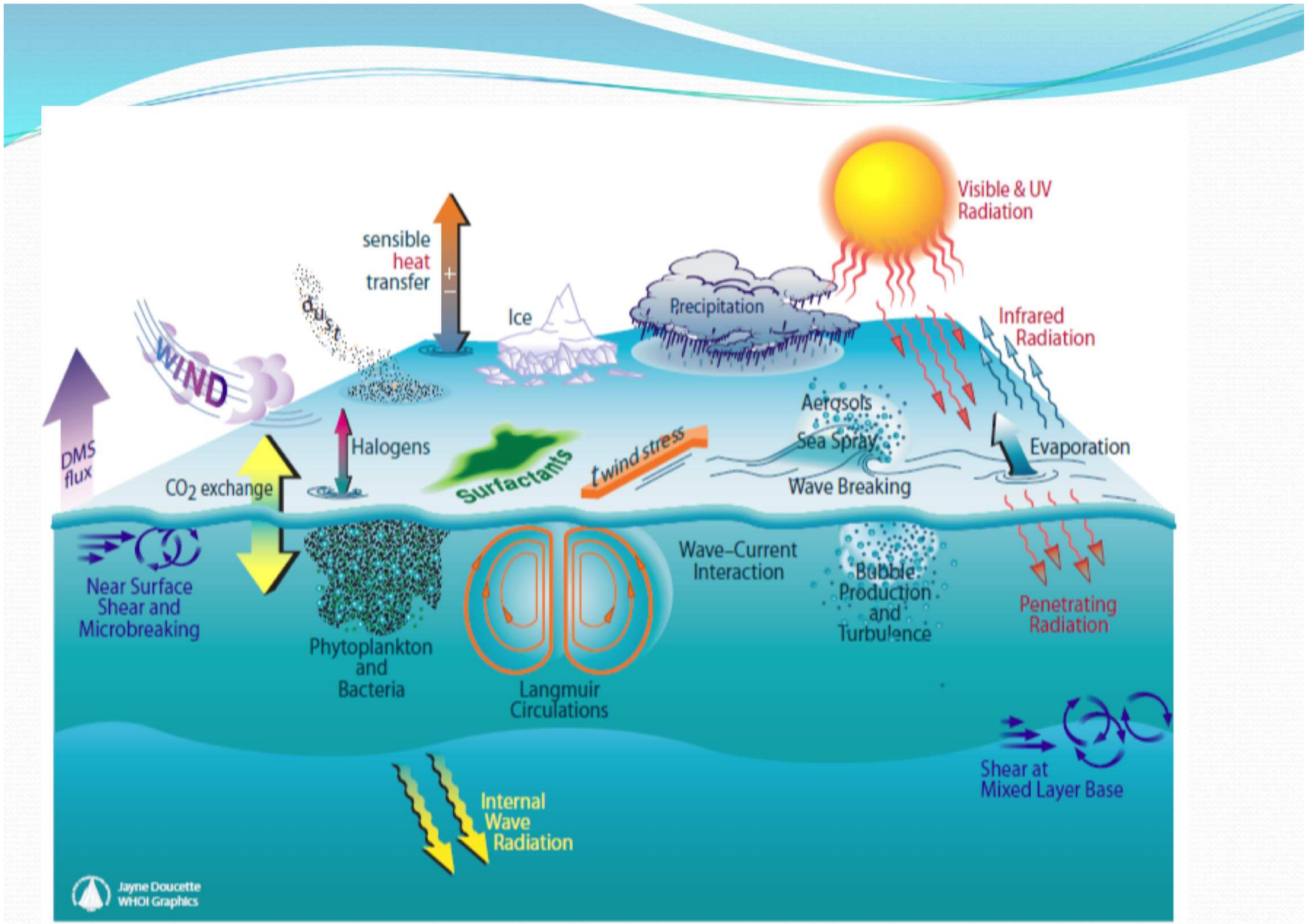
## Current: Communication is Ubiquitous

- The way to organise software is increasingly based on communications (Cloud Computing, many cores, message-passing parallel computation, ...)
- **Question**  $\implies$  **Multiparty session type theory**
  - How to **formally** abstract/specify/implement/control communications?
  - How to apply mobile processes and their type theories to real distributed applications and programming languages?
    - $\implies$  **large-scale cyberinfrastructure for e-Science**

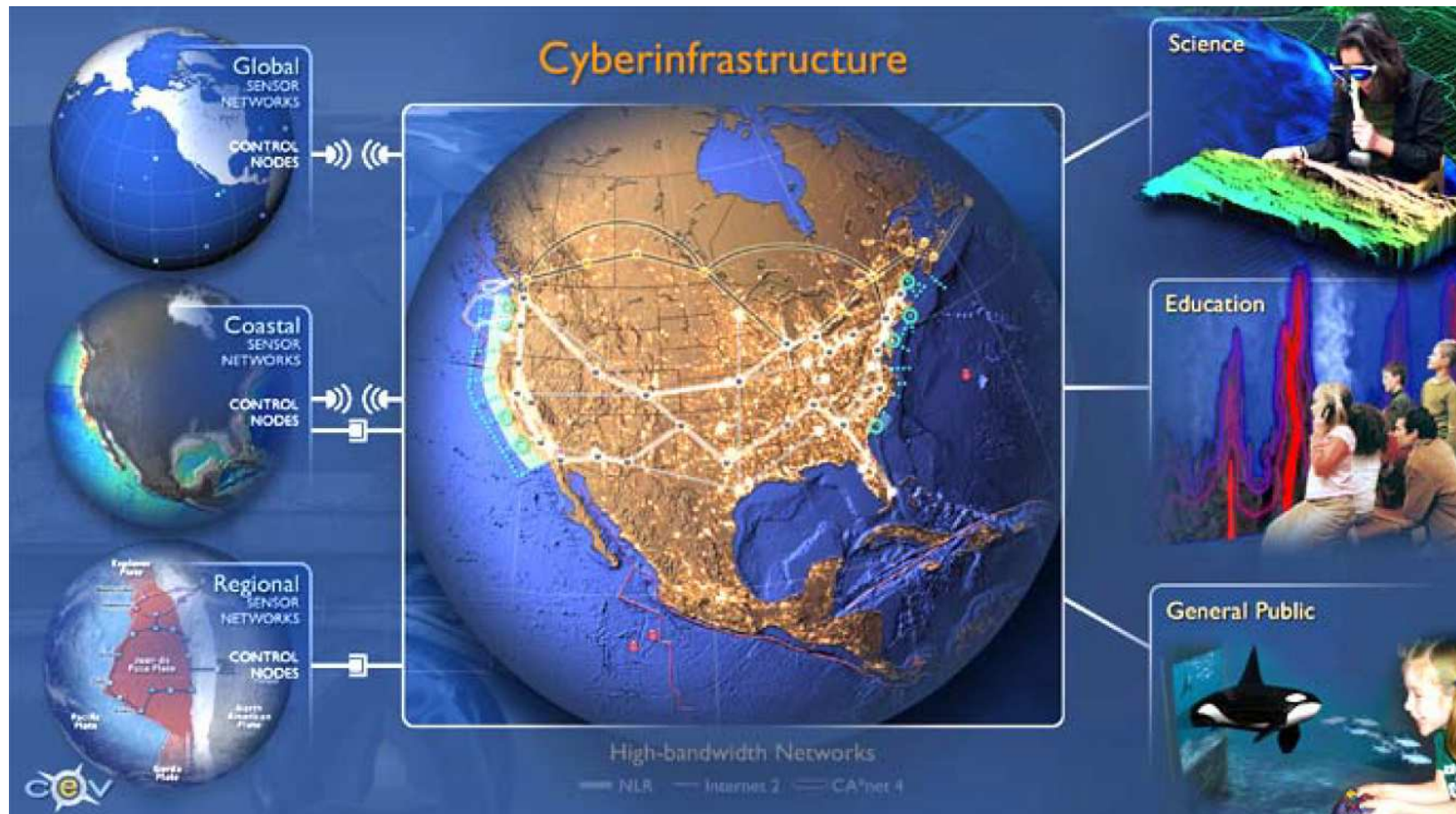
# Ocean Observatories Initiative

- 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





## 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

# Pi calculus versus Petri nets: Let us eat “humble pie” rather than further inflate the “Pi hype”

W.M.P. van der Aalst

**Abstract.** In the context of *Web Service Composition Languages* (WSCLs) there is an ongoing debate on the best foundation for *Process-Aware Information Systems* (PAISs): *Petri nets* or *Pi calculus*. Examples of PAISs are Workflow Management (WFM), Business Process Management (BPM), Business-to-Business (B2B), Customer Relationship Management (CRM), Enterprise Resource Planning (ERP) systems. Clearly, the web-service paradigm will change the architecture of these systems dramatically. Therefore, triggered by industry standards such as SOAP, WSDL, UDDI, etc., standards are being proposed for orchestrating web services. Examples of such WSCLs are BPEL4WS, BPML, WSFL, WSCI, and XLANG. In the debate on Petri nets versus Pi calculus many players in the “WSCL marketplace” are using demagogic arguments not based

**Petri-Pi** Working Group led by R. Milner and W.M.P van der Aalst started in 2003

# Beginning: Petri-Pi

From: Robin Milner

Date: Wed, February 11, 2004 1:02 pm

Steve

Thanks for that. I believe the pi-calculus team ought to be able to do something with it -- you seem to be taking it in that direction already.

Nobuko, Kohei: I thought we ought to try to model use-cases in pi-calculus, with copious explanations in natural language, aiming at seeing how various concepts like role, transaction, .. would be modelled in pi. I am hoping to try this one when I get time; you might like to try too, and see if we agree!

