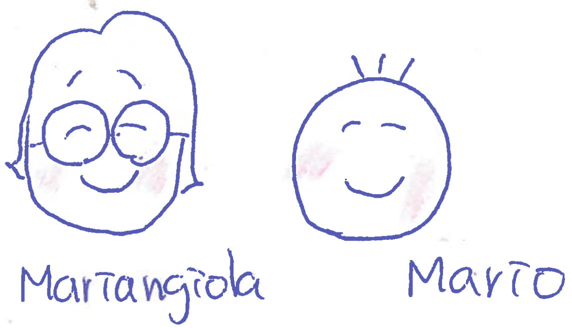
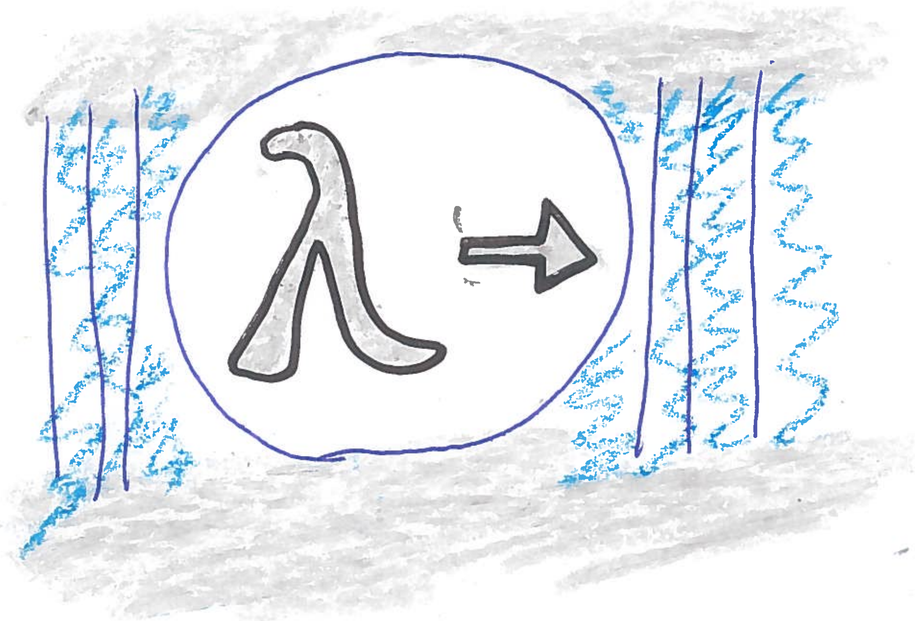
M & M's



Contributions to SESSION TYPES Theory and PRACTICE

NOBUKO YOSHIDA

MARIO



LOGIC AND
COMPUTER
SCIENCE

31

EDITED BY
P. ODIFREDDI

Two Extensions of Curry's Type Inference System

by F. Cardone and M. Coppo

MARIANGIOLA ♥

photo by J.W. Klop <u>1978</u>

ETAPS '02 ???

NY

MD

MARIANGIOLA ♥

photo by J.W. Klop 1978

ETAPS '02 ???

NY

MD

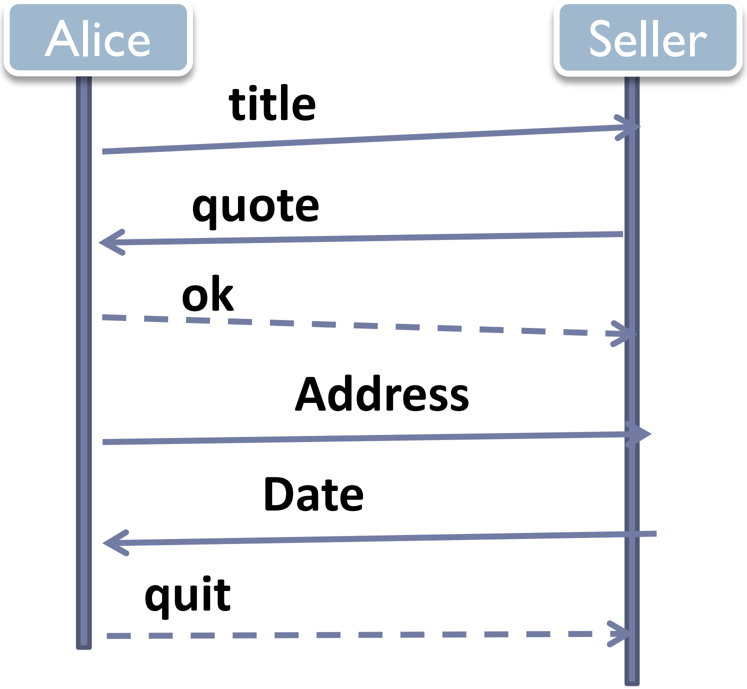
Session Types in a Nutshell

SESSION = STRUCTURED SEQUENCE OF COMMUNICATION

send(int).send(int).receive(bool)

"…Session Types ***structure*** a ***series of interactions*** in a simple and concise syntax and ensure ***type safe communication***."

A Protocol



- Protocol: Buyer-Seller
- Description: Alice buying a book

send(string).receive(int).⊕{ok: send(string).receive(date), quit:end}

receive(string).send(int).&{ok: receive(string).send(date), quit: end}

Are we compatible?

send(int).send(int).receive(bool)

receive(int).receive(int).send(bool)

It is all about duality!

Are we compatible?

receive(int).send(int).receive(bool)

receive(int).receive(int).send(bool)

What is a *type safe communication* ?

Communication safety

- No communication mismatch

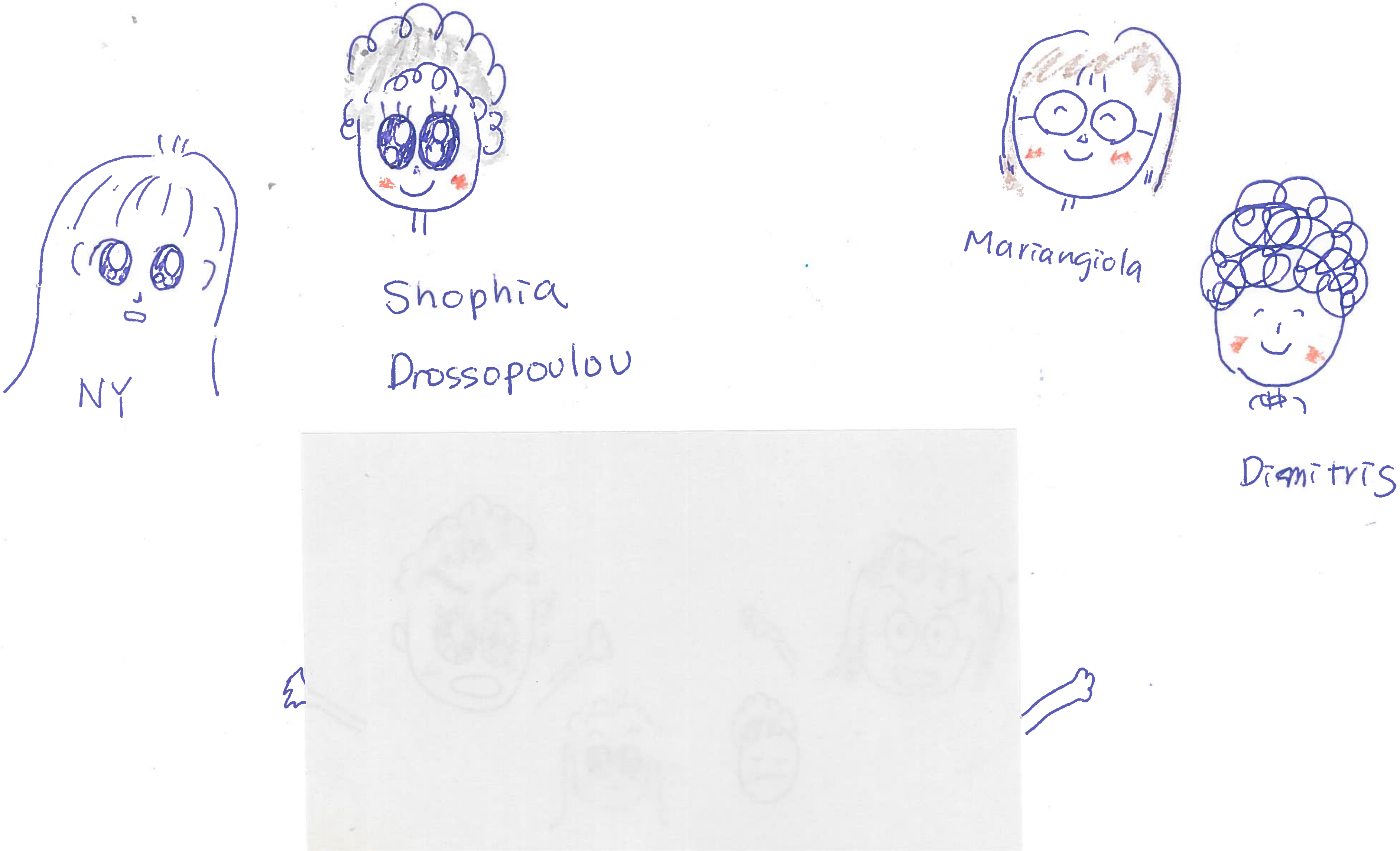
Session Fidelity

- Communications follow the desired protocol

Progress

- No deadlock/stuck in a session

Session Types and Objects



Session Types and Objects

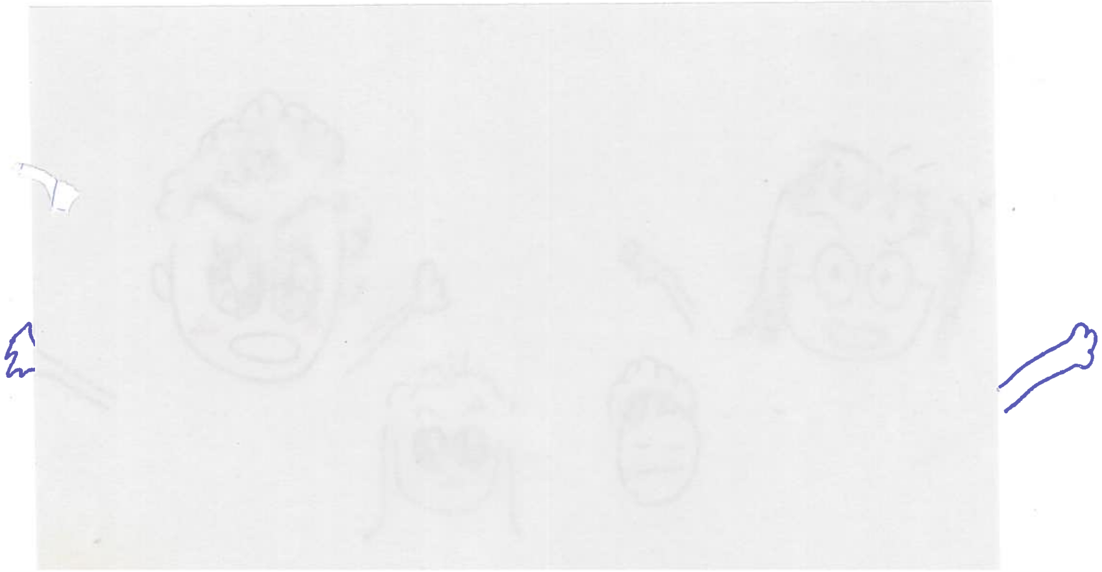
TGC
2005

ECOOP
2006

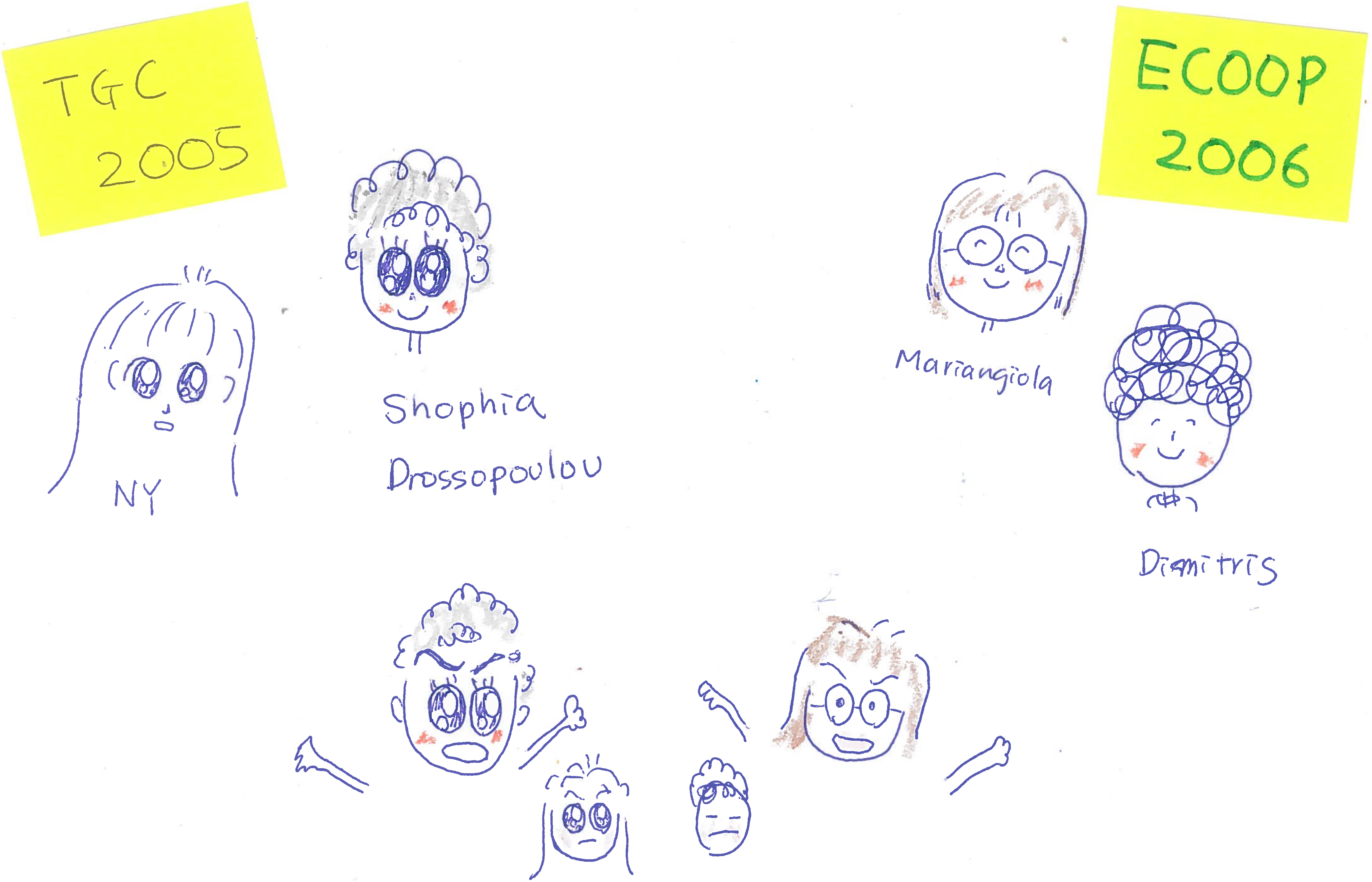
Shophia
Drossopoulou

Mariangiola

Dimitris



Session Types and Objects



Session Types and Objects

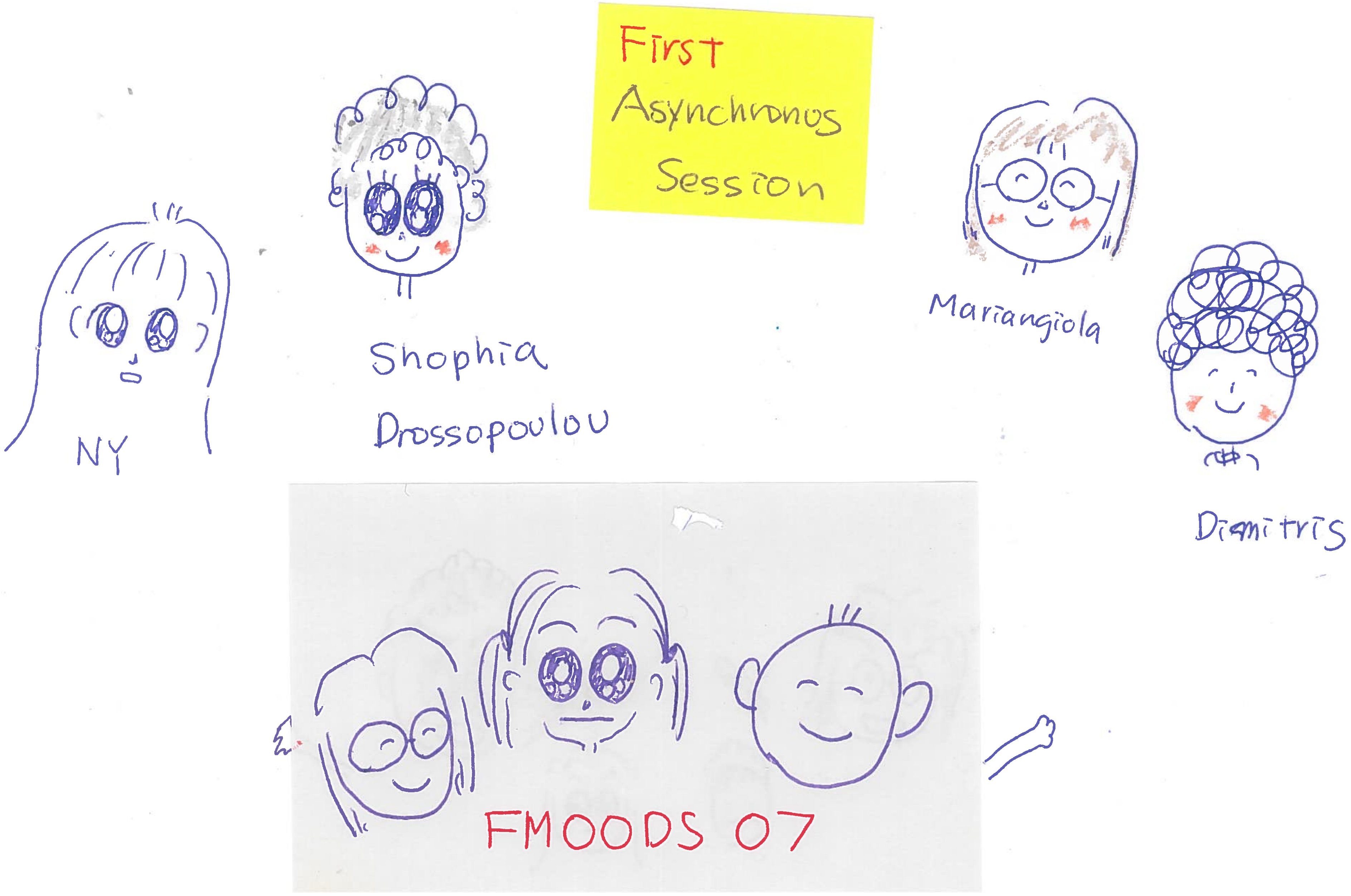
TGC 2005

First OOPL with Session

ECOOP 2006

Dezani 150+ 9th
NY 6th

Session Types and Objects



Session-based Distributed Programming in Java

Raymond Hu, Nobuko Yoshida

Kohei Honda

Imperial College London

Queen Mary University of London

Implementing Customer (4)

```
protocol p {
  begin.
  ![
    !<String>.
    ?(int)
  ]*
  !{
    ACCEPT: {
      !<Address>.
      ?(Date)
    },
    REJECT: { }
  }
}
```

```
SJSocket s = SJSocket.create(p, …);

s.request();
s.outwhile(…) {
  s.send(”PARIS/EUROSTAR”);
  cost = s.receive();
}
if (…) {
  s.outbranch(ACCEPT) {
    s.send(…);
    date = s.receive();
  }
} else {
  s.outbranch(REJECT) { }
}
```

