

Session TYPES

AS A

DESCRIPTIVE



Nobuko
Yoshida
&
Raymond Hu

25th
Oct
2017

TOOL

for DISTRIBUTED PROTOCOL

Awards



ETAPS 2019 TEST-OF-TIME AWARD

[Language primitives and type discipline for structured communication-based programming](#) by Kohei Honda,
Vasco T. Vasconcelos and Makoto Kubo

Awards

POPL 2008 MOST INFLUENTIAL PAPER AWARD



POPL 2008 Most Influential Paper Award

Kohei Honda, Nobuko Yoshida and Marco Carbone

Multiparty asynchronous session types





LICS 2018 TEST OF TIME AWARD

LICS'98

A fully abstract game semantics for general references

Samson Abramsky Kohei Honda
LFCS, University of Edinburgh
{samson,kohei}@dcs.ed.ac.uk

Guy McCusker
St John's College, Oxford
mccusker@comlab.ox.ac.uk

Abstract

A games model of a programming language with higher-order store in the style of ML-references is introduced. The category used for the model is obtained by relaxing certain behavioural conditions on a category of games previously used to provide fully abstract models of pure functional languages. The model is shown to be fully abstract by means of factorization arguments which reduce the question of definability for the language with higher-order store to that for its purely functional fragment.

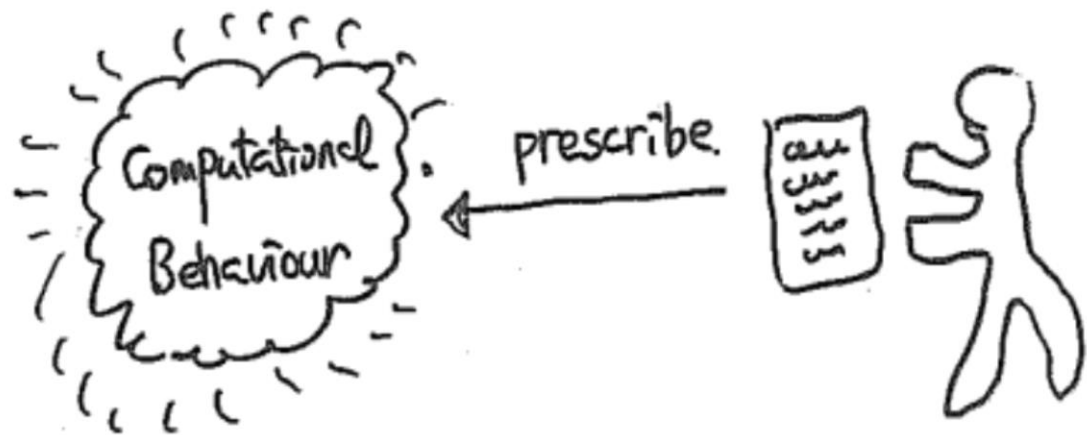
Idioms for Interaction

— Invitation to Hacking in π -calculus —

Programming languages are tools which offer frameworks of abstraction for such activities – promoting or limiting them

- Imperative
- Functional
- Logical

- Programs : prescription of computational behaviours based on a certain abstraction.



On Programs and Programming

- The most fundamental element of a PL in this context is a set of operations it is based on:

Imperative: assignment, jump.

Functional: β -reduction.

Logical: unification.

- Another element is how we can combine, on structure, these operations:

Imperative: sequential composition, if-then-else, while, procedures, module,

Functional: application, product, union, recursion, modules,

UNSTRUCTURED:

```

data _stkl, 48 } stacks.
data _stkr, 48 }
data _i, 4 } index
data _j, 4 }
data _l, 4 } left/right limit
data _r, 4 }
data _x, 4 } pivot
data _w, 4 } temporary value.
data _s, 4 } stack pointer
data _a, 48 } table to be sorted.

mov $0, _s
mov $0, _stkl } 2nd instruction
mov $11, _stkr

L1: mov _s, rax } Top loop.
    mov _stkl(, rax, 4), rcx } l := stkl(s)
    mov rcx, _l
    mov _s, rax
    mov _stkr(, rax, 4), rcx } r := stkr(s)
    mov rcx, _r
    dec _s

L2: mov _l, rcx } j := l.
    mov rcx, _i
    mov _r, rcx } i := r.
    mov rcx, _j
    mov _l, rcx } rdx := l+r
    add _r, rcx
    mov rcx, rax } rdx := (l+r)/2
    mov rax, rdx
    shr $31, rdx
    add rdx, rax
    mov rax, rdx
    sar $1, rdx
    mov _a(, rdx, 4), rcx } x := a(rdx)
    mov rcx, _x

L3: mov _i, rax } third loop
    mov _a(, rax, 4), rdx } rdx := a(i)
    cmp rdx, _x
    jle L4
    inc _i
    jmp L3 } if rdx >= x goto (4)
    } else i := i+1
    } goto (3) (loop)

L4: mov _j, rax
    mov _a(, rax, 4), rdx } rdx := a(j)

```

STRUCTURED:

```

Var a: array[MAX] of int;
Procedure sort(l, r: int);
  Var i, j, x: int;
  i := l; j := r;
  x := [(l+r) div 2];
  • Choose a pivot.
  repeat
    while a[i] < x do i := i+1 end
    while a[j] > x do j := j-1 end
    if i <= j then swap(i, j); i := i+1; j := j-1; end
  until i > j;
  if l < j then sort(l, j);
  if l < i then sort(i, r);
  end
  } Partition into two parts.
  } Recursively sort two parts.

Procedure swap(i, j: int)
  Var w: int;
  w := a[i]; a[i] := a[j]; a[j] := w;
end

```

Quicksort in pure lambda:

$((\lambda xy. y(xy))(\lambda xy. y(xy)))\lambda q. \lambda l.$
 $((\lambda x. x(\lambda xy. x))l)(\lambda x. x)$ } if l is not then nil.
 $((\lambda xy. y(xy))(\lambda xy. y(xy)))(\lambda c. \lambda xy. x((\lambda x. x(\lambda py. x))x)y$ } concat.
 $((\lambda xy. \lambda z. z(\lambda xy. y)xy)((\lambda x. x(\lambda xyz. y))x)(c((\lambda x. x(\lambda xyz. z))x)y$
 $(q(\lambda xy. y(xy))(\lambda xy. y(xy)))(\lambda f. \lambda px. ((\lambda x. x(\lambda py. x))x)(\lambda x. x))$ } sort and
 $(p((\lambda x. x(\lambda xyz. y))x))((\lambda xy. \lambda z. z(\lambda xy. y)xy)x$ Filter
 $(f((\lambda x. x(\lambda xyz. z))x)))(f((\lambda x. x(\lambda xyz. z))x)))$
 $(\lambda y. (((\lambda xy. y(xy))(\lambda xy. y(xy)))\lambda f'. \lambda xy. ((\lambda x. x(\lambda py. x))x)y$ } $(\lambda y. LT y. Car$
 $(\lambda xy. y)((\lambda x. x(\lambda py. x))x)(\lambda xy. y)(f'((\lambda x. x(\lambda py. y))x)((\lambda x. x(\lambda py. y)$
 $y((\lambda x. x(\lambda xyz. y))l))((\lambda x. x(\lambda xyz. z))l)))$ } cdr l
 $((\lambda xy. \lambda z. z(\lambda xy. y)xy)((\lambda x. x(\lambda xyz. y))l)$ } cons (car l)
 $(q((\lambda xy. y(xy))(\lambda xy. y(xy)))(\lambda f. \lambda px. ((\lambda x. x(\lambda py. x))x)$ } sort and
 $(\lambda x. x)(p((\lambda x. x(\lambda xyz. y))x))((\lambda xy. \lambda z. z(\lambda xy. y)xy)x$ Filter
 $(f((\lambda x. x(\lambda xyz. z))x)))(f((\lambda x. x(\lambda xyz. z))x)))$
 $(\lambda y. (((\lambda xy. y(xy))(\lambda xy. y(xy)))\lambda f''. \lambda xy. ((\lambda x. x(\lambda py. x))x)$ } $(\lambda y. ME y$
 $((\lambda x. x(\lambda py. x))y)(\lambda xy. x)(\lambda xy. y)$ Car l
 $((\lambda x. x(\lambda py. x))y)(\lambda xy. x)(f''((\lambda x. x(\lambda py. y))x)$
 $((\lambda x. x(\lambda py. y))y))y((\lambda x. x(\lambda xyz. y))l)((\lambda x. x(\lambda xyz. z))l)))$ } cdr l

