Open Problems and State-of-Art of Session Types

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

Nobuko Yoshida
Imperial College London

Questions on Mobile Processes

- How to apply mobile process theories to real distributed and concurrent applications and programming languages?
- Common Questions on Session (Protocol) Types
  - I wish to extend (Multiparty) Session Types to XXX.
  - I wish to learn (Multiparty) Session Types. What is the best paper?
  - (Multiparty) Session Types can verify/specify XXX?
  - What is a relationship between (Multiparty) Session Types and MSCs/Petri Nets/State Machines/...?

Usage on Multiparty Session Types

- Static Type Checking via End Point Projection

  $\Longrightarrow$ Java, C, Haskell, Ocaml

- Dynamic Type Checking

  $\Longrightarrow$ Runtime Monitoring, Python, Erlang, Java, ...

- Synthesis

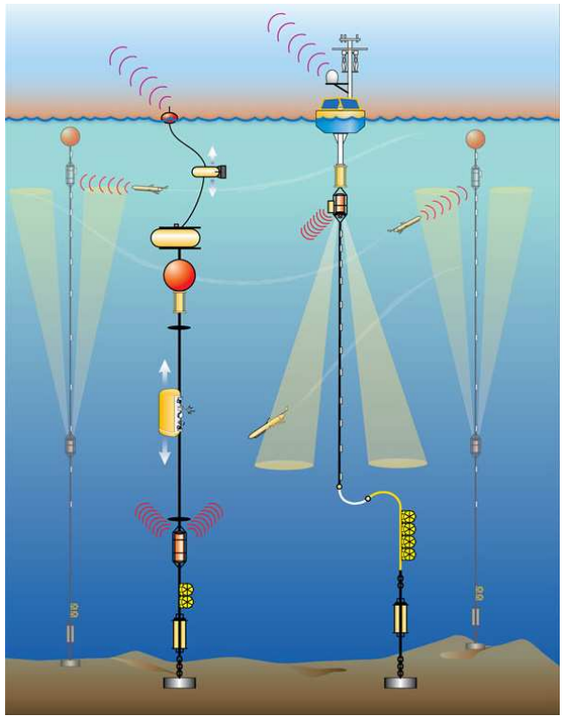
  $\Longrightarrow$ Generating BPMN Choreographies and Legacy Code Analysis

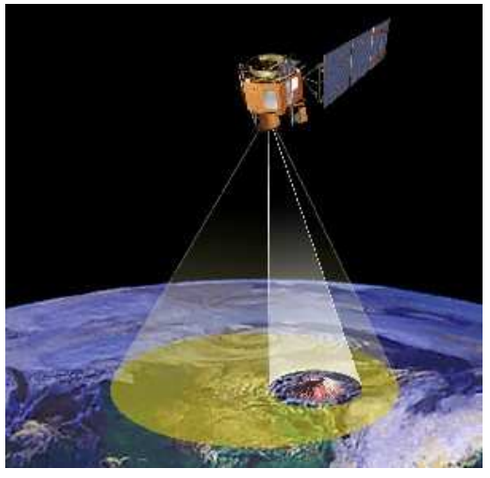
- Code Generation

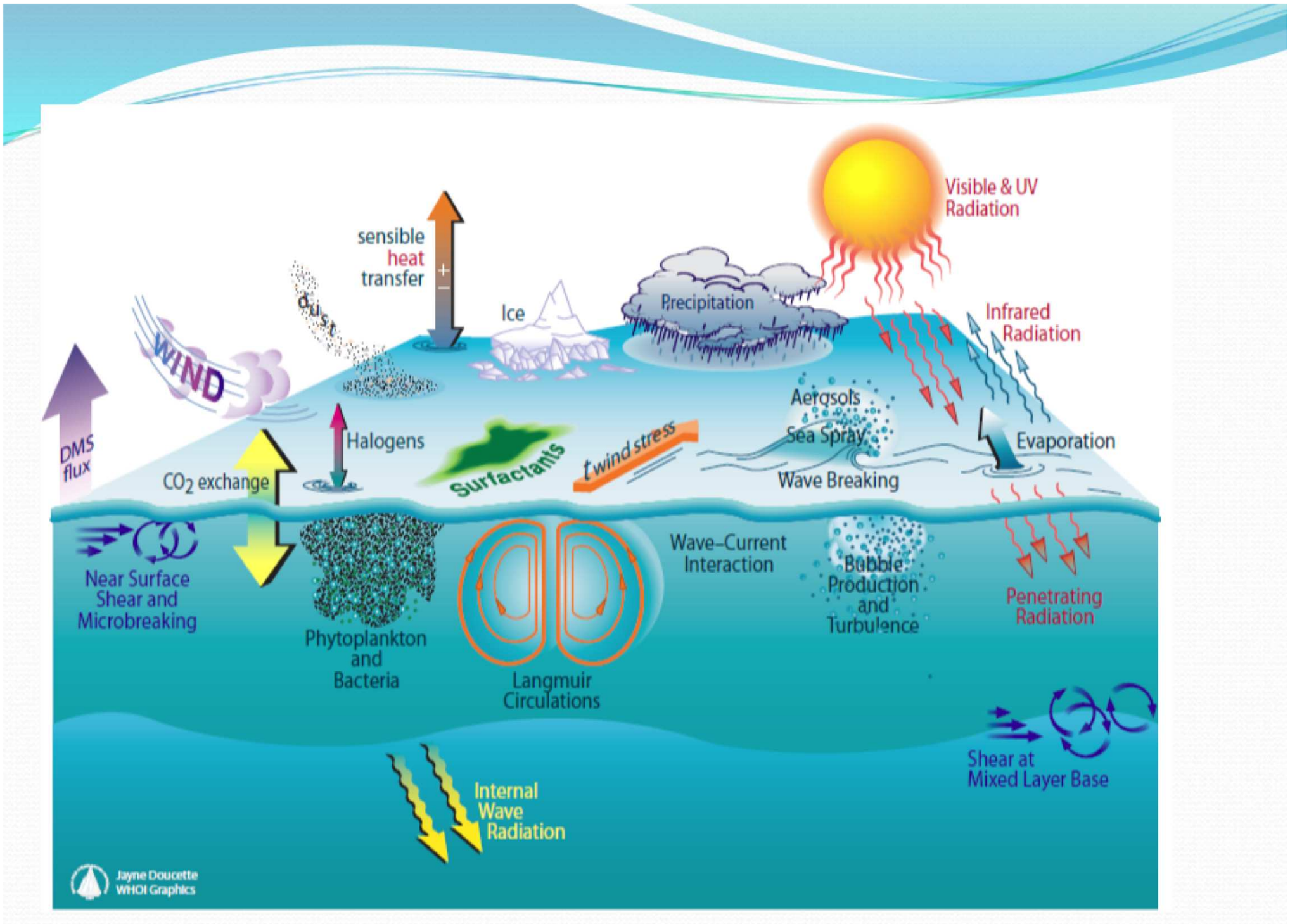
  $\Longrightarrow$ MPI Parallel Programming

Ocean Observatories Initiative

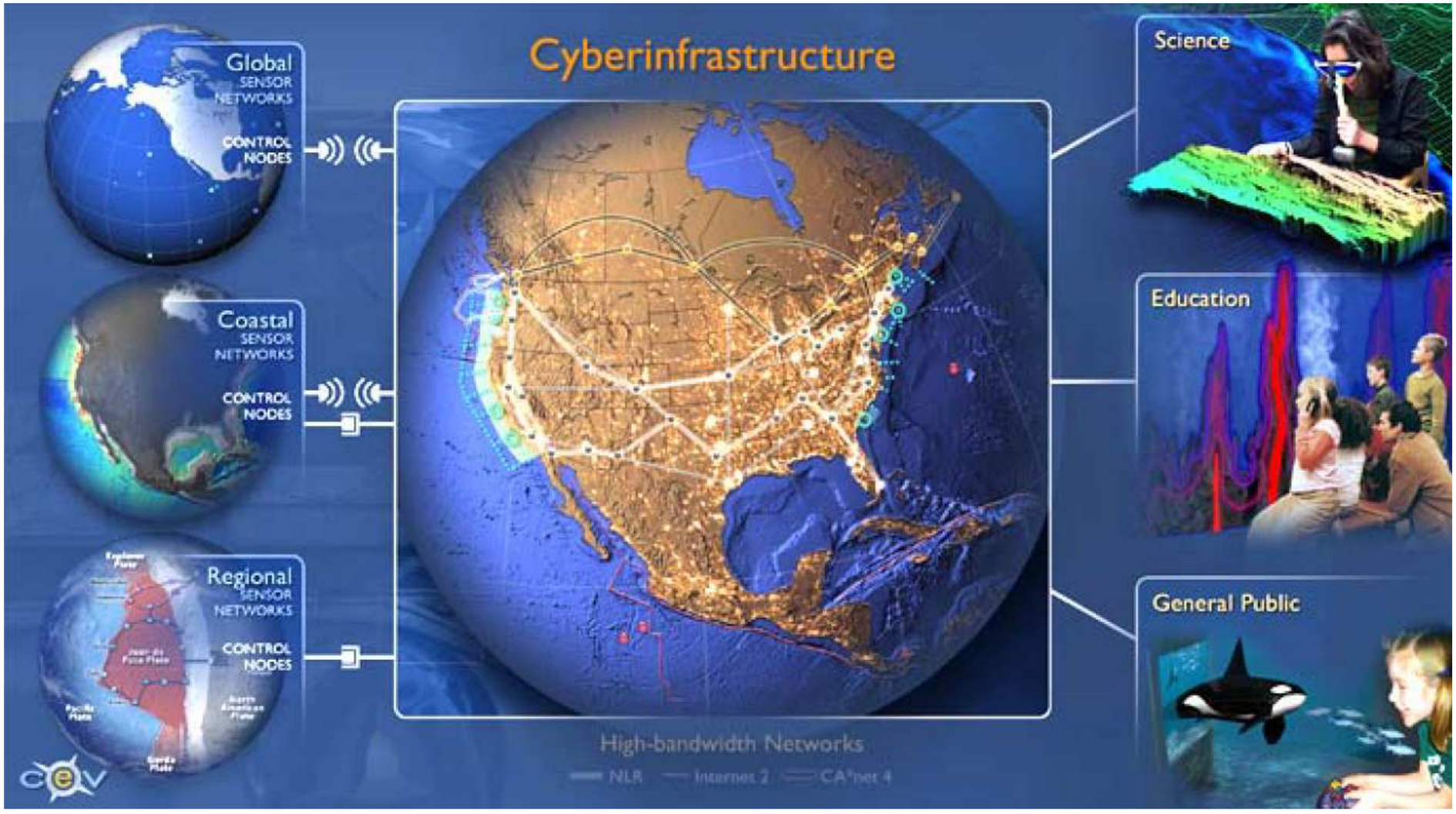
- A NSF project (400M$, 5 Years) to build a cyberinfrastructure for observing oceans around US and beyond.
- Real-time sensor data constantly coming from both off-shore and on-shore (e.g. buoys, submarines, under-water cameras, satellites), transmitted via high-speed networks.