Dialogue between Industry and Academia

Binary Session Types [PARL’94, ESOP’98]

⇓

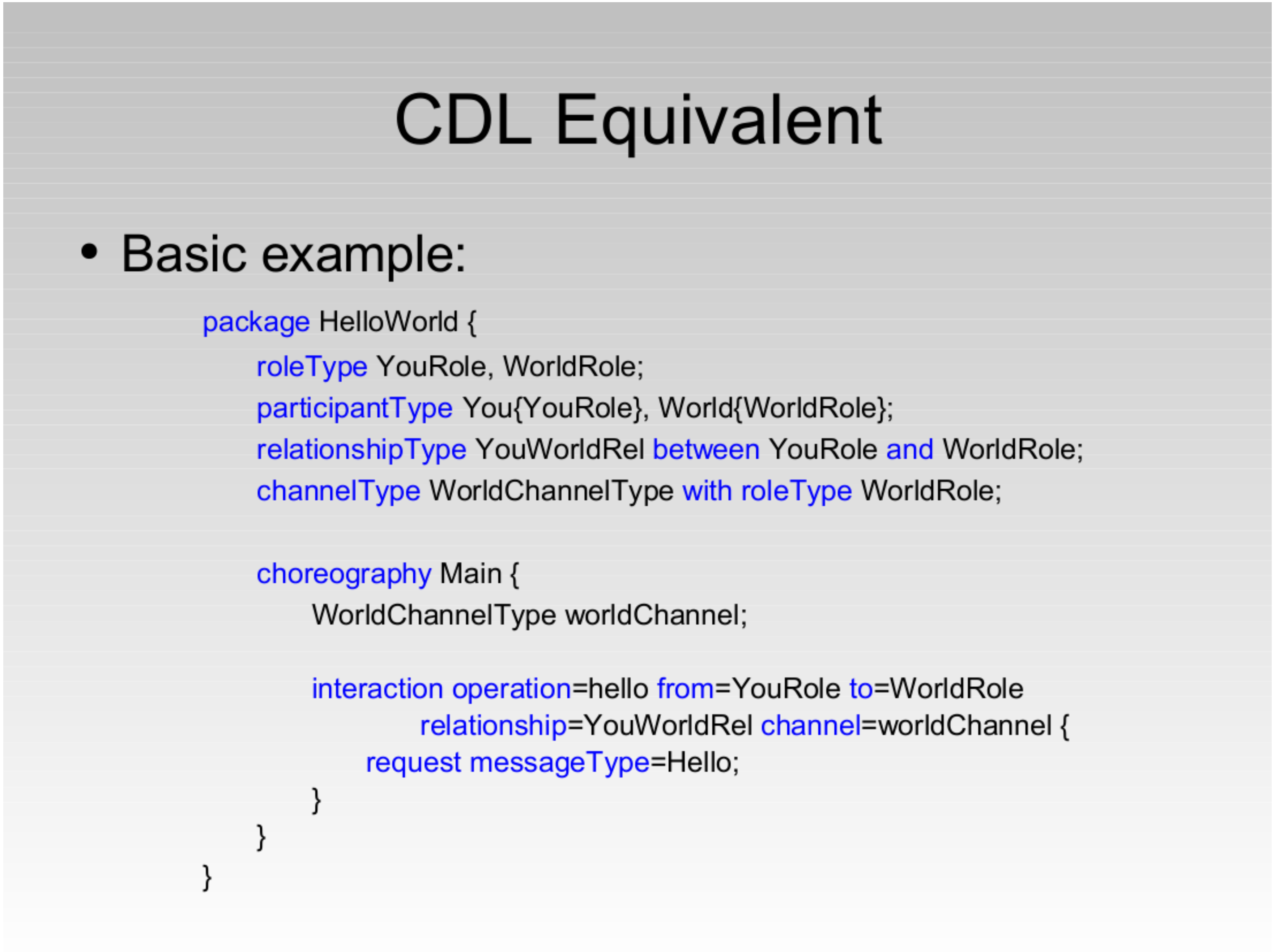
Milner, Honda and Yoshida joined W3C WS-CDL (2002)

⇓

Formalisation of W3C WS-CDL [ESOP’07]

⇓

Scribble at π4 Technology



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]

$\Downarrow$

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

$\Downarrow$

Formalisation of W3C WS-CDL [ESOP’07]

$\Downarrow$

Scribble at π4 Technology

$\Downarrow$

Multiparty Session Types [POPL’08]

$\Downarrow$

Scribble redhat

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]

⇓

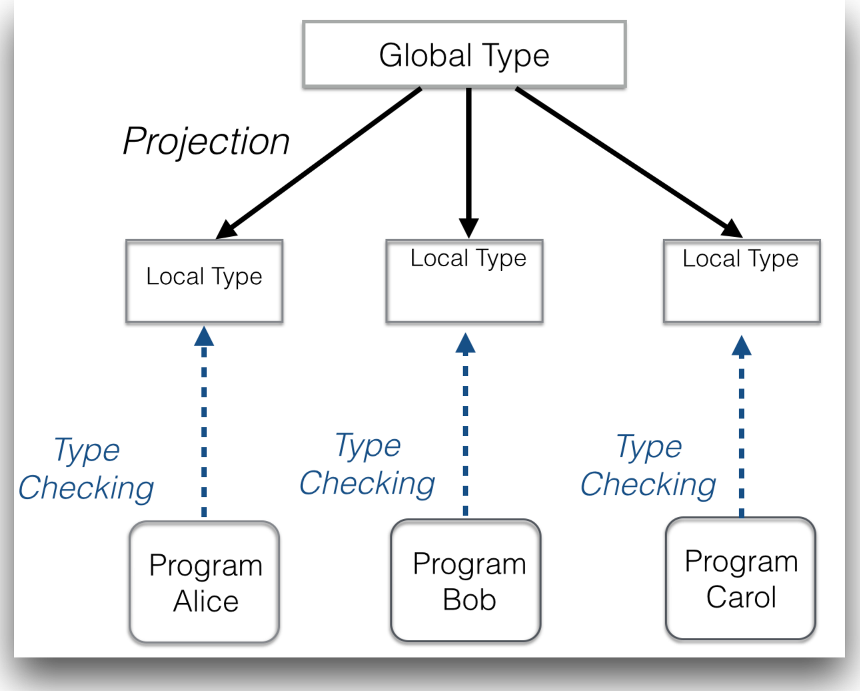
Scribble at π4 Technology

⇓

Multiparty Session Types [POPL'08]

⇓

Session Types Overview



- Global session type

$$G = A \to B : \langle U_1 \rangle . B \to C : \langle U_2 \rangle . C \to A : \langle U_3 \rangle$$

- Local session type
  - Slice of global protocol relevant to one role
  - Mechanically derived from a global protocol

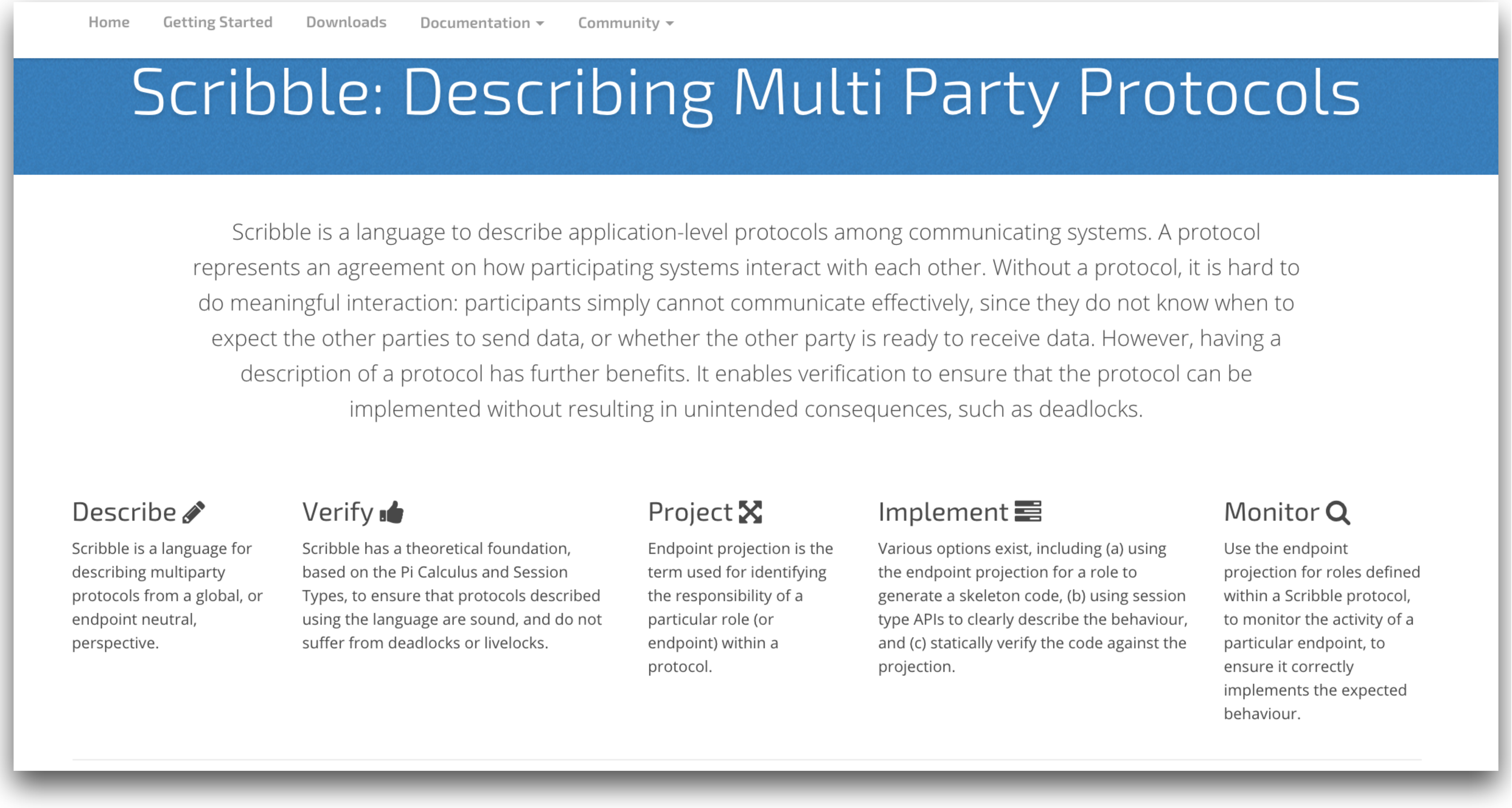
$$T_A = !\langle B, U_1 \rangle . ?\langle C, U_3 \rangle$$

- Process language
  - Execution model of I/O actions by session participants
  - Mechanically derived from a global protocol

$$P_A = a[A](x). x!\langle B, u_1 \rangle . x?(C, y)$$

- (Static) type checking for communication safety and progress

www.scribble.org



Scribble: Describing Multi Party Protocols

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.

Describe

Scribble is a language for describing multiparty protocols from a global, or endpoint neutral, perspective.

Verify

Scribble has a theoretical foundation, based on the Pi Calculus and Session Types, to ensure that protocols described using the language are sound, and do not suffer from deadlocks or livelocks.

Project

Endpoint projection is the term used for identifying the responsibility of a particular role (or endpoint) within a protocol.

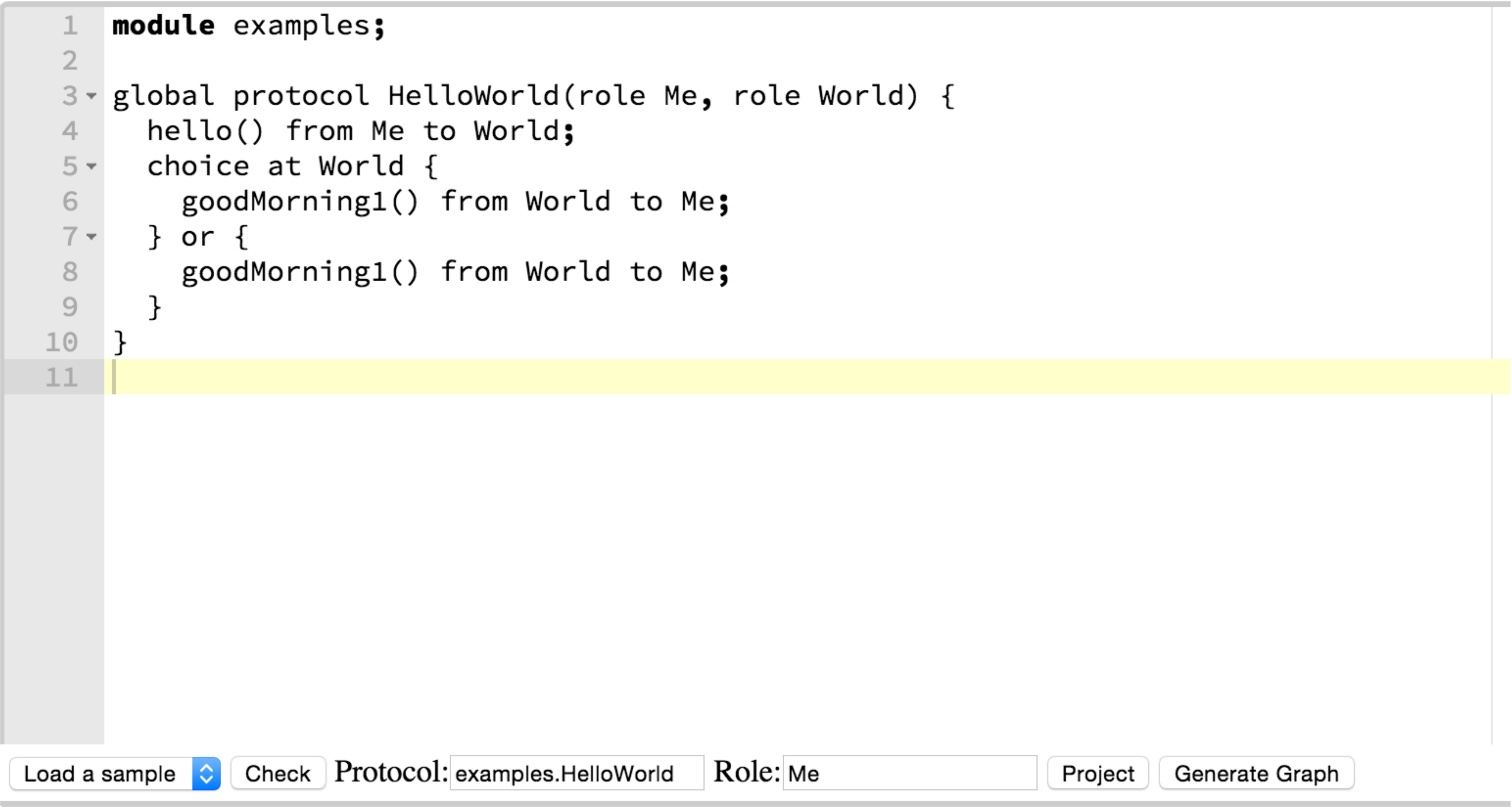
Implement

Various options exist, including (a) using the endpoint projection for a role to generate a skeleton code, (b) using session type APIs to clearly describe the behaviour, and (c) statically verify the code against the projection.

Monitor

Use the endpoint projection for roles defined within a Scribble protocol, to monitor the activity of a particular endpoint, to ensure it correctly implements the expected behaviour.

Online tool : http://scribble.doc.ic.ac.uk/



```
module examples;

global protocol HelloWorld(role Me, role World) {
  hello() from Me to World;
  choice at World {
    goodMorning1() from World to Me;
  } or {
    goodMorning1() from World to Me;
  }
}
```

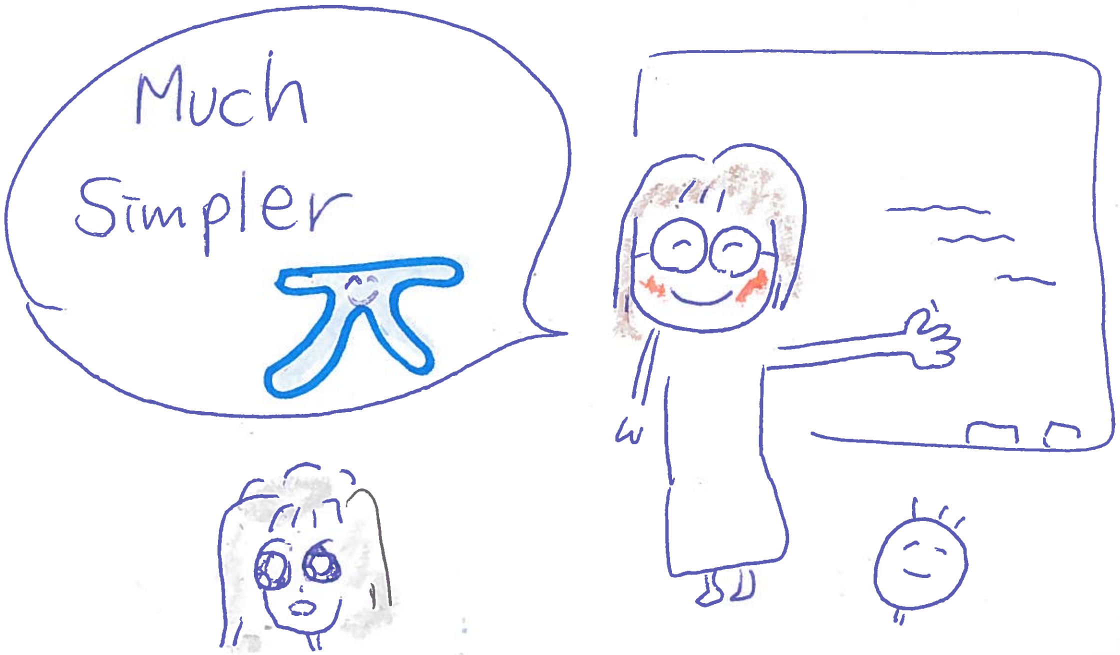
Protocol: examples.HelloWorld Role: Me

Multiparty Session Types (1)