Quicksort with combinators:

$Y(\lambda f. \lambda l.$
 $(Isnil l)$
 $(Concat (f Filter (\lambda y. LT y (Car l))$
 $(Cdr l))$
 $(Cons (Car l)$
 $(f Filter (\lambda y. ME y$
 $(Car l)$
 $(Cdr l))))))$

$I = \lambda x. x$ $T = \lambda xy. x$ $F = \lambda xy. y$ $Y = (\lambda xy. y(xy))(\lambda xy. y(xy))$
 $Cons = \lambda xy. \lambda z. z F xy$ $Isnil = \lambda x. x T$
 $Car = \lambda x. x (\lambda yz. y)$ $Cdr = \lambda x. x (\lambda yz. z)$
 $Concat = Y (\lambda c. \lambda xy. x (Isnil x) y (Cons (Car x) (c (Cdr x) y))$
 $Filter = Y (\lambda f. \lambda px. (Isnil x) I (p (Car x)) (Cons x (Filter (Cdr x))))$
 $Iszero = \lambda x. x T$ $Pred = \lambda x. x F$ $(Filter (Cdr x))$
 $LT = Y (\lambda f. \lambda xy. (Iszero y) F ((Iszero x) F (f (Pred x) (Pred y)$
 $ME = Y (\lambda f. \lambda xy. (Iszero x) ((Iszero y) I F) ((Iszero y) (T) y))$
 $(f (Pred x) (Pred y))$

Quicksort in ML:

```
fun qs nil: int list = nil
```

```
| qs (x::r) = let val small =  
                filter (fn y => y < x) r  
                and large =  
                filter (fn y => y >= x) r
```

```
    in qs small @ [x] @ qs large
```

```
    end
```

```
fun filter p nil = nil
```

```
| filter p (x::r) =
```

```
    if p x then x := filter p r
```

```
    else filter p r
```

The π -calculus as a Descriptive Tool

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

$$\pi \quad P ::= \sum \pi_i.P_i \mid P|Q \mid \nu x.P \mid !P \mid \emptyset.$$

$$\text{with } \pi ::= x(\bar{y}) \mid \bar{x}(y).$$

λ in π

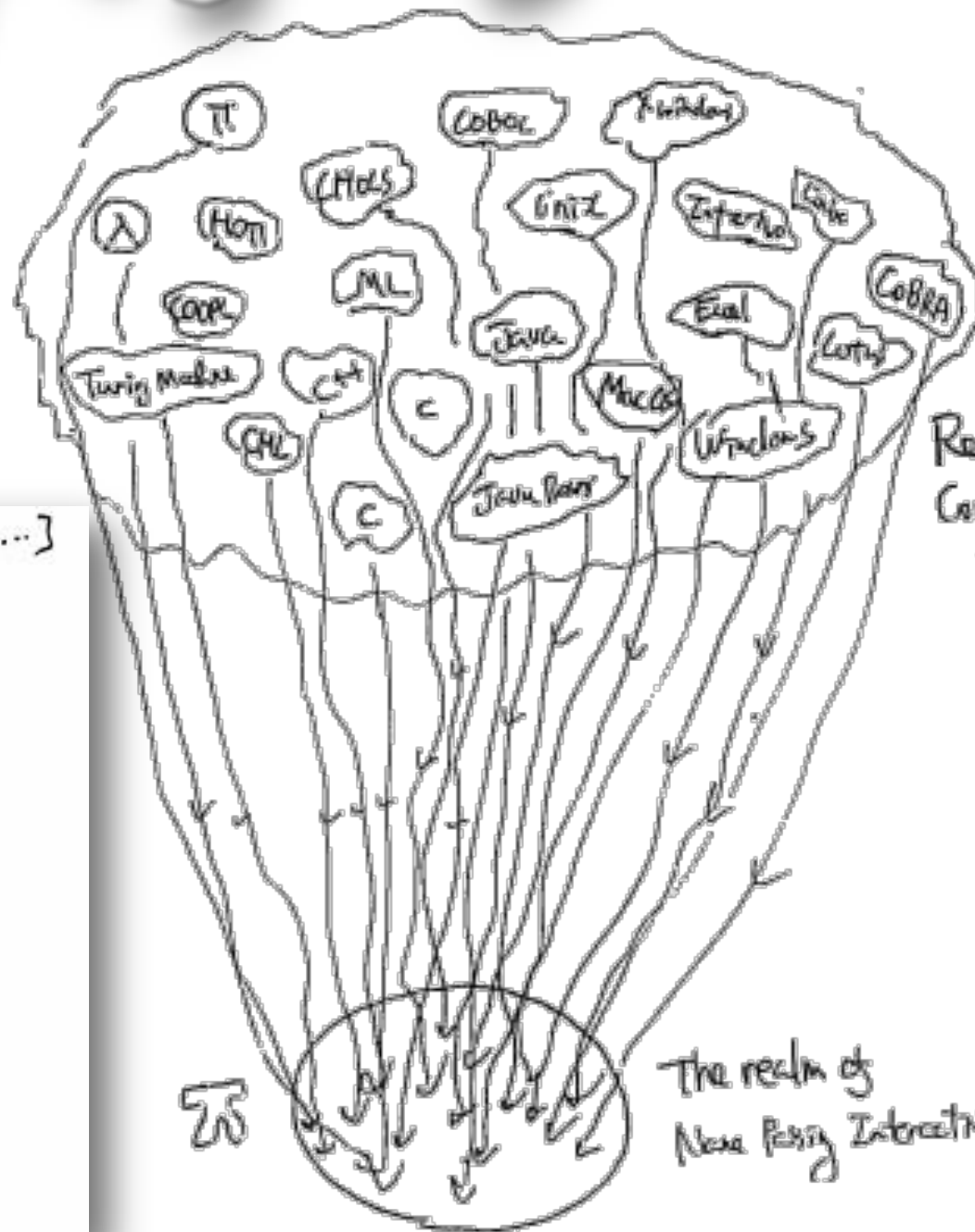
$$[x]_u \stackrel{\text{def}}{=} \bar{x}(u).$$

$$[\lambda x.M]_u \stackrel{\text{def}}{=} u(x).[M]_u.$$

$$[MN]_u \stackrel{\text{def}}{=} (\nu f_x)([M]_f \mid \bar{f}(x) \mid [x=N]_u)$$

$$\text{with } [x=N]_u \stackrel{\text{def}}{=} !x(u).[N]_u.$$

* Examples of Representable Computation.