Robin

# 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;  
}
```

# 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]



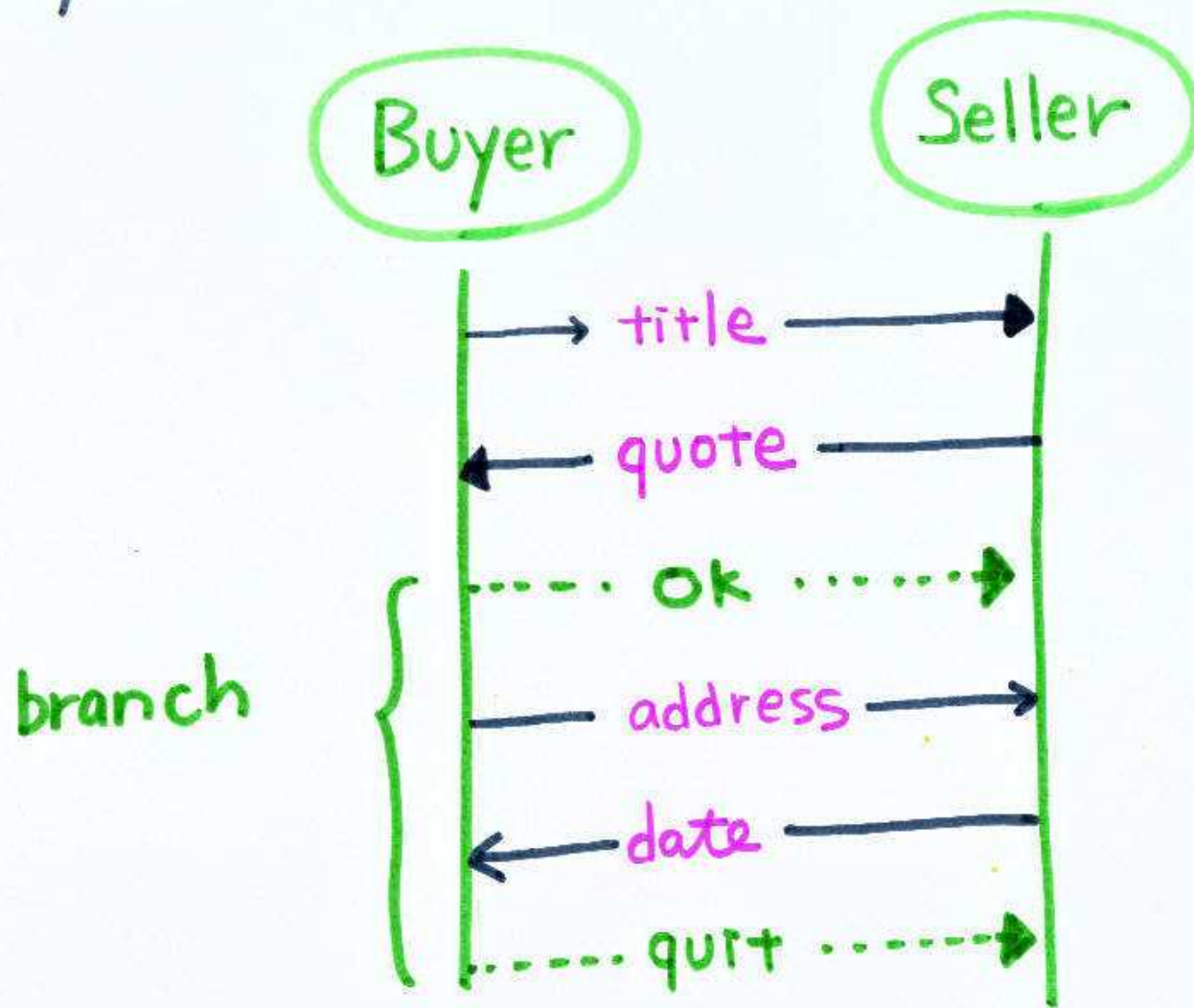
Scribble at  $\pi^4$  Technology

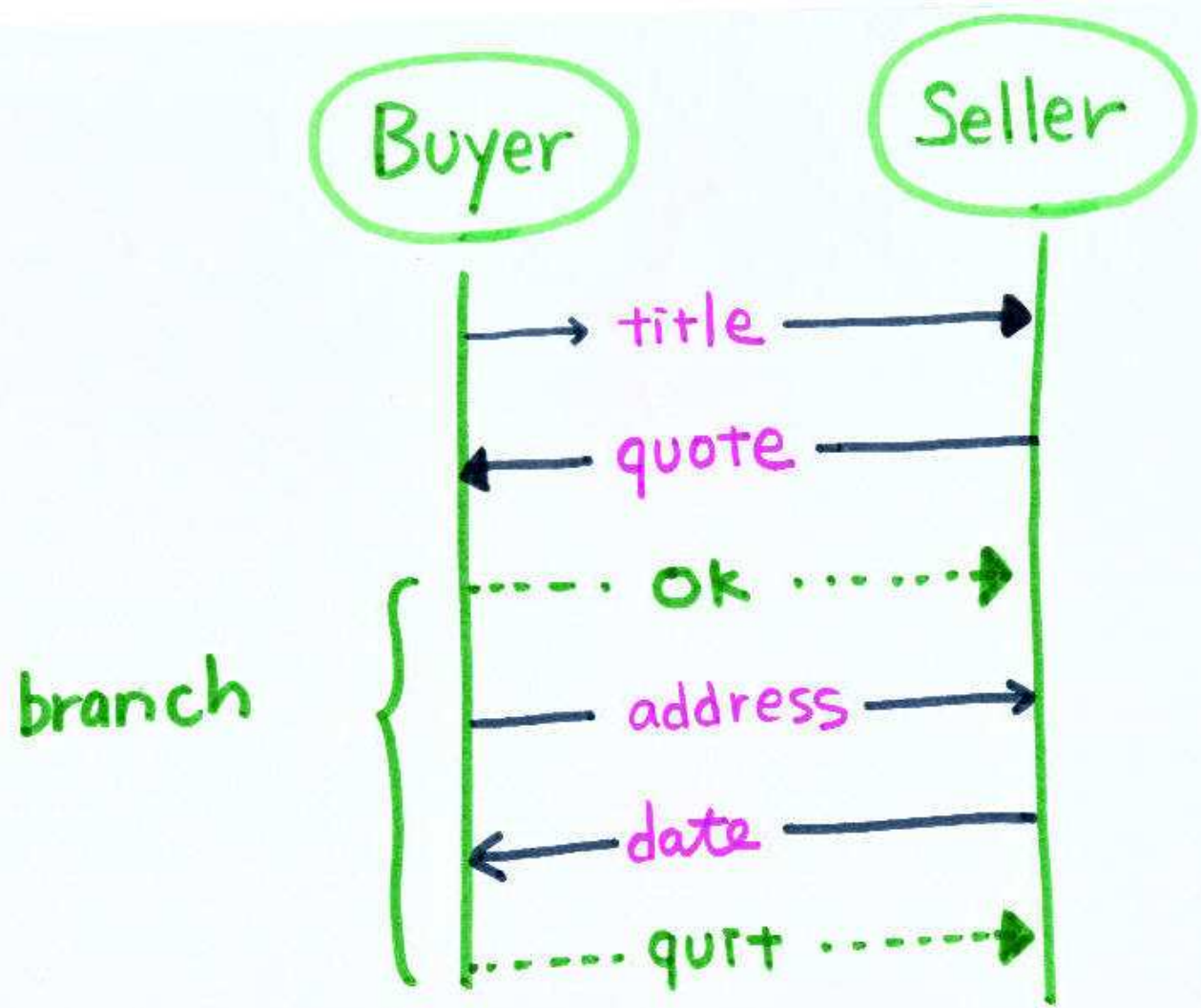


Multiparty Session Types [POPL'08]

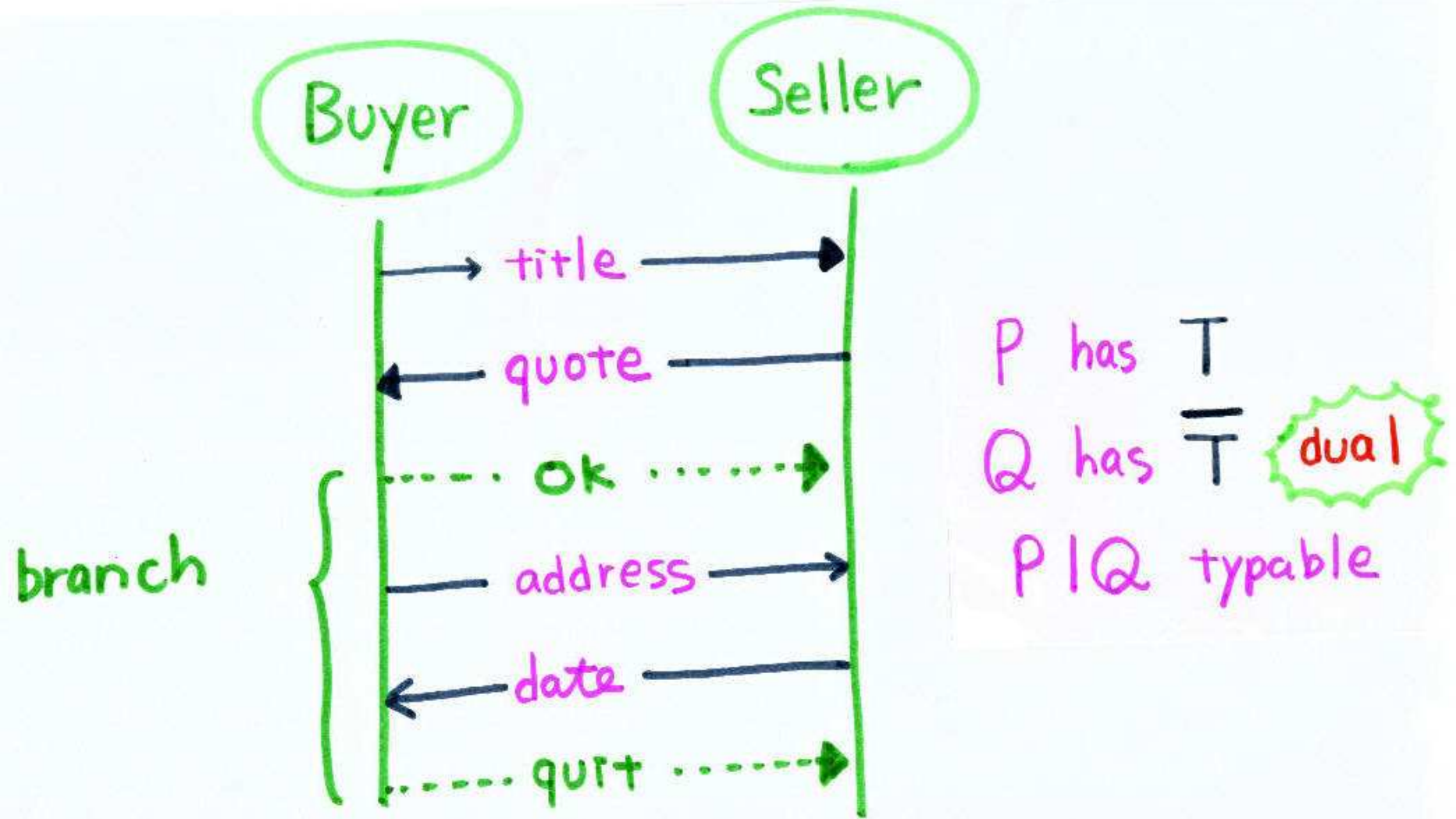


# Binary Session Types : Buyer-Seller Protocol





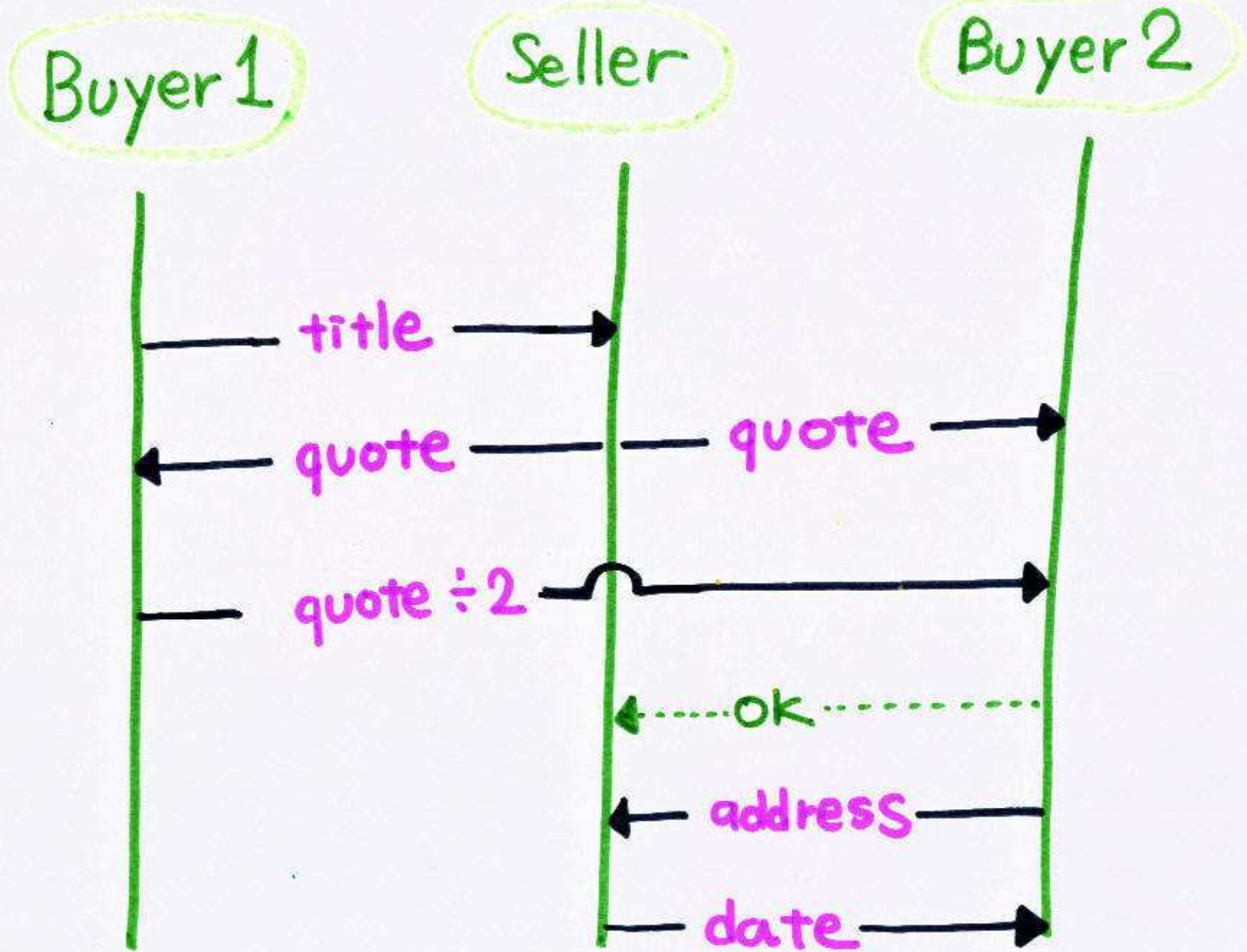
! String ; ? Int ; ⊕ { ok : !String ; ? Date ; end , quit : end }

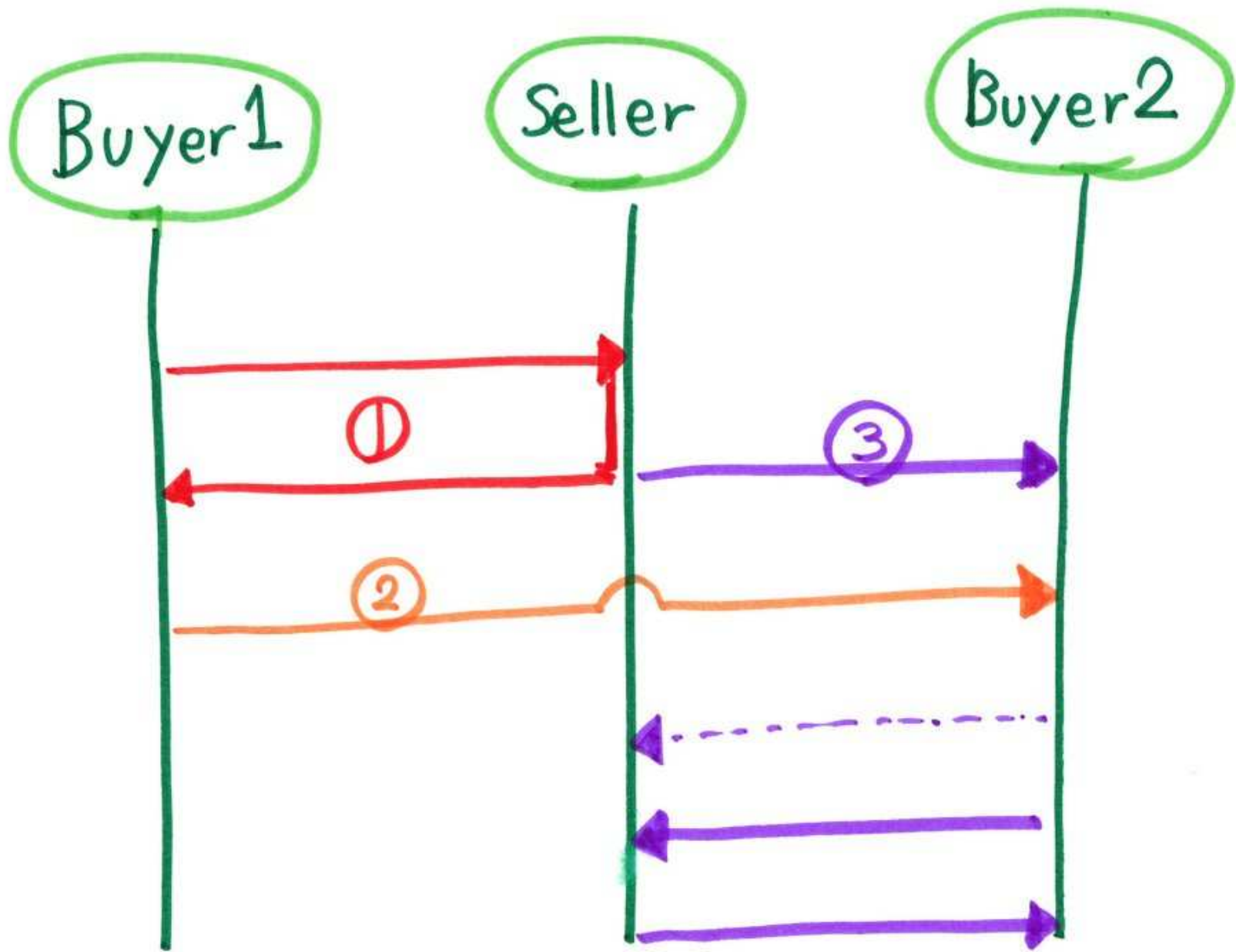


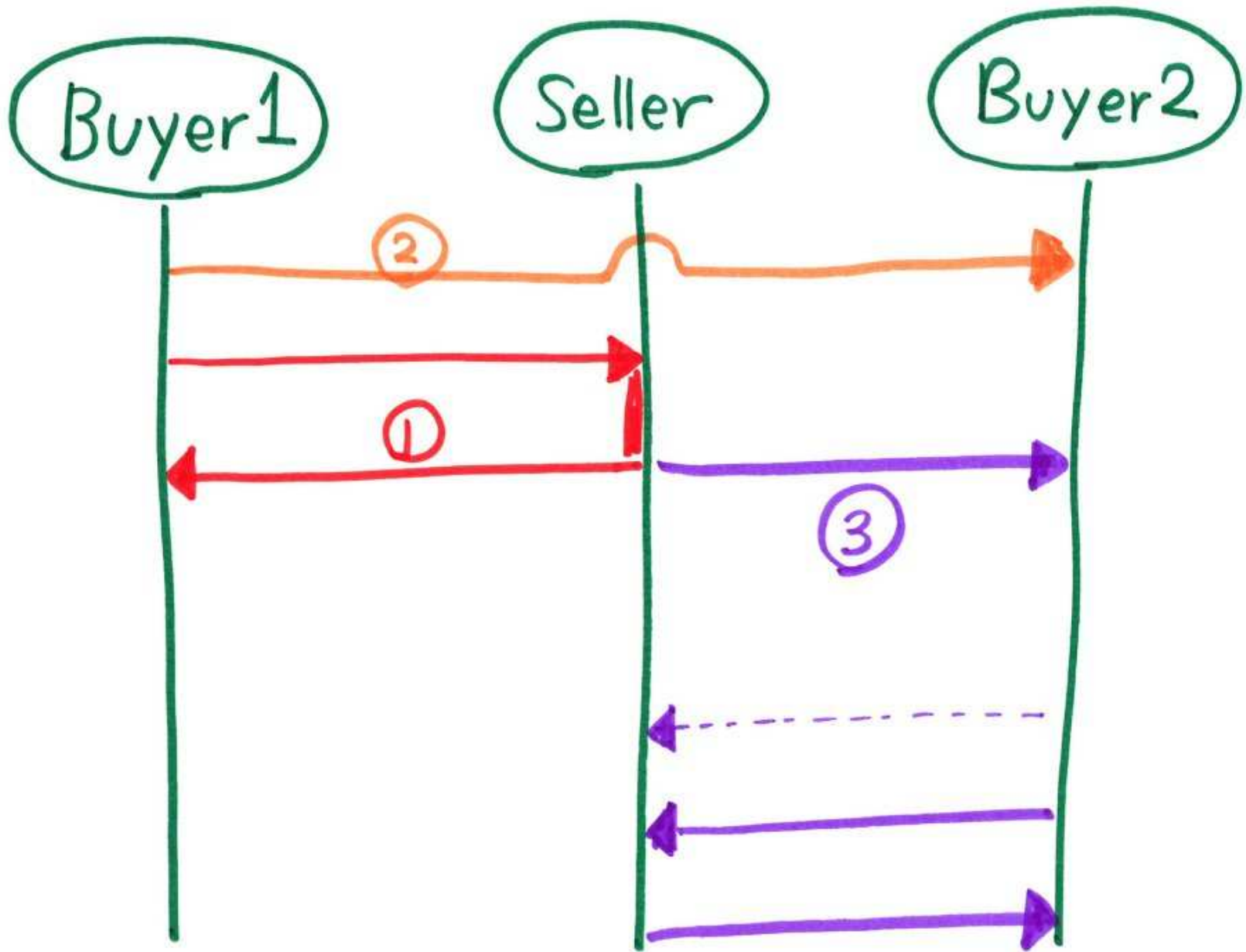
! String ; ? Int ; ⊕ { ok : !String ; ? Date ; end , quit : end }

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

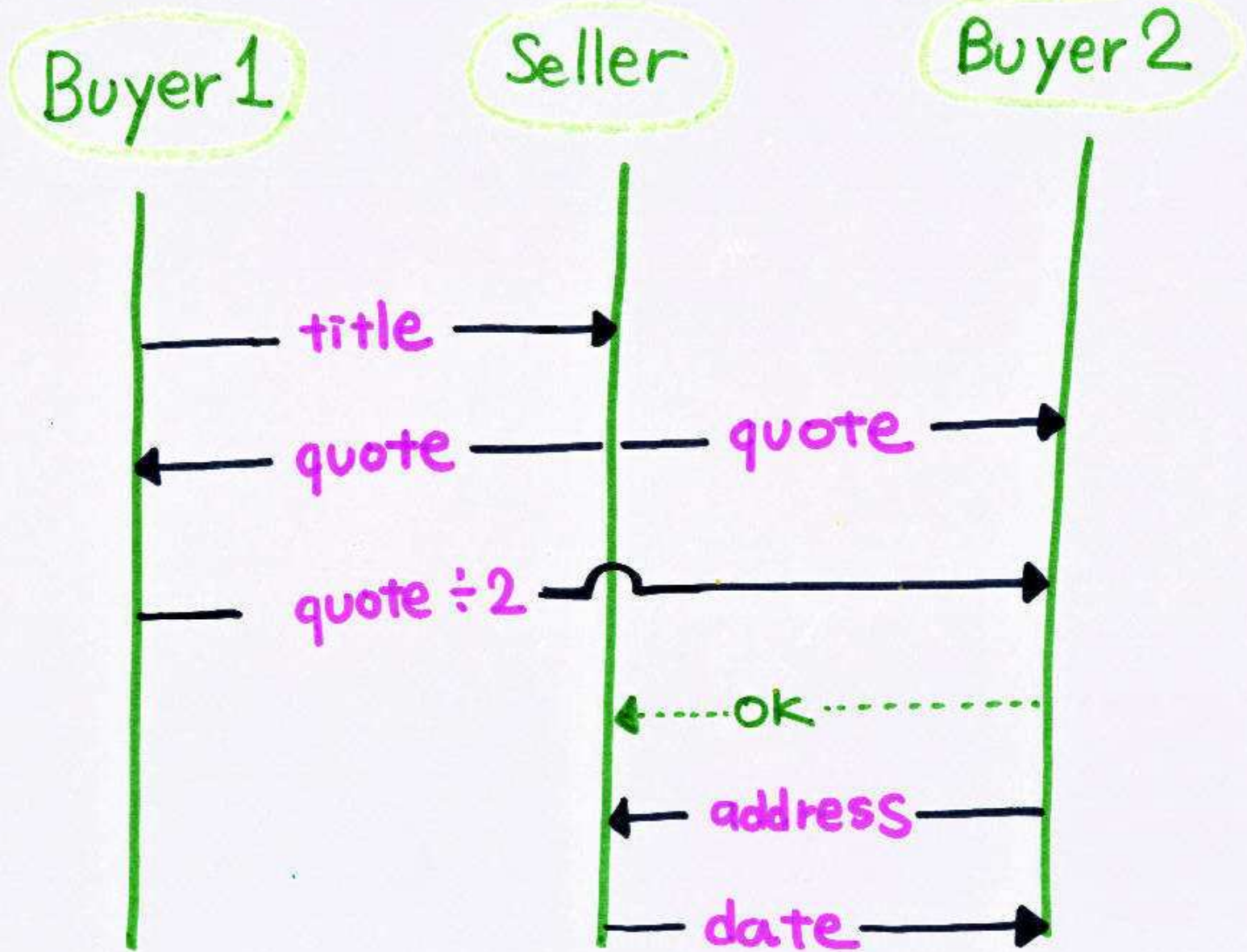
# Multiparty Session Types



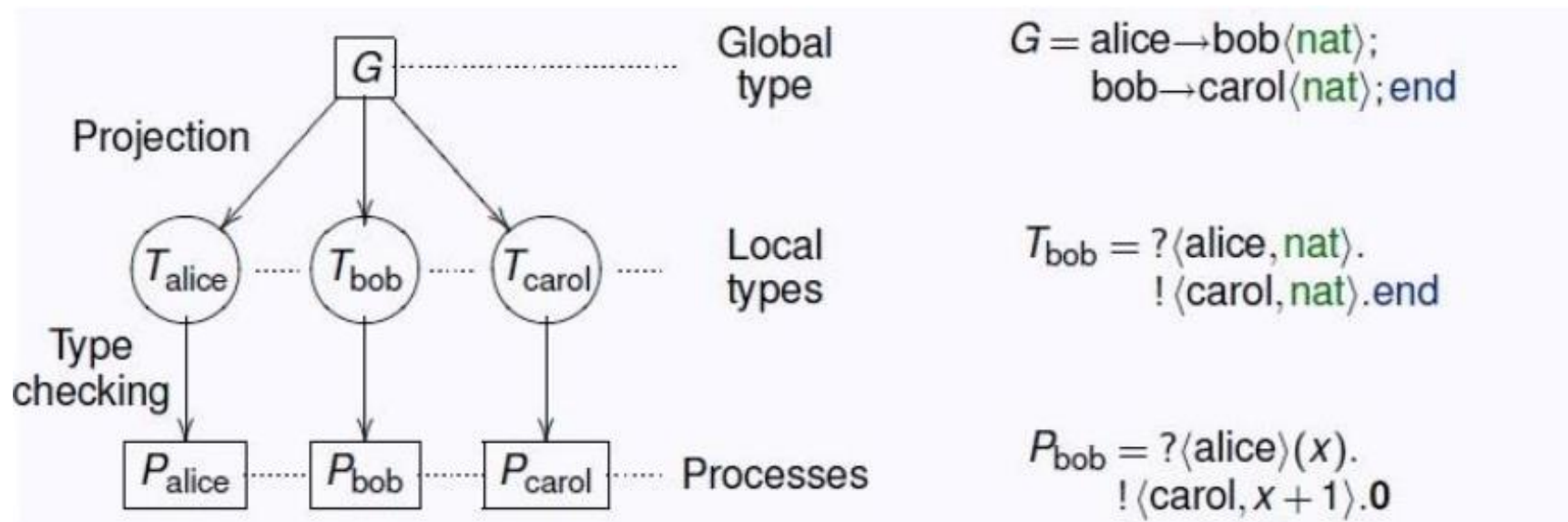




# Multiparty Session Types



# Session Types Overview



## ► Properties

- Communication safety (no communication mismatch)
- Communication fidelity (the communication follow the protocol)
- Progress (no deadlock/stuck in a session)

















# Evolution Of MPST

---

- ▶ Binary Session Types [THK98, HVK98]



- ▶ Multiparty Session Types [POPL'08]



- ▶ A Theory of Design-by-Contract for Distributed Multiparty Interactions [Concur'11]