POPL'08

Alice → Bob : K⟨Nat⟩

Bob → Alice : K'⟨Int⟩

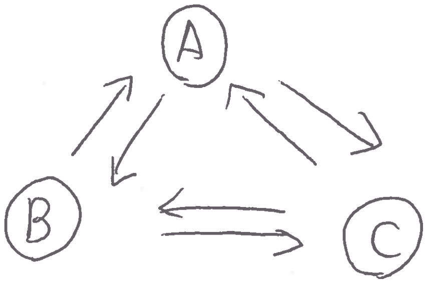


30 Minutes

CONCUR'08

Alice → Bob : ⟨Nat⟩

Bob → Alice : ⟨Int⟩



TCP connection

Multiparty Session Types (1)

POPL'08

$$\text{Alice} \to \text{Bob} : k\langle \text{Nat} \rangle$$

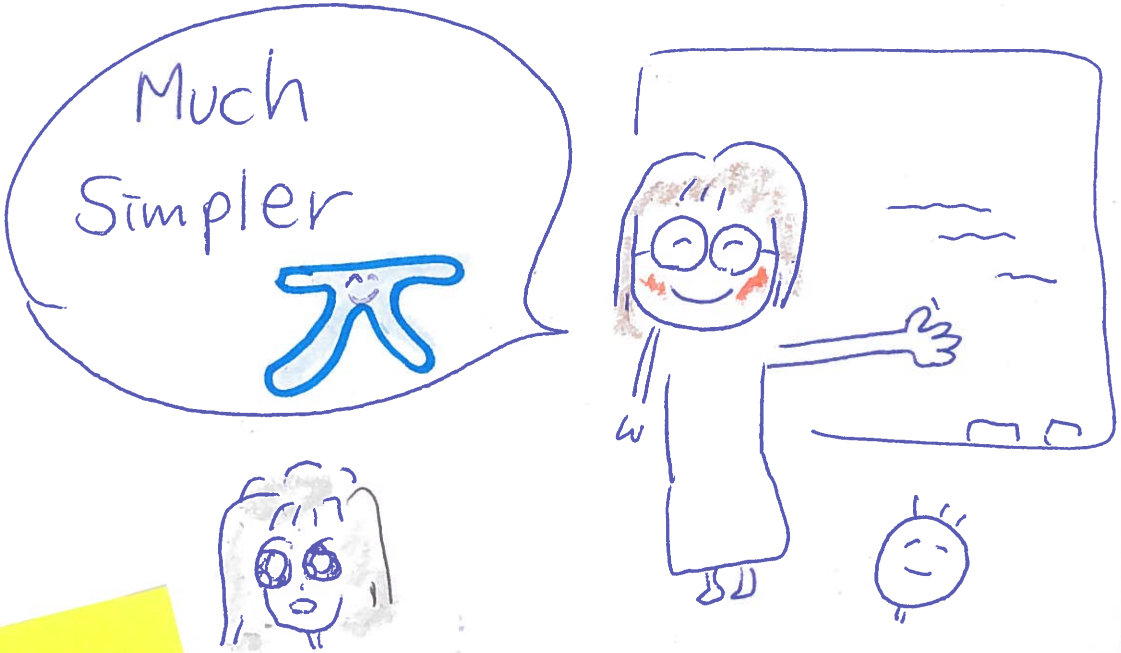
$$\text{Bob} \to \text{Alice} : k'\langle \text{Int} \rangle$$

CONCUR'08

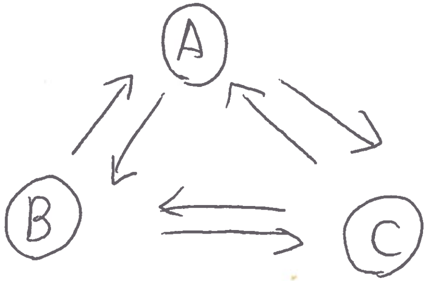
$$\text{Alice} \to \text{Bob} : \langle \text{Nat} \rangle$$

$$\text{Bob} \to \text{Alice} : \langle \text{Int} \rangle$$

No channel!



30 Minutes



TCP connection

Multiparty Session Types (1)

POPL'08

$\text{Alice} \to \text{Bob} : K \langle \text{Nat} \rangle$

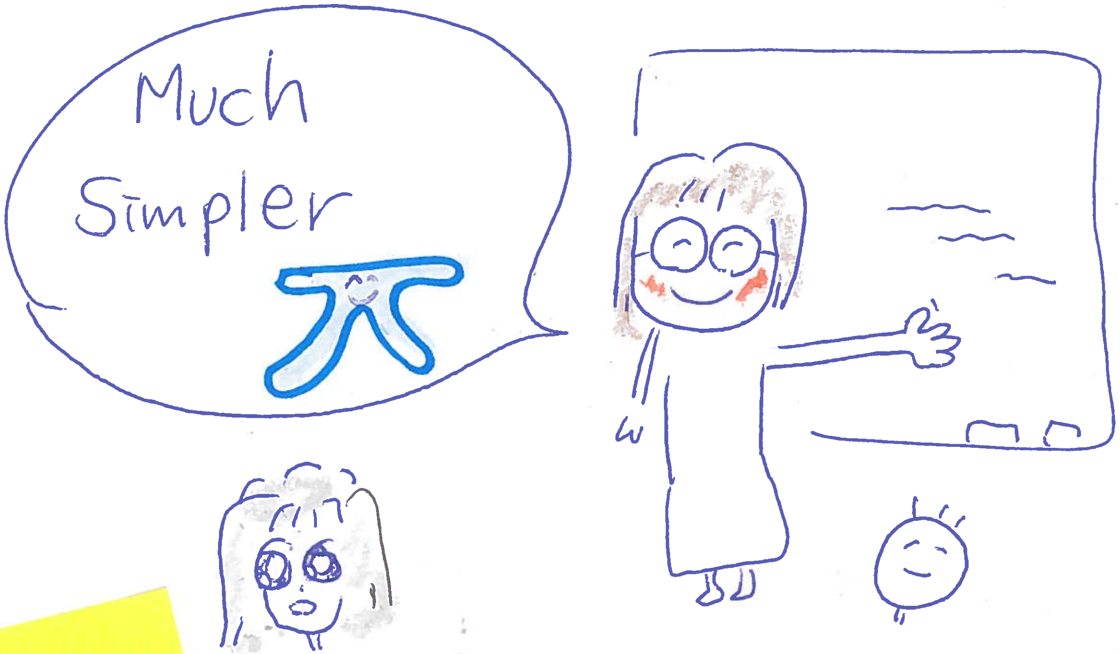
$\text{Bob} \to \text{Alice} : K' \langle \text{Int} \rangle$

CONCUR'08

$\text{Alice} \to \text{Bob} : \langle \text{Nat} \rangle$

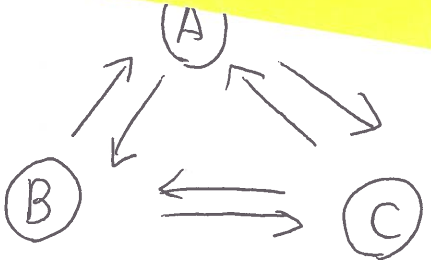
$\text{Bob} \to \text{Alice} : \langle \text{Int} \rangle$

No channel!



My 4th
180+

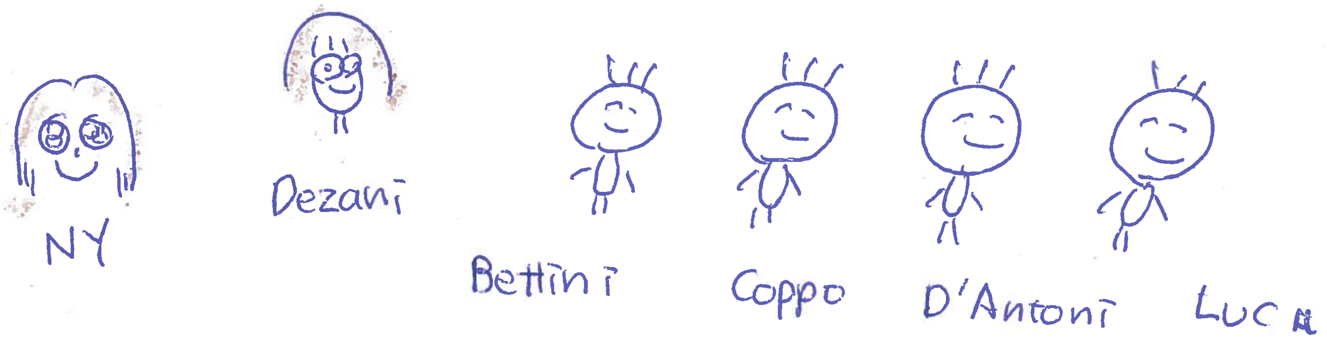
30 Minutes



TCP
connection

CONCUR'08

Global Progress of MPST



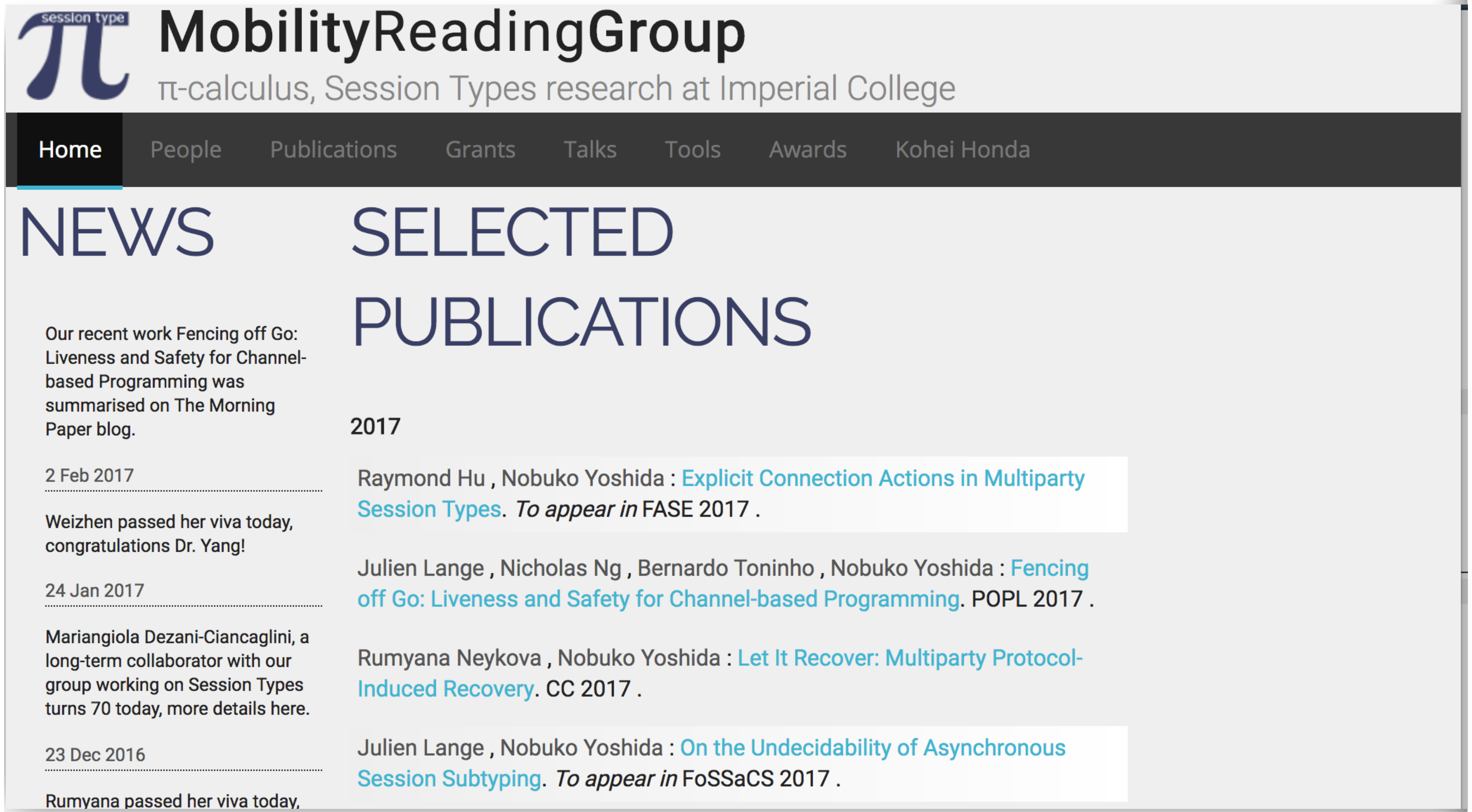
COORDINATION'13

Inference of Global Progress in MPST

MSCS

Gentle Introduction of MPST

Us ∈ Mobility Research Group



MobilityReadingGroup

π-calculus, Session Types research at Imperial College

NEWS

Our recent work Fencing off Go: Liveness and Safety for Channel-based Programming was summarised on The Morning Paper blog.

2 Feb 2017

Weizhen passed her viva today, congratulations Dr. Yang!

24 Jan 2017

Mariangiola Dezani-Ciancaglini, a long-term collaborator with our group working on Session Types turns 70 today, more details here.

23 Dec 2016

Rumyana passed her viva today,

SELECTED PUBLICATIONS

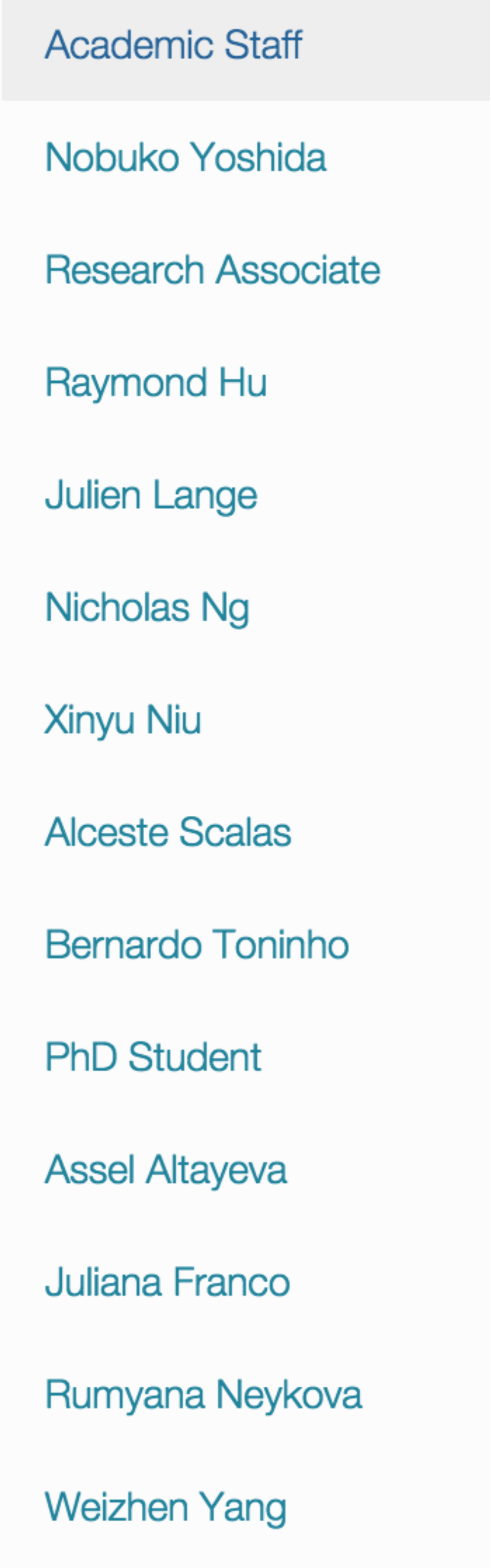
2017

Raymond Hu , Nobuko Yoshida : Explicit Connection Actions in Multiparty Session Types. *To appear in* FASE 2017 .

Julien Lange , Nicholas Ng , Bernardo Toninho , Nobuko Yoshida : Fencing off Go: Liveness and Safety for Channel-based Programming. POPL 2017 .

Rumyana Neykova , Nobuko Yoshida : Let It Recover: Multiparty Protocol-Induced Recovery. CC 2017 .

Julien Lange , Nobuko Yoshida : On the Undecidability of Asynchronous Session Subtyping. *To appear in* FoSSaCS 2017 .