- λ -calculus [MPW89, Milner90, Milner92, ...]
- Concurrent Object [Walker91]
- ω -order term passing [Sangiorgi 92]
- Various data structures [Milner 92, ...]
- Proof Nets [Bellon and Scott 93]
- Abstract "constant" interaction [HRS4]
- Strategies on Games [HO95]

The realm of
Non-Passive Interactions

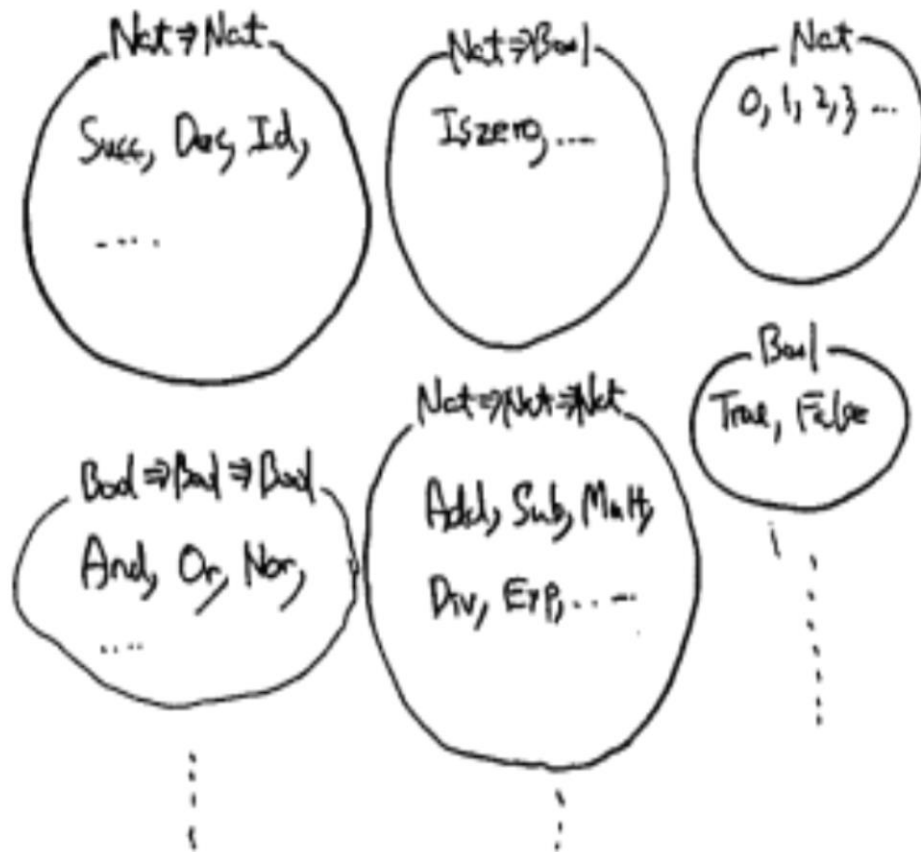
Realms of
Computational Behaviors

The Role of Types in π -Calculus.

- (classification) How can we classify name-passing interactive behaviours, i.e. behaviours representable in π -calculus? What classes ("types") of behaviours can we find in the calculus?

- (safety) Is this program/system in the safe (or correct, relevant,...) classes of behaviours? Can the safety be preserved compositionally?

Functional Types



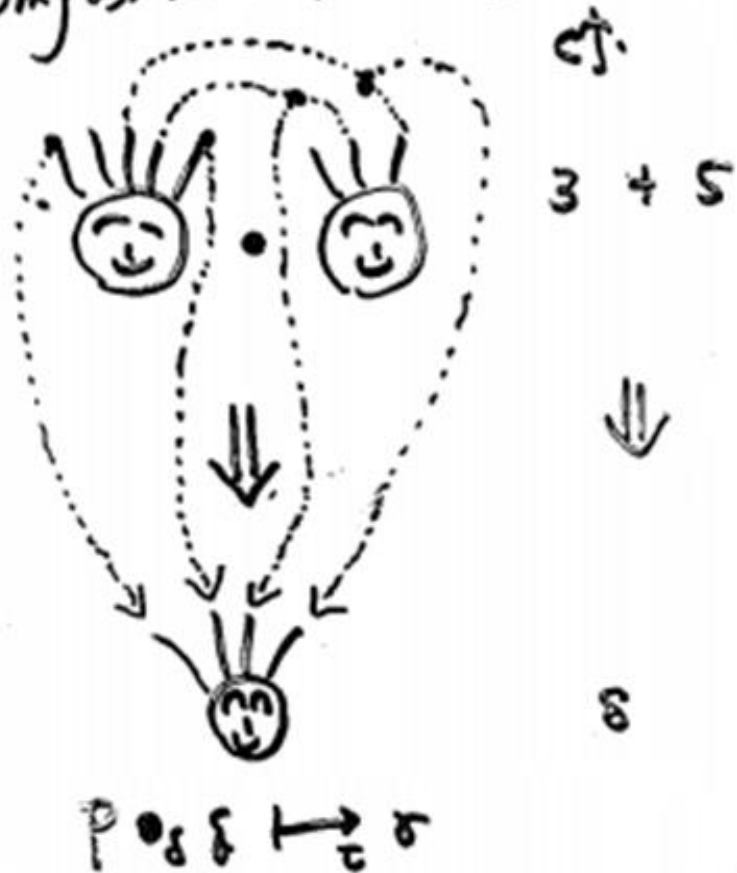
with operation!

$$\left\{ \begin{array}{l} f : \alpha \rightarrow \beta \bullet e : \alpha = f \bullet e : \beta. \\ \text{else undefined.} \end{array} \right.$$

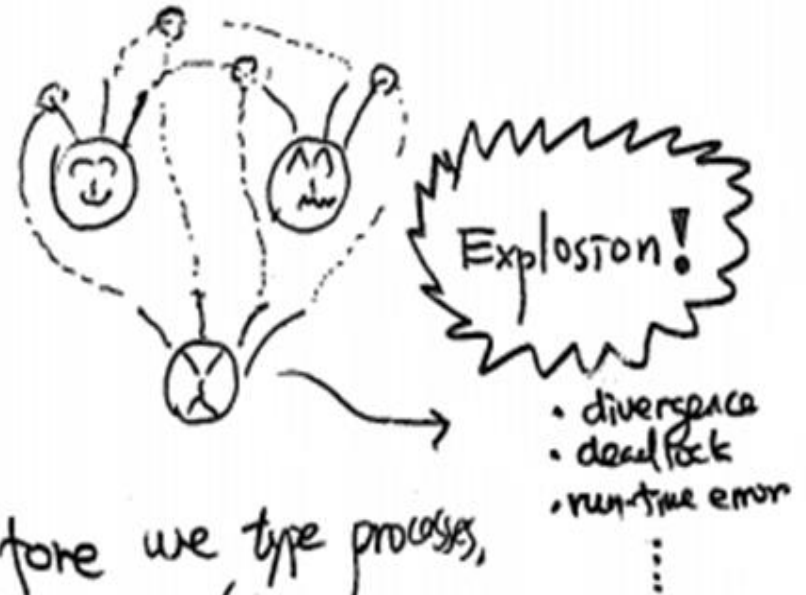
function application.