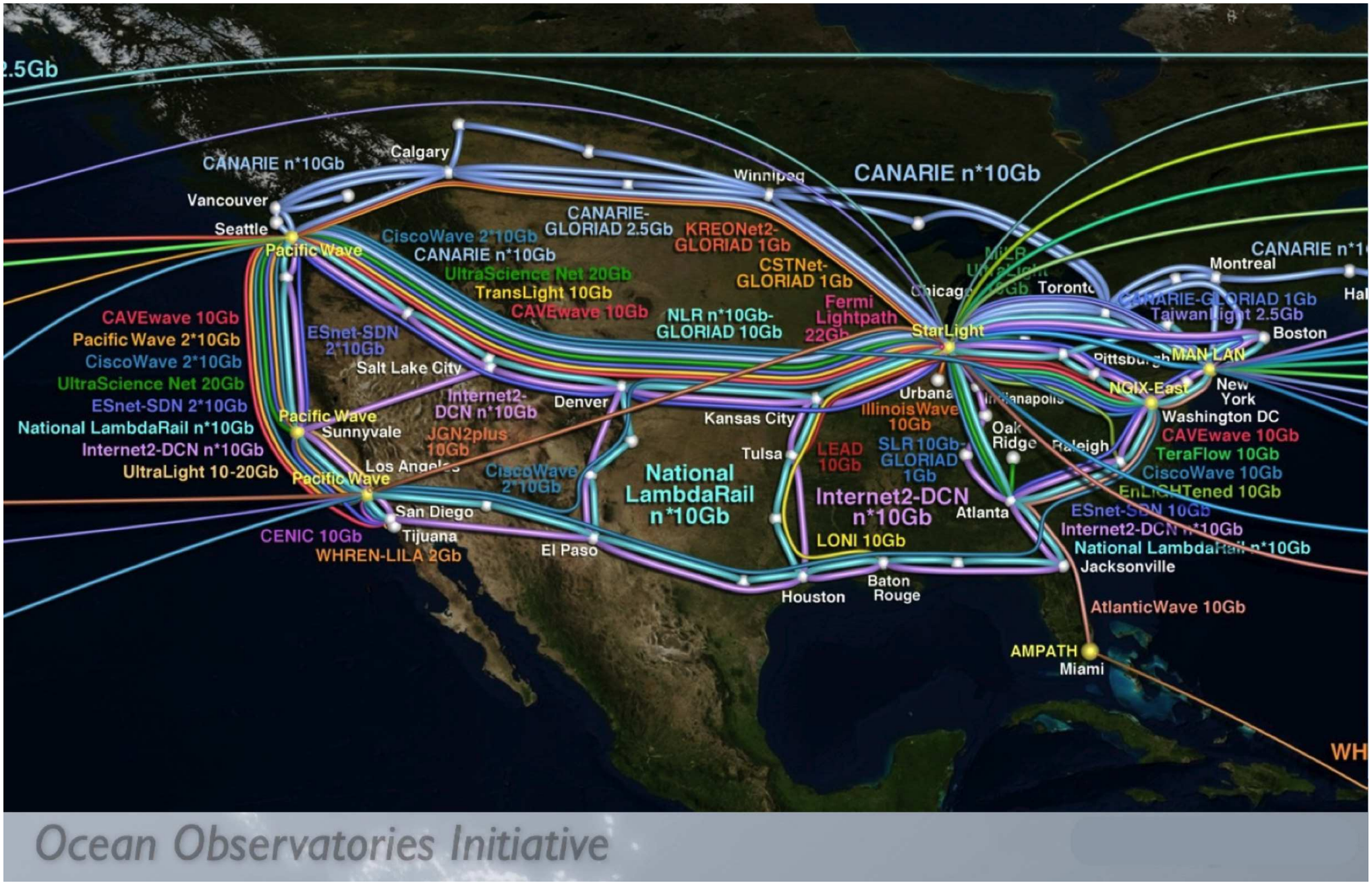





sensible
heat
transfer
+
−
Visible & UV
Radiation
Infrared
Radiation
Ice
Precipitation
dust
WIND
DMS
flux
Halogens
CO2 exchange
Surfactants
t wind stress
Aerosols
Sea Spray
Wave Breaking
Evaporation
Near Surface
Shear and
Microbreaking
Phytoplankton
and
Bacteria
Langmuir
Circulations
Wave–Current
Interaction
Bubble
Production
and
Turbulence
Penetrating
Radiation
Shear at
Mixed Layer Base
Internal
Wave
Radiation
Jayne Doucette
WHOI Graphics

Ocean Observatories Initiative




2.5Gb
CANARIE n*10Gb
Calgary
Vancouver
Seattle
Pacific Wave
Winnipeg
CANARIE n*10Gb
CANARIE-
GLORIAD 2.5Gb
KREONet2-
GLORIAD 1Gb
CiscoWave 2*10Gb
CANARIE n*10Gb
UltraScience Net 20Gb
TransLight 10Gb
CSTNet-
GLORIAD 1Gb
CAVEwave 10Gb
NLR n*10Gb-
GLORIAD 10Gb
Fermi
Lightpath
22Gb
Chicago
Toronto
Montreal
CANARIE-GLORIAD 1Gb
TaiwanLight 2.5Gb
Boston
StarLight
MAN LAN
CAVEwave 10Gb
Pacific Wave 2*10Gb
CiscoWave 2*10Gb
UltraScience Net 20Gb
ESnet-SDN 2*10Gb
National LambdaRail n*10Gb
Internet2-DCN n*10Gb
UltraLight 10-20Gb
ESnet-SDN
2*10Gb
Salt Lake City
Internet2-
DCN n*10Gb
Denver
Pacific Wave
Sunnyvale
JGN2plus
10Gb
Los Angeles
Pacific Wave
CiscoWave
2*10Gb
San Diego
Tijuana
CENIC 10Gb
WHREN-LILA 2Gb
El Paso
Kansas City
Tulsa
National
LambdaRail
n*10Gb
Houston
Baton
Rouge
Urbana
IllinoisWave
10Gb
LEAD
10Gb
SLR 10Gb-
GLORIAD
1Gb
Internet2-DCN
n*10Gb
LONI 10Gb
Atlanta
Oak
Ridge
Raleigh
NGIX-East
New
York
Washington DC
CAVEwave 10Gb
TeraFlow 10Gb
CiscoWave 10Gb
ESnet-SDN 10Gb
Internet2-DCN n*10Gb
Jacksonville
AtlanticWave 10Gb
AMPATH
Miami
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*.

  $\Longrightarrow$ Can describe communication protocols as *types*

  $\Longrightarrow$ 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]

$\Downarrow$

Milner, Honda and Yoshida joined W3C WS-CDL (2002)

$\Downarrow$

Formalisation of W3C WS-CDL [ESOP'07]

$\Downarrow$

Scribble at $\pi 4$ Technology

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;
        }
    }
}
```

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:

```
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]

⇓

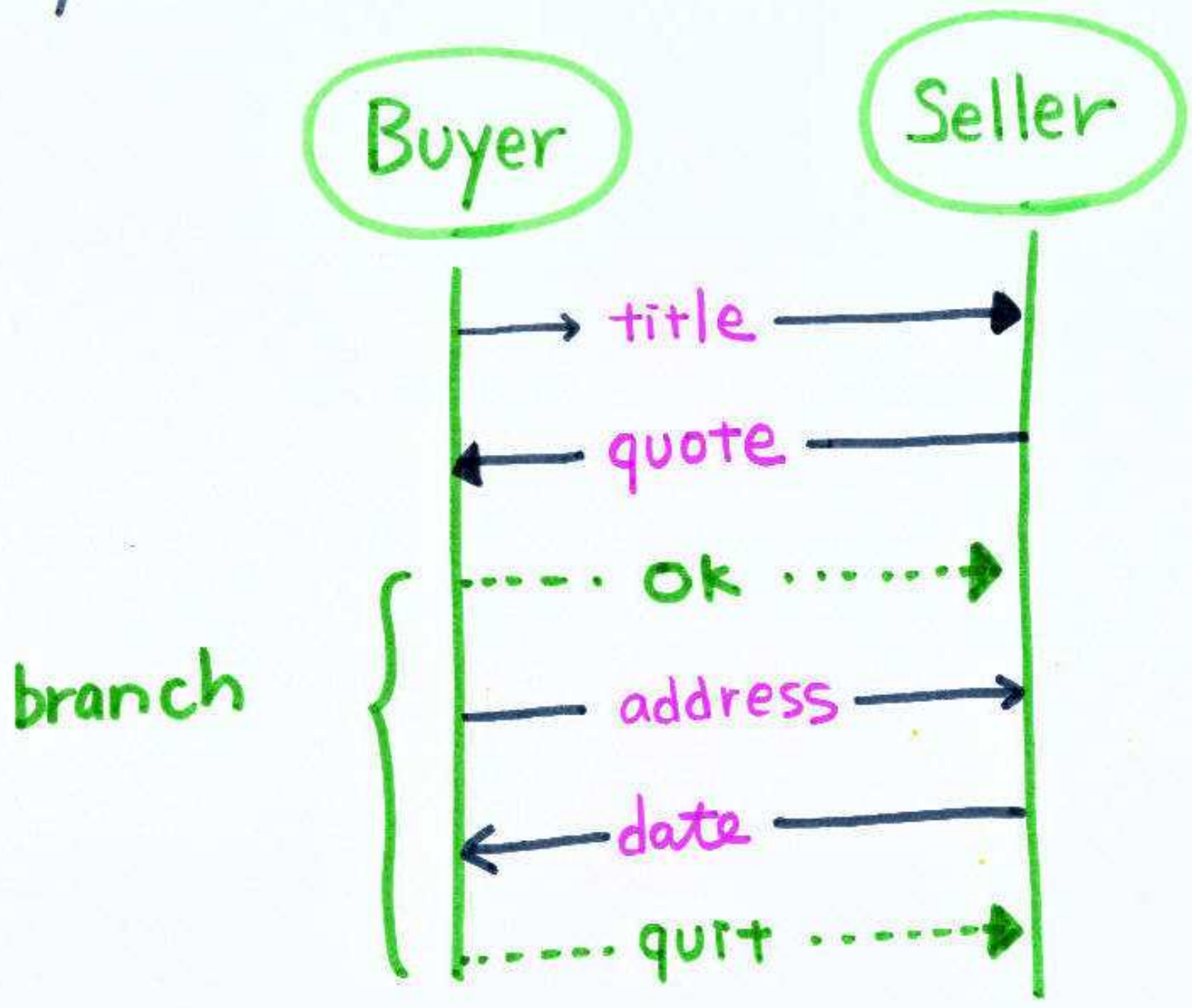
Scribble at $\pi 4$ Technology

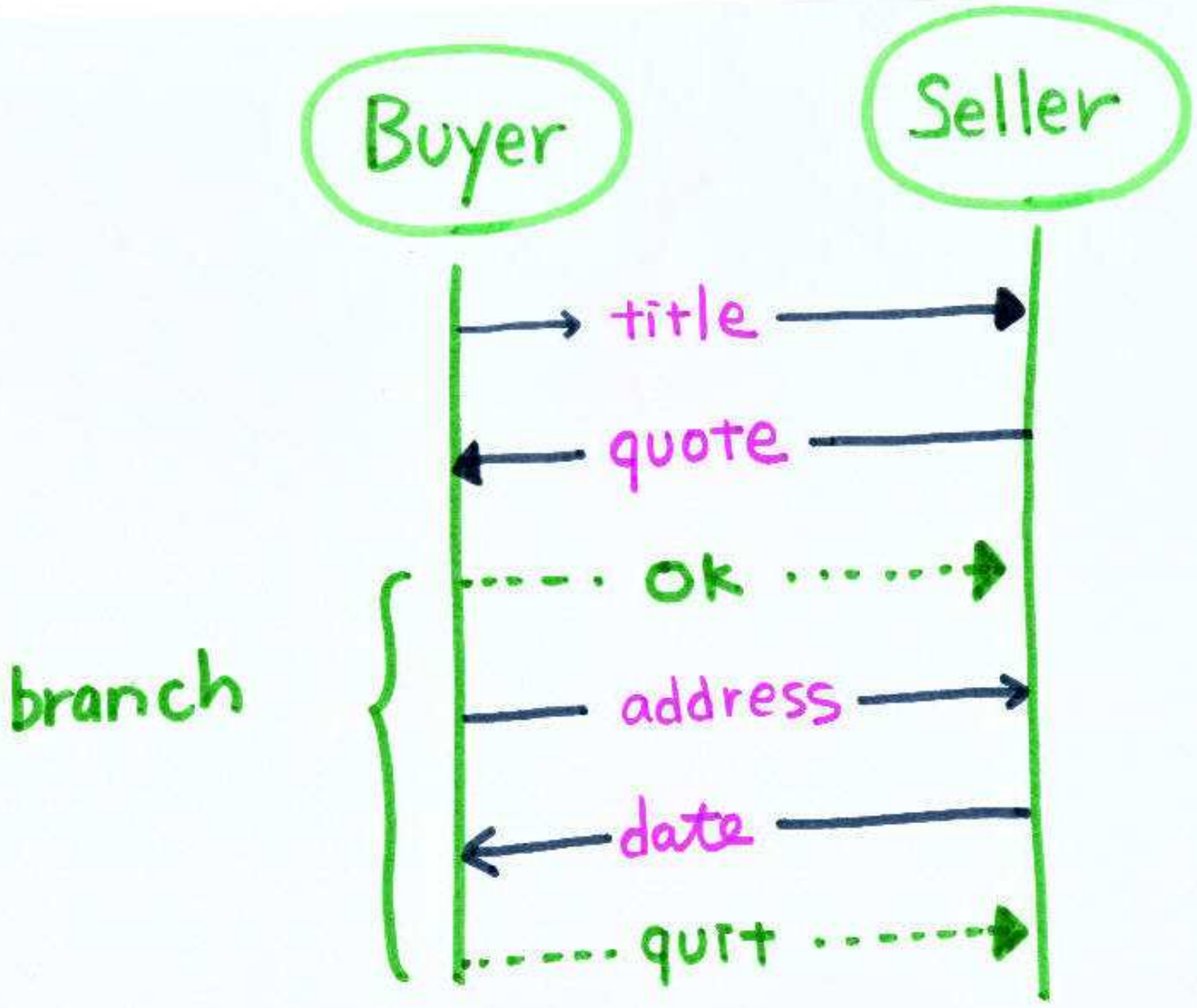
⇓

Multiparty Session Types [POPL'08]