Academic Staff

Nobuko Yoshida

Research Associate

Raymond Hu

Julien Lange

Nicholas Ng

Xinyu Niu

Alceste Scalas

Bernardo Toninho

PhD Student

Assel Altayeva

Juliana Franco

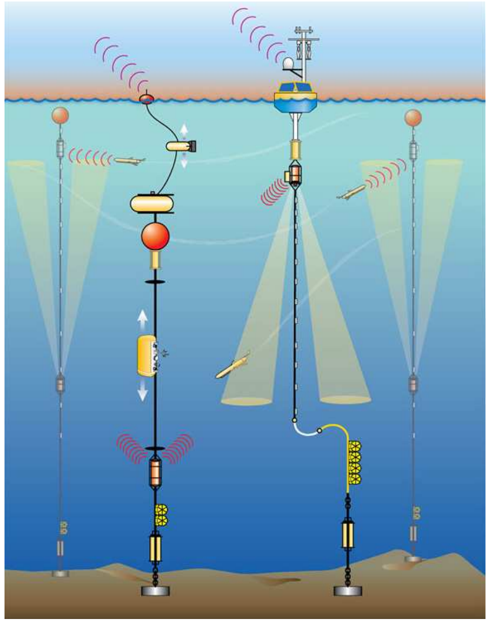
Rumyana Neykova

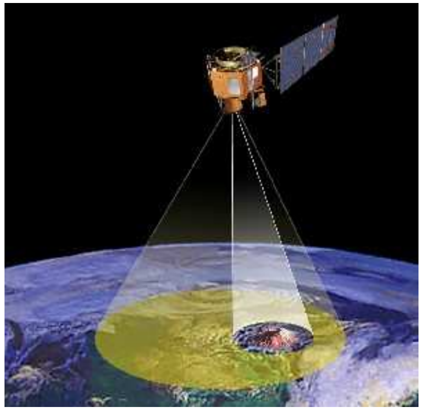
Weizhen Yang

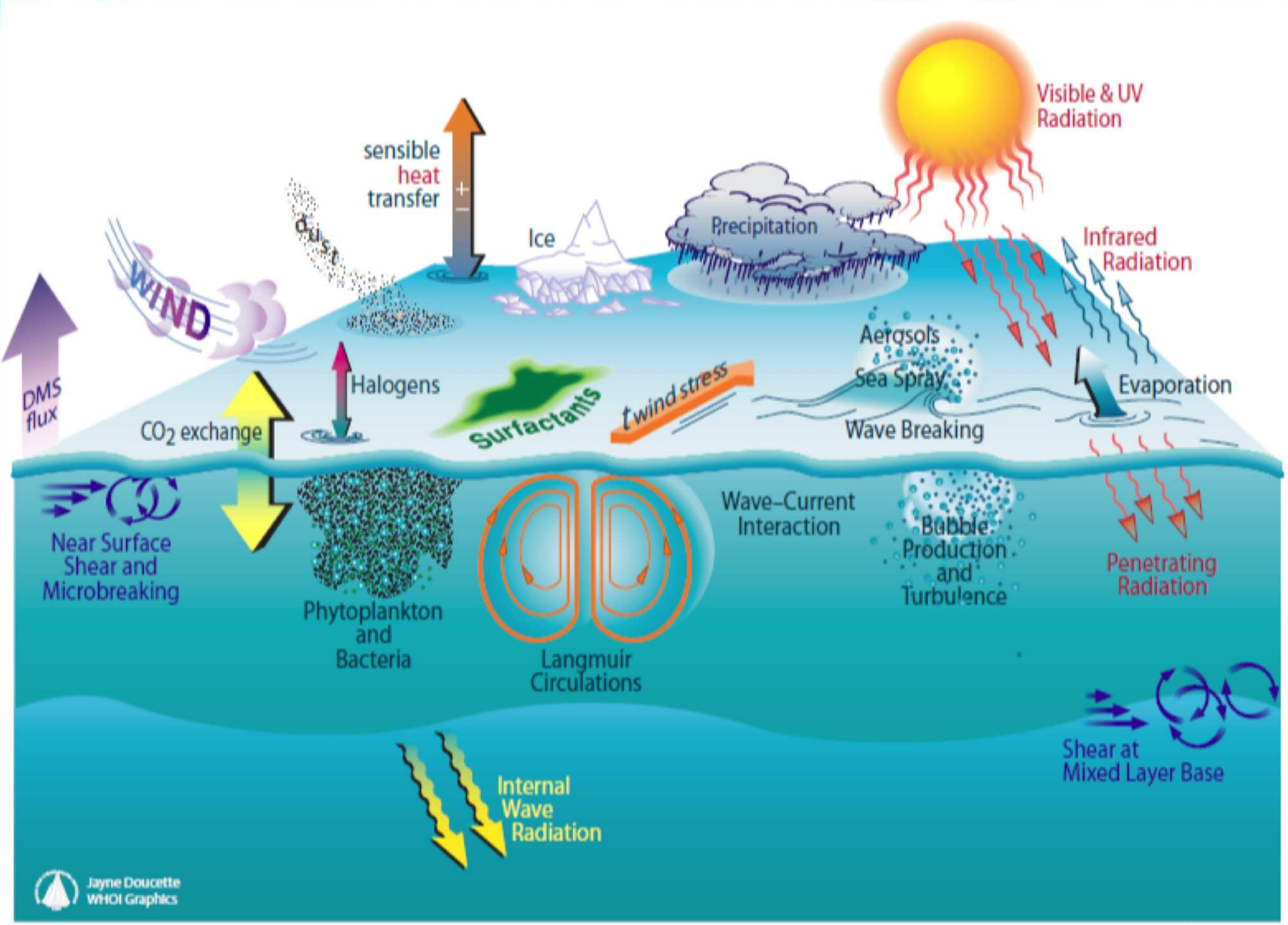
http://mrg.doc.ic.ac.uk/

Ocean Observatories Initiative

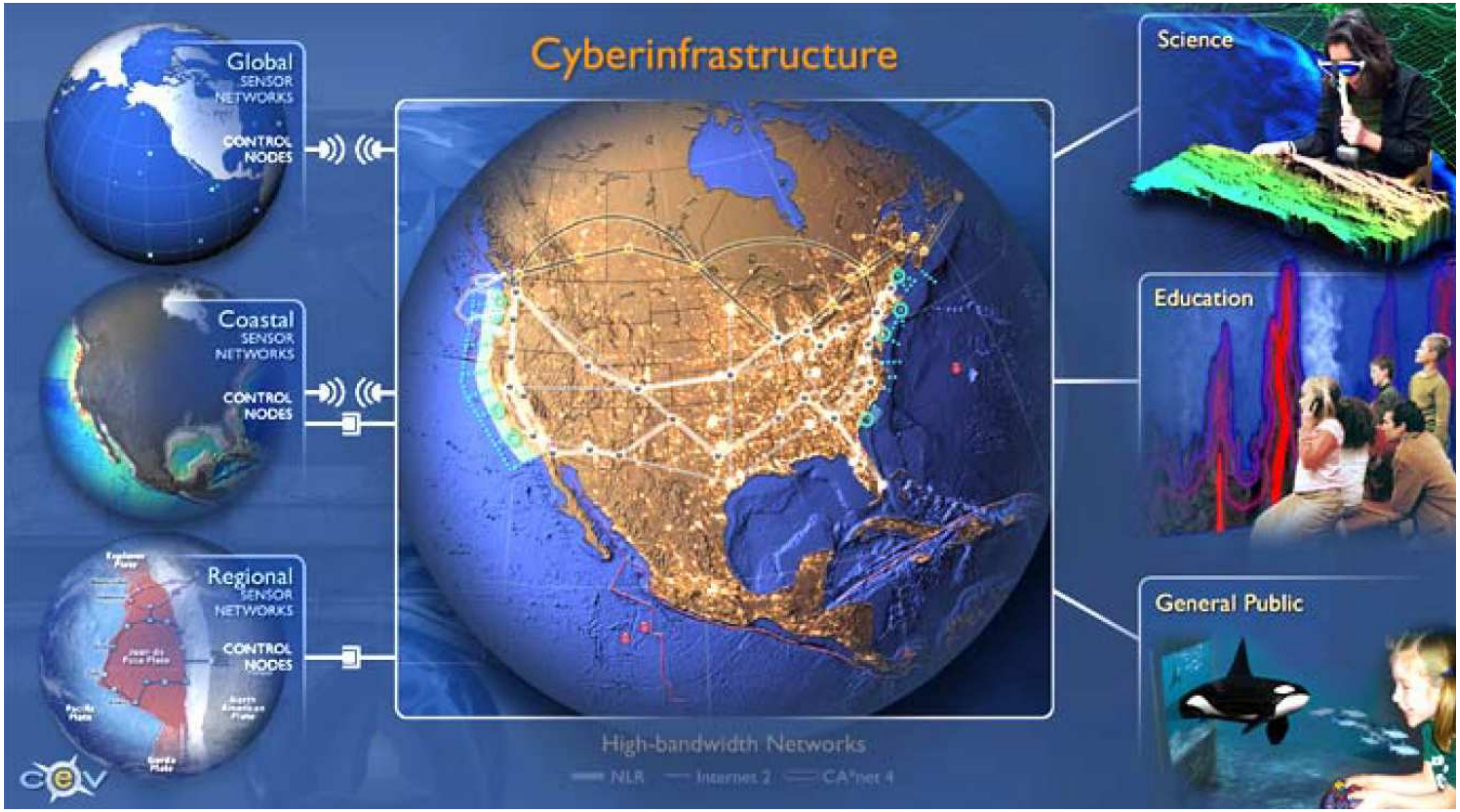
- A NSF project (400M$, 5 Years) to build a cyberinfrastructure for observing oceans around US and beyond.
- Real-time sensor data constantly coming from both off-shore and on-shore (e.g. buoys, submarines, under-water cameras, satellites), transmitted via high-speed networks.

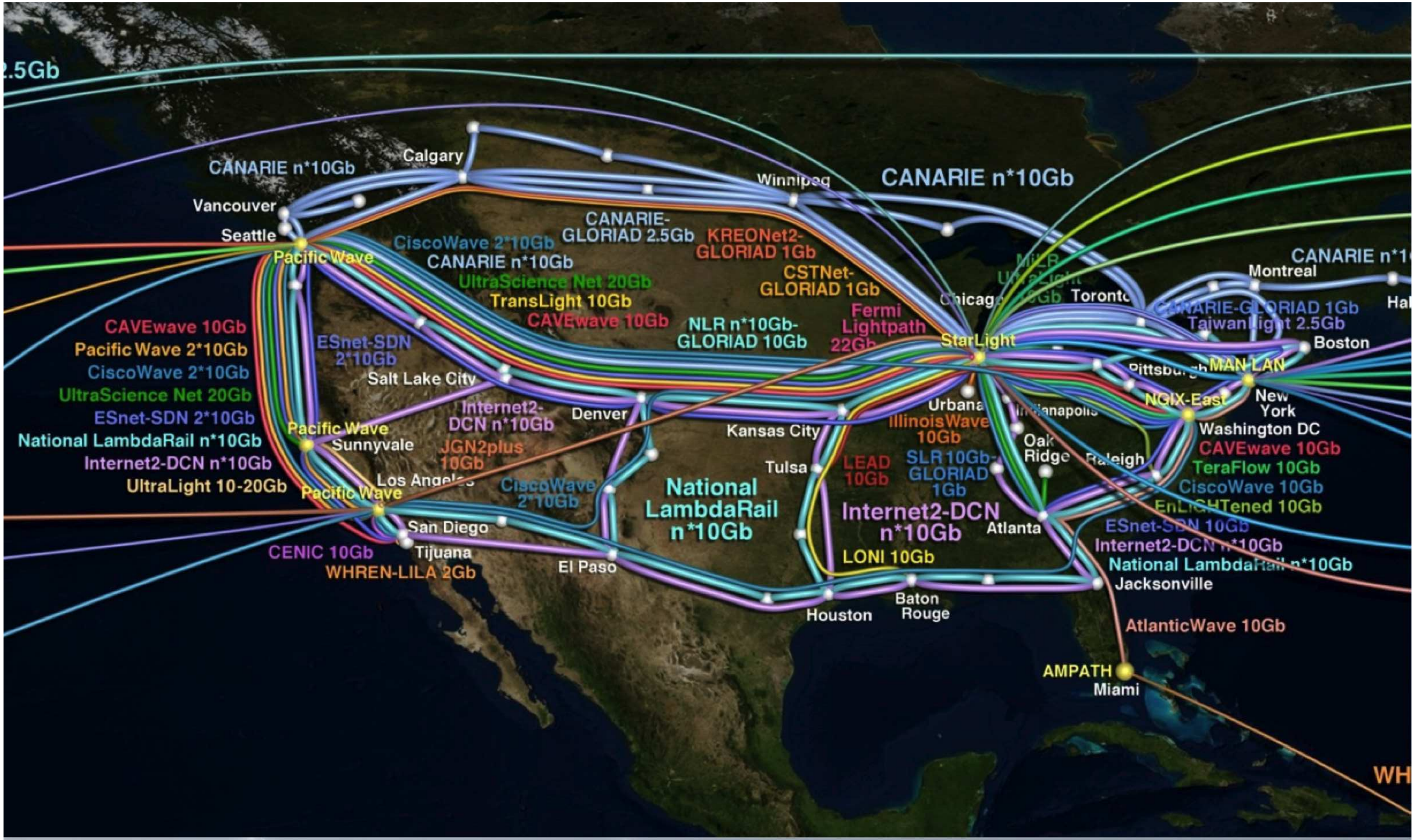





Visible & UV Radiation
sensible heat transfer
+
–
dust
Ice
Precipitation
Infrared Radiation
WIND
DMS flux
Aerosols
Sea Spray
Halogens
Evaporation
Surfactants
τ wind stress
CO2 exchange
Wave Breaking
Wave–Current Interaction
Near Surface Shear and Microbreaking
Bubble Production and Turbulence
Penetrating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

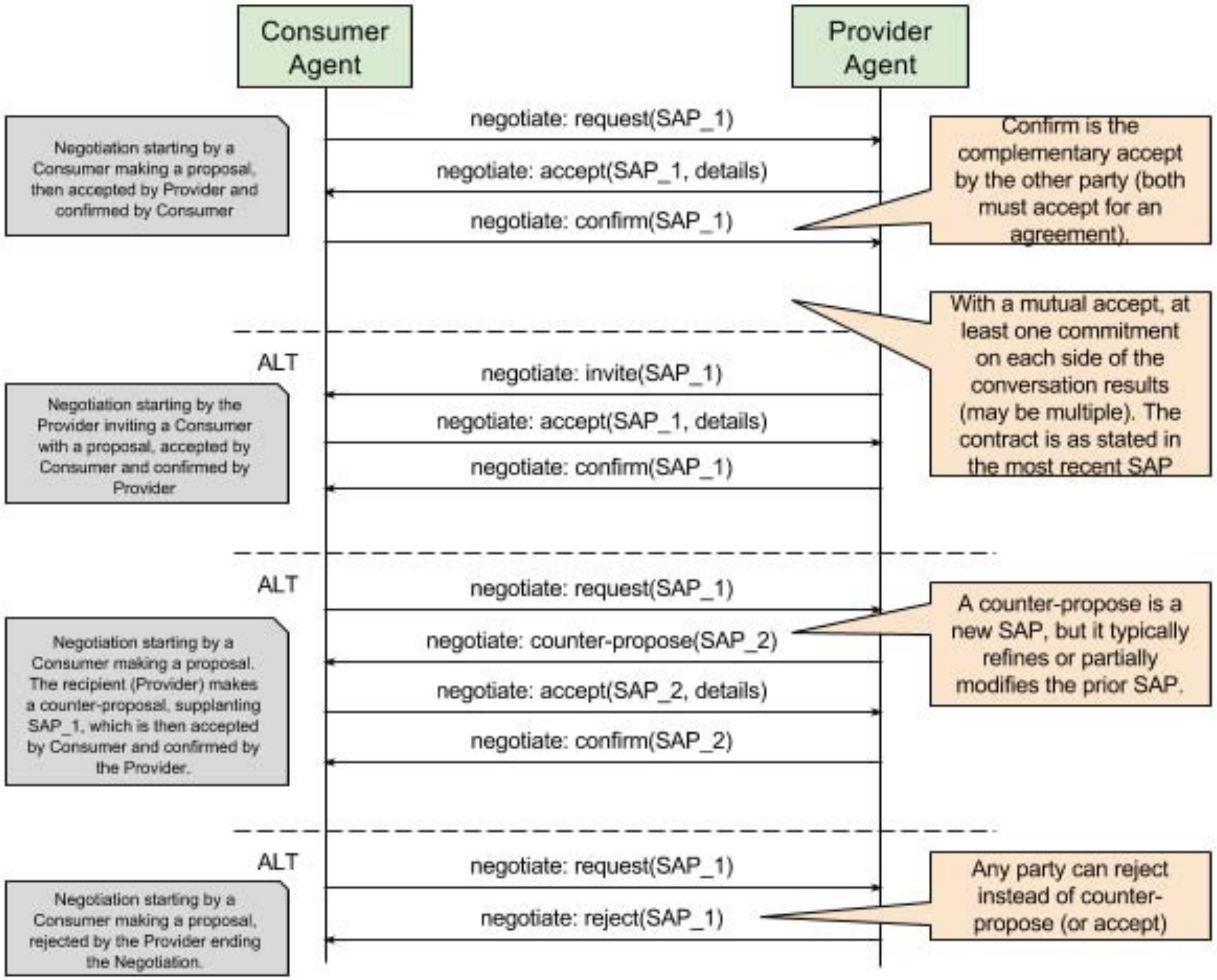
Ocean Observatories Initiative





Ocean Observatories Initiative

OOI agent negotiation 1/5


Consumer Agent
Provider Agent
negotiate: request(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: invite(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: request(SAP_1)
negotiate: counter-propose(SAP_2)
negotiate: accept(SAP_2, details)
negotiate: confirm(SAP_2)
ALT
negotiate: request(SAP_1)
negotiate: reject(SAP_1)

Negotiation starting by a Consumer making a proposal, then accepted by Provider and confirmed by Consumer

Confirm is the complementary accept by the other party (both must accept for an agreement).

With a mutual accept, at least one commitment on each side of the conversation results (may be multiple). The contract is as stated in the most recent SAP

Negotiation starting by the Provider inviting a Consumer with a proposal, accepted by Consumer and confirmed by Provider

Negotiation starting by a Consumer making a proposal. The recipient (Provider) makes a counter-proposal, supplanting SAP_1, which is then accepted by Consumer and confirmed by the Provider.

A counter-propose is a new SAP, but it typically refines or partially modifies the prior SAP.

Negotiation starting by a Consumer making a proposal, rejected by the Provider ending the Negotiation.

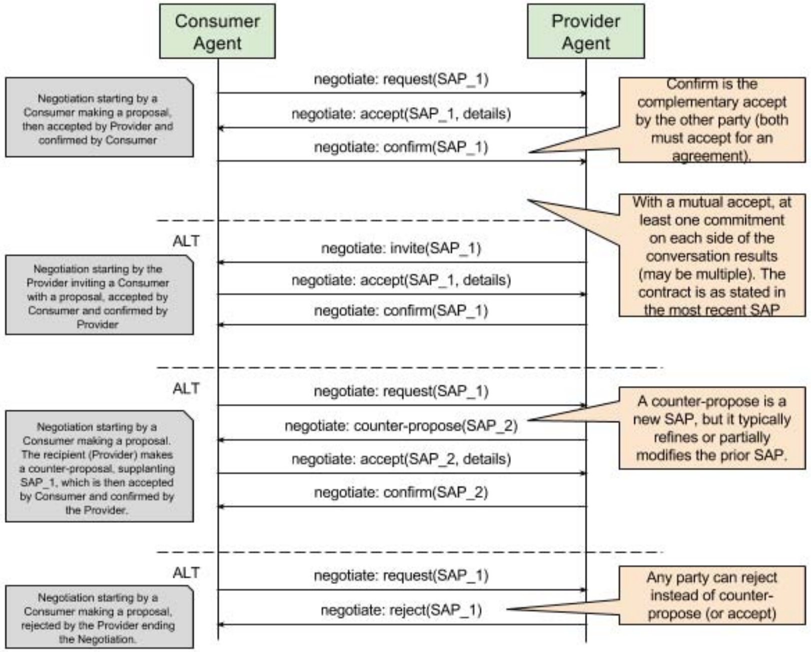
Any party can reject instead of counter-propose (or accept)

- `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) {
```



Consumer Agent
Provider Agent
negotiate: request(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: invite(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: request(SAP_1)
negotiate: counter-propose(SAP_2)
negotiate: accept(SAP_2, details)
negotiate: confirm(SAP_2)
ALT
negotiate: request(SAP_1)
negotiate: reject(SAP_1)

Negotiation starting by a Consumer making a proposal, then accepted by Provider and confirmed by Consumer

Confirm is the complementary accept by the other party (both must accept for an agreement).

With a mutual accept, at least one commitment on each side of the conversation results (may be multiple). The contract is as stated in the most recent SAP

Negotiation starting by the Provider inviting a Consumer with a proposal, accepted by Consumer and confirmed by Provider

Negotiation starting by a Consumer making a proposal. The recipient (Provider) makes a counter-proposal, supplanting SAP_1, which is then accepted by Consumer and confirmed by the Provider.

A counter-propose is a new SAP, but it typically refines or partially modifies the prior SAP.

Negotiation starting by a Consumer making a proposal, rejected by the Provider ending the Negotiation.

Any party can reject instead of counter-propose (or accept)

```
}
```

OOI agent negotiation 3/5 (choice)

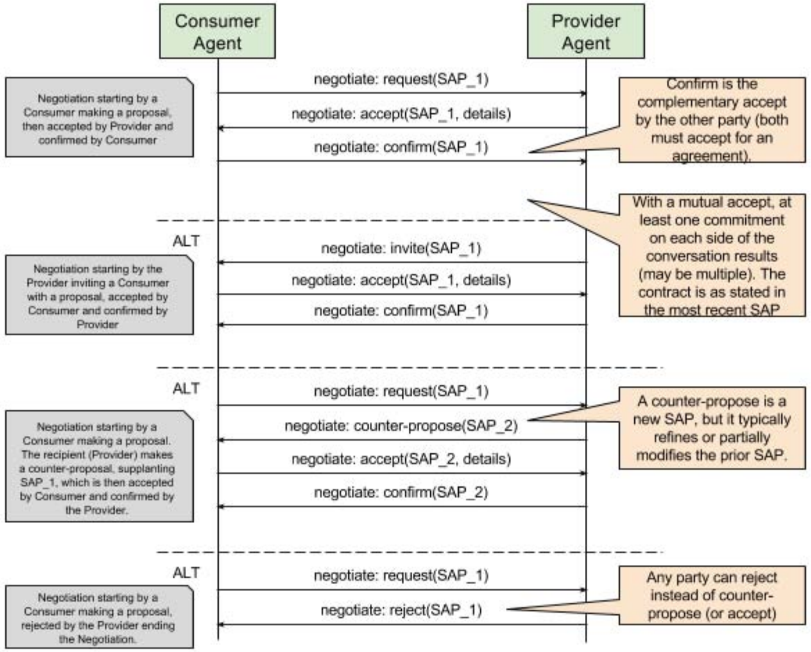
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

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;




} }
```


Consumer Agent
Provider Agent
negotiate: request(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: invite(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: request(SAP_1)
negotiate: counter-propose(SAP_2)
negotiate: accept(SAP_2, details)
negotiate: confirm(SAP_2)
ALT
negotiate: request(SAP_1)
negotiate: reject(SAP_1)

Negotiation starting by a Consumer making a proposal, then accepted by Provider and confirmed by Consumer

Confirm is the complementary accept by the other party (both must accept for an agreement).