- ▶ Multiparty Session Types Meet Communicating Automata [ESOP'12, ICALP'13]



- ▶ Network Monitoring through Multiparty Session Types [FMOODS'13]



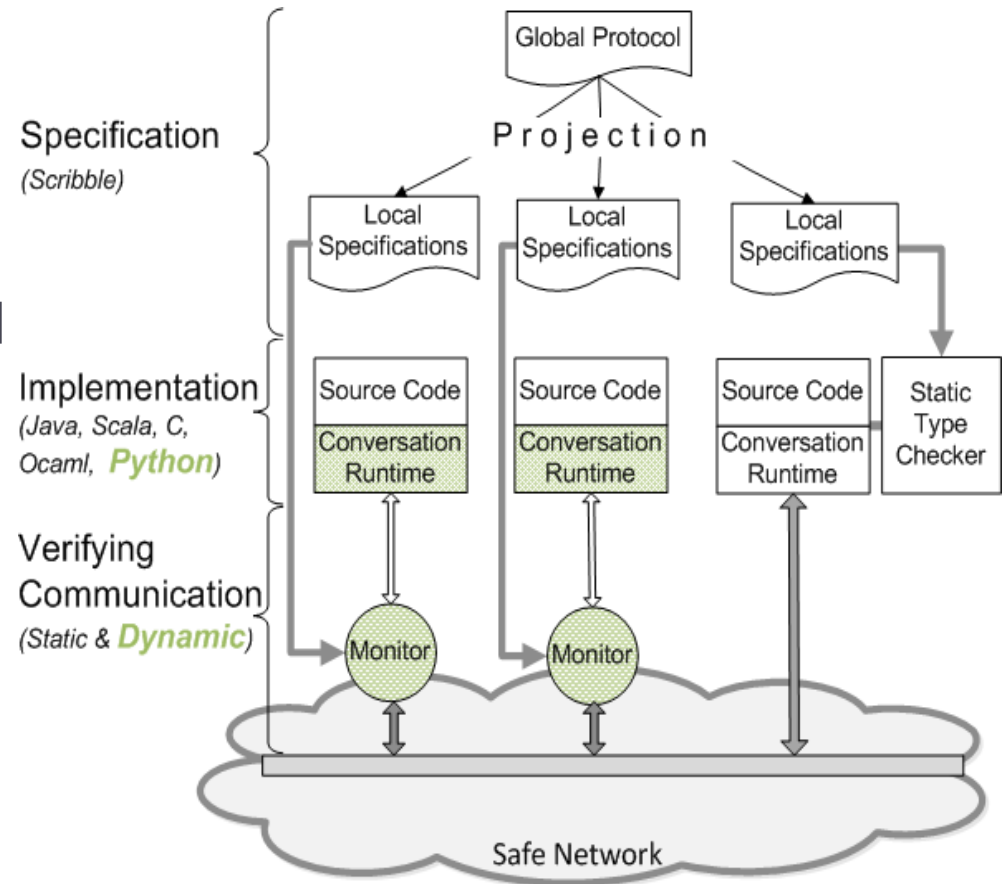
- ▶ SPY: Local Verification of Global Protocols [RV'13]
- ▶ Distributed Runtime Verification with Session Types and Python [RV'13]



# Session Types for Runtime Verification

## ▶ Methodology

- ▶ Developers design protocols in a dedicated language - Scribble
- ▶ Well-formedness is checked by Scribble tools
- ▶ Protocols are projected into local types
- ▶ Local types generate monitors





## What is Scribble?

Scribble is a language to describe application-level protocols among communicating systems. A protocol represents an agreement on how participating systems interact with each other. Without a protocol, it is hard to do meaningful interaction: participants simply cannot communicate effectively, since they do not know when to expect the other parties to send data, or whether the other party is ready to receive data.

However, having a description of a protocol has further benefits. It enables verification to ensure that the protocol can be implemented without resulting in unintended consequences, such as deadlocks.

Find out more ...

[Language Guide](#)[Tools -](#)[Specification](#)[Forum](#)

## An example