Process Types

- When it comes to processes, composition becomes:



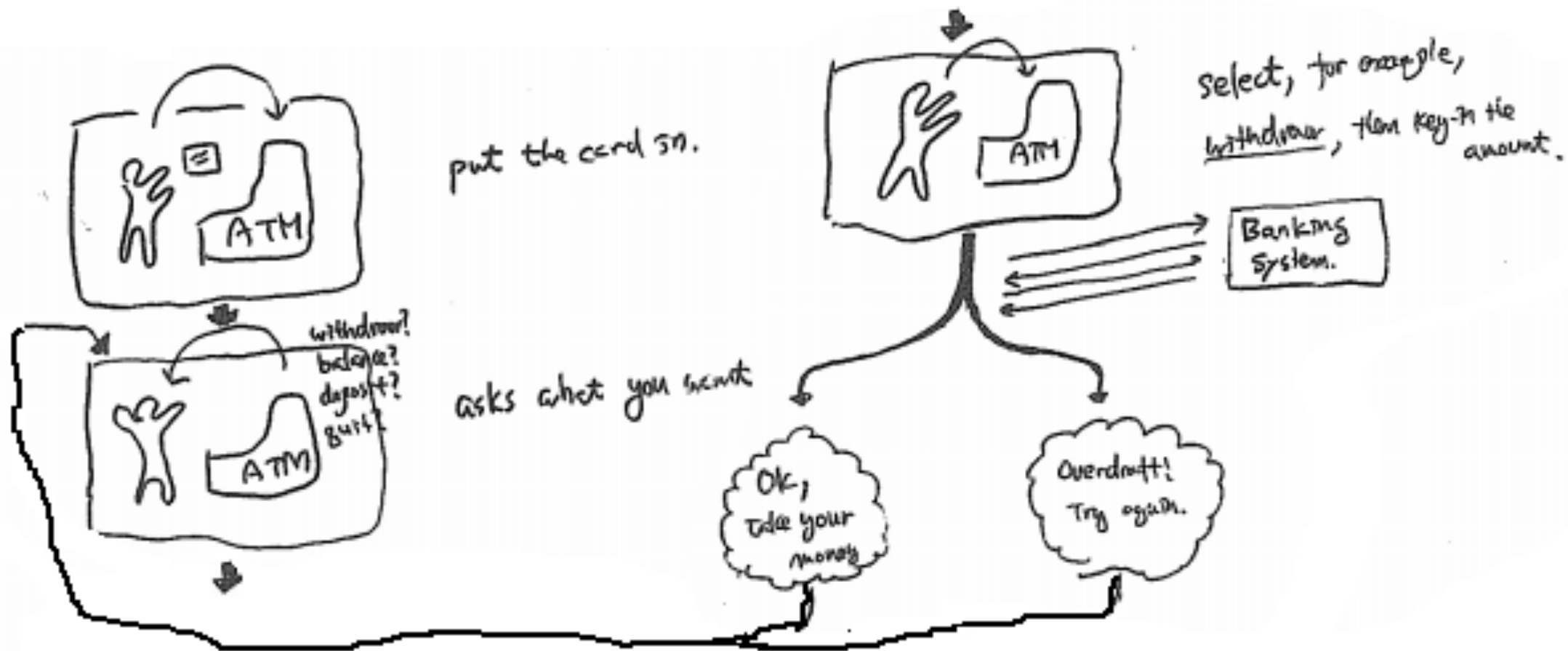
- But some composition is dangerous!