With a mutual accept, at least one commitment on each side of the conversation results (may be multiple). The contract is as stated in the most recent SAP

Negotiation starting by the Provider inviting a Consumer with a proposal, accepted by Consumer and confirmed by Provider

Negotiation starting by a Consumer making a proposal. The recipient (Provider) makes a counter-proposal, supplanting SAP_1, which is then accepted by Consumer and confirmed by the Provider.

A counter-propose is a new SAP, but it typically refines or partially modifies the prior SAP.

Negotiation starting by a Consumer making a proposal, rejected by the Provider ending the Negotiation.

Any party can reject instead of counter-propose (or accept)

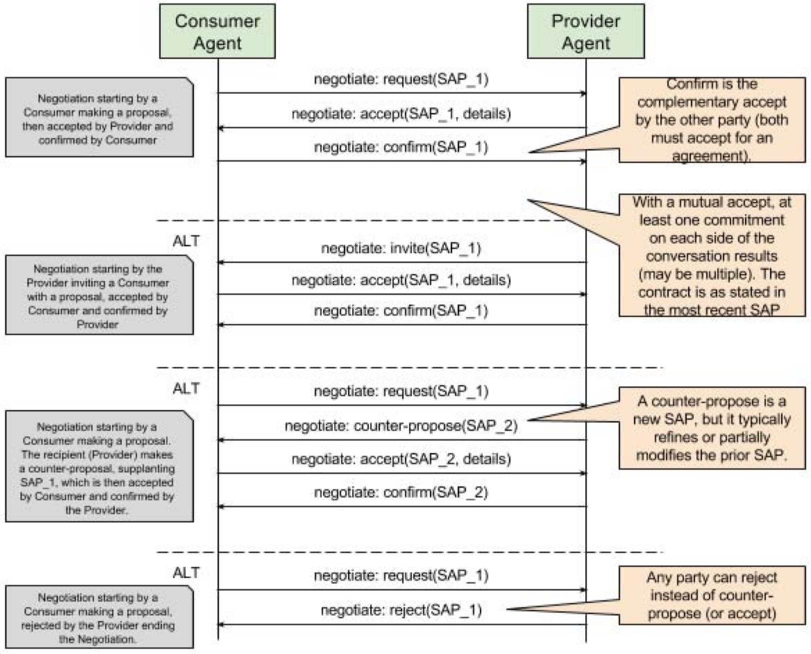
OOI agent negotiation 4/5

```
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;

} } }
```


Consumer Agent
Provider Agent
negotiate: request(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: invite(SAP_1)
negotiate: accept(SAP_1, details)
negotiate: confirm(SAP_1)
ALT
negotiate: request(SAP_1)
negotiate: counter-propose(SAP_2)
negotiate: accept(SAP_2, details)
negotiate: confirm(SAP_2)
ALT
negotiate: request(SAP_1)
negotiate: reject(SAP_1)

Negotiation starting by a Consumer making a proposal, then accepted by Provider and confirmed by Consumer

Confirm is the complementary accept by the other party (both must accept for an agreement).

With a mutual accept, at least one commitment on each side of the conversation results (may be multiple). The contract is as stated in the most recent SAP

Negotiation starting by the Provider inviting a Consumer with a proposal, accepted by Consumer and confirmed by Provider

Negotiation starting by a Consumer making a proposal. The recipient (Provider) makes a counter-proposal, supplanting SAP_1, which is then accepted by Consumer and confirmed by the Provider.

A counter-propose is a new SAP, but it typically refines or partially modifies the prior SAP.

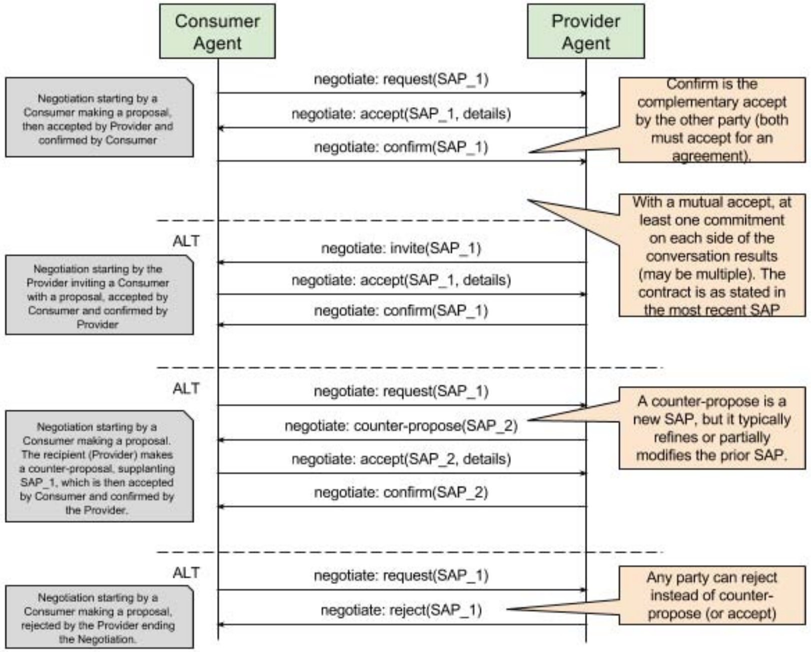
Negotiation starting by a Consumer making a proposal, rejected by the Provider ending the Negotiation.

Any party can reject instead of counter-propose (or accept)

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;
      }
    }
```



Consumer Agent

Provider Agent

Negotiation starting by a Consumer making a proposal, then accepted by Provider and confirmed by Consumer

negotiate: request(SAP_1)

negotiate: accept(SAP_1, details)

negotiate: confirm(SAP_1)

Confirm is the complementary accept by the other party (both must accept for an agreement).

ALT

Negotiation starting by the Provider inviting a Consumer with a proposal, accepted by Consumer and confirmed by Provider

negotiate: invite(SAP_1)

negotiate: accept(SAP_1, details)

negotiate: confirm(SAP_1)

With a mutual accept, at least one commitment on each side of the conversation results (may be multiple). The contract is as stated in the most recent SAP

ALT

Negotiation starting by a Consumer making a proposal. The recipient (Provider) makes a counter-proposal, supplanting SAP_1, which is then accepted by Consumer and confirmed by the Provider.

negotiate: request(SAP_1)

negotiate: counter-propose(SAP_2)

negotiate: accept(SAP_2, details)

negotiate: confirm(SAP_2)

A counter-propose is a new SAP, but it typically refines or partially modifies the prior SAP.

ALT

Negotiation starting by a Consumer making a proposal, rejected by the Provider ending the Negotiation.

negotiate: request(SAP_1)

negotiate: reject(SAP_1)

Any party can reject instead of counter-propose (or accept)

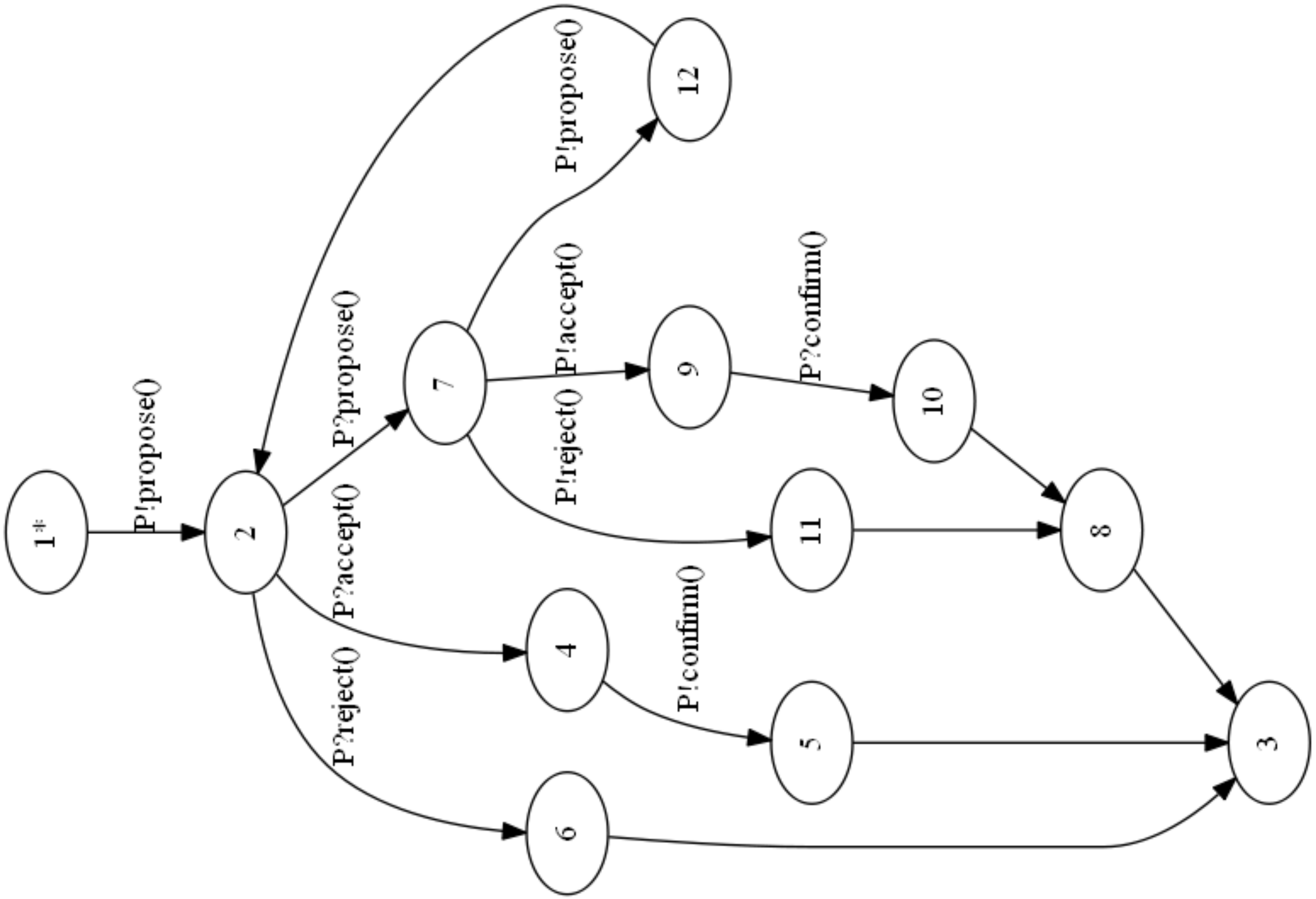
```
  }
```

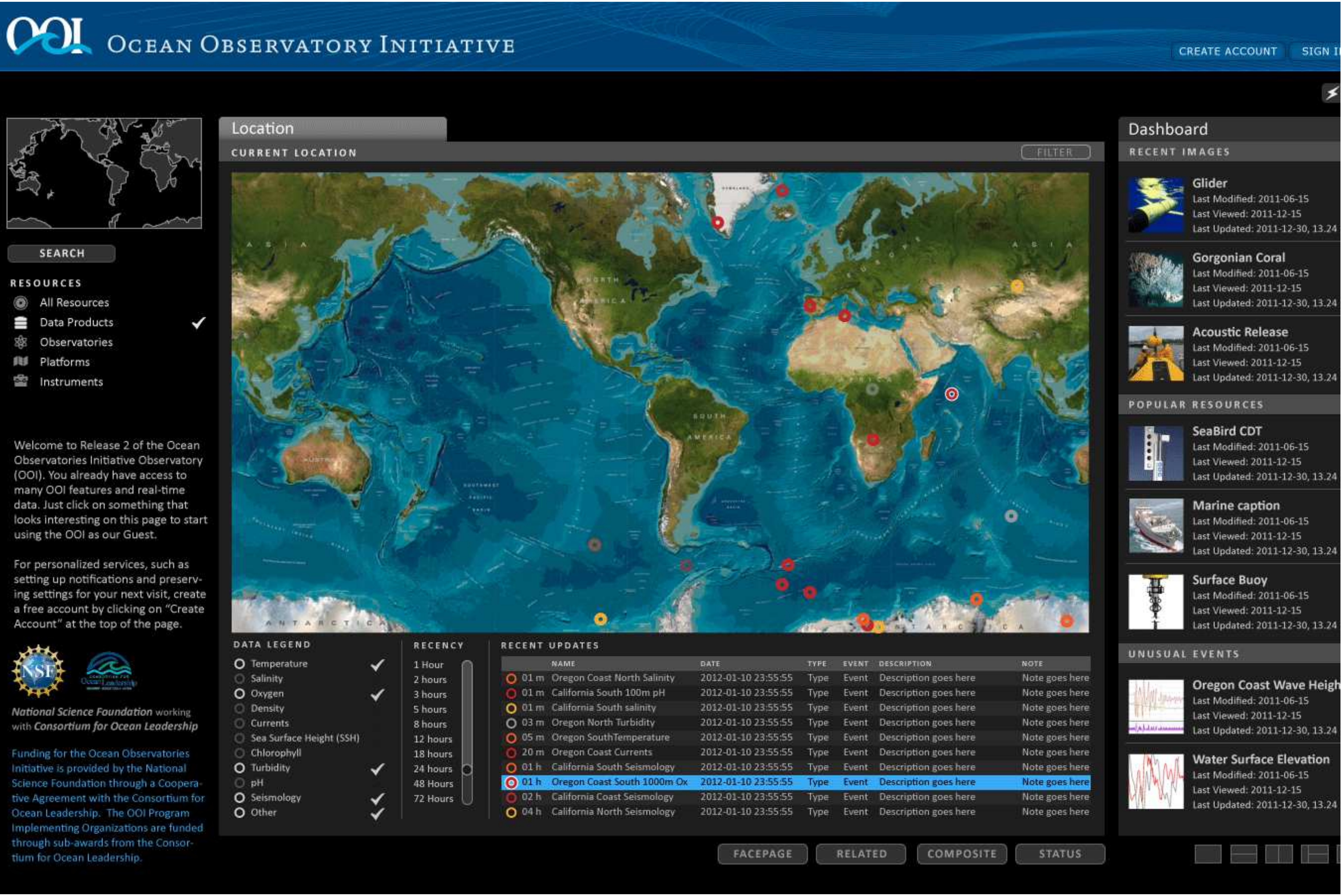
Local protocol projection (Negotiation Consumer)

```
// 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)





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.

DATA LEGEND

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

RECENCY

- 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 SouthTemperature	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

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

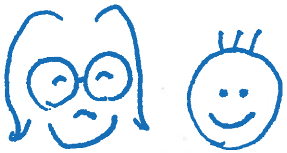
Oregon Coast Wave Heigh
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

Water Surface Elevation
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24

DOZEN MONITORING and ADAPTATIONS

based on Session Types

by Coppo & Dezani



[PPDP'14]

[PDP'14]

[SOCA'15]

[PLACES'16]

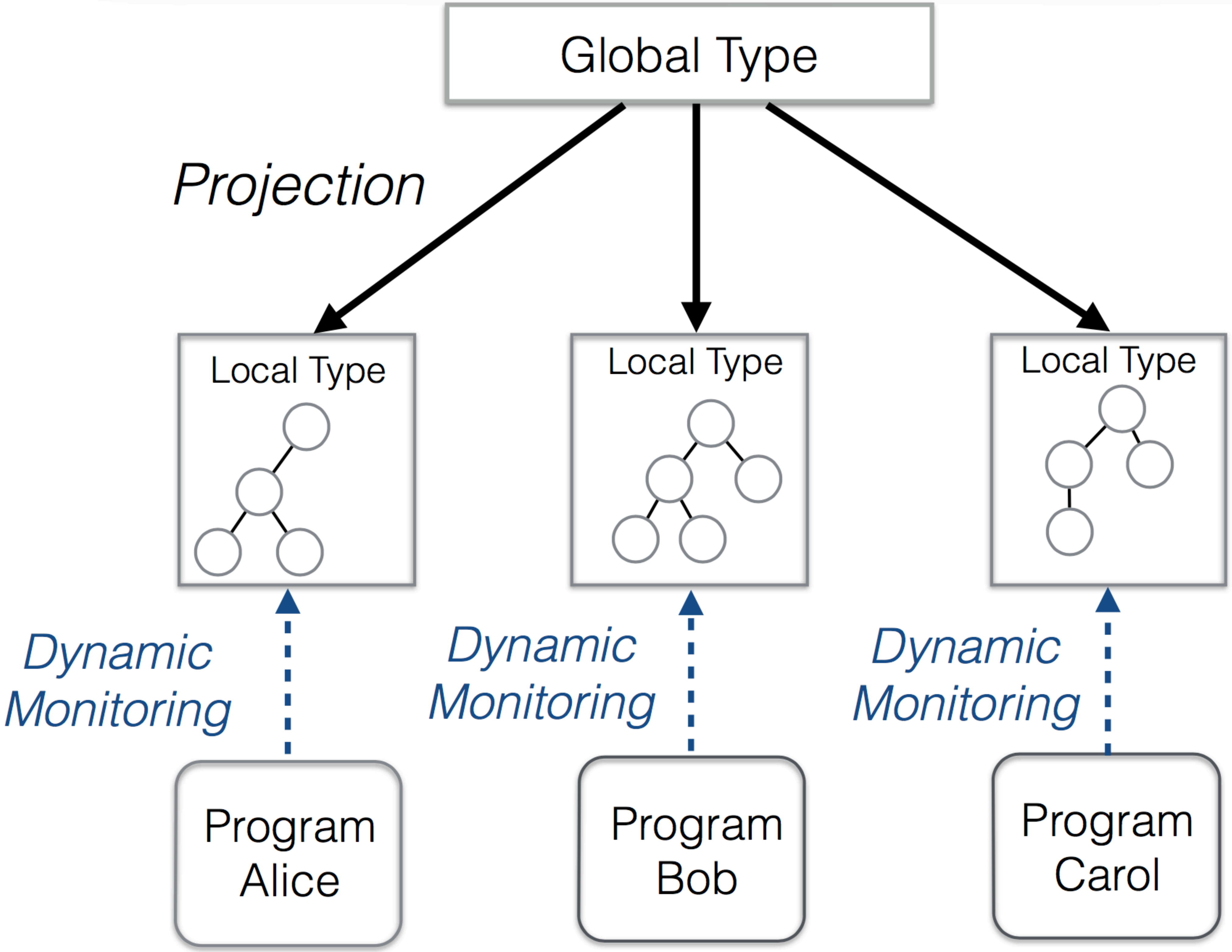
[FAC'16]