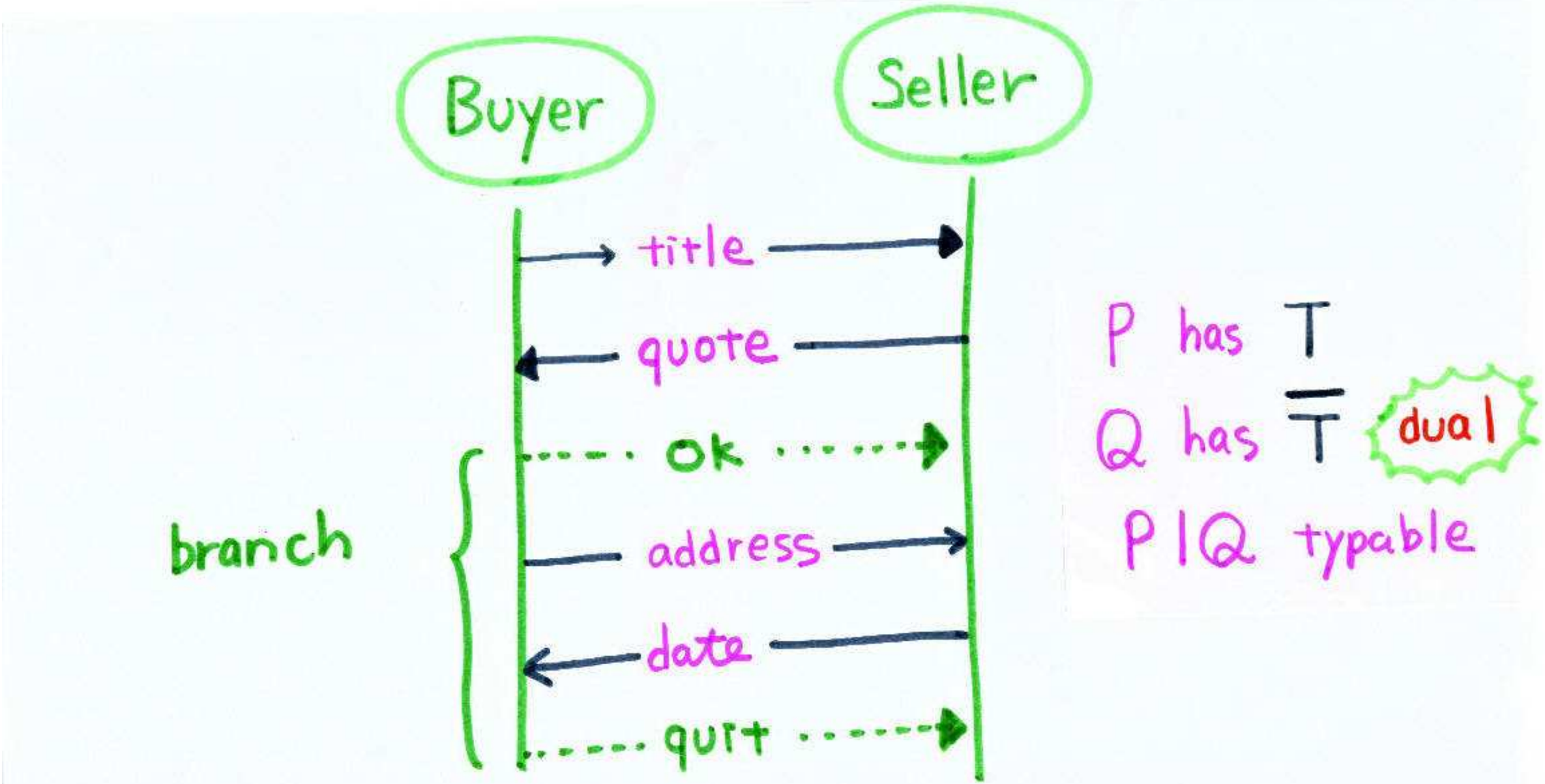
⇓

Binary Session Types : Buyer - Seller Protocol





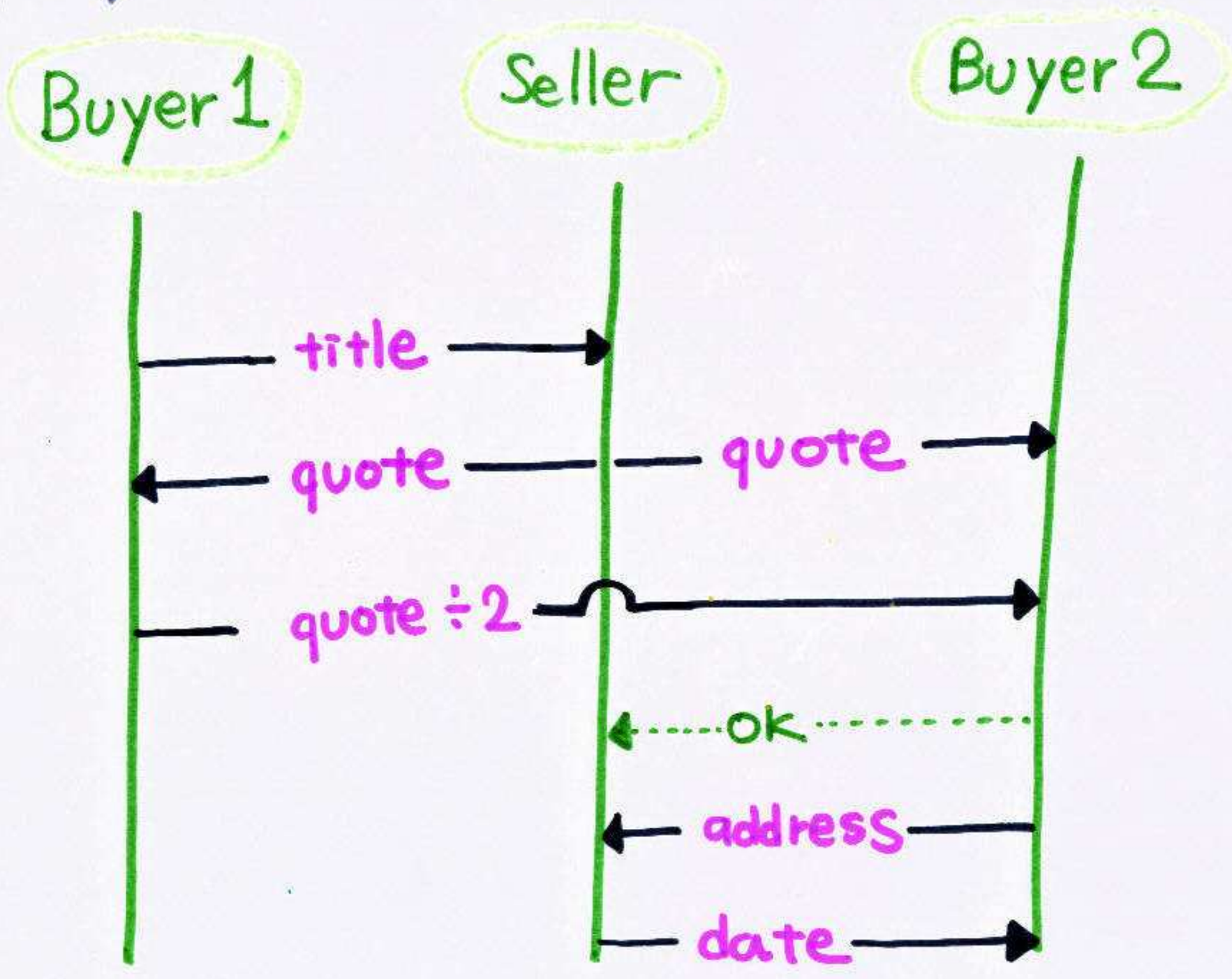
! String ; ? Int ; ⊕{ ok : ! String ; ? Date ; end , quit : end }