- Therefore we type processes,



Implementing ATM



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 π^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;
        }
    }
}
```

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 π^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 π^4 Technology



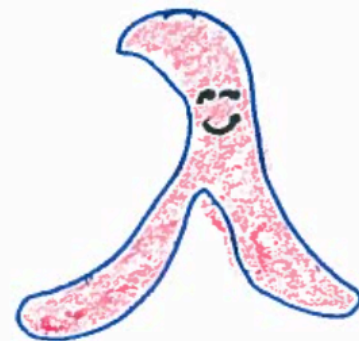
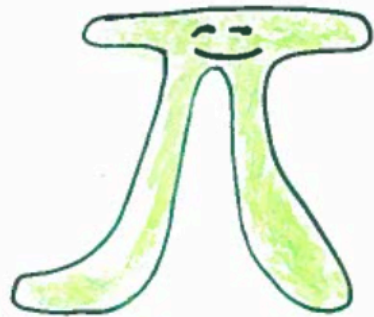
Multiparty Session Types [POPL'08]



PHIL &

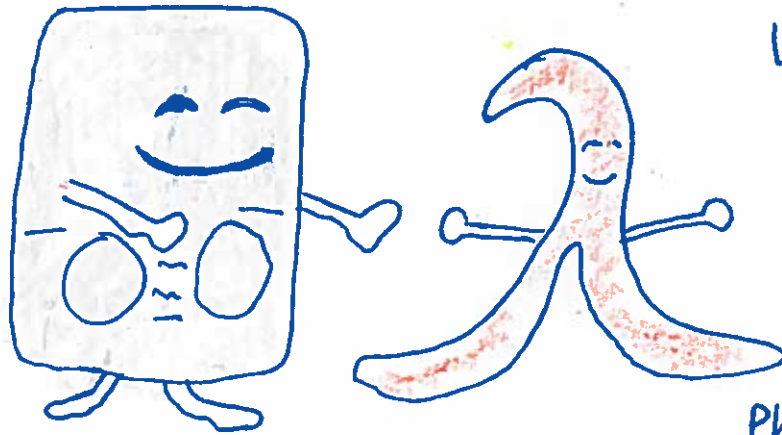
SESSION

TYPES