[JLAMP'16]

etc etc

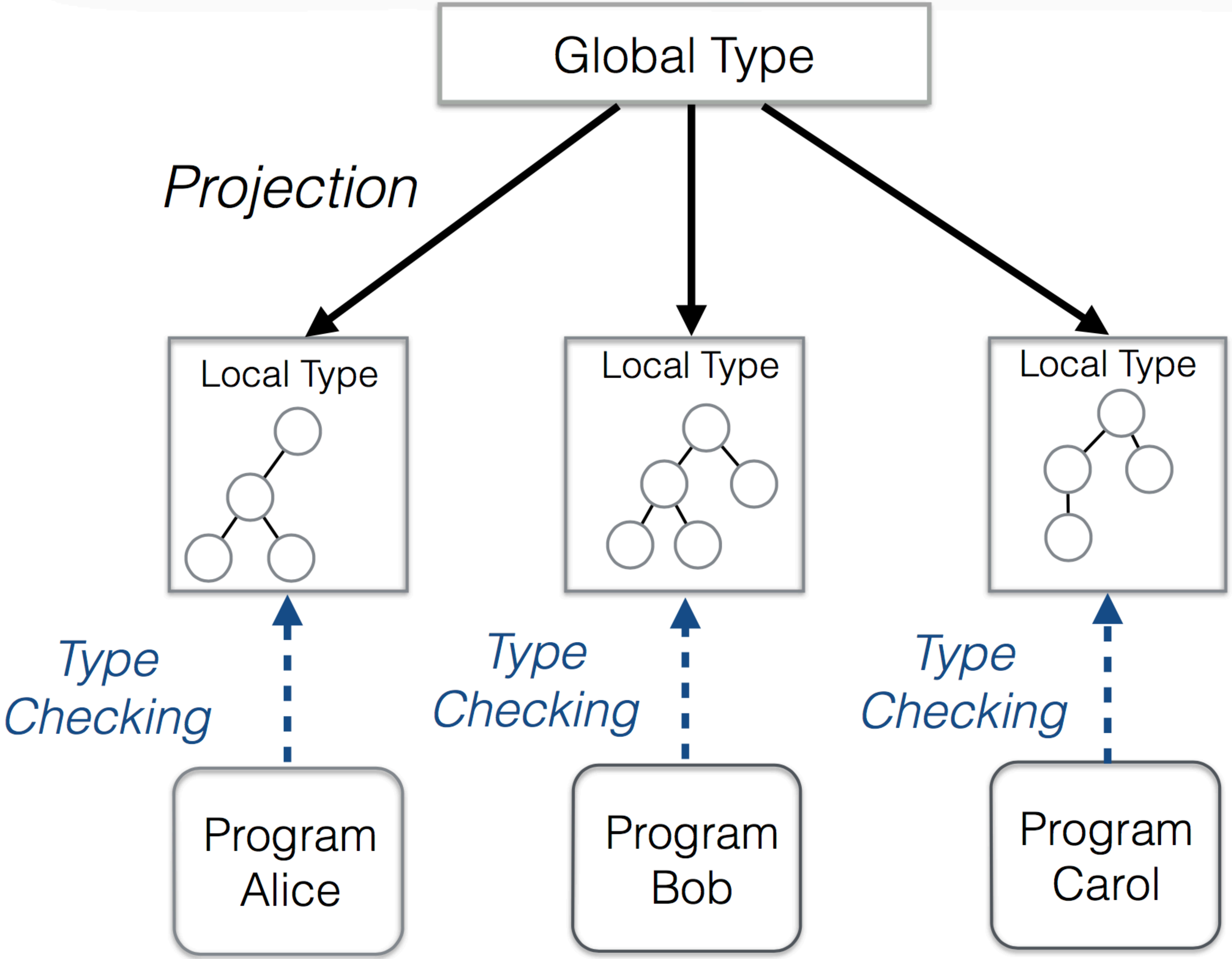
Dynamic Monitoring

[RV'13, COORDINATION'14, FMSD'15, LMCS'17, CC'17]



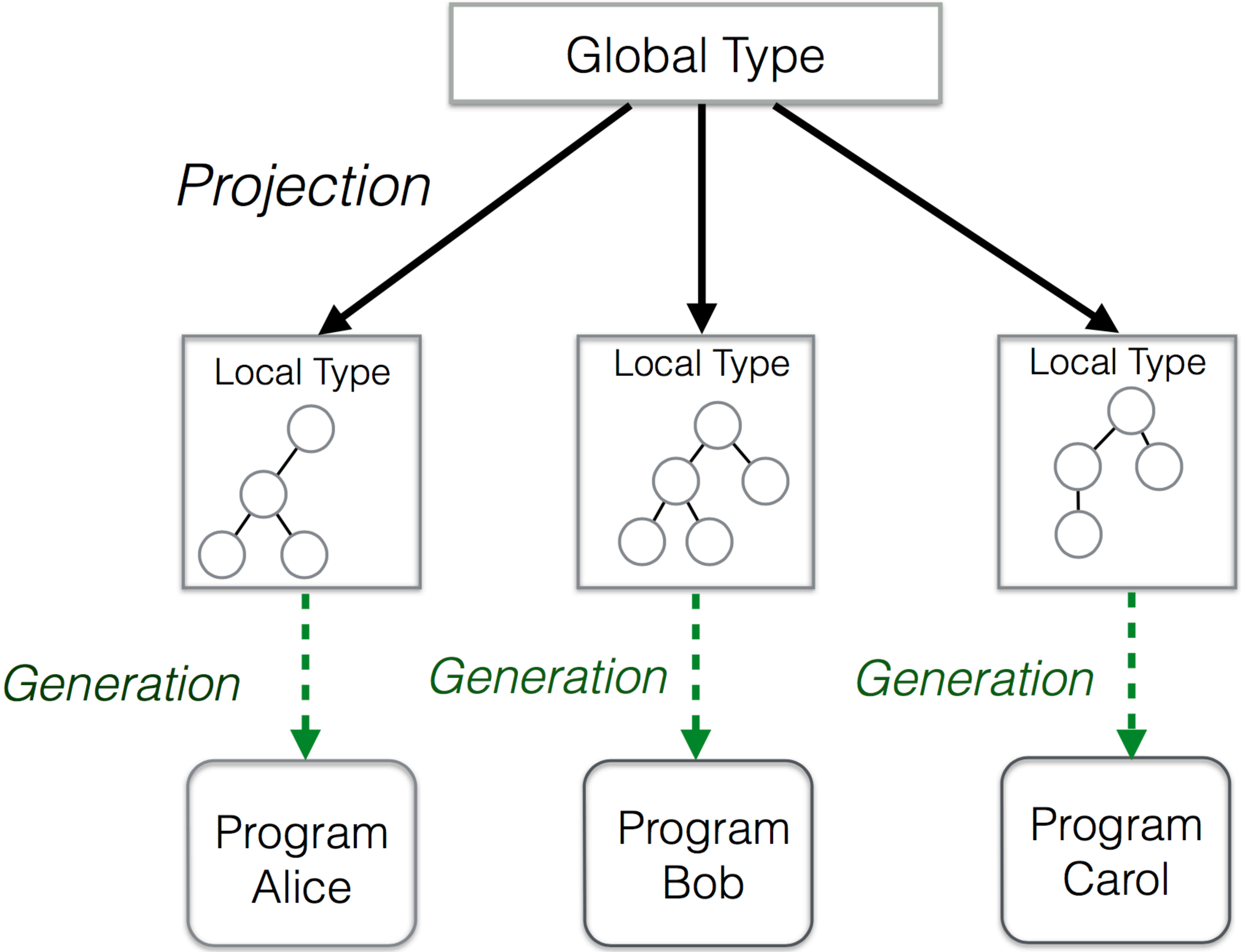
Type Checking

[OOPSLA'15, ECOOP'16, ECOOP'17, COORDINATION'17]



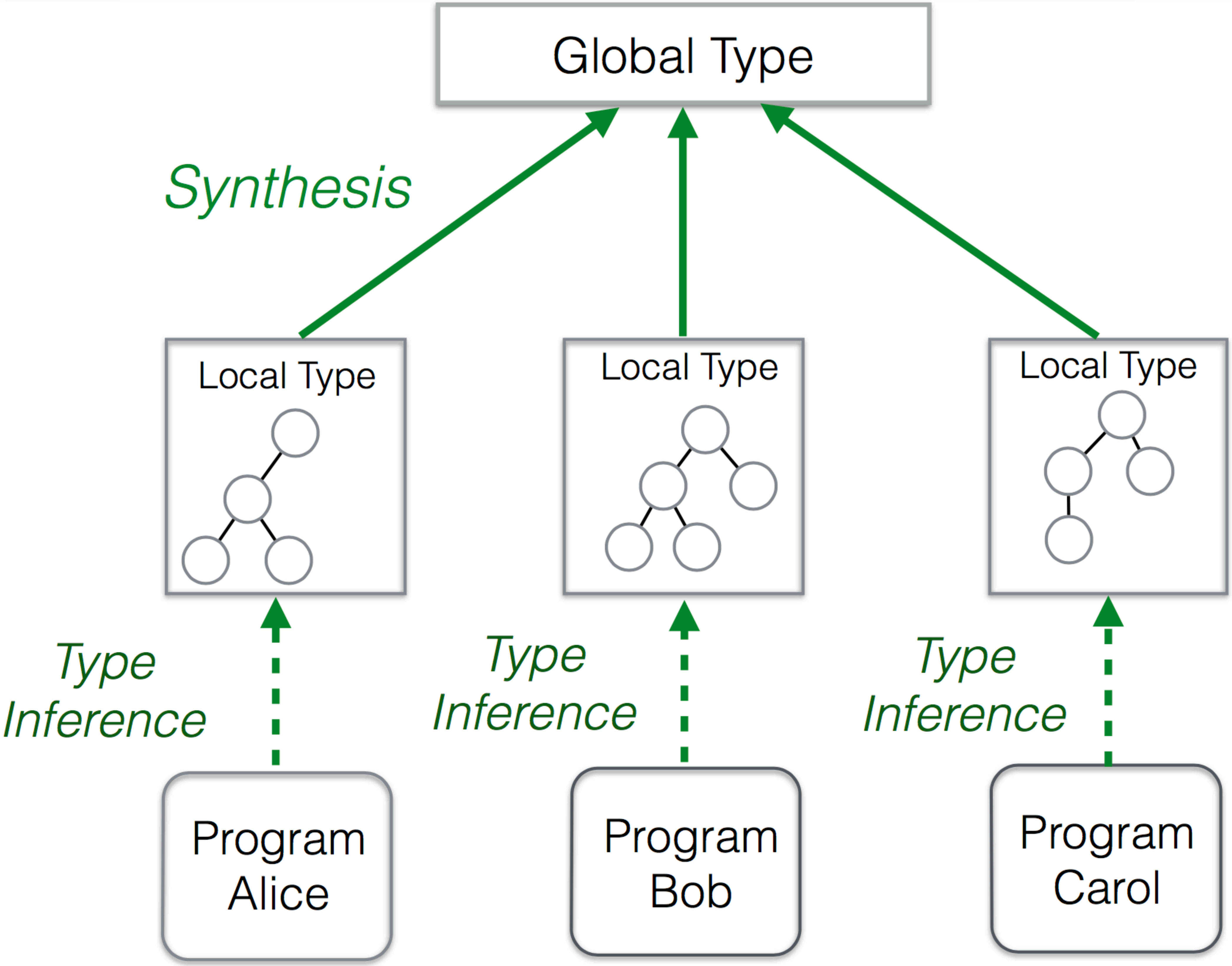
Code Generation

[CC'15, FASE'16, FASE'17]



Synthesis

[ICALP’13, POPL’15, CONCUR’15, TACAS’16, CC’16]



Global Type

Synthesis

Local Type

Local Type

Local Type

Type Inference

Type Inference

Type Inference

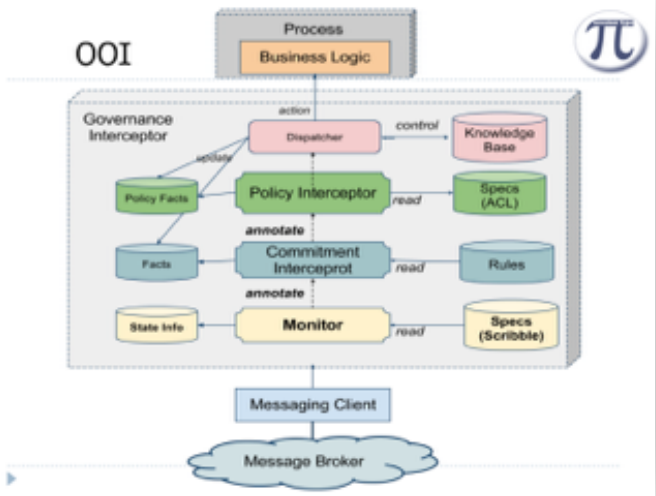
Program Alice

Program Bob

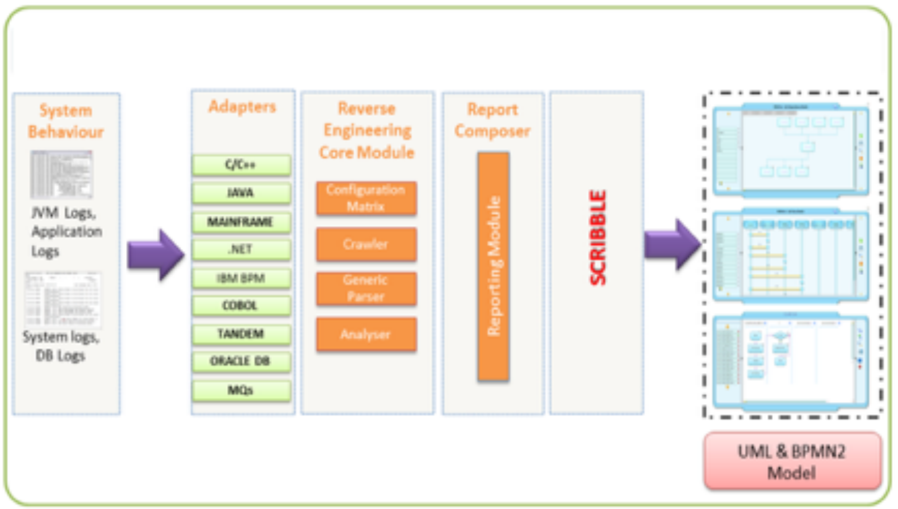
Program Carol

Session Type based Tools

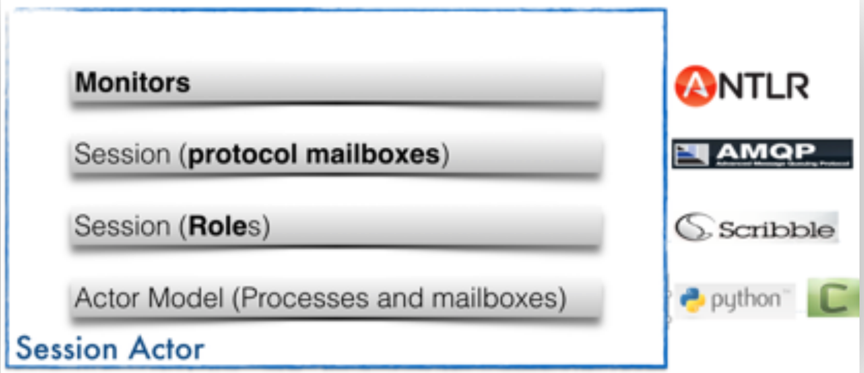
OOI Governance



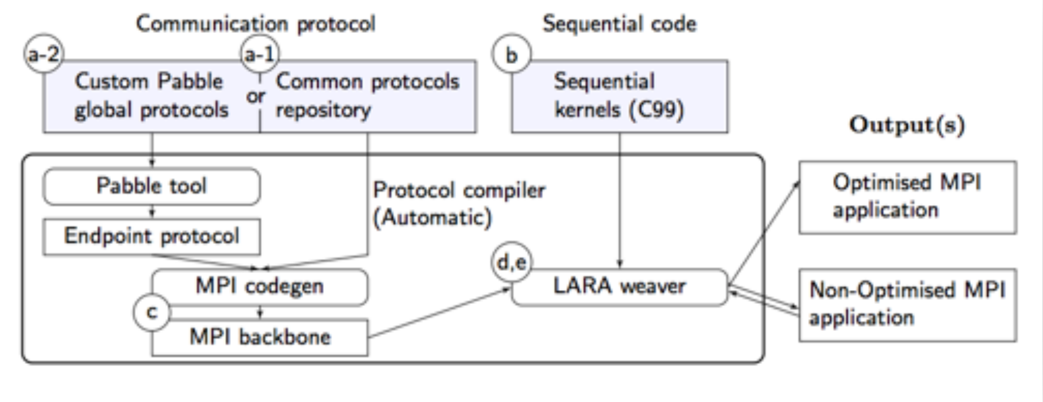
ZDLC: Process Modeling



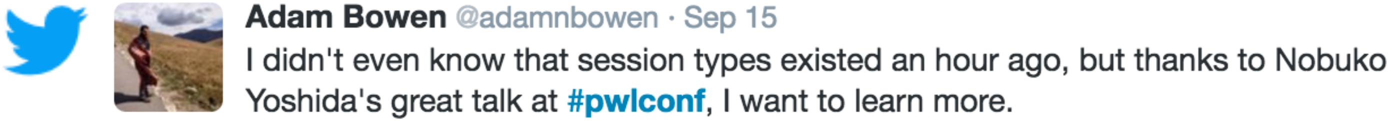
Actor Verification


Monitors
Session (protocol mailboxes)
Session (Roles)
Actor Model (Processes and mailboxes)
Session Actor
ANTLR
AMQP
Scribble
python
C

MPI code generations



Interactions with Industries

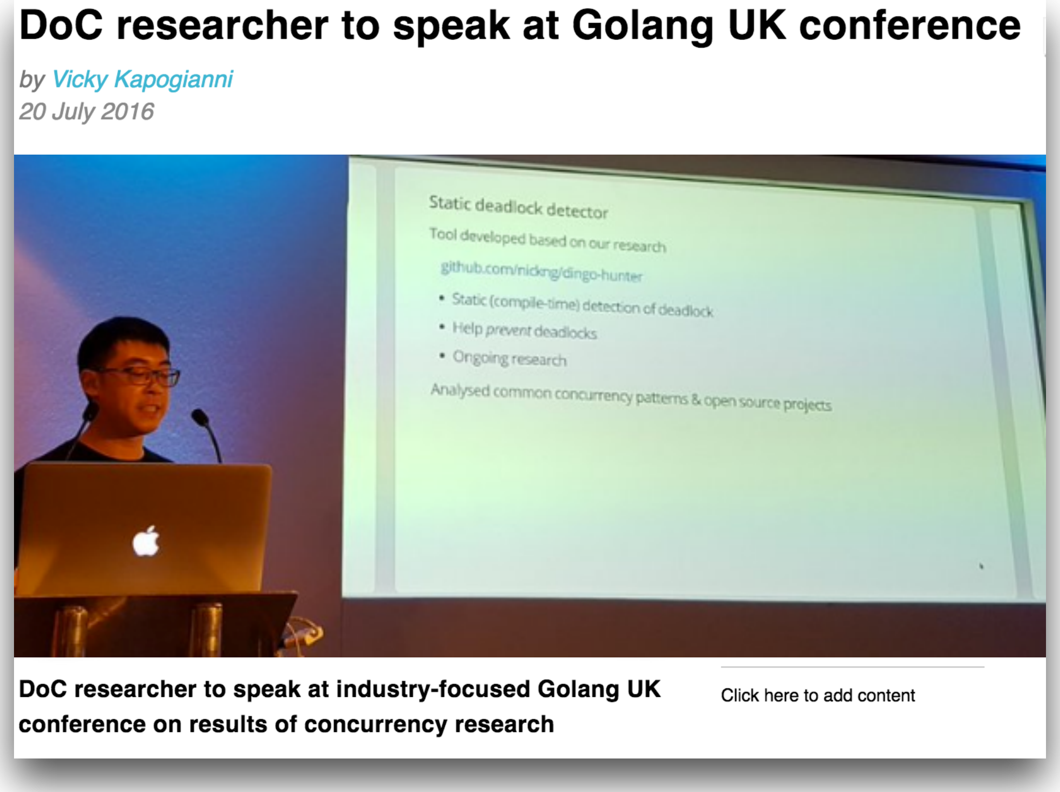


Adam Bowen @adamnbowen · Sep 15

I didn't even know that session types existed an hour ago, but thanks to Nobuko Yoshida's great talk at #pwlconf, I want to learn more.