```
module examples;

global protocol HelloWorld(role Me, role World) {
  hello(Greetings) from Me to World;
  choice at World {
    hello(GoodMorning) from World to Me;
  } or {
    hello(GoodAfternoon) from World to Me;
  }
}
```

A very simply example, but this illustrates the basic syntax for a hello world interaction, where a party performing the role *Me* sends a message of type *Greetings* to another party performing the role "World", who subsequently makes a decision which determines which path of the choice will be followed, resulting in a *GoodMorning* or *GoodAfternoon* message being exchanged.

### Describe

Scribble is a language for describing multiparty protocols

### Verify

Scribble has a theoretical foundation, based on the Pi Calculus and Session Types, to ensure that protocols

### Project

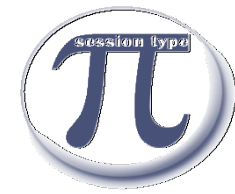
Endpoint projection is the term used for identifying the

### Implement

Various options exist, including (a) using the endpoint projection for a role to generate a skeleton code, (b)

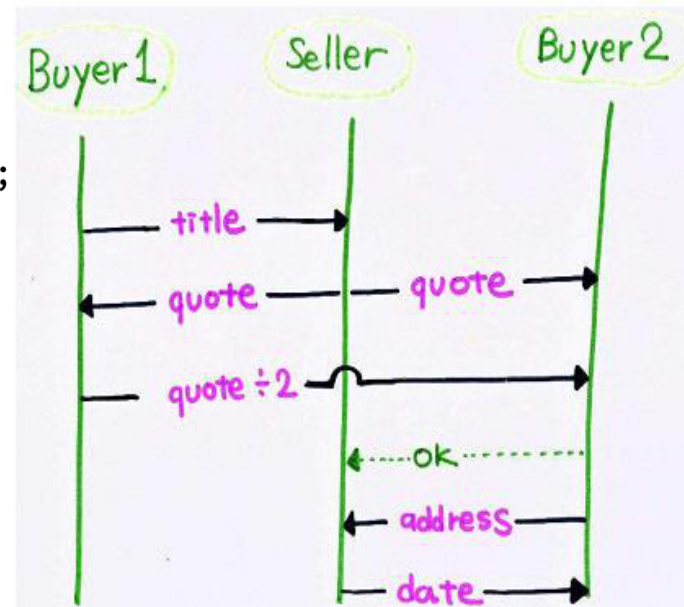
### Monitor

Use the endpoint projection for roles defined within a



# Two Buyer Protocol in Scribble

```
module Bookstore;  
  
type <java> "java.lang.Integer" from "rt.jar" as Integer;  
type <java> "java.lang.String" from "rt.jar" as String;  
  
global protocol TwoBuyers(role A, role B, role S) {  
  title(String) from A to S;  
  quote(Integer) from S to A, B;  
  rec LOOP {  
    share(Integer) from A to B;  
    choice at B {  
      accept(address:String) from B to A, S;  
      date(String) from S to B;  
    } or {  
      retry() from B to A, S;  
      continue LOOP;  
    } or {  
      quit() from B to A, S;  
    }  
  }  
}
```





# Buyer: A local projection

---

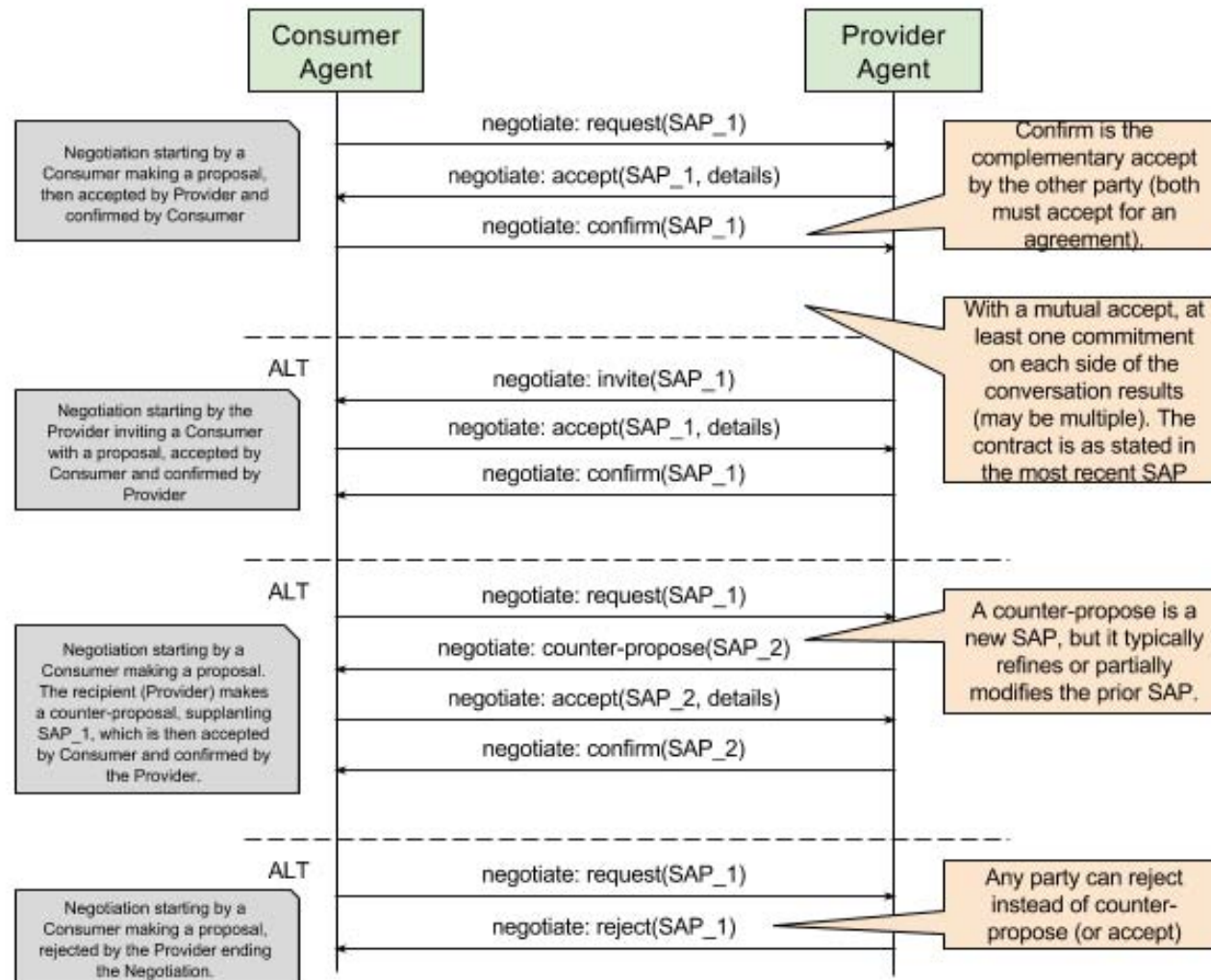
```
module Bookstore_TwoBuyers_A;

type <java> "java.lang.Integer" from "rt.jar" as Integer;
type <java> "java.lang.String" from "rt.jar" as String;

local protocol TwoBuyers_A at A(role A, role B, role S) {
  title(String) to S;
  quote(Integer) from S;
  rec LOOP {
    share(Integer) to B;
    choice at B {
      accept(address:String) from B;
    } or {
      retry() from B;
      continue LOOP;
    } or {
      quit() from B;
    }
  }
}
```



# OOI agent negotiation 1/5

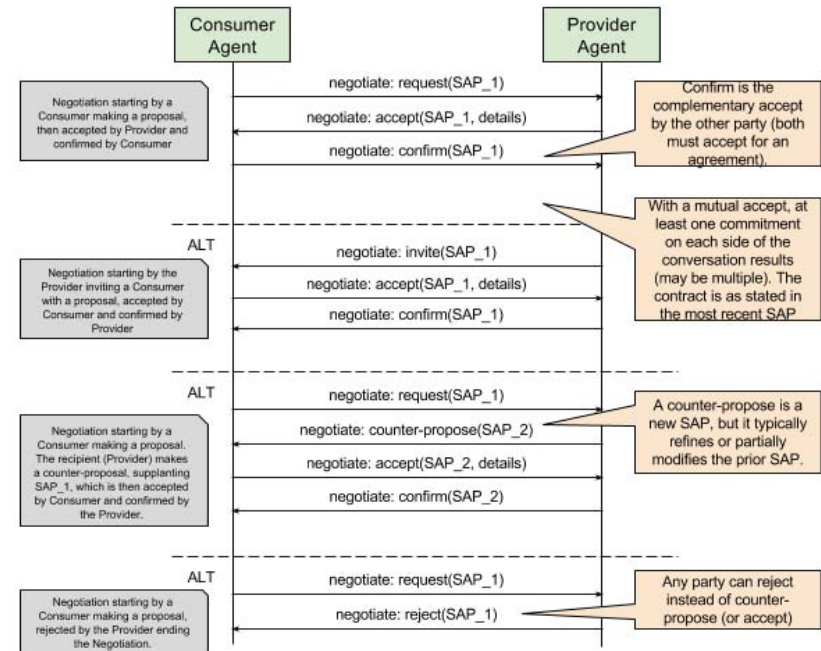


- ▶ <https://confluence.oceanobservatories.org/display/syseng/CIAD+COI+OV+Negotiate+Protocol>

# OOI agent negotiation 2/5