Imperial
College
London

1999?

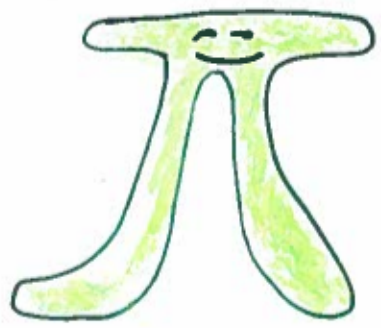


LOGO

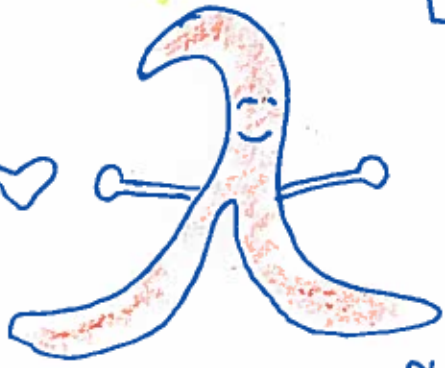
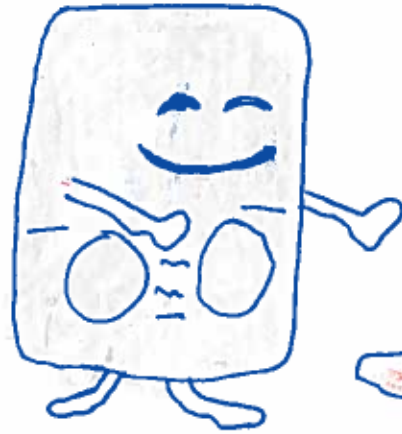
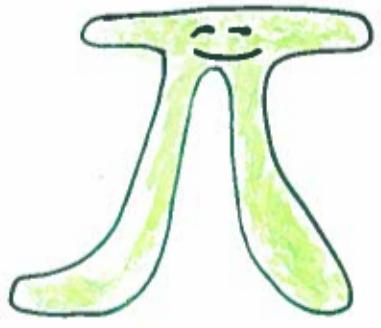
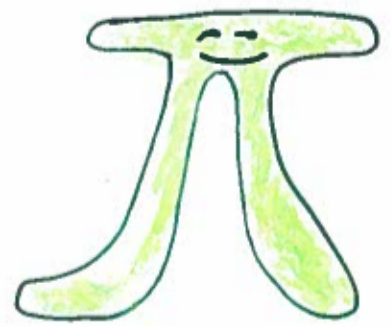
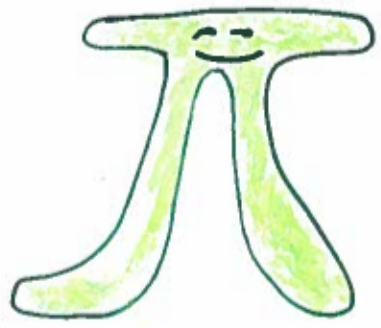
Phil Wadler

FPCA

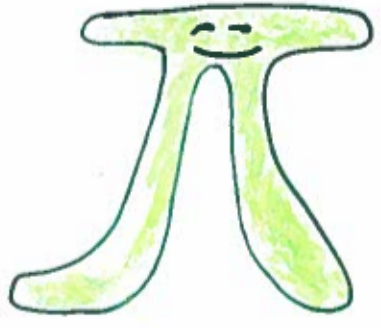
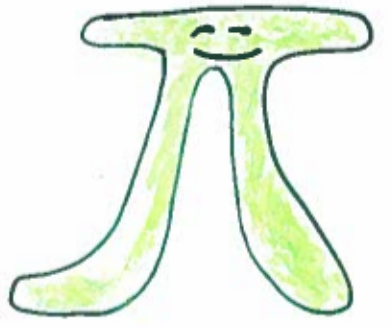
Functional
Programming
Languages and
Computer
Architectures



1999?

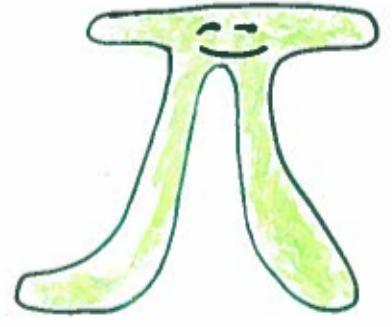


LOGO