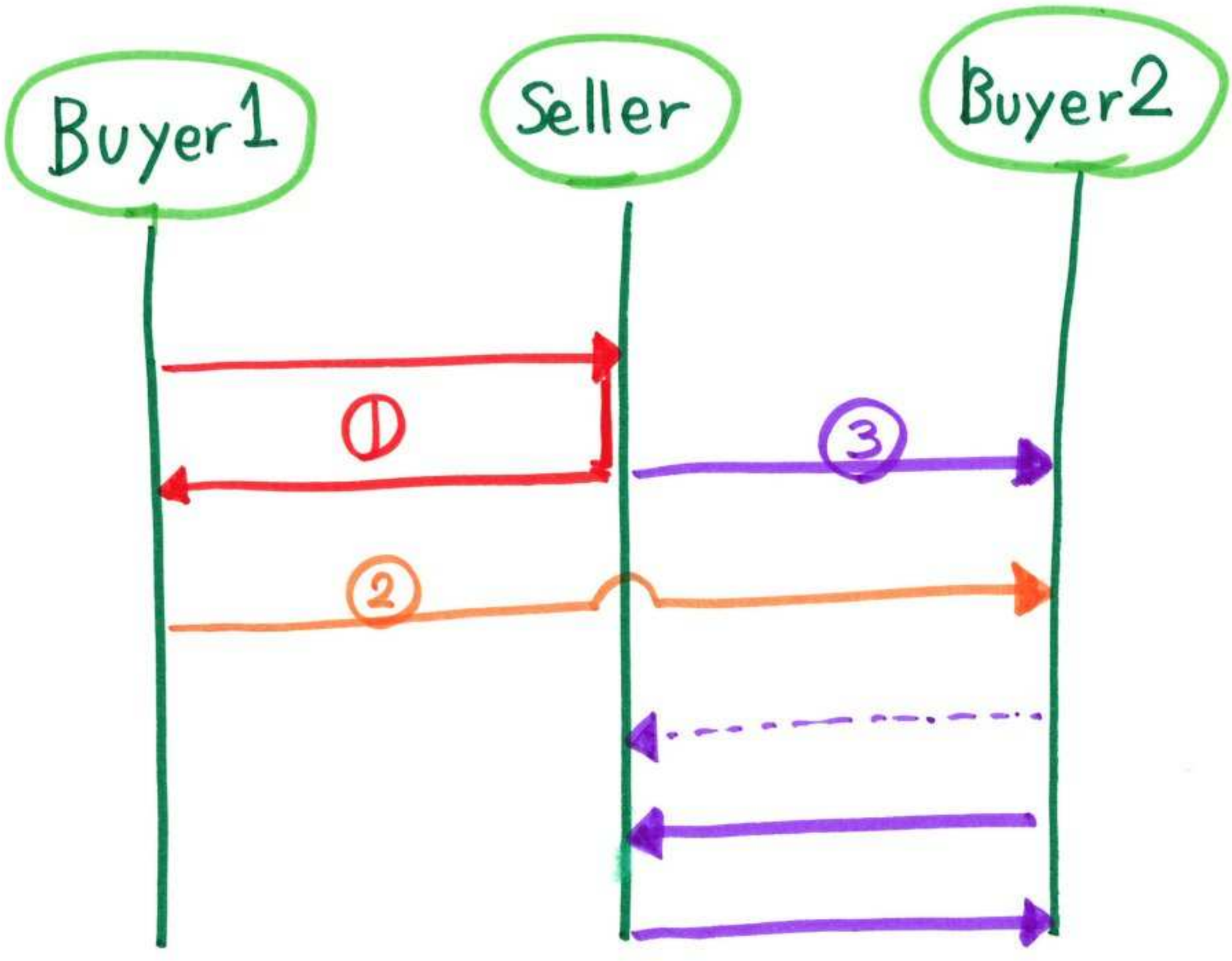


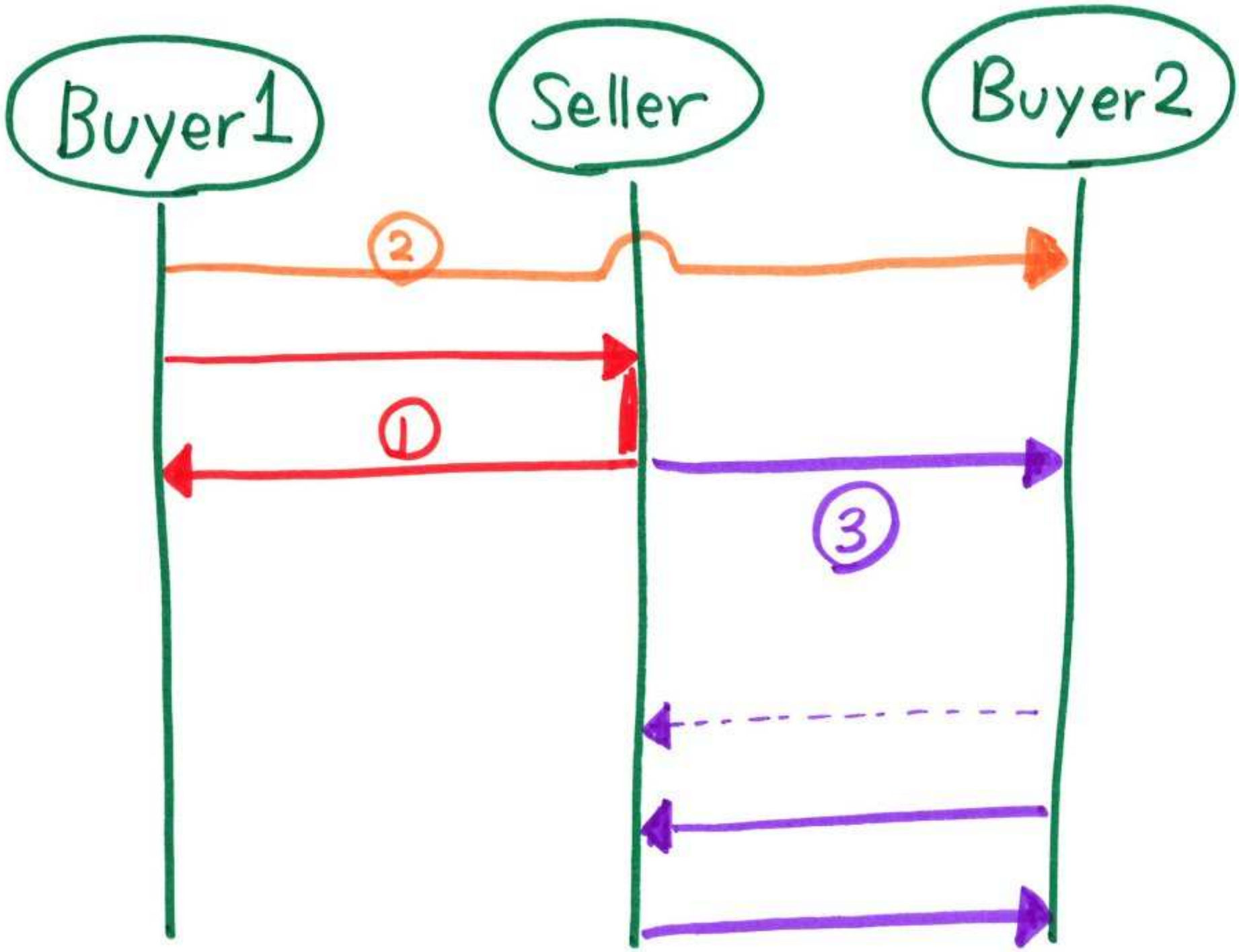
! String ; ? Int ; ⊕{ ok : ! String ; ? Date ; end , quit : end }