```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;
```

```
global protocol Negotiate(role Consumer as C, role Producer as P) {
```



```
}
```

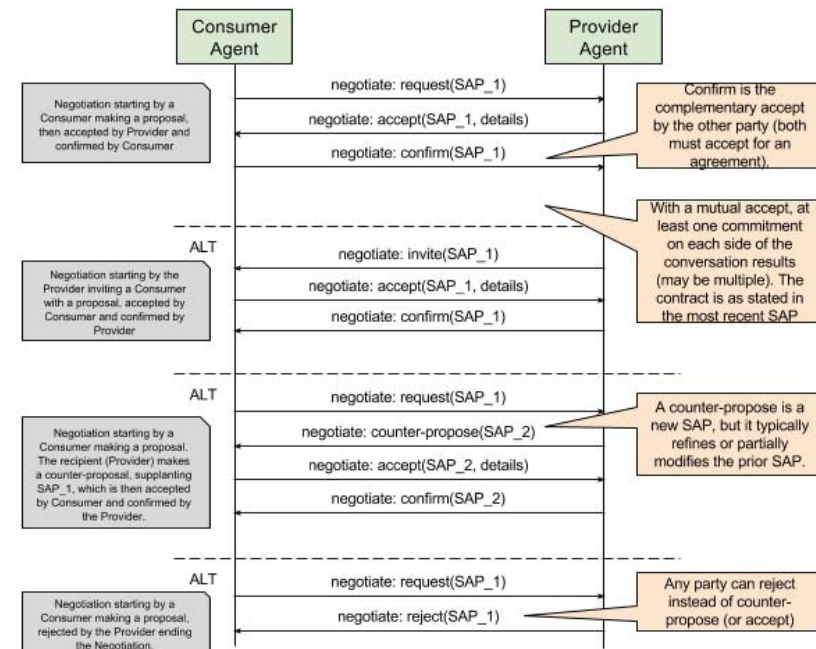
# OOI agent negotiation 3/5 (choice)

```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;
```

```
global protocol Negotiate(role Consumer as C, role Producer as P) {  
  propose(SAP) from C to P;
```

```
  choice at P {  
    accept() from P to C;  
    confirm() from C to P;  
  } or {  
    reject() from P to C;  
  } or {  
    propose(SAP) from P to C;
```

```
  } }  
}
```

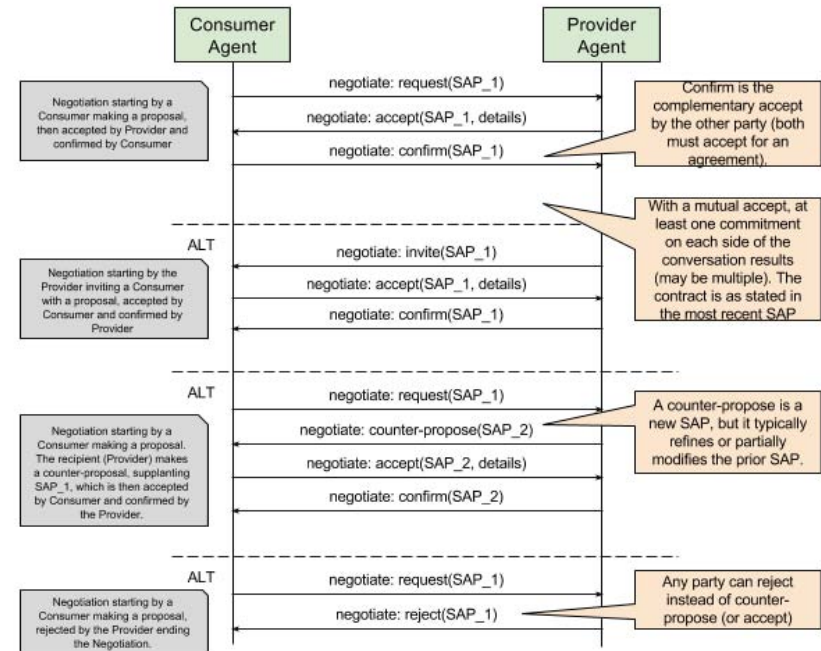


# OOI agent negotiation 4/5

```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;
```

```
global protocol Negotiate(role Consumer as C, role Producer as P) {
  propose(SAP) from C to P;
```

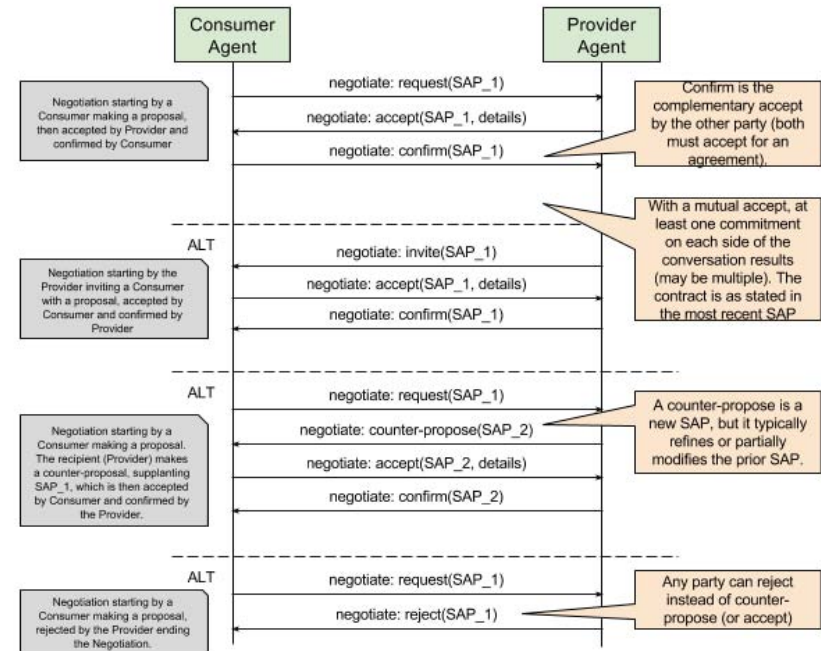
```
  choice at P {
    accept() from P to C;
    confirm() from C to P;
  } or {
    reject() from P to C;
  } or {
    propose(SAP) from P to C;
    choice at C {
      accept() from C to P;
      confirm() from P to C;
    } or {
      reject() from C to P;
    } or {
      propose(SAP) from C to P;
    }
  }
}
```



# OOI agent negotiation 5/5 (recursion)

```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;
```

```
global protocol Negotiate(role Consumer as C, role Producer as P) {
  propose(SAP) from C to P;
  rec X {
    choice at P {
      accept() from P to C;
      confirm() from C to P;
    } or {
      reject() from P to C;
    } or {
      propose(SAP) from P to C;
      choice at C {
        accept() from C to P;
        confirm() from P to C;
      } or {
        reject() from C to P;
      } or {
        propose(SAP) from C to P;
        continue X;
      }
    }
  }
}
```

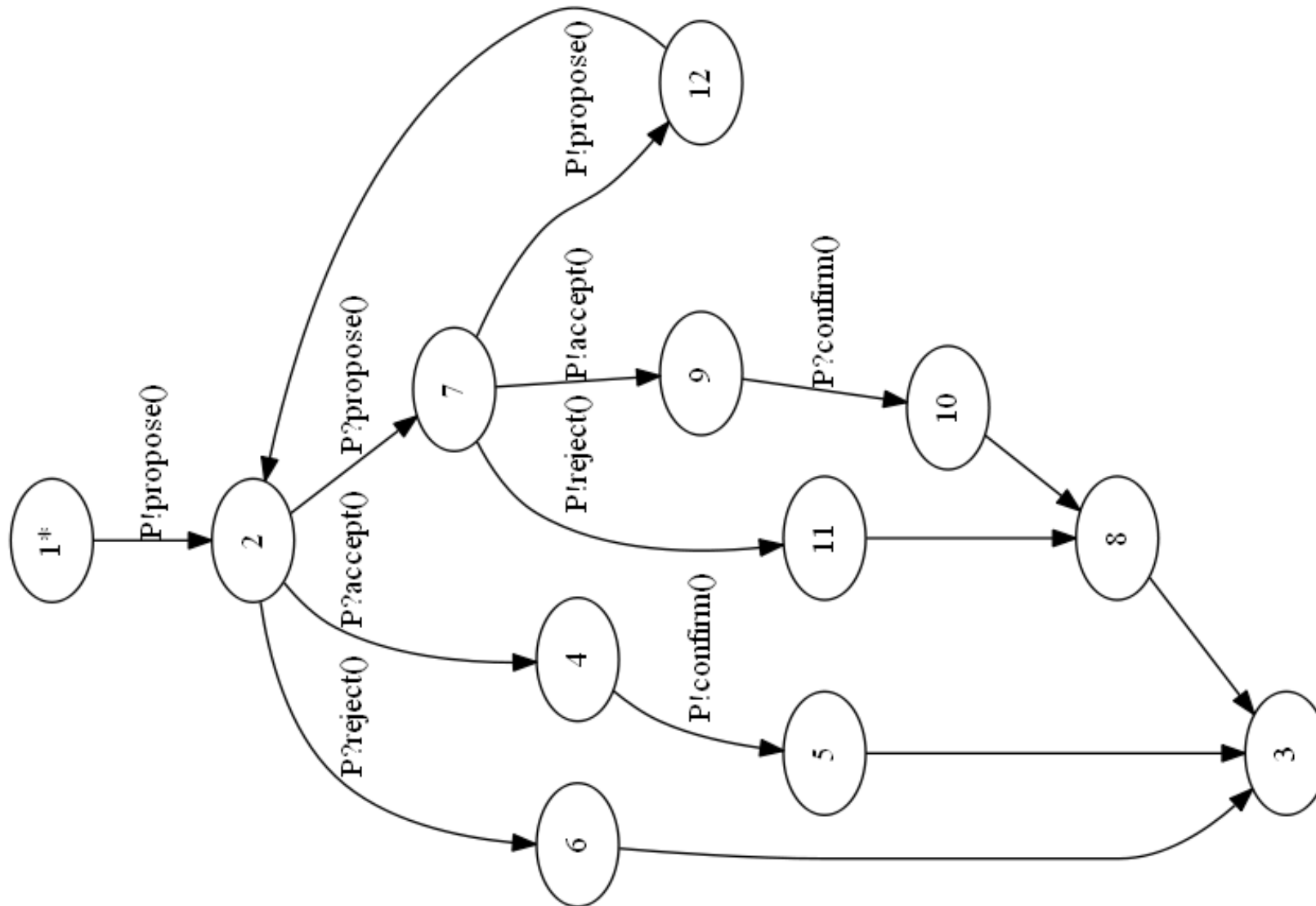


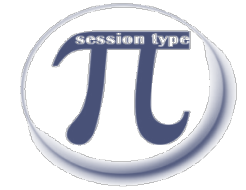
# Local protocol projection (Negotiation Consumer)

```
// Global
propose(SAP) from C to P;
rec START {
  choice at P {
    accept() from P to C;
    confirm() from C to P;
  } or {
    reject() from P to C;
  } or {
    propose(SAP) from P to C;
    choice at C {
      accept() from C to P;
      confirm() from P to C;
    } or {
      reject() from C to P;
    } or {
      propose(SAP) from C to P;
      continue START;
    }
  }
}
```