Nobuko Yoshida

Imperial College, London



DoC researcher to speak at Golang UK conference

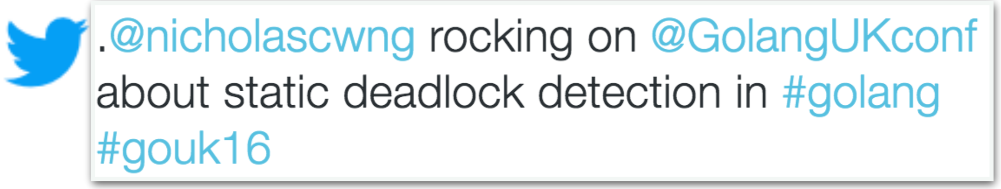
by Vicky Kapogianni

20 July 2016

Static deadlock detector
Tool developed based on our research
github.com/nickng/dingo-hunter
• Static (compile-time) detection of deadlock
• Help prevent deadlocks
• Ongoing research
Analysed common concurrency patterns & open source projects

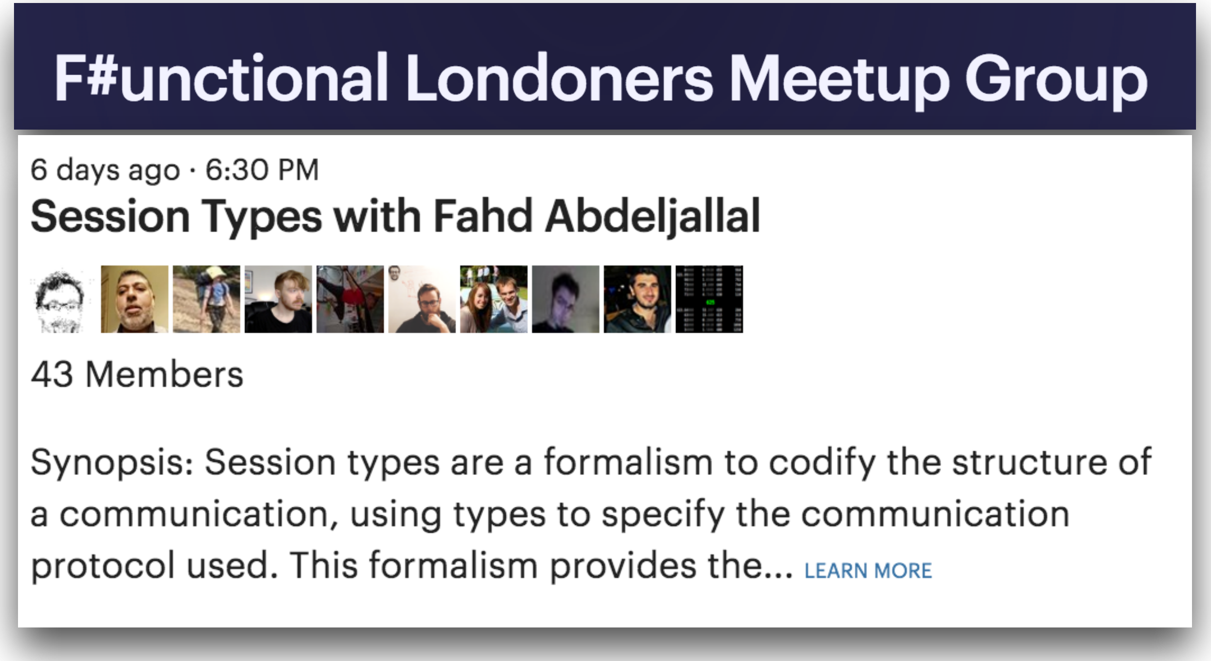
DoC researcher to speak at industry-focused Golang UK conference on results of concurrency research

Click here to add content



.@nicholascwng rocking on @GolangUKconf about static deadlock detection in #golang #gouk16

Interactions with Industries



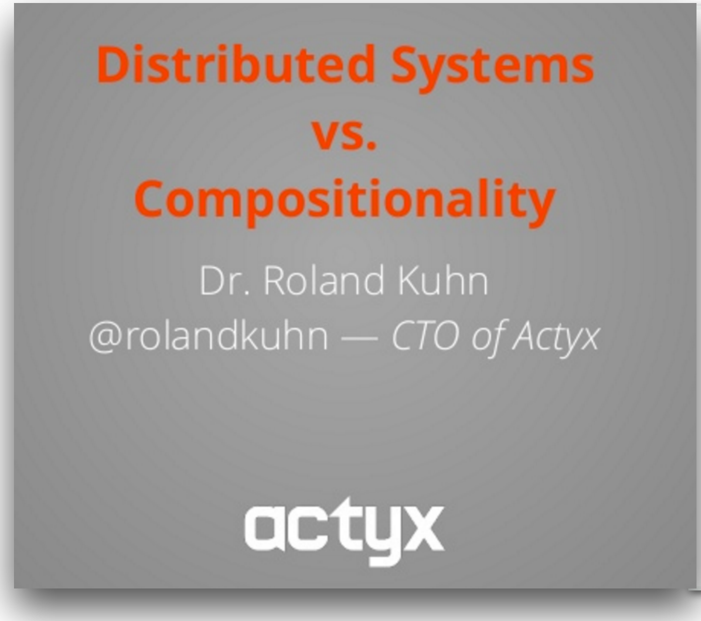
F#unctional Londoners Meetup Group

6 days ago · 6:30 PM

Session Types with Fahd Abdeljallal

43 Members

Synopsis: Session types are a formalism to codify the structure of a communication, using types to specify the communication protocol used. This formalism provides the... LEARN MORE

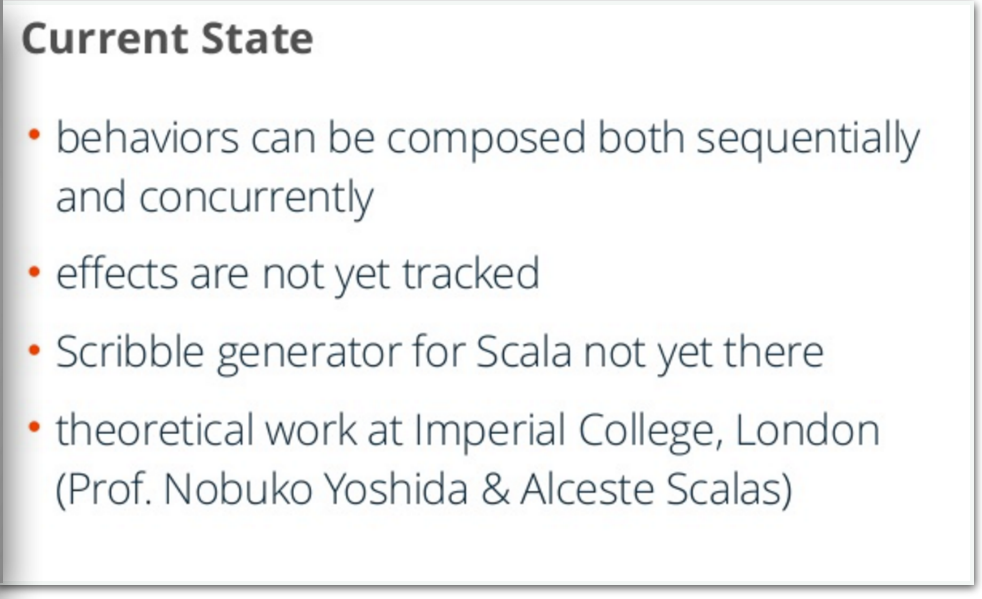


Distributed Systems vs. Compositionality

Dr. Roland Kuhn

@rolandkuhn — *CTO of Actyx*

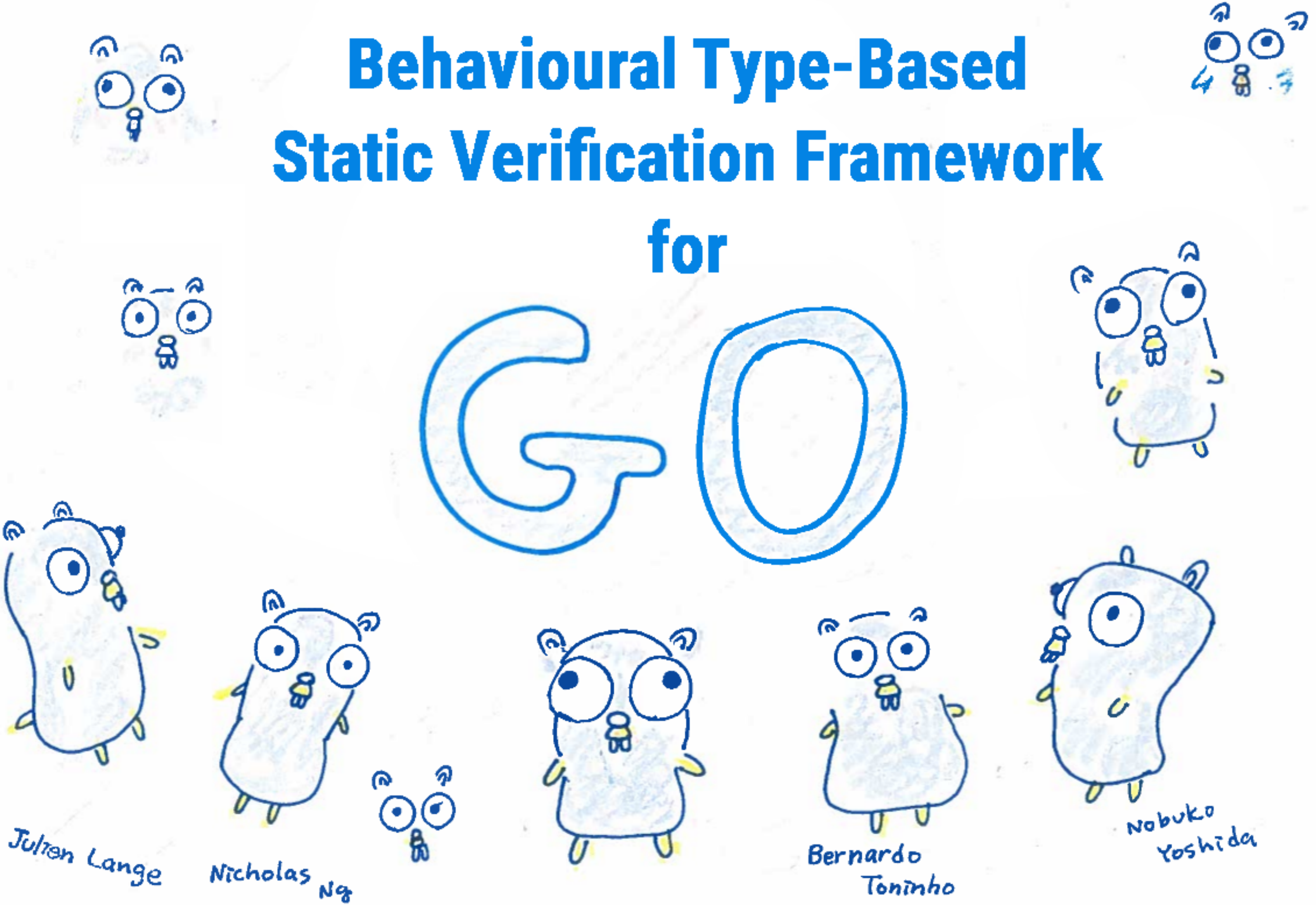
actyx

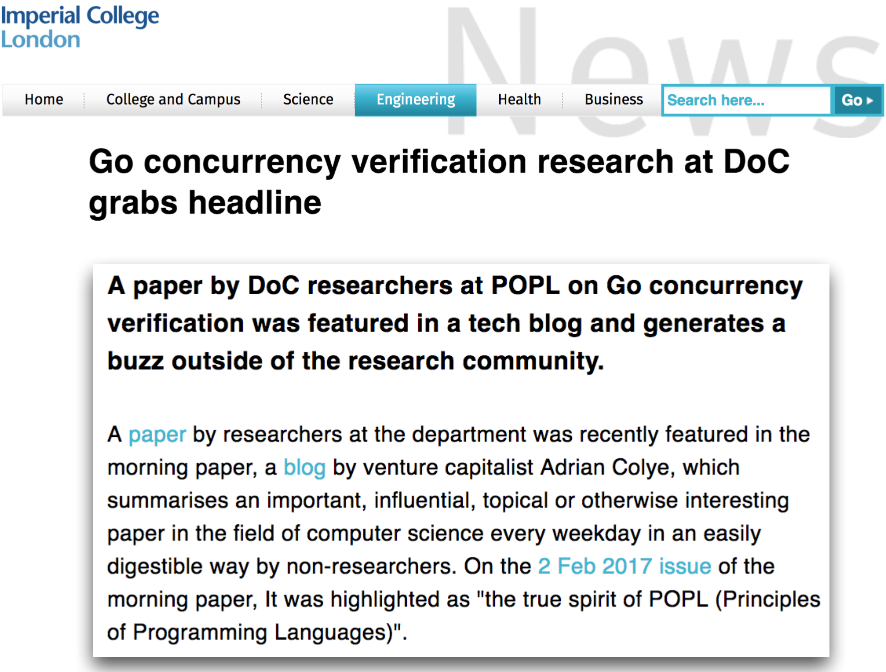


Current State

- behaviors can be composed both sequentially and concurrently
- effects are not yet tracked
- Scribble generator for Scala not yet there
- theoretical work at Imperial College, London (Prof. Nobuko Yoshida & Alceste Scalas)

Behavioural Type-Based Static Verification Framework for



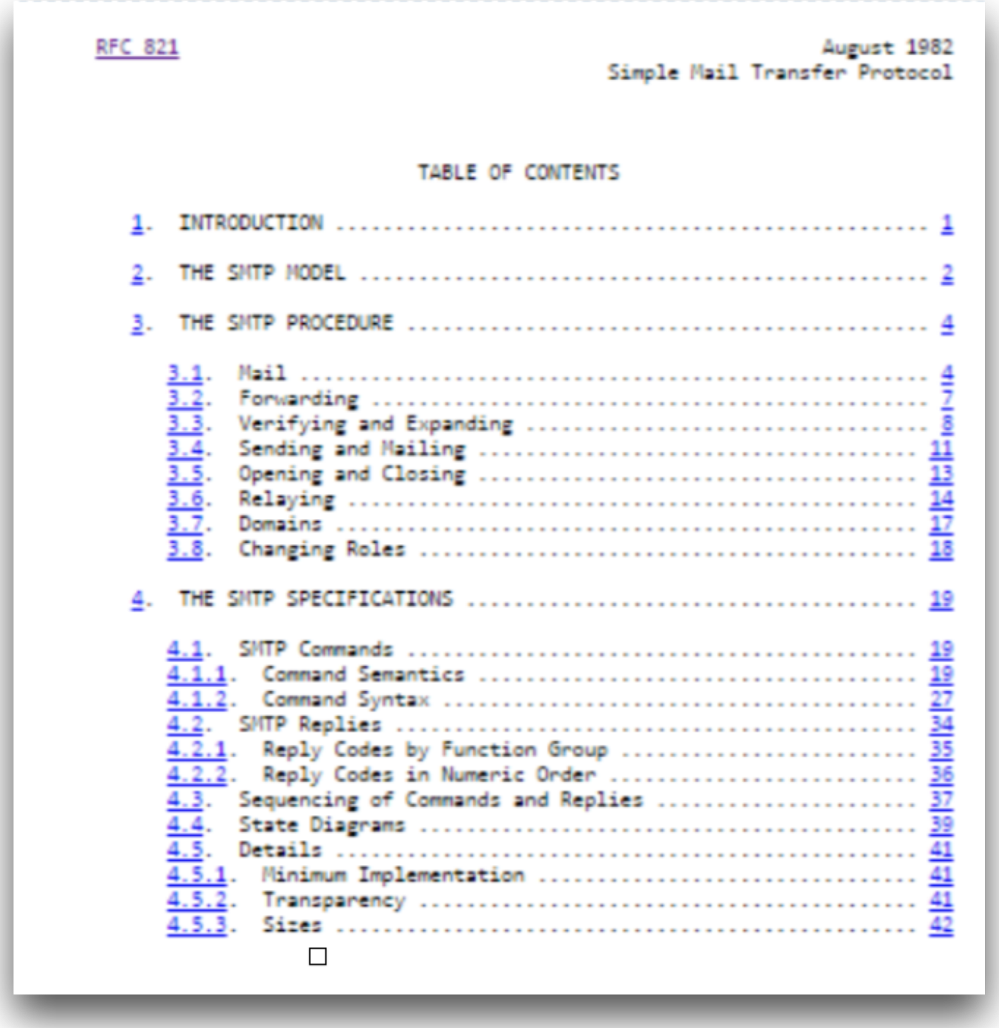


Go concurrency verification research at DoC grabs headline

A paper by DoC researchers at POPL on Go concurrency verification was featured in a tech blog and generates a buzz outside of the research community.