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

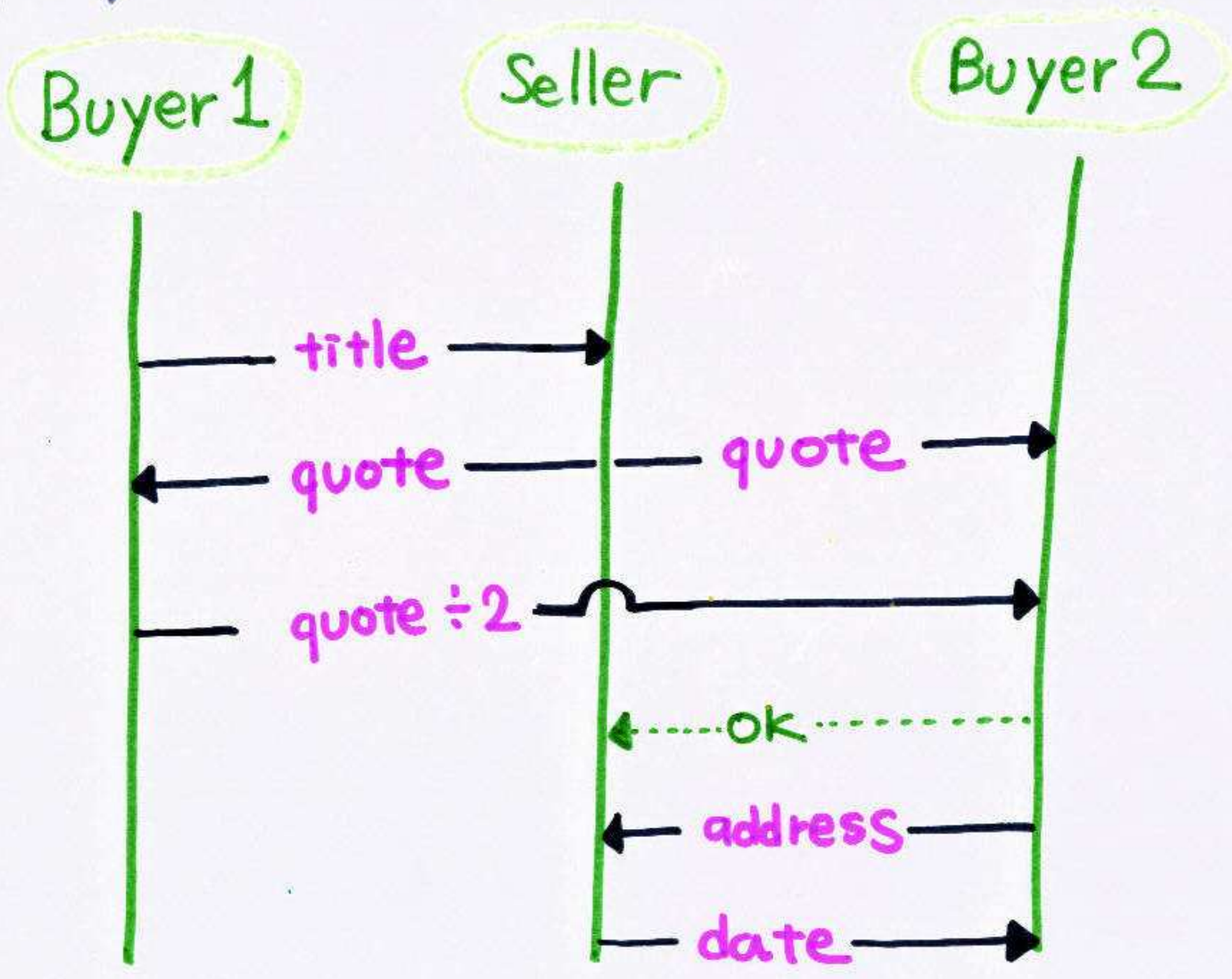
Multiparty Session Types



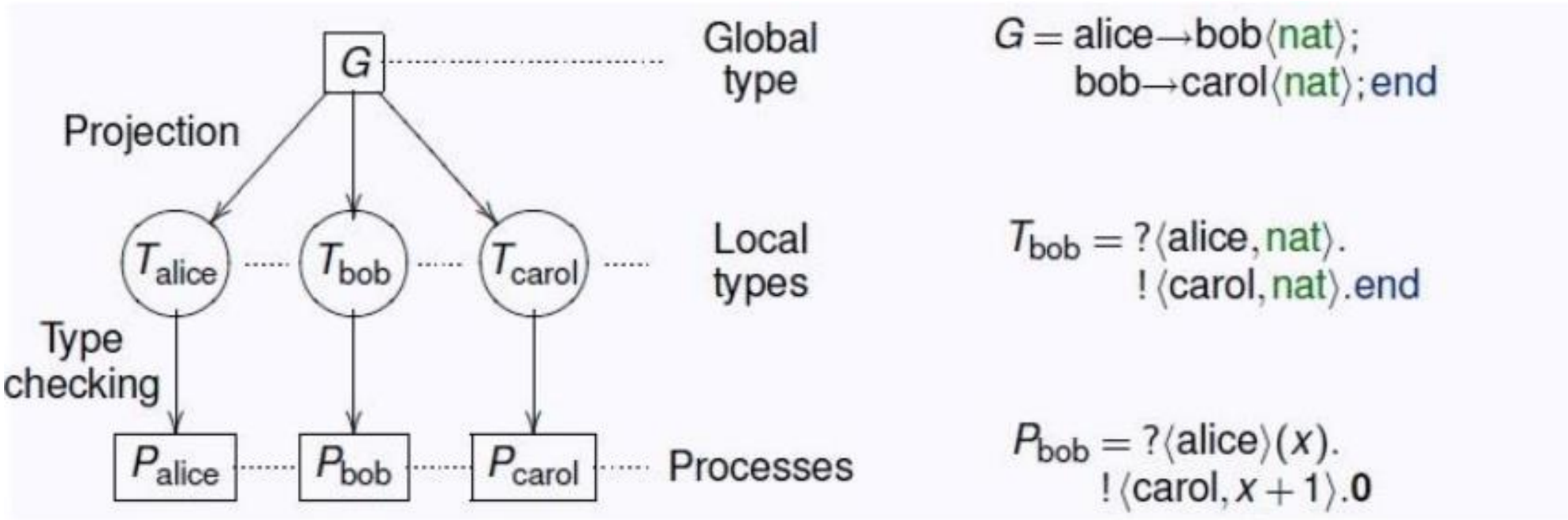

Buyer1
Seller
Buyer2
1
2
3


Buyer1
Seller
Buyer2
2
1
3

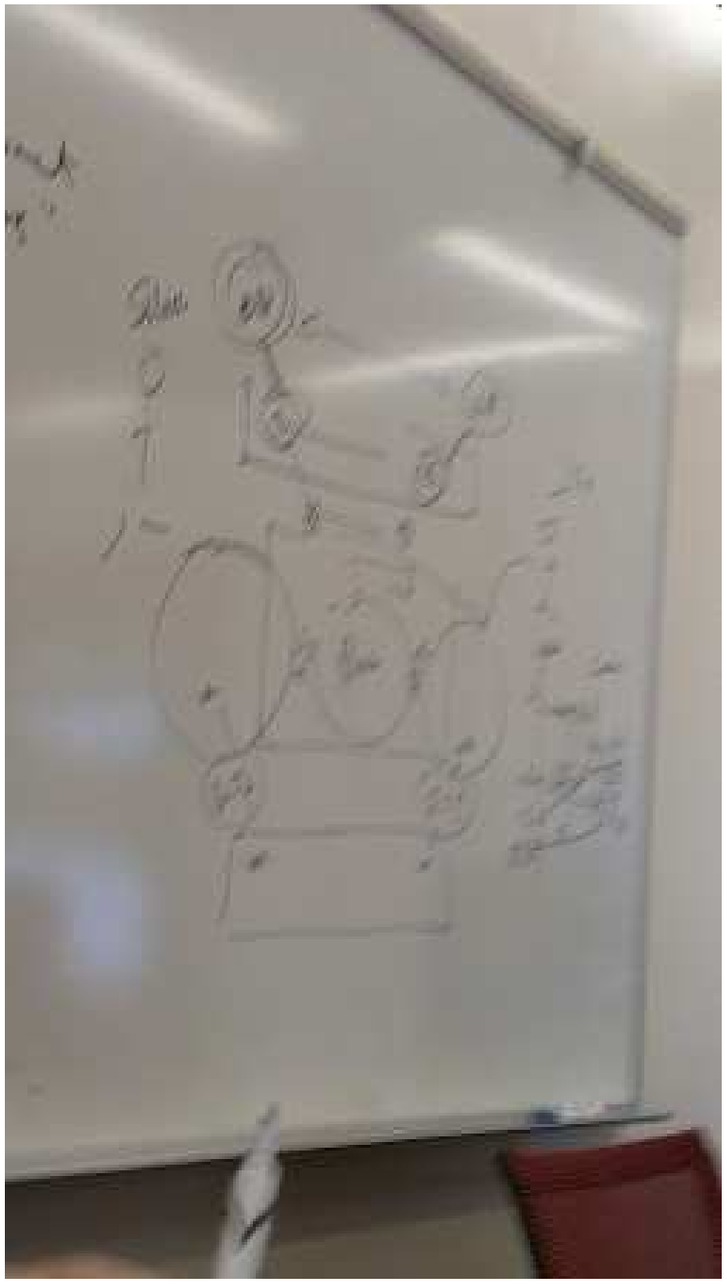
Multiparty Session Types



Session Types Overview



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



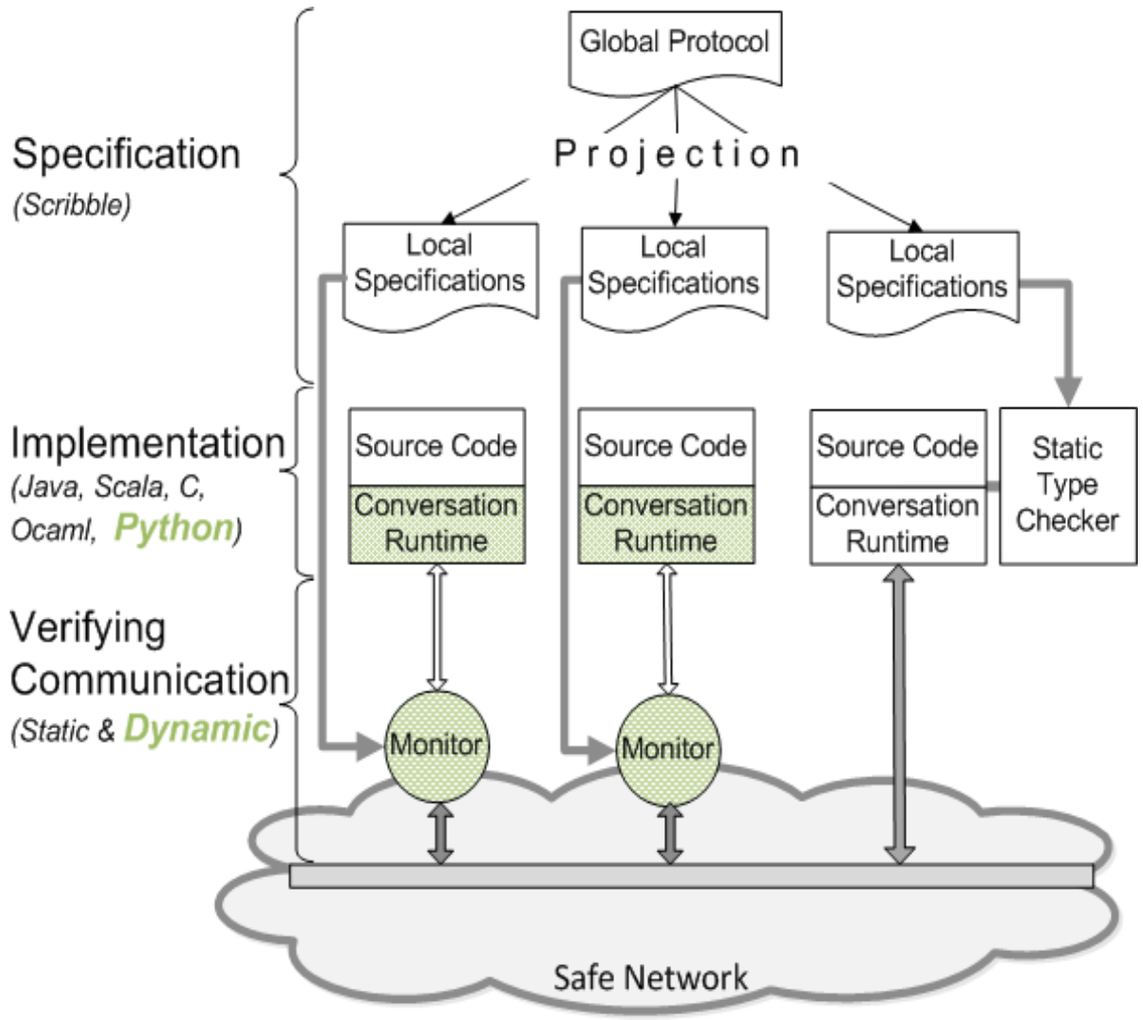
Evolution Of MPST

- 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]

- Distributed Runtime Verification with Session Types and Python [FMSD'15]
- Multiparty Session Actors [COORDINATION'14]

Session Types for Runtime Verification

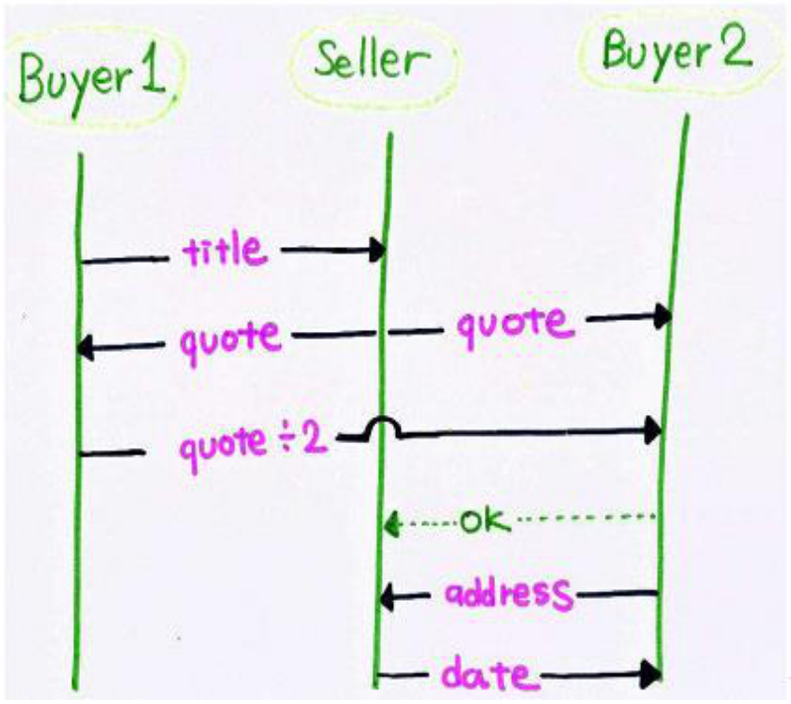
- Methodology
  - Developers design protocols in a dedicated language - Scribble
  - Well-fomedness is checked by Scribble tools
  - Protocols are projected into local types
  - Local types generate monitors



www.scribble.org

Two Buyer Protocol in Scribble

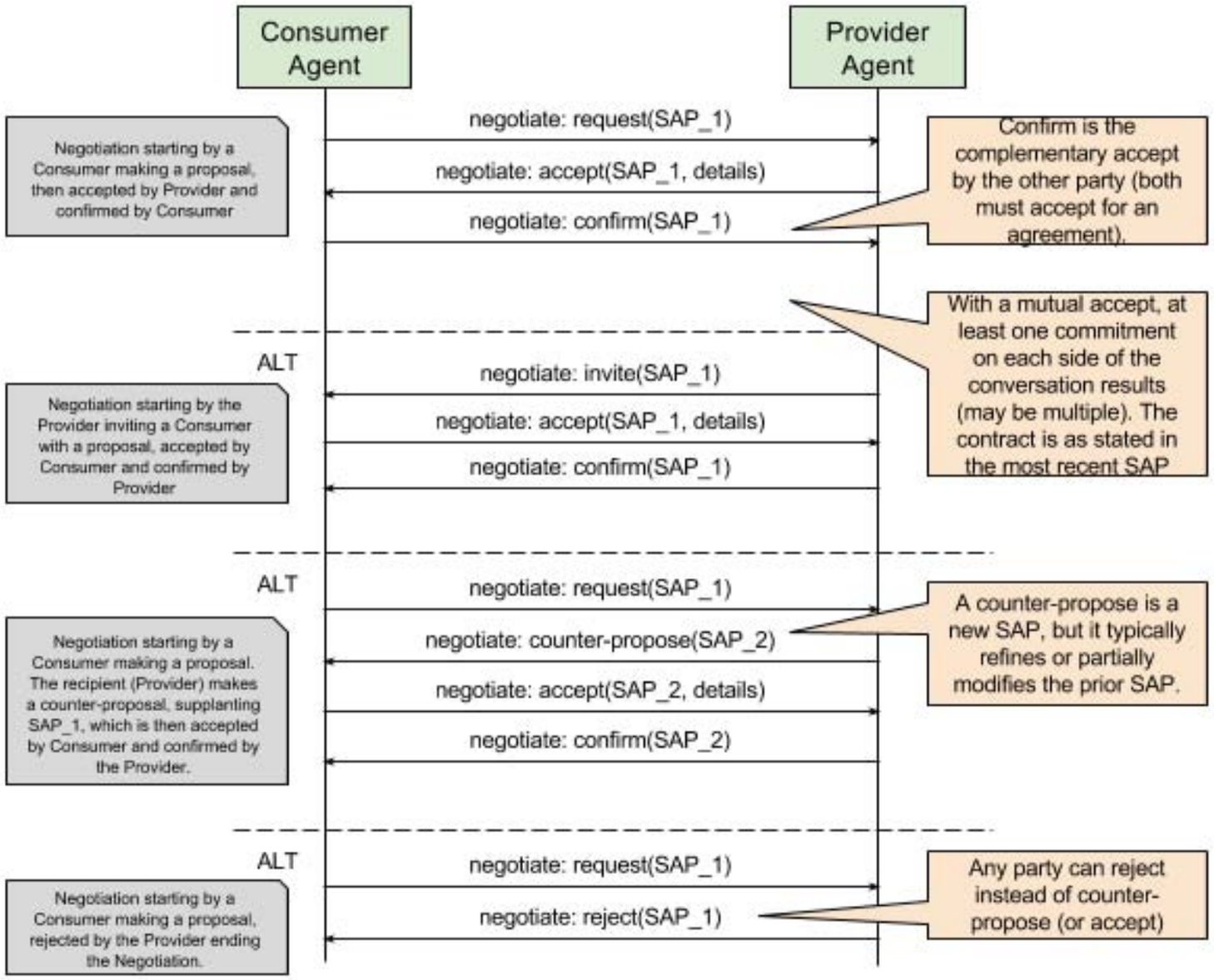
```
type <java> "java.lang.String" from "rt.jar" as String

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,
            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;
 } } }
```