```
// Projection for Consumer
propose(SAP) to P;
rec START {
  choice at P {
    accept() from P;
    confirm() to P;
  } or {
    reject() from P;
  } or {
    propose(SAP) from P;
    choice at C {
      accept() to P;
      confirm() from P;
    } or {
      reject() to P;
    } or {
      propose(SAP) to P;
      continue START;
    }
  }
}
```

# FSM generation (Negotiation Consumer)





# Scribble Community

---

- ▶ **Webpage:**

- ▶ [www.scribble.org](http://www.scribble.org)

- ▶ **GitHub:**

- ▶ <https://github.com/scribble>

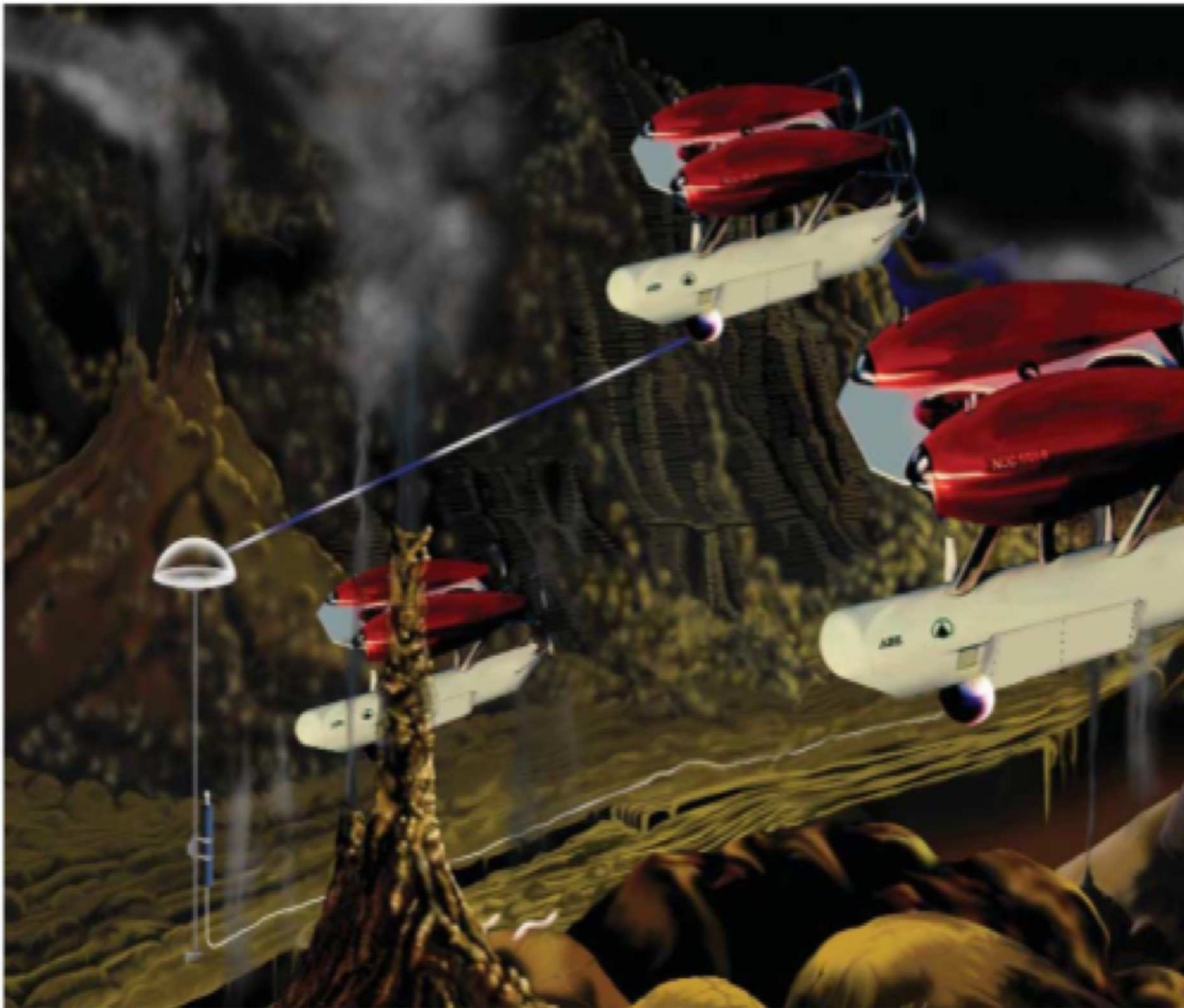
- ▶ **Tutorial:**

- ▶ [www.doc.ic.ac.uk/~rhu/scribble/tutorial.html](http://www.doc.ic.ac.uk/~rhu/scribble/tutorial.html)

- ▶ **Specification (0.3)**

- ▶ [www.doc.ic.ac.uk/~rhu/scribble/langref.html](http://www.doc.ic.ac.uk/~rhu/scribble/langref.html)





**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**



SEARCH

RESOURCES

- All Resources
- Data Products
- Observatories
- Platforms
- Instruments

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.

Location

CURRENT LOCATION

FILTER



DATA LEGEND

- Temperature
- Salinity
- Oxygen
- Density
- Currents
- Sea Surface Height (SSH)
- Chlorophyll
- Turbidity
- pH
- Seismology
- Other

REQUENCY

- 1 Hour
- 2 hours
- 3 hours
- 5 hours
- 8 hours
- 12 hours
- 18 hours
- 24 hours
- 48 Hours
- 72 Hours

RECENT UPDATES

NAME	DATE	TYPE	EVENT	DESCRIPTION	NOTE
01 m Oregon Coast North Salinity	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 m California South 100m pH	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 m California South salinity	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
03 m Oregon North Turbidity	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
05 m Oregon South Temperature	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
20 m Oregon Coast Currents	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 h California South Seismology	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
01 h Oregon Coast South 1000m Ox	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
02 h California Coast Seismology	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here
04 h California North Seismology	2012-01-10 23:55:55	Type	Event	Description goes here	Note goes here

Dashboard

RECENT IMAGES