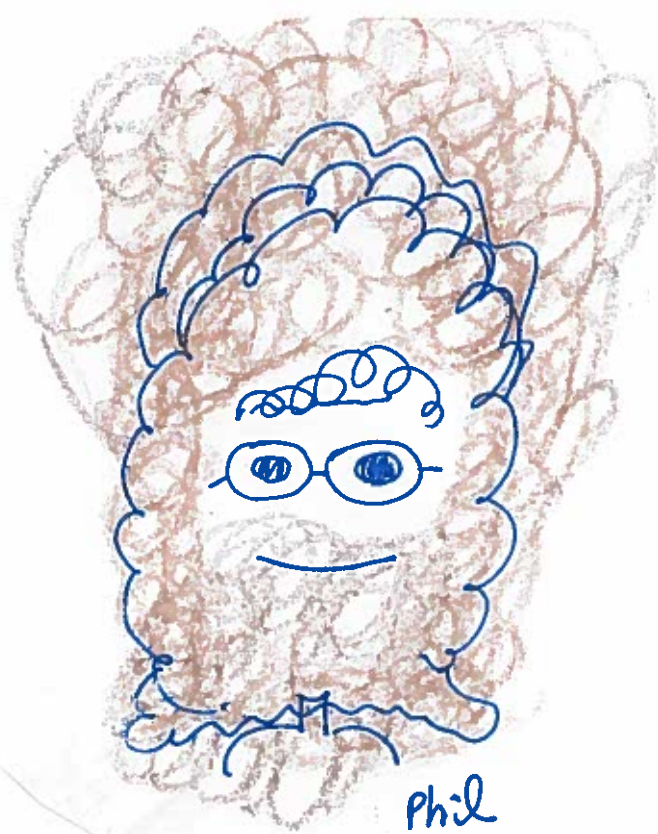


FPCA

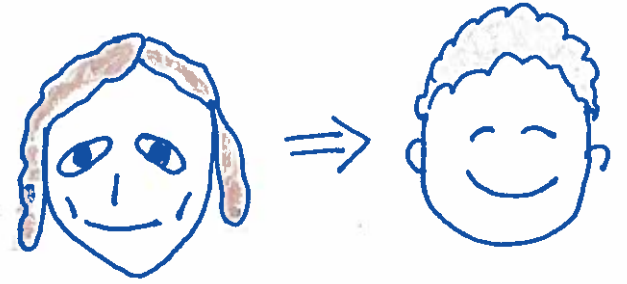
Phil Wadler



Functional Programming Languages and Computer Architectures



Phil



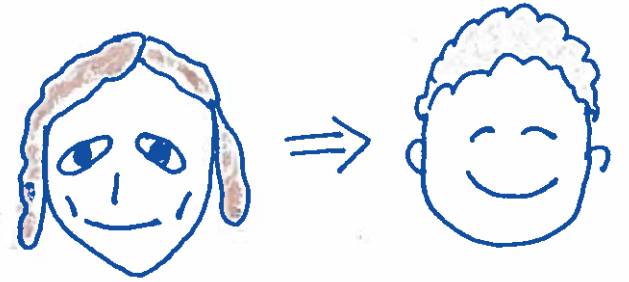
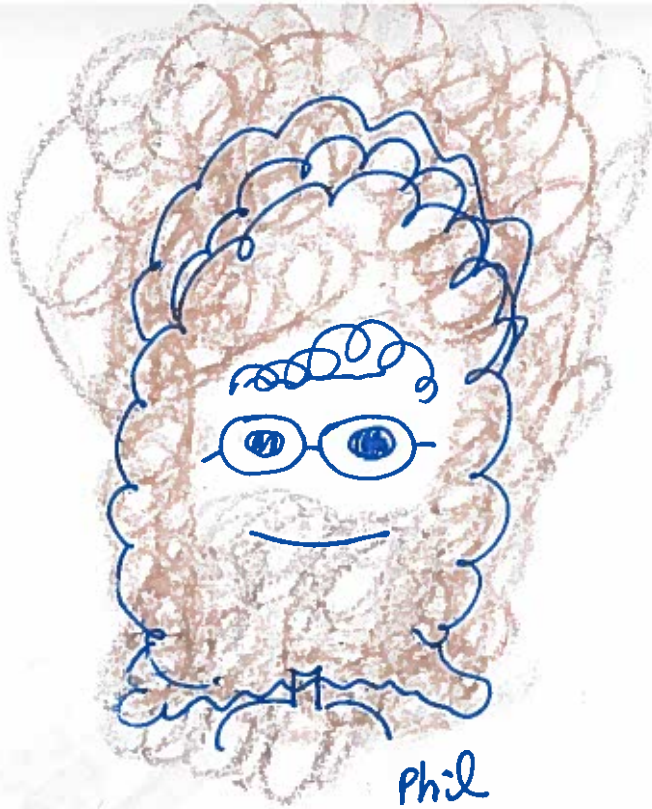
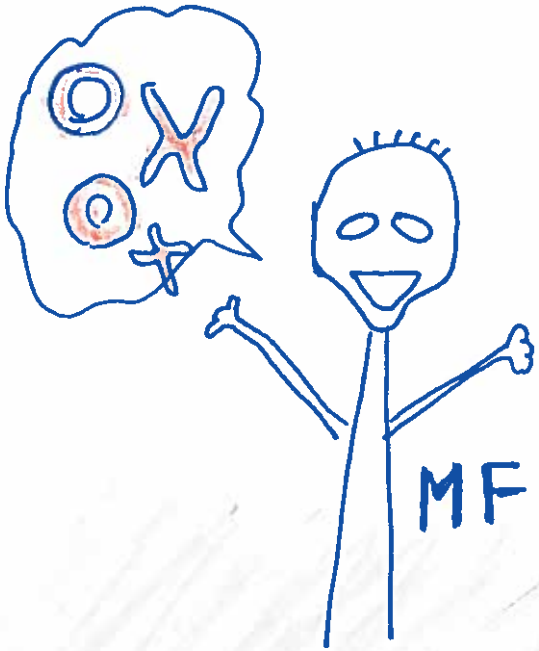
Jean - Jacques Levy



Andy Gordon



Luc Maranget



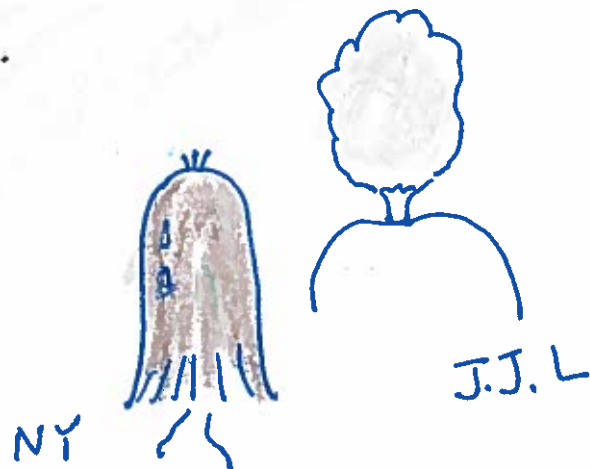
Jean - Jacque Levy



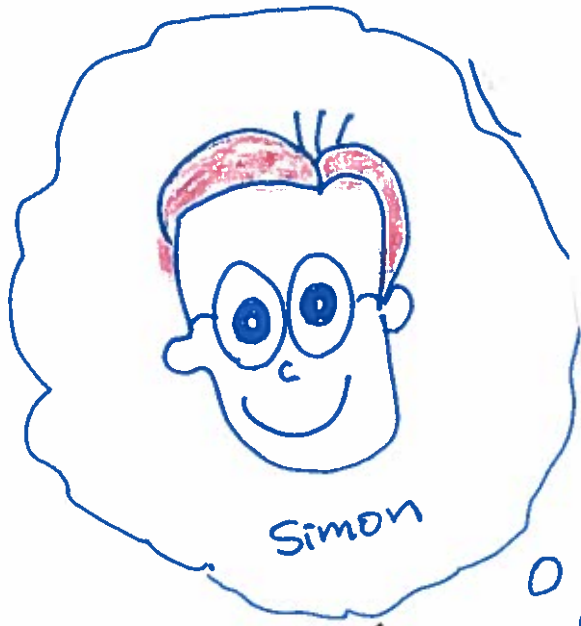
Andy Gordon



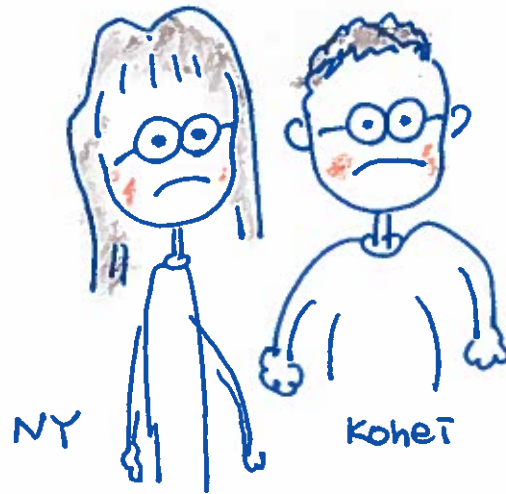