A paper by researchers at the department was recently featured in the morning paper, a blog by venture capitalist Adrian Colye, which summarises an important, influential, topical or otherwise interesting paper in the field of computer science every weekday in an easily digestible way by non-researchers. On the 2 Feb 2017 issue of the morning paper, It was highlighted as "the true spirit of POPL (Principles of Programming Languages)".

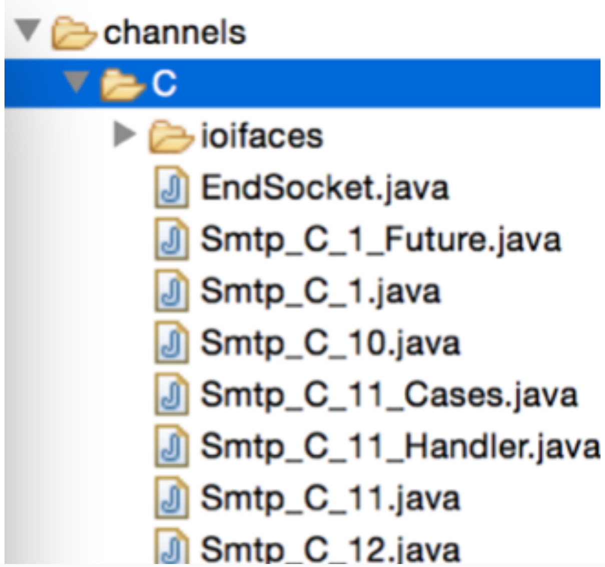
Java API Generation [FASE'16]

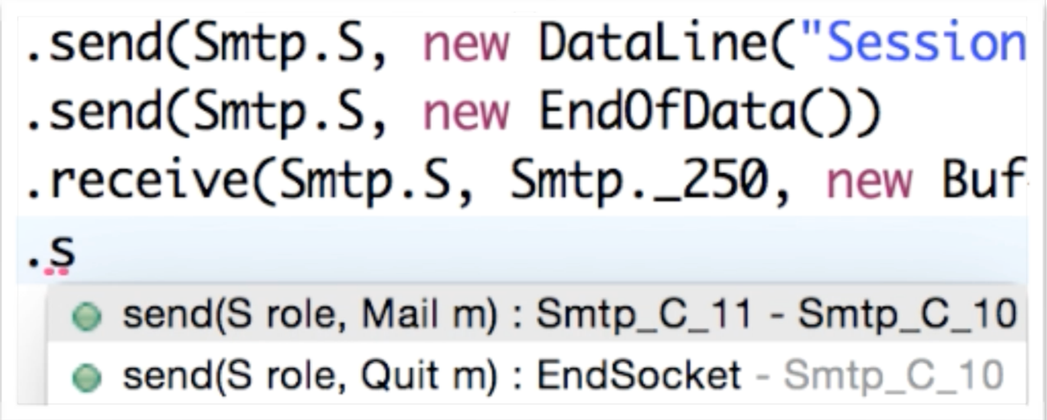


RFC 821 August 1982
Simple Mail Transfer Protocol

TABLE OF CONTENTS

□

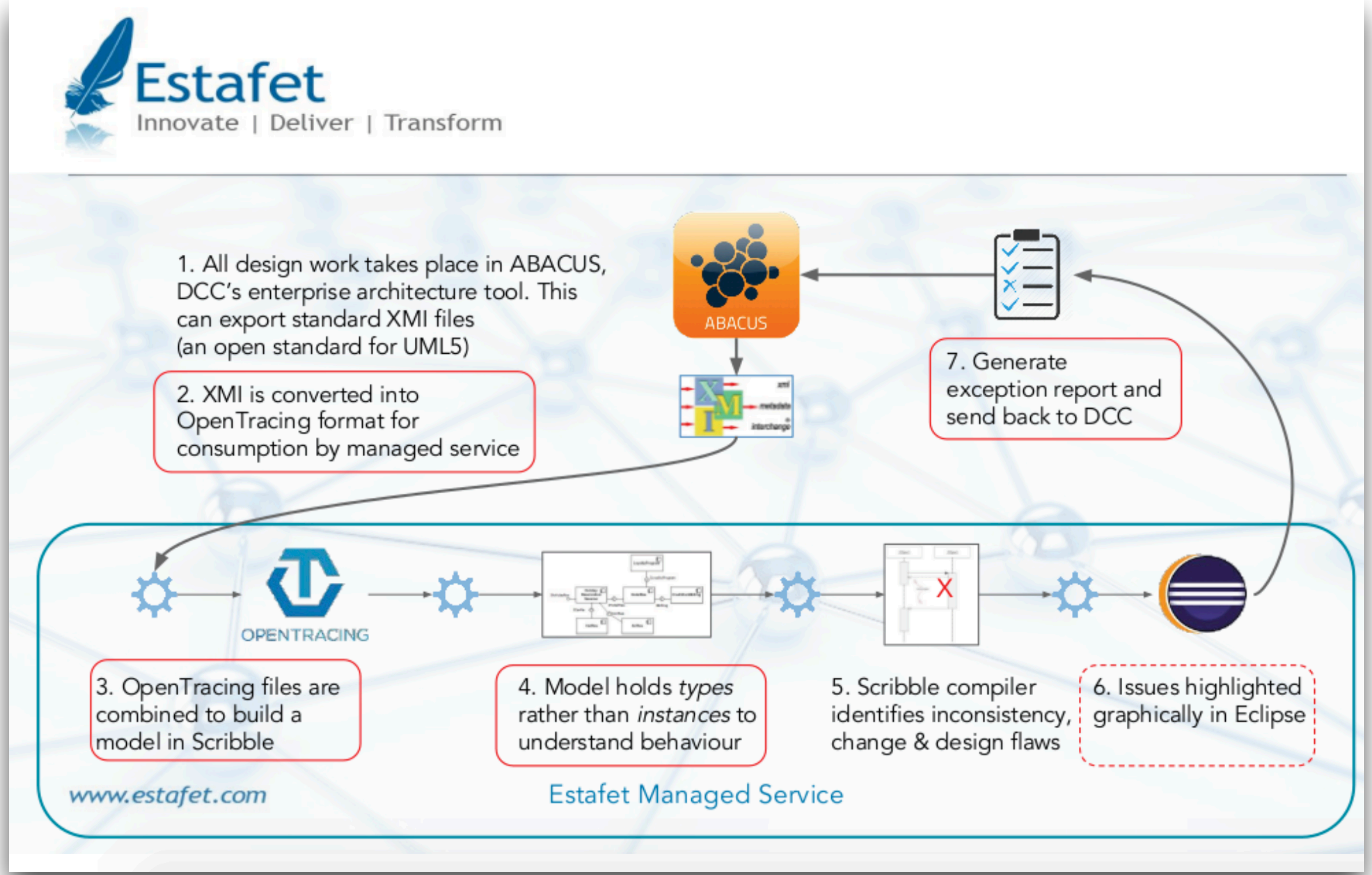




```
.send(Smtp.S, new DataLine("Session
.send(Smtp.S, new EndOfData())
.receive(Smtp.S, Smtp._250, new Buf
.s
```

send(S role, Mail m) : Smtp_C_11 - Smtp_C_10
send(S role, Quit m) : EndSocket - Smtp_C_10

Scribble – Proving a distributed design



Estafet

Innovate | Deliver | Transform

1. All design work takes place in ABACUS, DCC's enterprise architecture tool. This can export standard XMI files (an open standard for UML5)

2. XMI is converted into OpenTracing format for consumption by managed service

3. OpenTracing files are combined to build a model in Scribble

4. Model holds *types* rather than *instances* to understand behaviour

5. Scribble compiler identifies inconsistency, change & design flaws

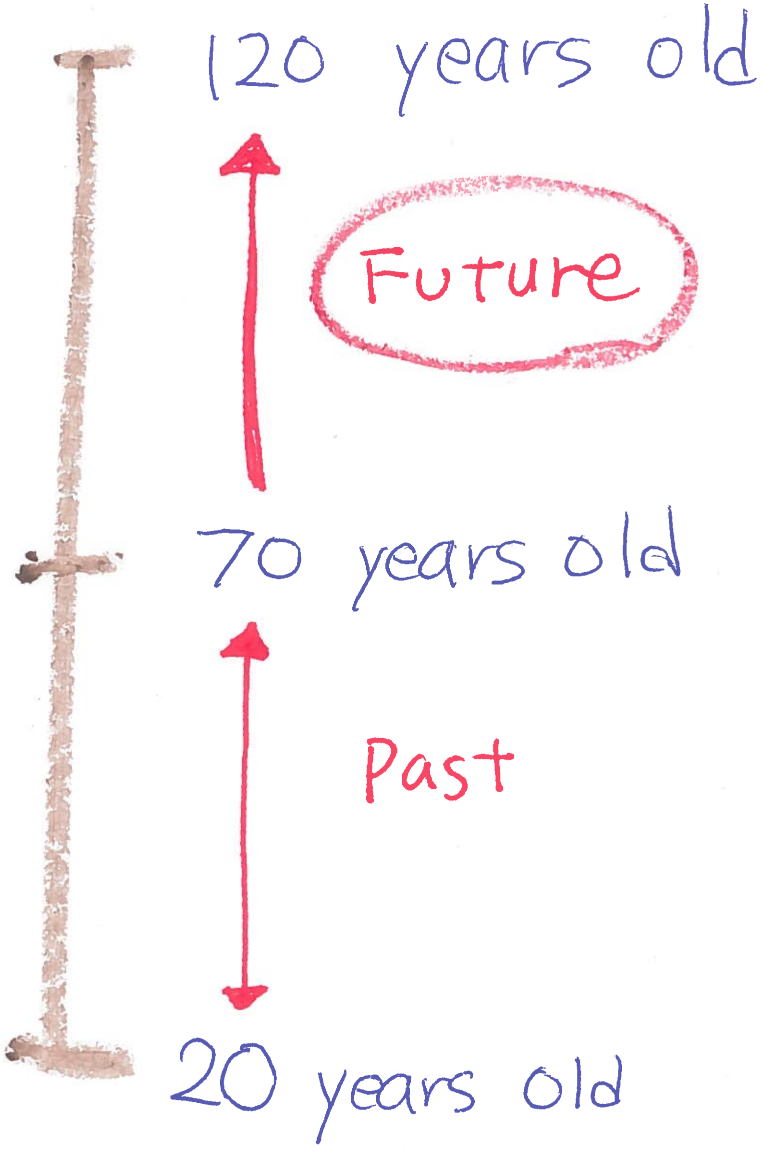
6. Issues highlighted graphically in Eclipse

7. Generate exception report and send back to DCC

www.estafet.com

Estafet Managed Service

LIVE Researchers



120 years old

Future

70 years old

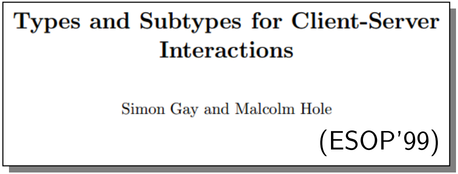
Past

20 years old

- Information Flow and Security
  [CONCUR'10]....
- Reversible Computation
  [CONCUR'17]...
- Preciseness
  [PPDP'14] [LMCS]...

:

Background: session subtyping

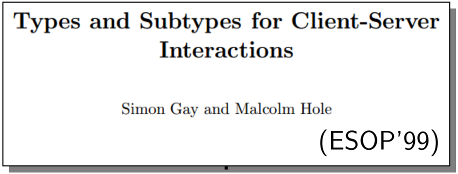


Types and Subtypes for Client-Server Interactions

Simon Gay and Malcolm Hole

(ESOP’99)

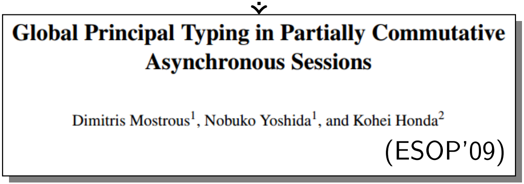
Background: session subtyping



Types and Subtypes for Client-Server Interactions

Simon Gay and Malcolm Hole

(ESOP'99)

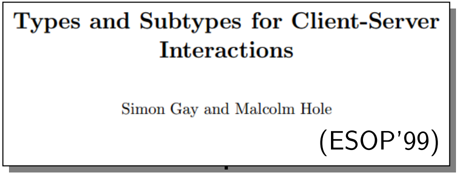


Global Principal Typing in Partially Commutative Asynchronous Sessions

Dimitris Mostrous[1], Nobuko Yoshida[1], and Kohei Honda[2]

(ESOP'09)

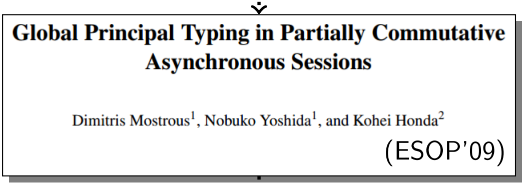
Background: session subtyping



Types and Subtypes for Client-Server Interactions

Simon Gay and Malcolm Hole

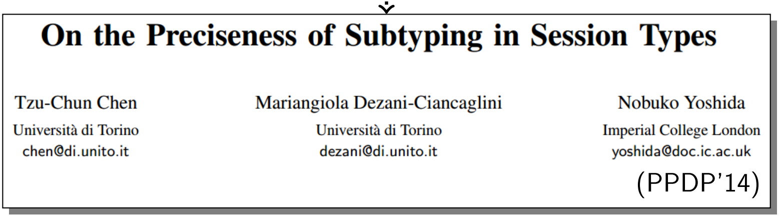
(ESOP'99)



Global Principal Typing in Partially Commutative Asynchronous Sessions

Dimitris Mostrous[1], Nobuko Yoshida[1], and Kohei Honda[2]

(ESOP'09)



On the Preciseness of Subtyping in Session Types

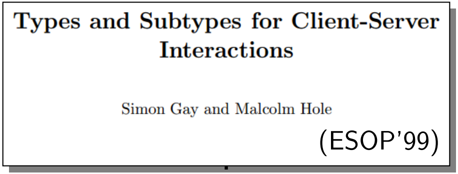
Tzu-Chun Chen
Università di Torino
chen@di.unito.it

Mariangiola Dezani-Ciancaglini
Università di Torino
dezani@di.unito.it

Nobuko Yoshida
Imperial College London
yoshida@doc.ic.ac.uk

(PPDP'14)

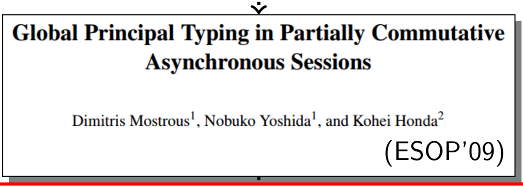
Background: session subtyping



Types and Subtypes for Client-Server Interactions

Simon Gay and Malcolm Hole

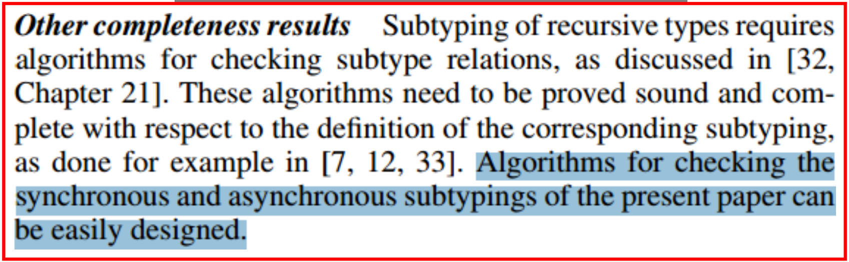
(ESOP'99)



Global Principal Typing in Partially Commutative Asynchronous Sessions

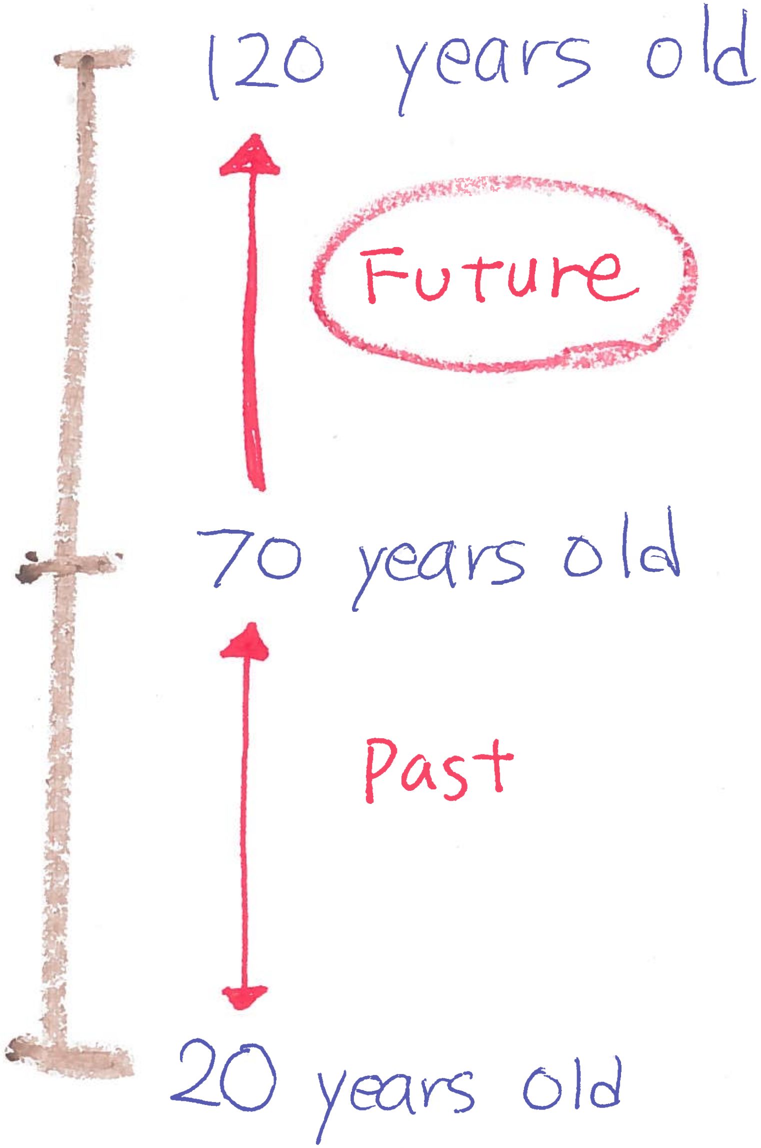
Dimitris Mostrous[1], Nobuko Yoshida[1], and Kohei Honda[2]

(ESOP'09)



Other completeness results Subtyping of recursive types requires algorithms for checking subtype relations, as discussed in [32, Chapter 21]. These algorithms need to be proved sound and complete with respect to the definition of the corresponding subtyping, as done for example in [7, 12, 33]. Algorithms for checking the synchronous and asynchronous subtypings of the present paper can be easily designed.

LIVE Researchers



120 years old

Future

70 years old

Past

20 years old

- × Information Flow and Security [CONCUR'10]....
- × Reversible Computation [CONCUR'17]...
- × Preciseness [PPDP'14] [LMCS]...

:

HAPPY 70