- Glider**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24
- Gorgonian Coral**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24
- Acoustic Release**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24

POPULAR RESOURCES

- SeaBird CDT**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24
- Marine caption**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24
- Surface Buoy**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24

UNUSUAL EVENTS

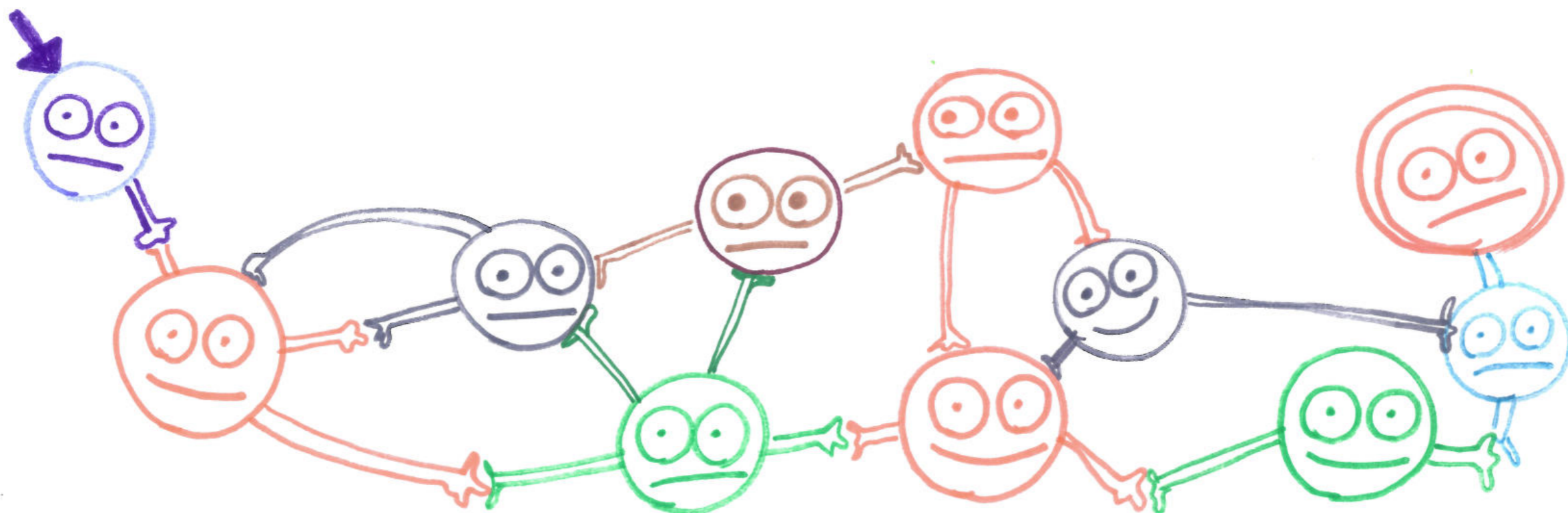
- Oregon Coast Wave Height**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24
- Water Surface Elevation**  
Last Modified: 2011-06-15  
Last Viewed: 2011-12-15  
Last Updated: 2011-12-30, 13.24

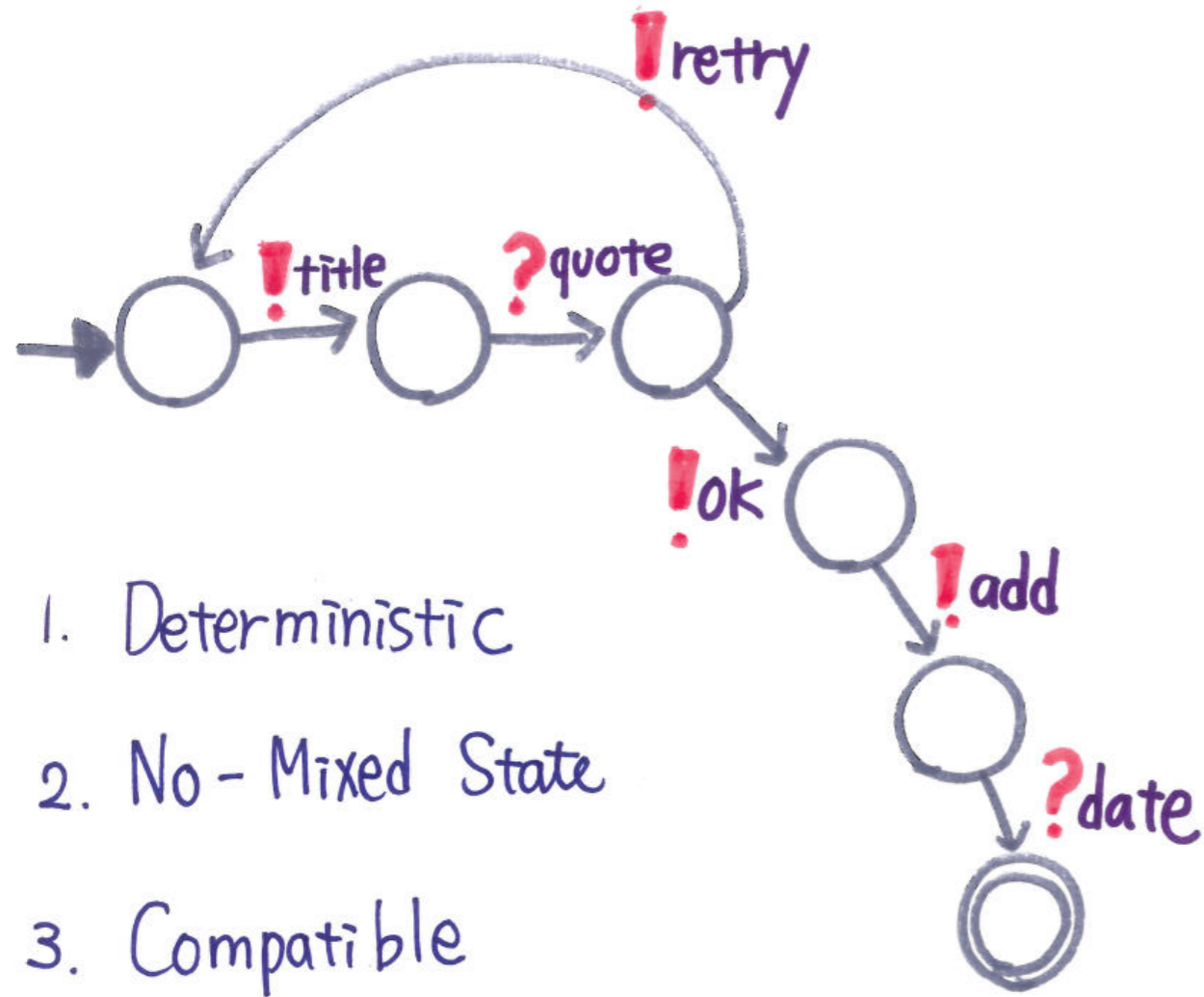
# Multiparty Compatibility in Communicating Automata

## Synthesis and Characterisation of Multiparty Session Types

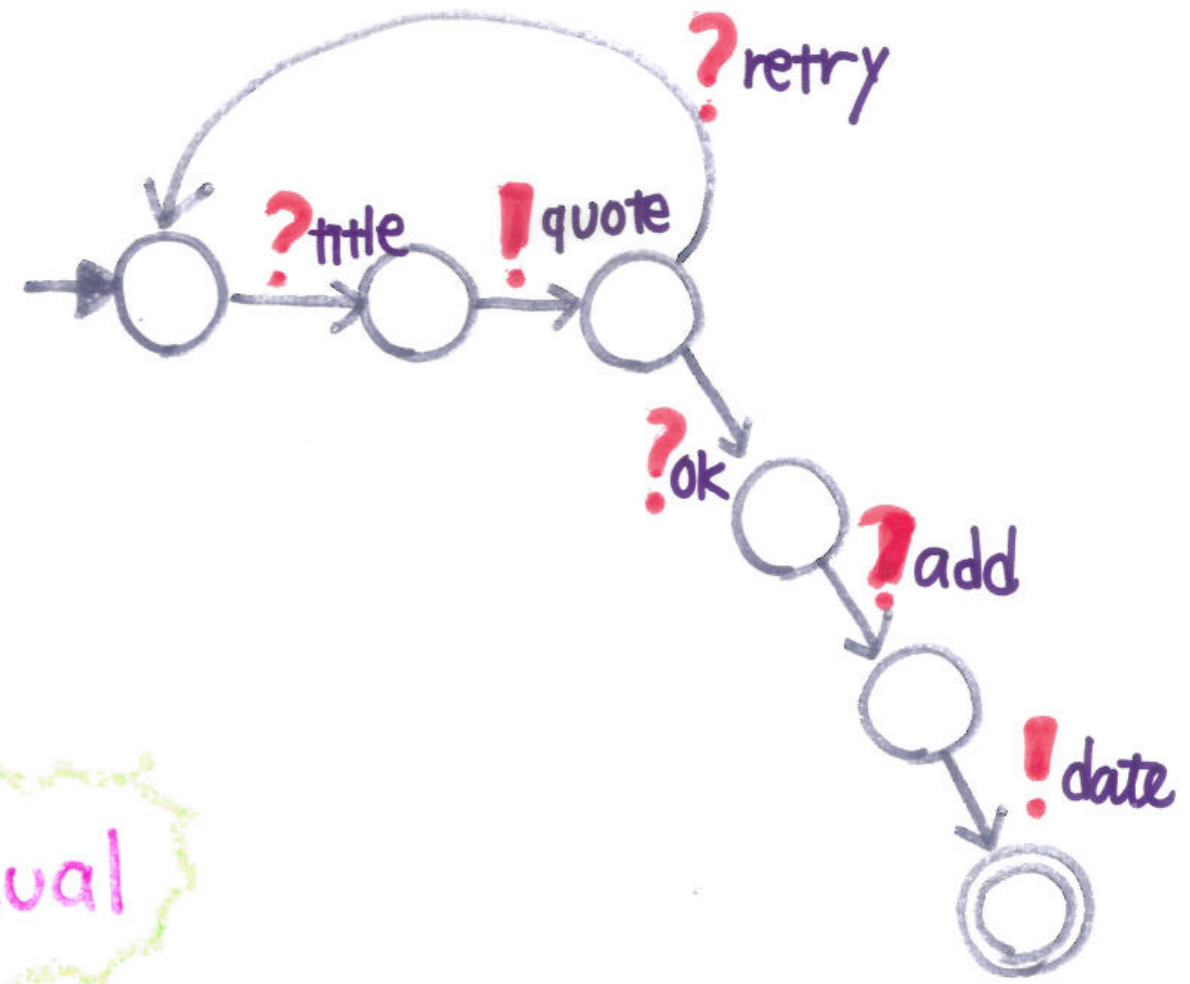
Nobuko Yoshida

Pierre-Malo Denielóu **ICALP'13**





1. Deterministic
2. No-Mixed State
3. Compatible



dual

**[Gouda et al 1986]** Two compatible machines without mixed states which are deterministic satisfy deadlock-freedom.

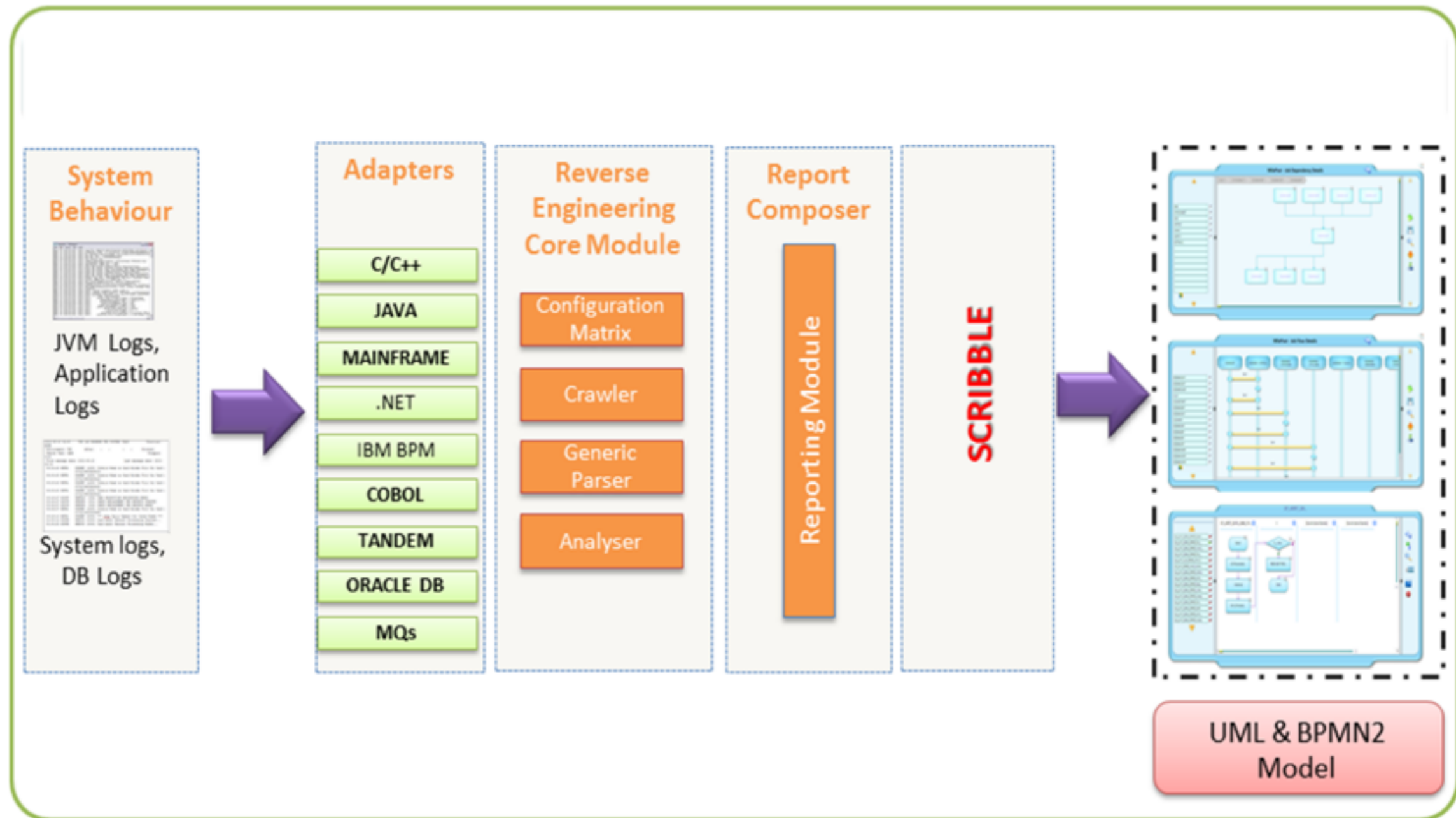
<http://www.zdlc.co/faq/>



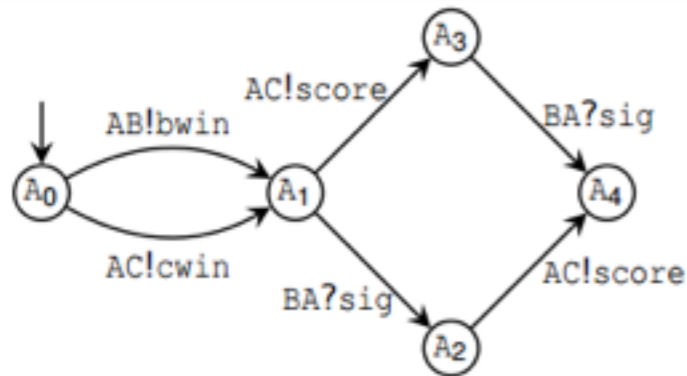
[Home](#) [ZDLC Solutions](#) [FAQ](#) [Resources](#) [Events](#) [Blog](#) [Contact](#) [Partners](#) [Cognizant](#)



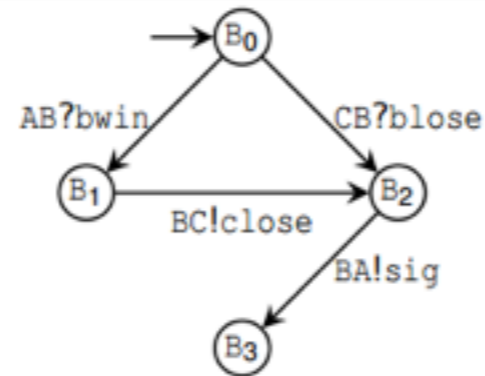
# Zero Deviation Life Cycle Platform



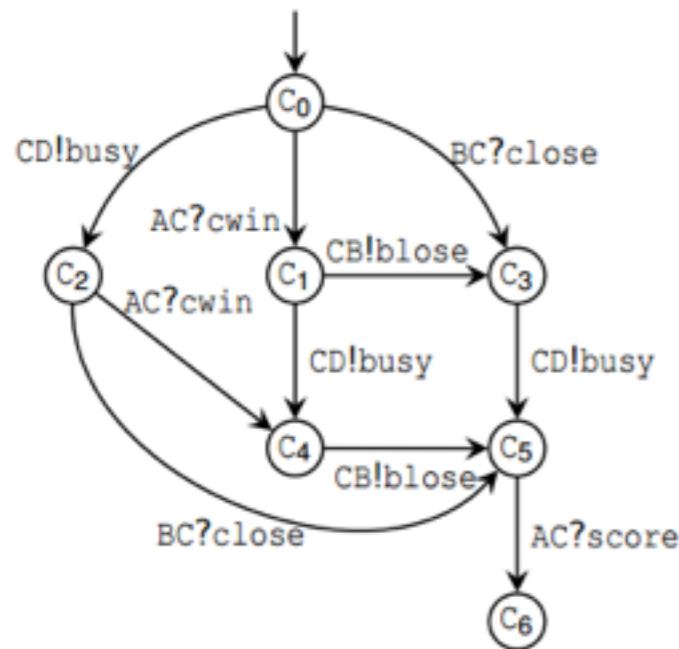
# From Communicating Machines to Graphical Choreographies [POPL'15]



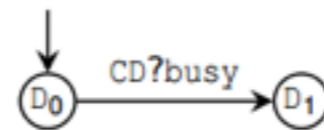
Alice



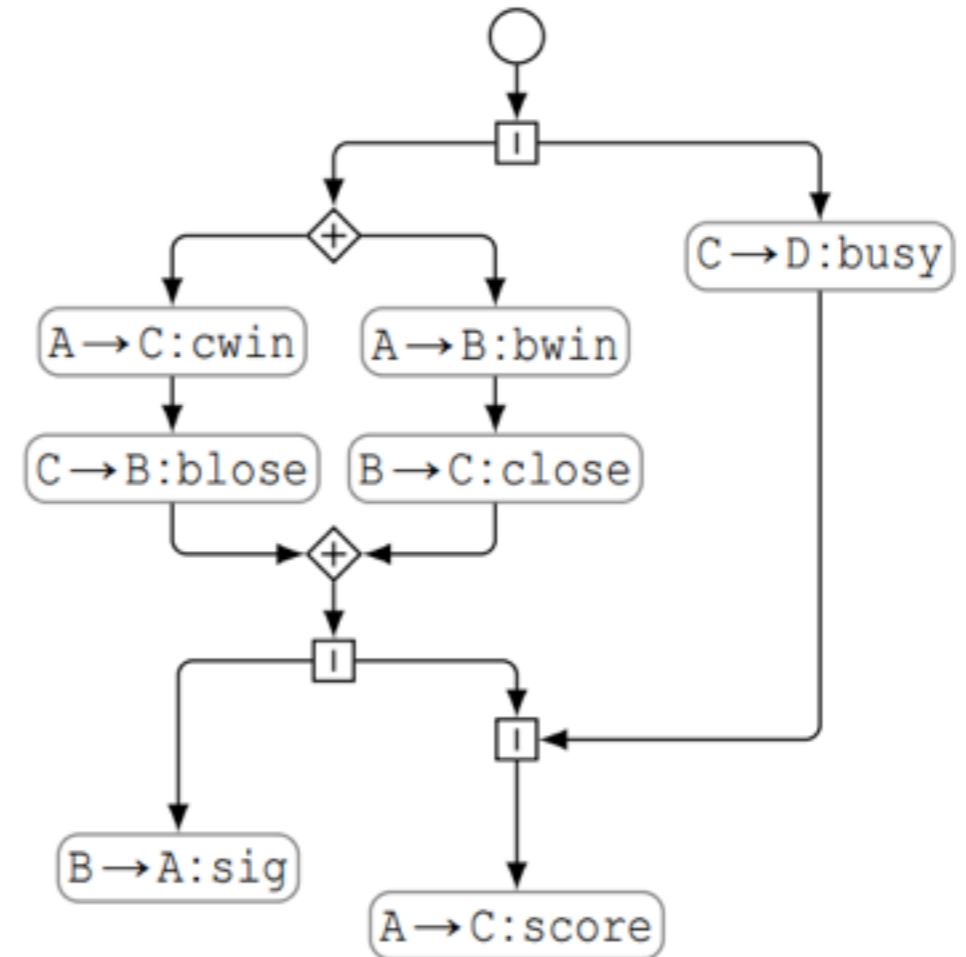
Bob



Carol



Dave

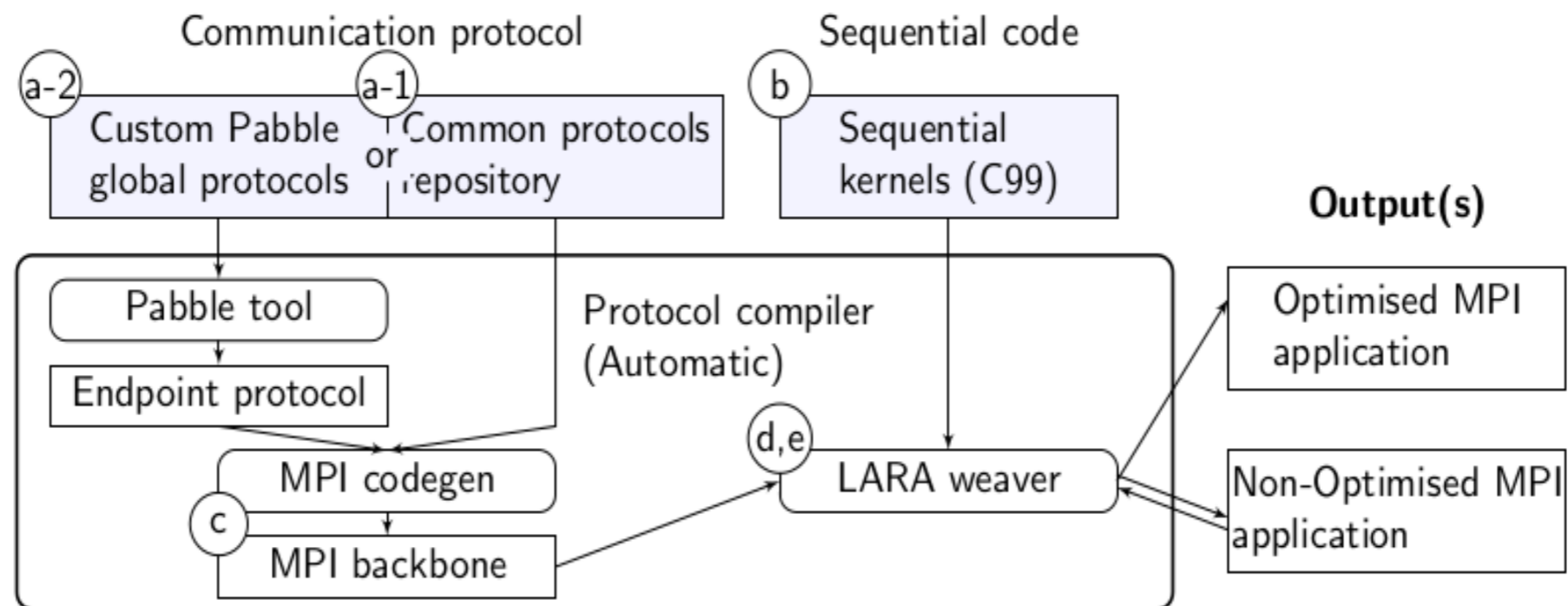


► [ESOP'10, ESOP'12, CONCUR'12, CONCUR'14]

# Message Passing Algorithms [CC'15]

A complete parallel programming workflow

- Captures **parallel interaction patterns** by Pabble language
- Combines with **sequential computation kernels** in C
- Generates **communication safe & deadlock free** MPI programs
- Optimisation as part of merging technique



# Example: Simple search engine

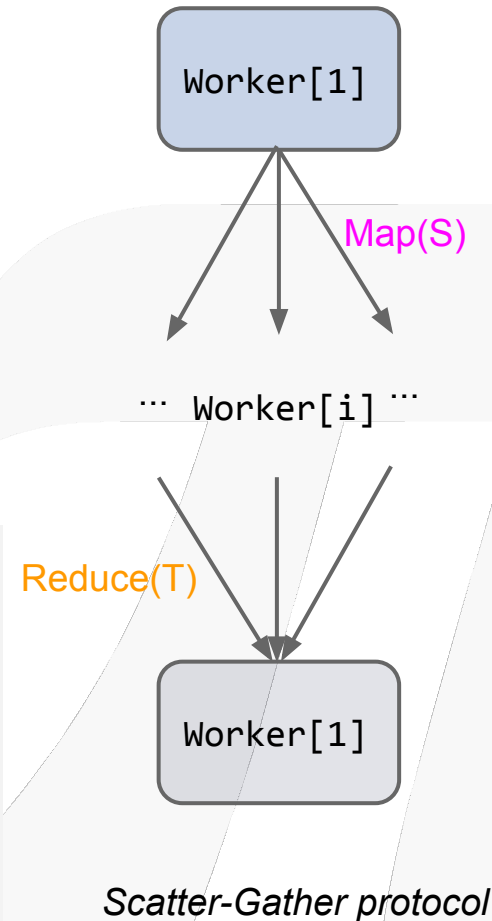
## Scatter-Gather protocol

- Distribute query to all nodes
- Nodes collect relevant records
- Results gathered and merged
- Display results to user