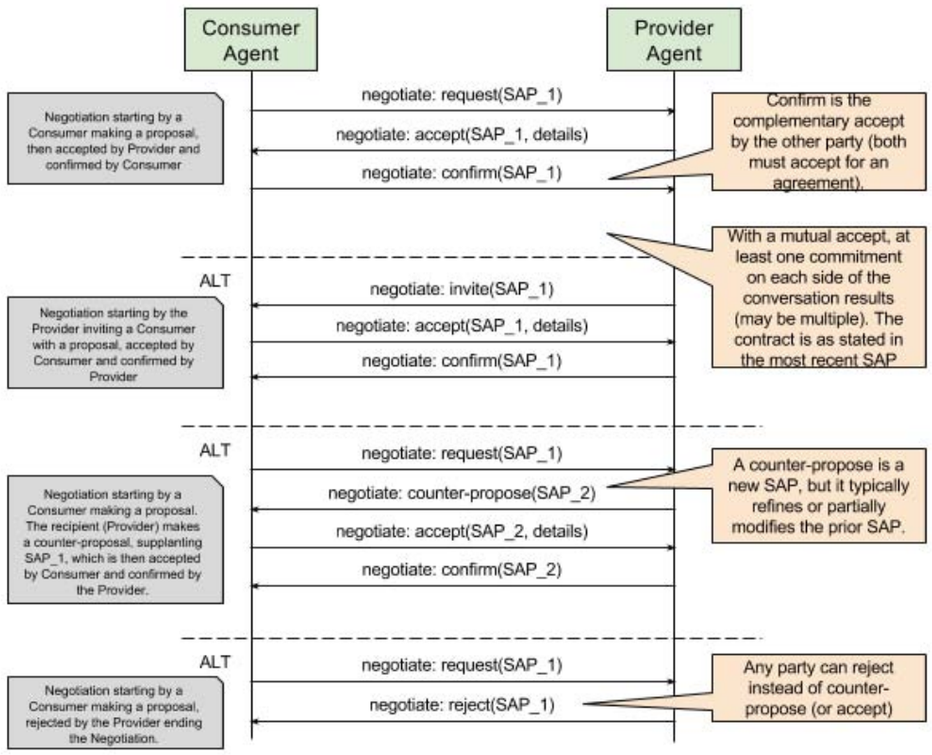
OOI agent negotiation 1/5



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

OOI agent negotiation 2/5

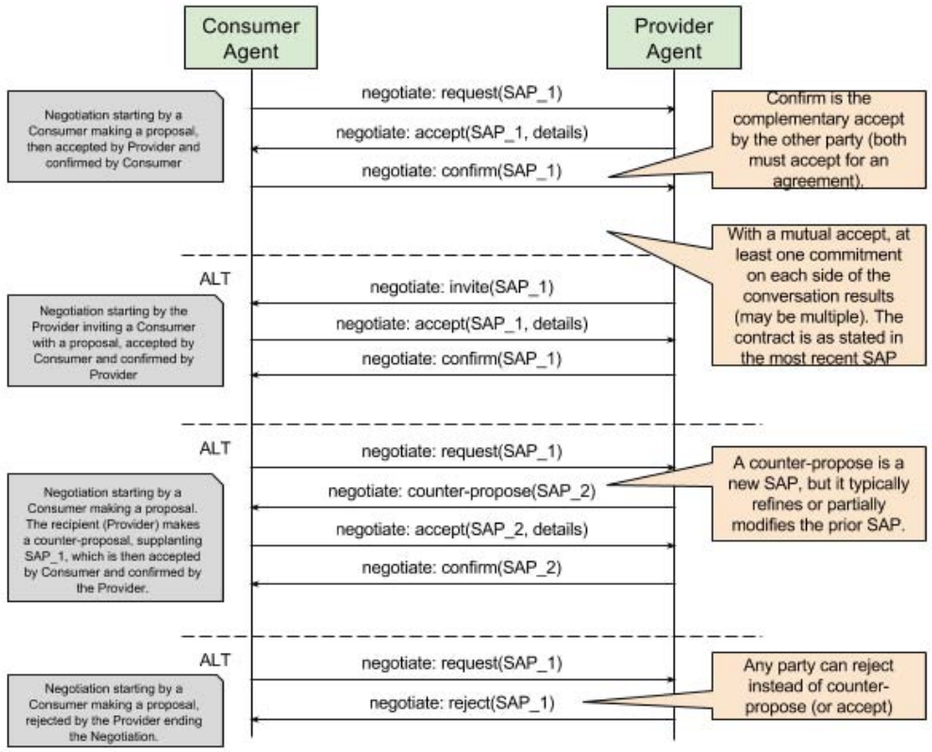
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

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




```
}
```

OOI agent negotiation 3/5 (choice)

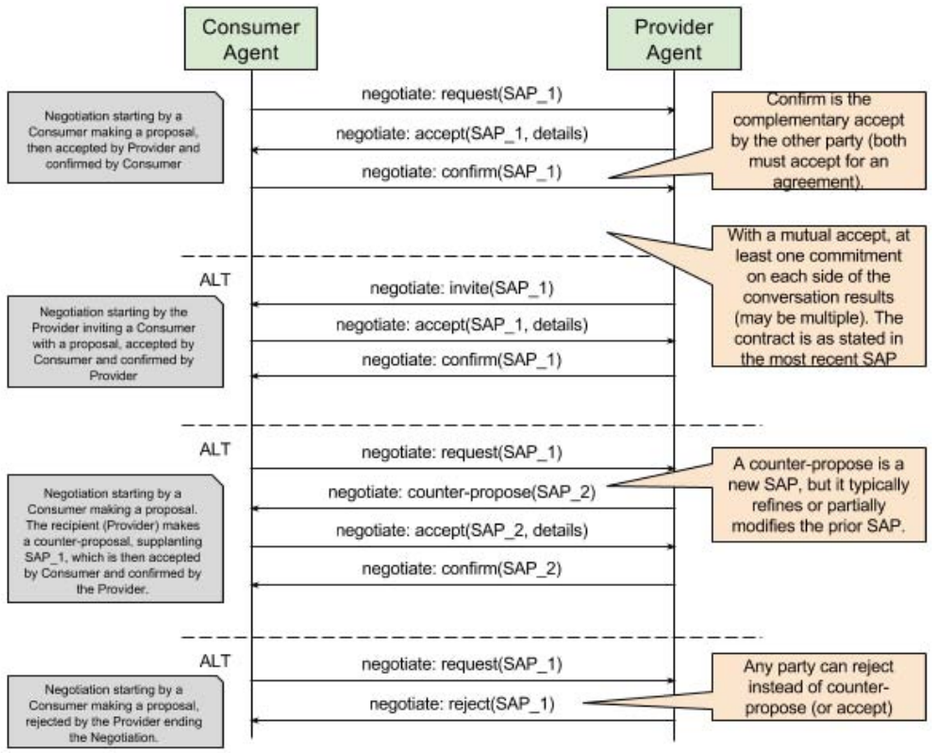
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