```
const N = 2..max;
global protocol ScatterGather(role Worker[1..N])
{
  Init()    from __self    to __self;

  Map(S)    from Worker[1] to __All;
  Reduce(T) from __All     to Worker[1];

  Finish()  from __self    to __self;
}
```



# Example: search engine

## Merging backbone with kernels

```
#pragma pabble type S
  typedef void S; MPI_Datatype MPI_S = MPI_BYTE;
#pragma pabble type T
  typedef void T; MPI_Datatype MPI_T = MPI_BYTE;
#pragma pabble Init
  bufMap0_r = calloc(meta.buflen(Map), sizeof(S));
#pragma pabble Map
  bufMap0_s = pabble_sendq_dequeue();
  MPI_Scatter( ..., MPI_S, Worker_RANK(1), ... );
  free(bufMap0_s);
  pabble_recvq_enqueue(Map, bufMap0_r);
#pragma pabble Map
  bufReduce1_r = calloc(meta.buflen(Reduce)*meta.
nprocs, sizeof(T));
#pragma pabble Reduce
  bufReduce1_s = pabble_sendq_dequeue();
  MPI_Gather( ... , MPI_T, Worker_RANK(1) ... );
  free(bufReduce1_s);
  pabble_recvq_enqueue(Reduce, bufReduce1_r);
#pragma pabble Reduce
#pragma pabble Finish
```

```
typedef char *S; MPI_Datatype MPI_S = MPI_CHAR;

typedef char *T; MPI_Datatype MPI_T = MPI_CHAR;
load_data();
bufMap0_r = calloc(meta.buflen(Map), sizeof(S));
distribute_data();
bufMap0_s = pabble_sendq_dequeue();
MPI_Scatter( ..., MPI_S, Worker_RANK(1), ... );
free(bufMap0_s);
pabble_recvq_enqueue(Map, bufMap0_r);
distribute_data();
bufReduce1_r = calloc(meta.buflen(Reduce)*meta.
nprocs, sizeof(T));
collect_records();
bufReduce1_s = pabble_sendq_dequeue();
MPI_Gather( ... , MPI_T, Worker_RANK(1) ... );
free(bufReduce1_s);
pabble_recvq_enqueue(Reduce, bufReduce1_r);
collect_records();
display_cleanup();
```

# Evaluation

Productivity: Flexibility

Reusable protocols

- e.g. scatter-gather
- e.g. stencil

Berkeley Dwarfs [CACM'09]

- Representative parallel computing patterns
- 4 of 5 HPC patterns

		Repository	Berkeley HPC Dwarfs
<b>heateq</b>	stencil*	Yes	Structured Grid
<b>nbody</b>	ring*	Yes	Particle Methods
<b>wordcount</b>	scatter-gather*	Yes	
<b>adpredictor</b>	scatter-gather*	Yes	
<b>montecarlo</b>	scatter-gather*	Yes	
<b>montecarlo-mw</b>	master-worker*	Yes	
<b>LESolver</b>	wraparound mesh		Structured Grid
<b>matvec</b>	custom		Dense Matrix
<b>fft64</b>	6-step butterfly		Spectral (FFT)

# Evaluation

## Productivity: Effort

### Protocols in repository

- Use backbone directly
- Write kernel
- Effort =  $K / B+K$

### Custom protocols

- Write Pabble protocol
- Tool generate backbone
- Write kernel
- Effort =  $P+K / B+K$

		Pabble LOC(P)	Backbone LOC (B)	Kernel LOC(K)	Effort
<b>heateq</b>	stencil*	15	154	335	0.69
<b>nbody</b>	ring*	15	93	228	0.71
<b>wordcount</b>	scatter-gather*	8	76	176	0.70
<b>adpredictor</b>	scatter-gather*	8	76	182	0.71
<b>montecarlo</b>	scatter-gather*	8	76	70	0.48
<b>montecarlo-mw</b>	master-worker*	10	82	70	0.46
<b>LEsolver</b>	wraparound mesh	15	132	208	0.66
<b>matvec</b>	custom	15	130	117	0.41
<b>fft64</b>	6-step butterfly	11	64	134	0.68

## Effort ratio

 LOC savings

# Language and Implementations

- ▶ **Carrying out large-scale experiences** with OOI, VMWare, Red Hat, Cognizant, UNIFI
  - JBoss **SCRIBBLE** [ICDCIT'10, COB'12, TGC'13] and **SAVARA** projects
- ▶ **High-performance computing**
  - Session Java [ECOOP'08, ECOOP'10, Coordination'11]
  - ⇒ Multiparty Session C and MPI
  - [TOOLS'12, Hearts'12, EuroMPI'12, PDP'14, CC'15]
- ▶ **Multiparty session languages** Ocaml, Java, C, MPI, Python, Scala, Jolie, Haskell, Erlang
  - Trustworthy Pervasive Healthcare Services [FHIES'12]
  - SPY: Local Verification of Global Protocols [RV'13, FMCD'15]
  - Practical interruptible conversations: Distributed dynamic verification with session types and Python [RV'13]
  - Multiparty Session Actors [COORDINATION'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, Info.&Comp]
  - Buffered Communication Analysis in Distributed Multiparty Sessions [CONCUR'10]

- ▶ 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]
  - Meeting Deadlines Together [CONCUR'15]
- ▶ Petri Nets
  - Multiparty Session Nets [TGC'14]
- ▶ Typed Behavioural Theories
  - On Asynchronous Eventful Session Semantics [FORTE'11]  
[Math. Struct. Comp. Sci.]
  - Governed Session Semantics [CONCUR'13]
  - Characteristic Bisimulations for Higher-Order Session Processes  
[CONCUR'15]
- ▶ Choreography Languages
  - Compositional Choreographies [CONCUR'13]

▶ Logics

- ▶ Design-by-Contract for Distributed Multiparty Interactions [[CONCUR'10](#)]
- ▶ Specifying Stateful Asynchronous Properties [[CONCUR'12](#)]
- ▶ Multiparty, Multi-session Logic [[TGC'12](#)]
- ▶ Multiparty Session Types as Coherence Proofs [[CONCUR'15](#)]

## 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.

## A rare cluster of qualities

From the team of OOI CI:

*Kohei has lead us deep into the nature of communication and processing. His esthetics, precision and enthusiasm for our mutual pursuit of formal Session (Conversation) Types and specifically for our OOI collaboration to realize this vision in very concrete terms were, as penned by Henry James, lessons in seeing the nuances of both beauty and craft, through a rare cluster of qualities - curiosity, patience and perception; all at the perfect pitch of passion and expression.*