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;
```




```
} }
```

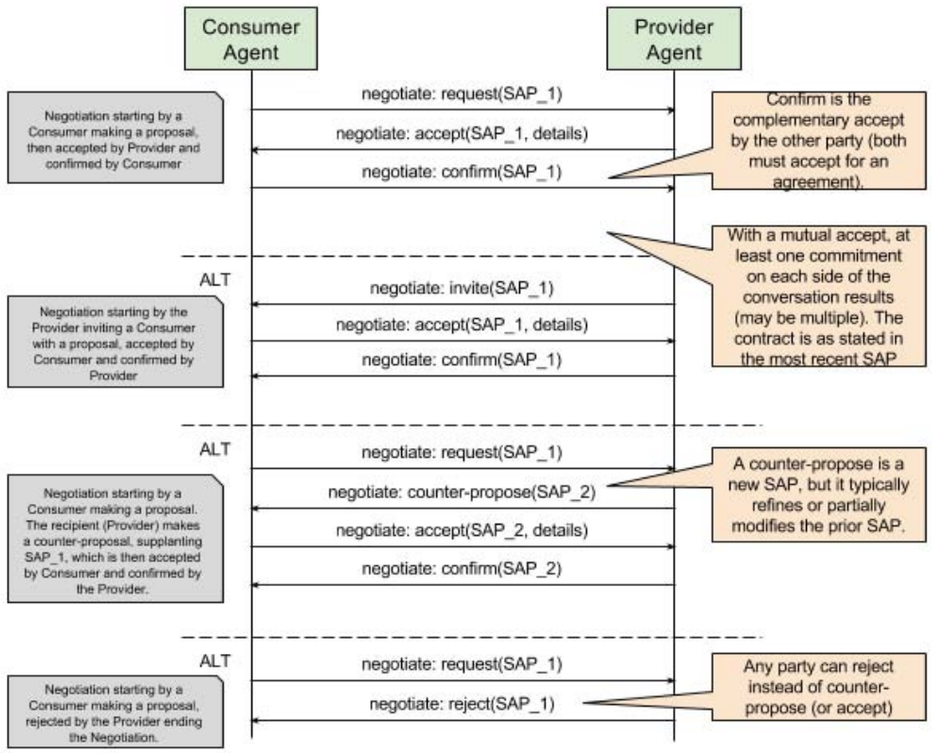
OOI agent negotiation 4/5

```
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;
      }
    }
  }
```

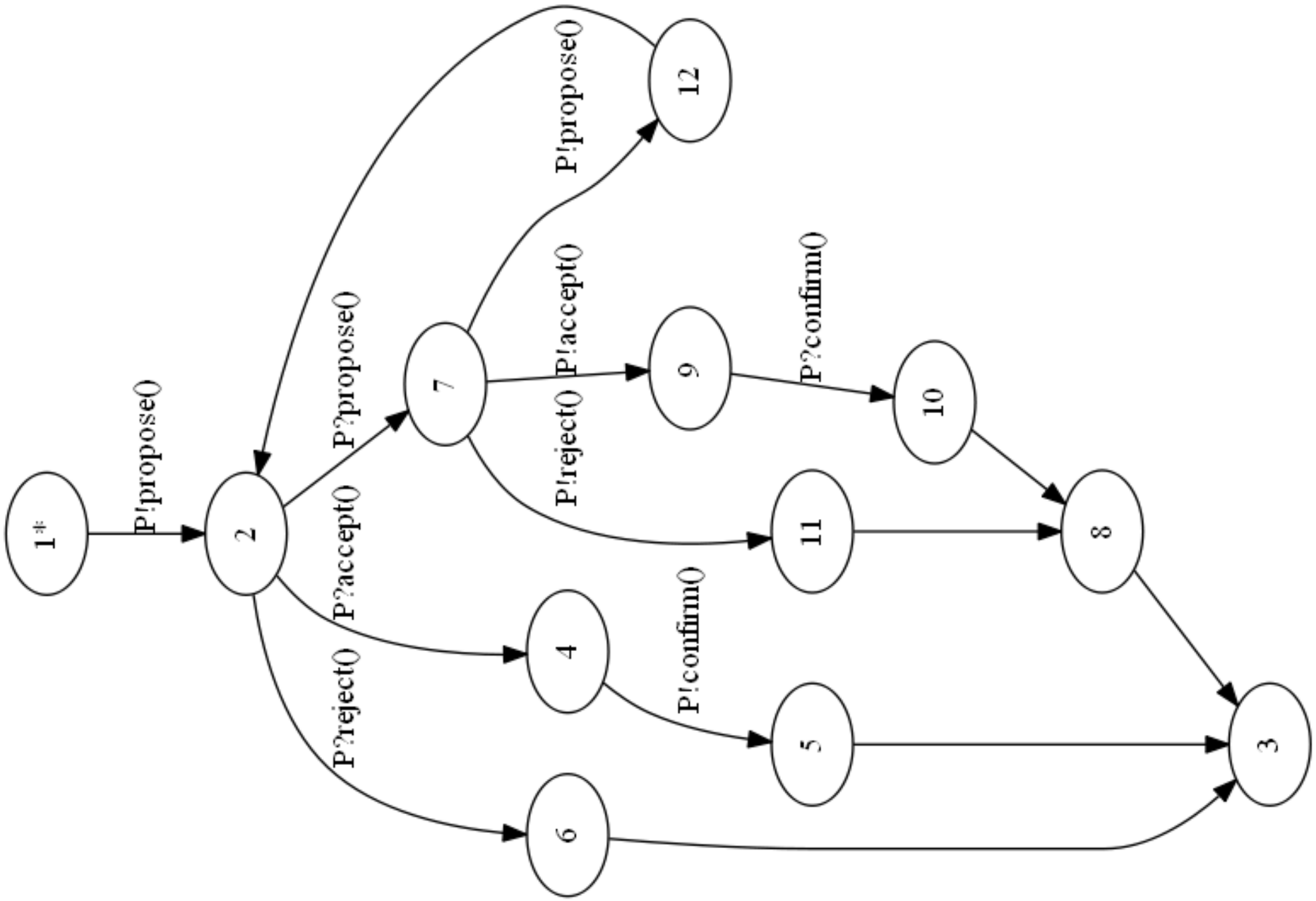


Local protocol projection (Negotiation Consumer)

```
// 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
- GitHub:
  - https://github.com/scribble
- Tutorial:
  - www.doc.ic.ac.uk/~rhu/scribble/tutorial.html
- Specification (0.3)
  - www.doc.ic.ac.uk/~rhu/scribble/langref.html

Scribble online checker: scribble.doc.ic.ac.uk

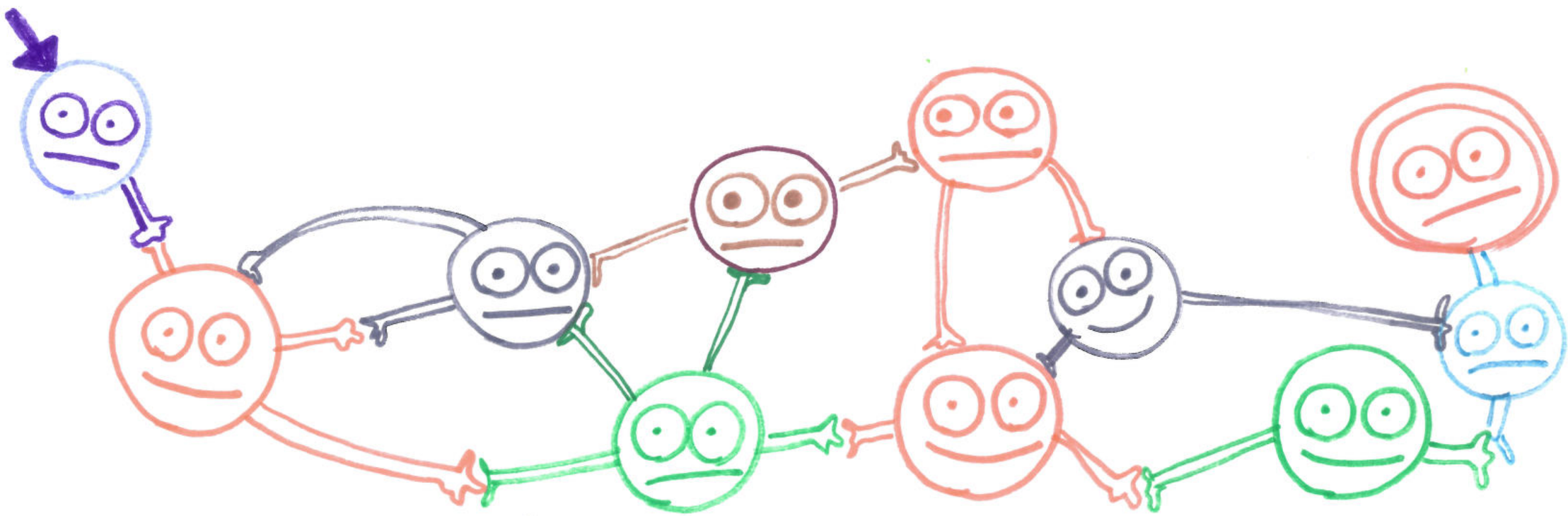
Multiparty Compatibility in Communicating Automata

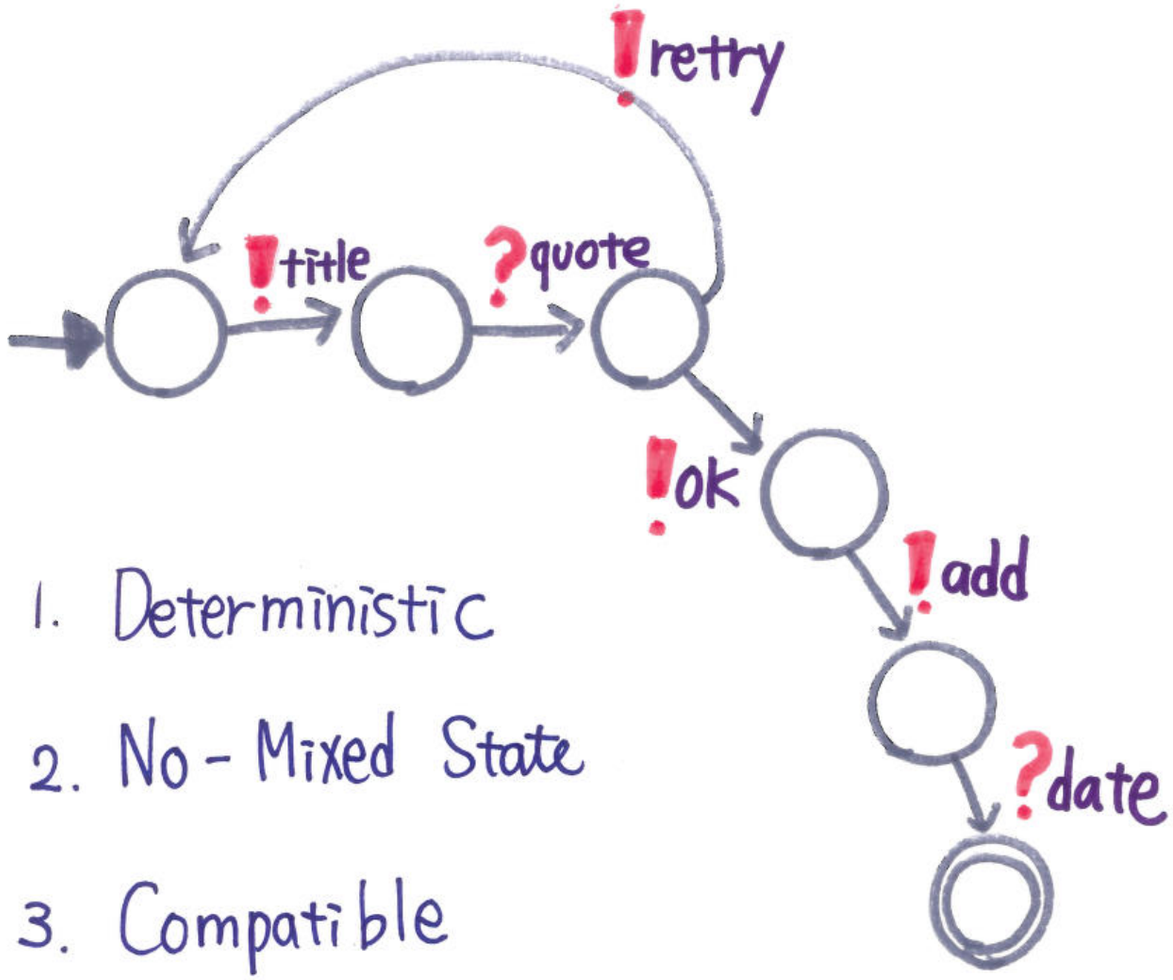
Synthesis and Characterisation of Multiparty Session Types

Nobuko Yoshida

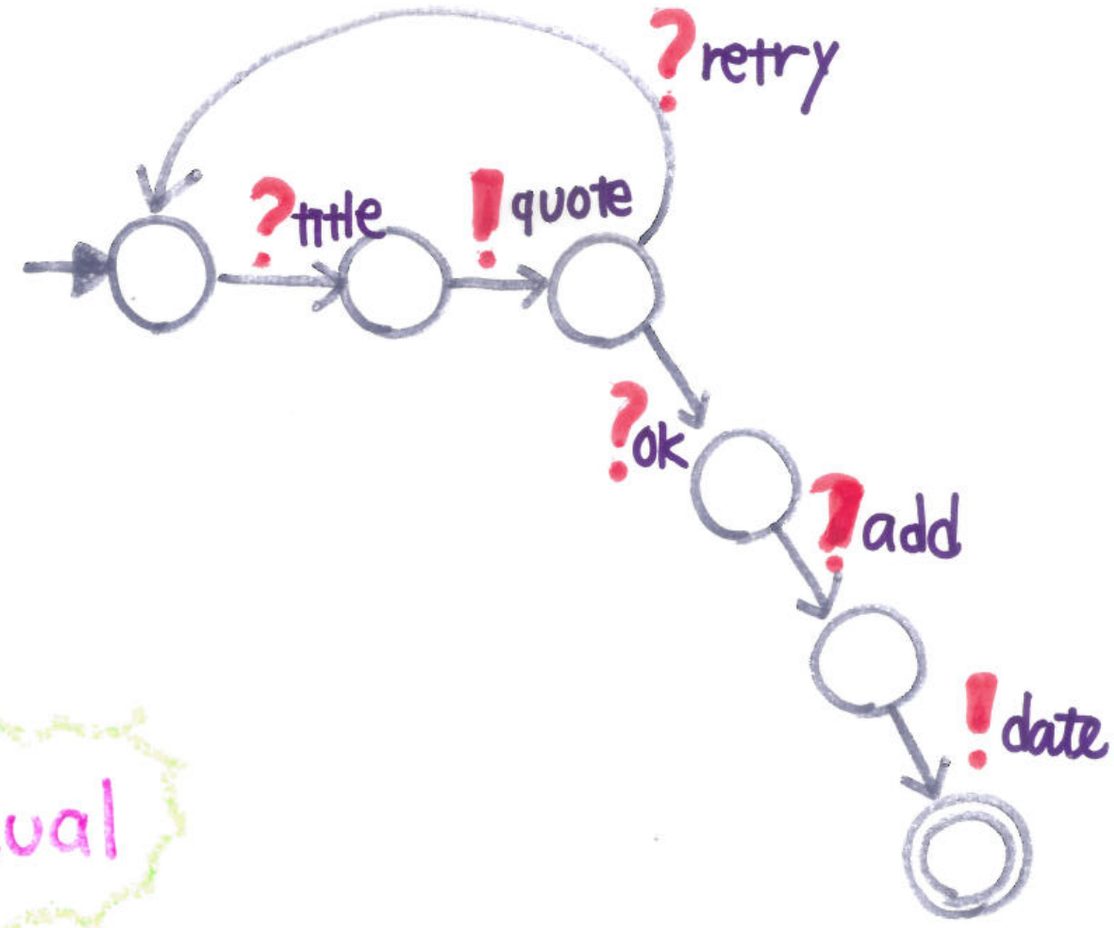
Pierre-Malo Deniélou

ICALP'13





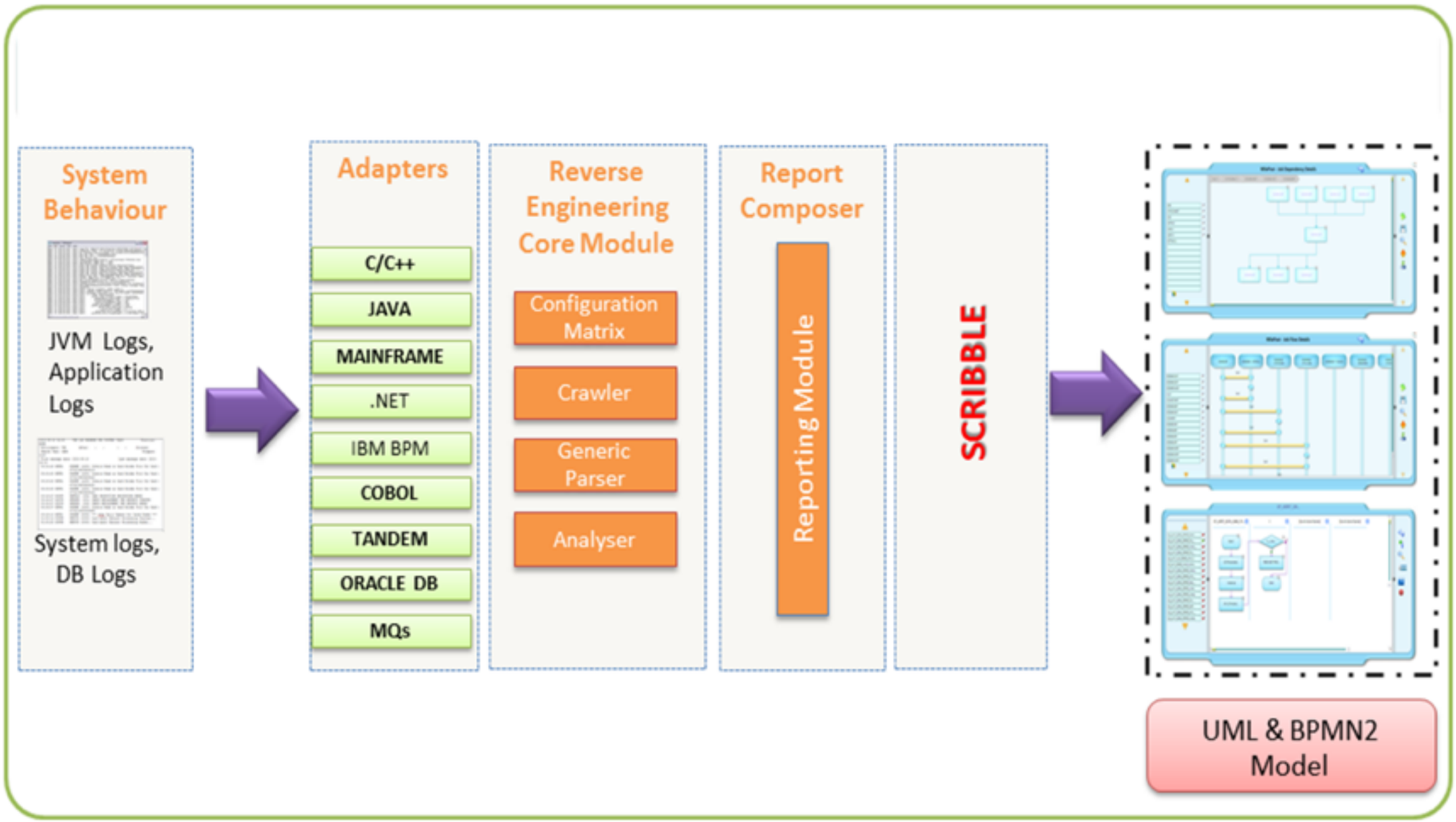
1. Deterministic
2. No-Mixed State
3. Compatible



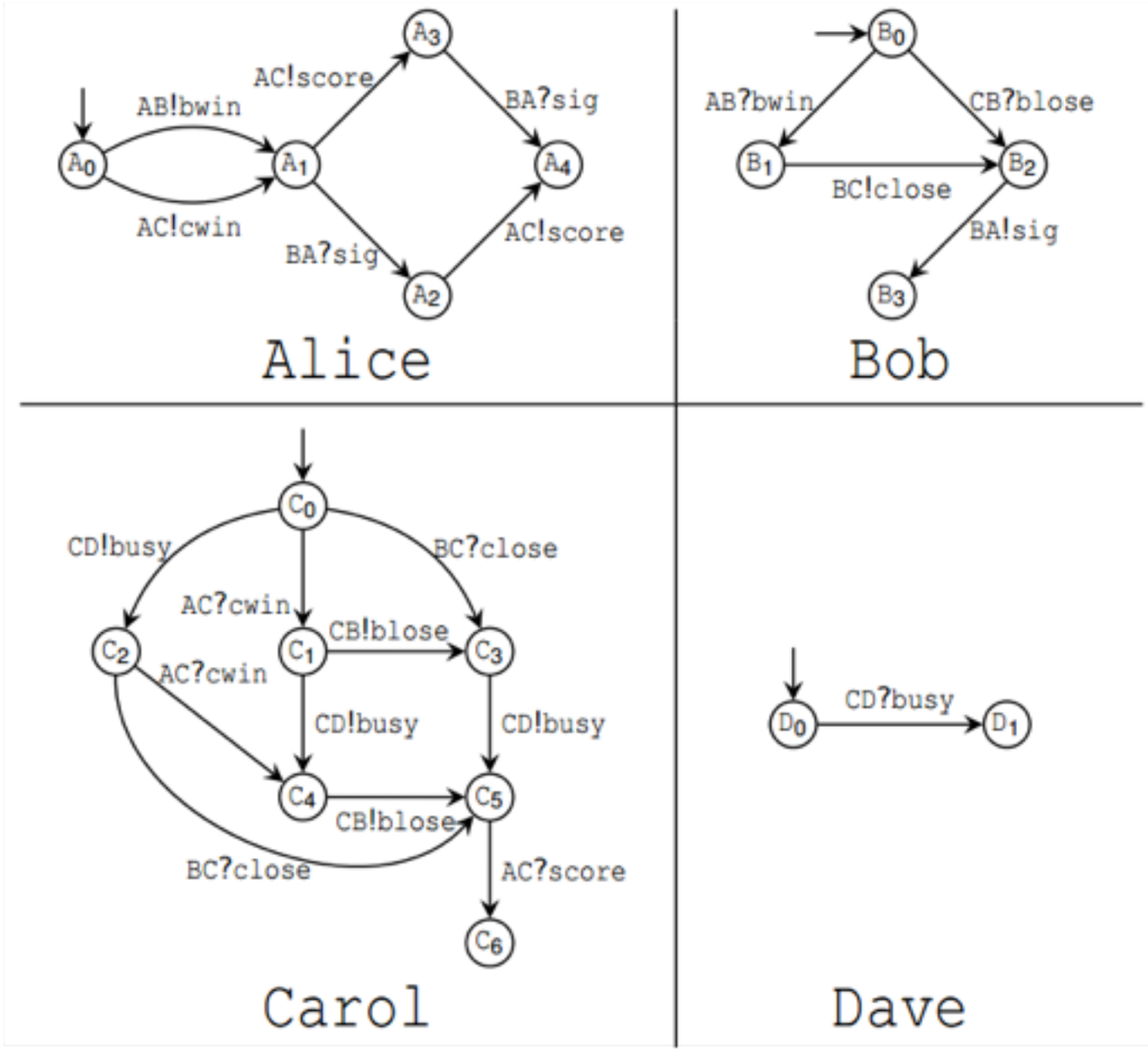
[Gouda et al 1986] Two compatible machines without mixed states which are deterministic satisfy deadlock-freedom.

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

Zero Deviation Life Cycle Platform



From Communicating Machines to Graphical Choreographies [POPL'15]

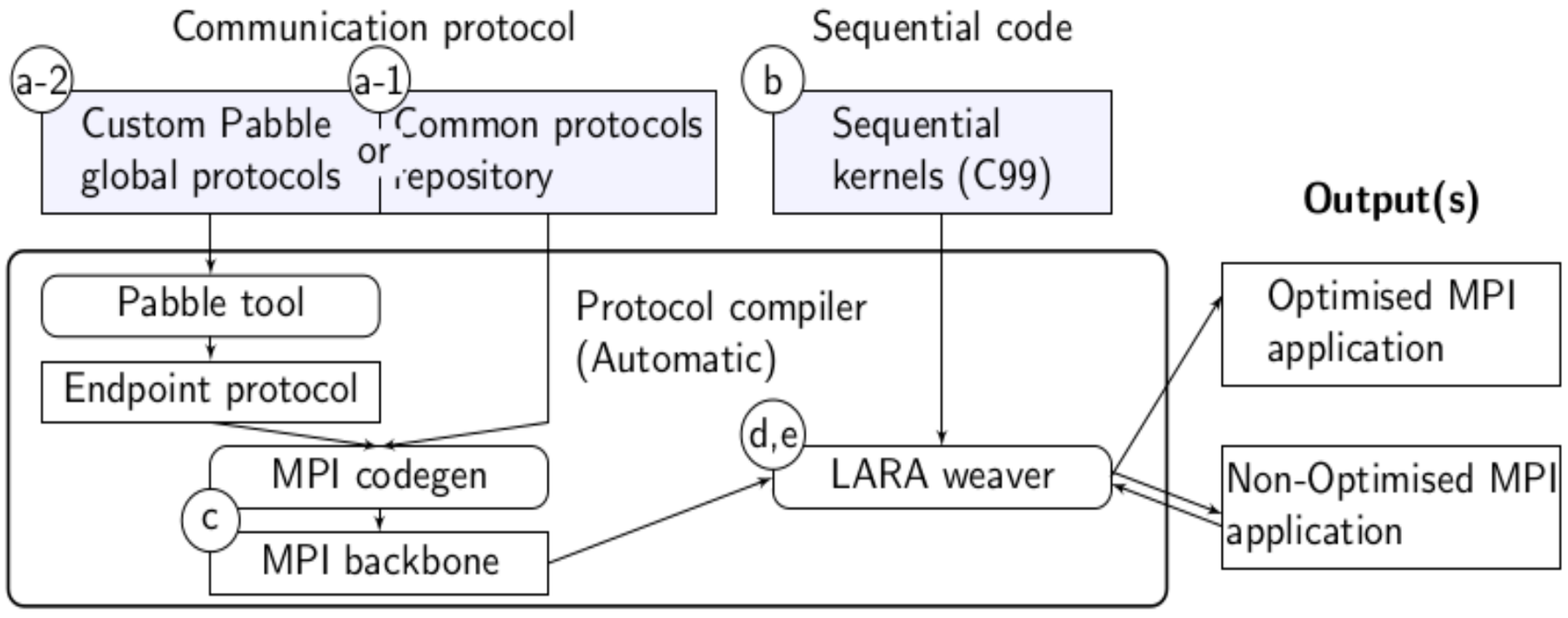


[ESOP'10,ESOP'12,CONCUR'12,CONCUR'14]

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



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
heateq	stencil*	Yes	Structured Grid
nbody	ring*	Yes	Particle Methods
wordcount	scatter-gather*	Yes	
adpredictor	scatter-gather*	Yes	
montecarlo	scatter-gather*	Yes	
montecarlo-mw	master-worker*	Yes	
LEsolver	wraparound mesh		Structured Grid
matvec	custom		Dense Matrix
fft64	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

Effort ratio

- LOC savings

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

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.