Luc Maranget



ABCD



o
o
o



SUSAN E

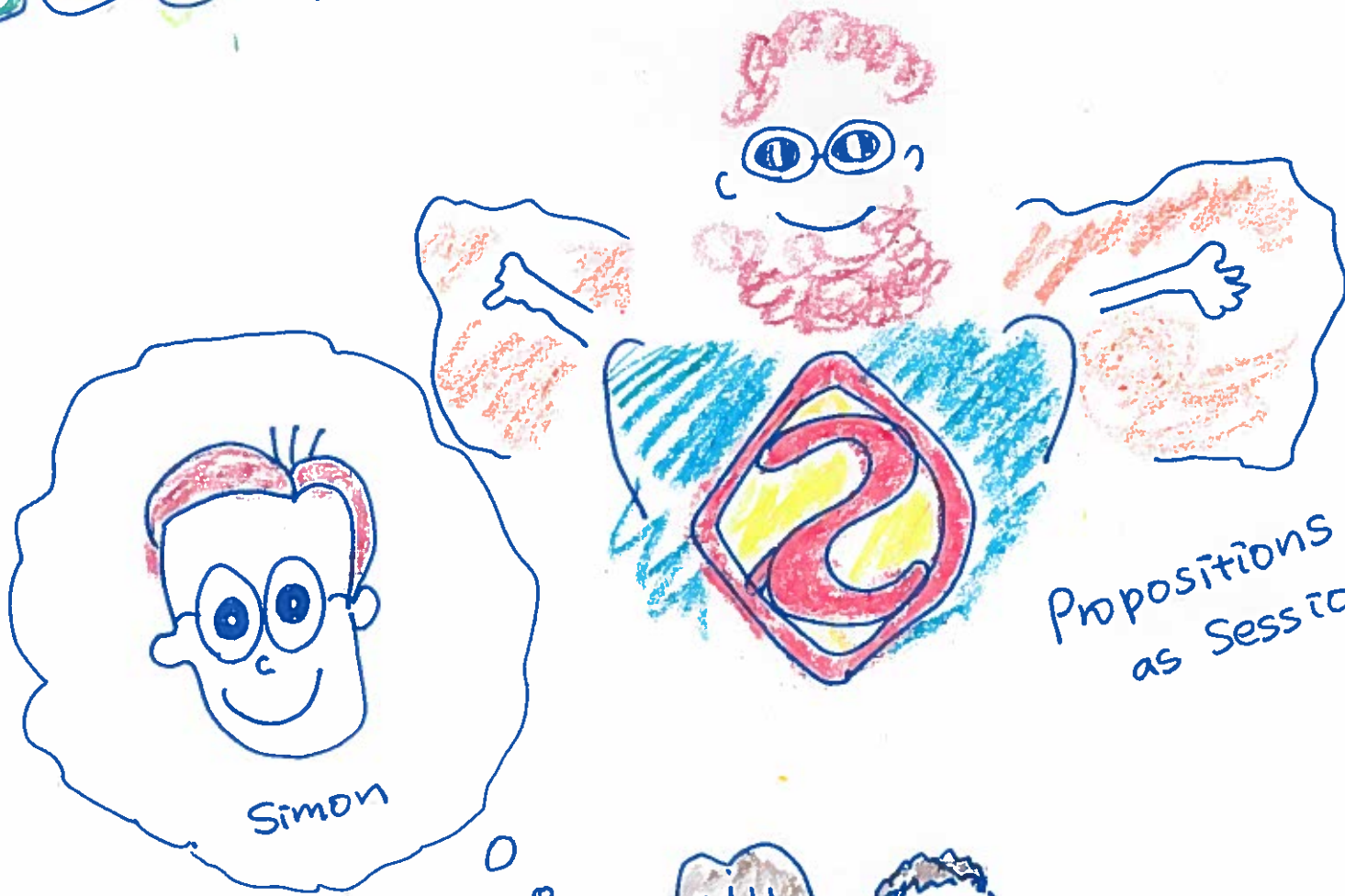
HOD

Imperial College

ABCD

program grant 2013-2018

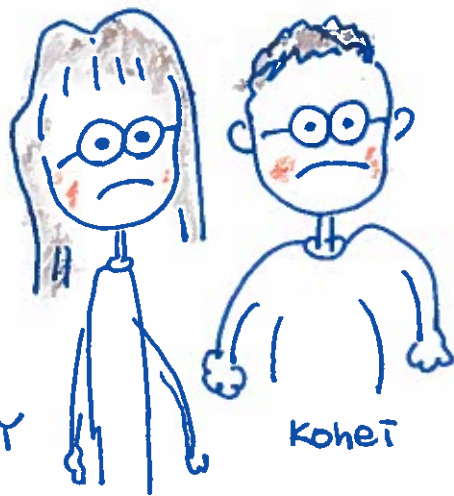
EPSRC



Propositions
as Sessions @ ICFP'12



Simon



NY

Kohei

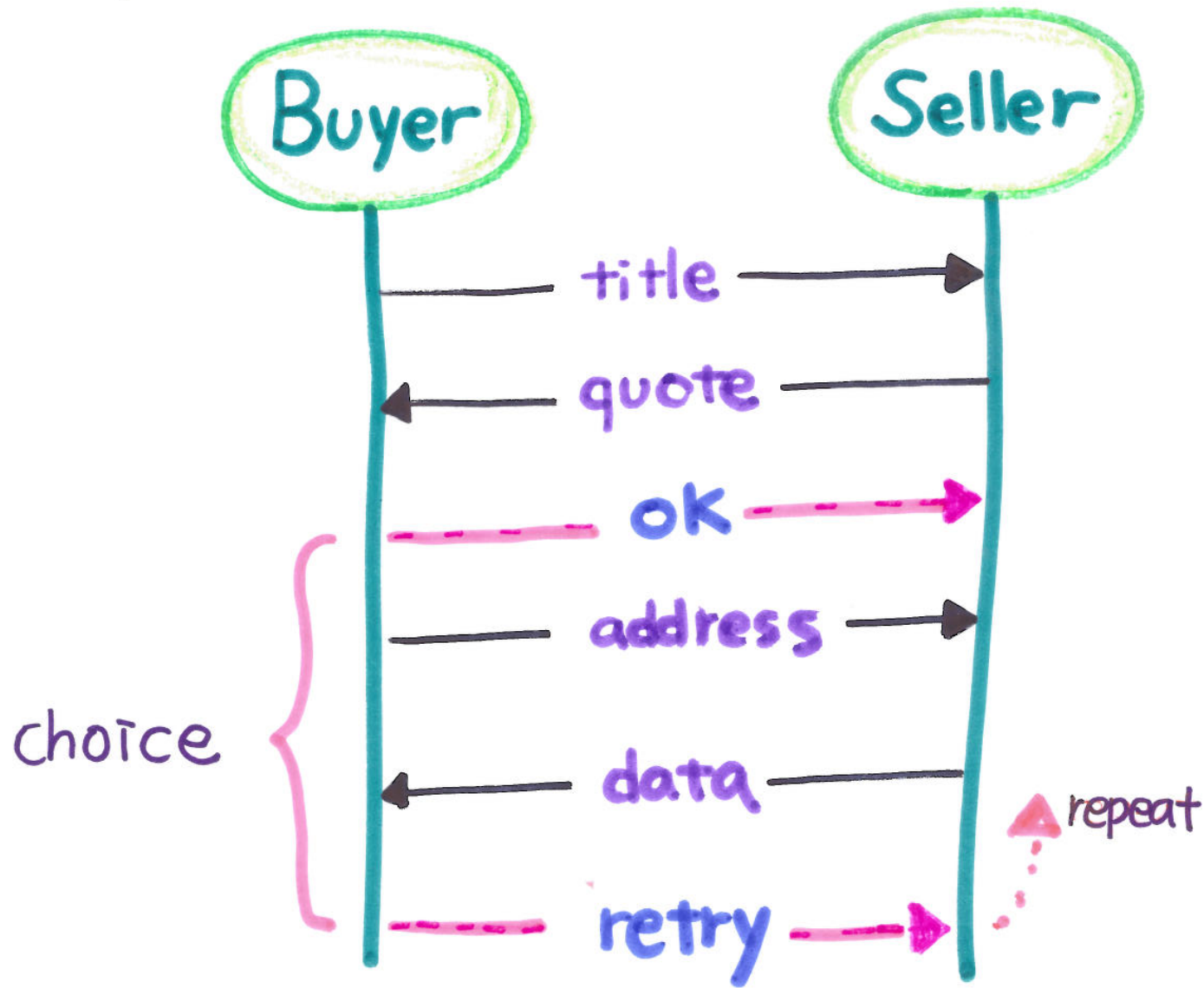


SUSAN E

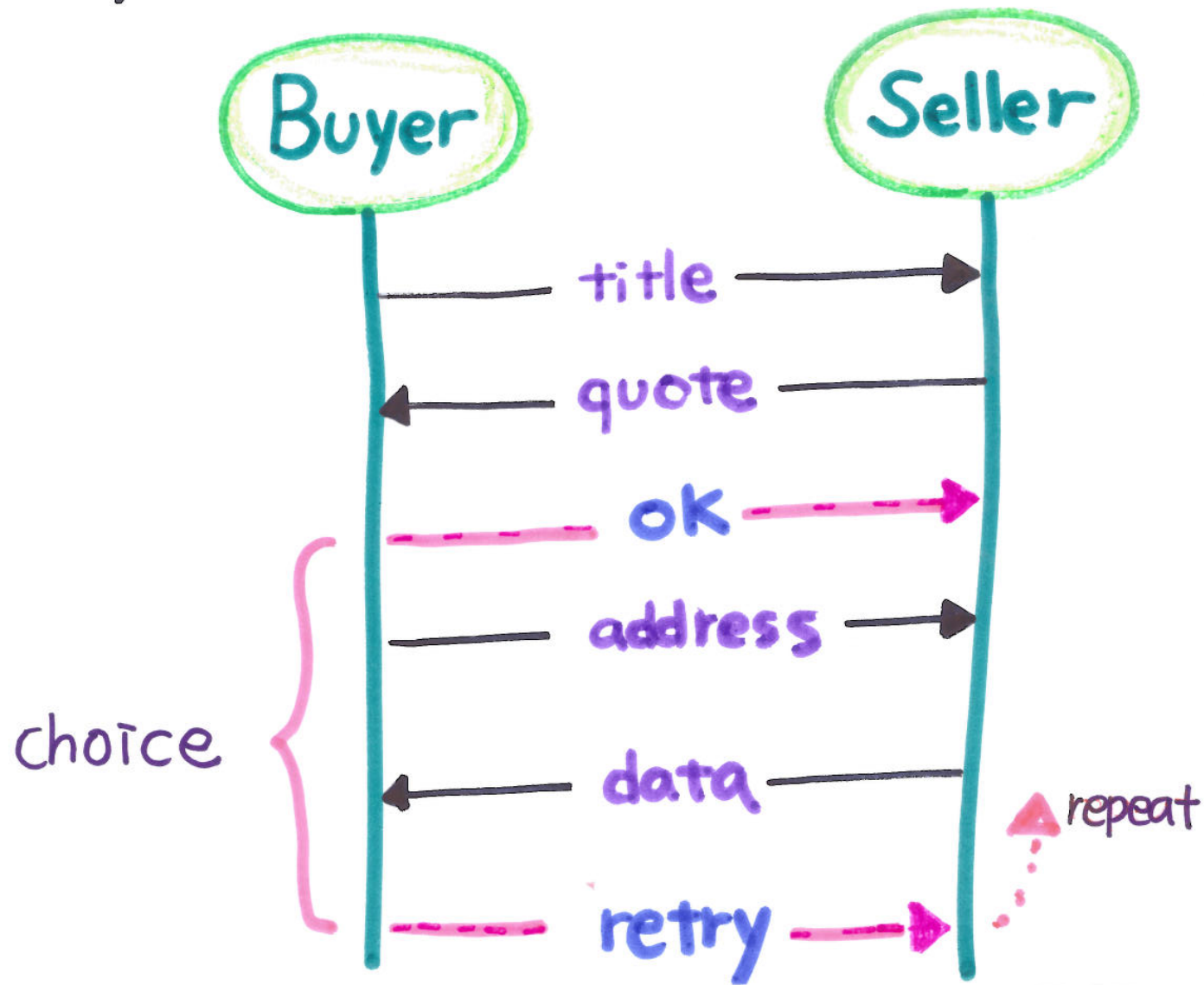
HOD

Imperial College

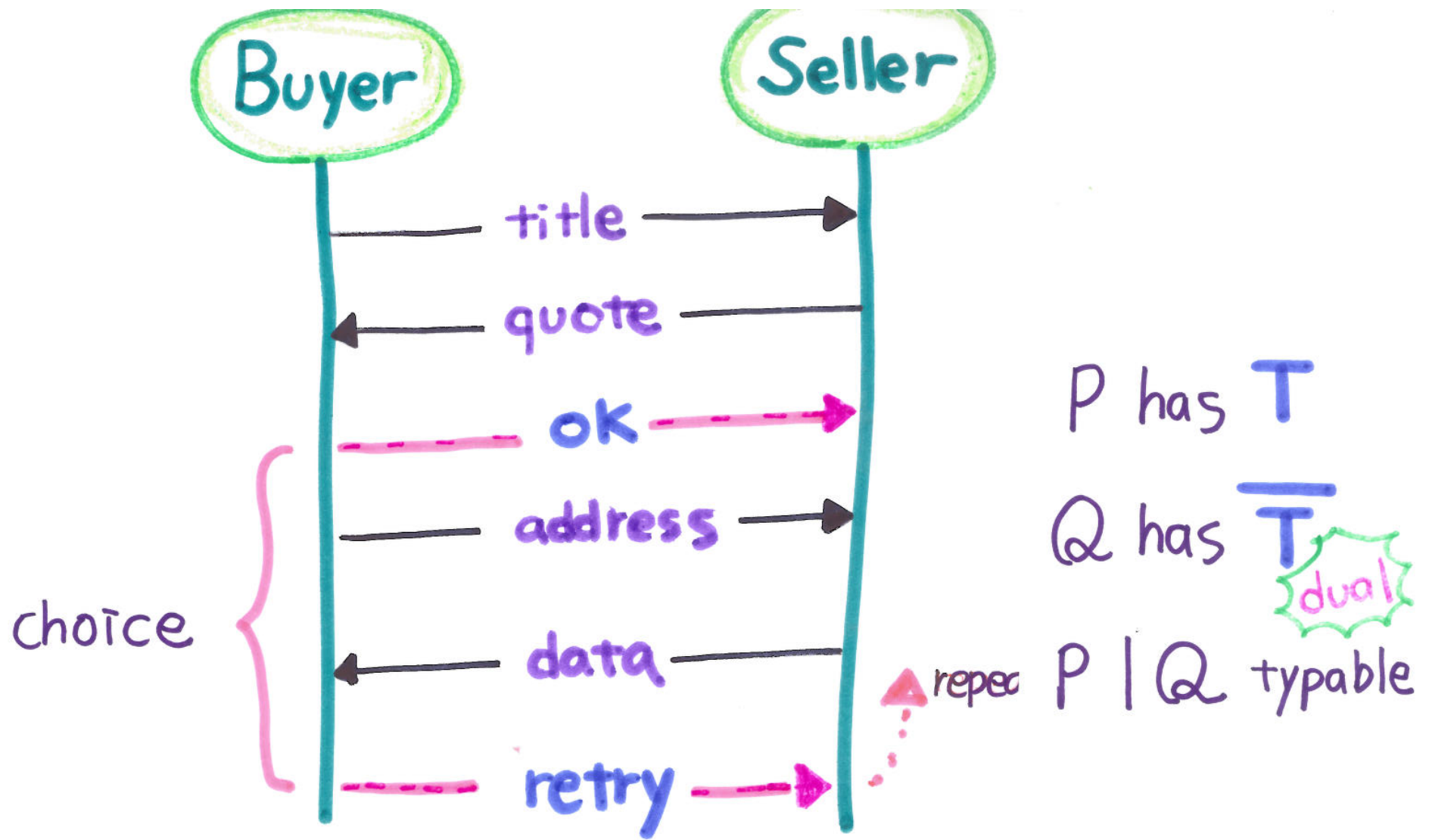
Binary Session Types: Buyer - Seller Protocol



Binary Session Types: Buyer - Seller Protocol



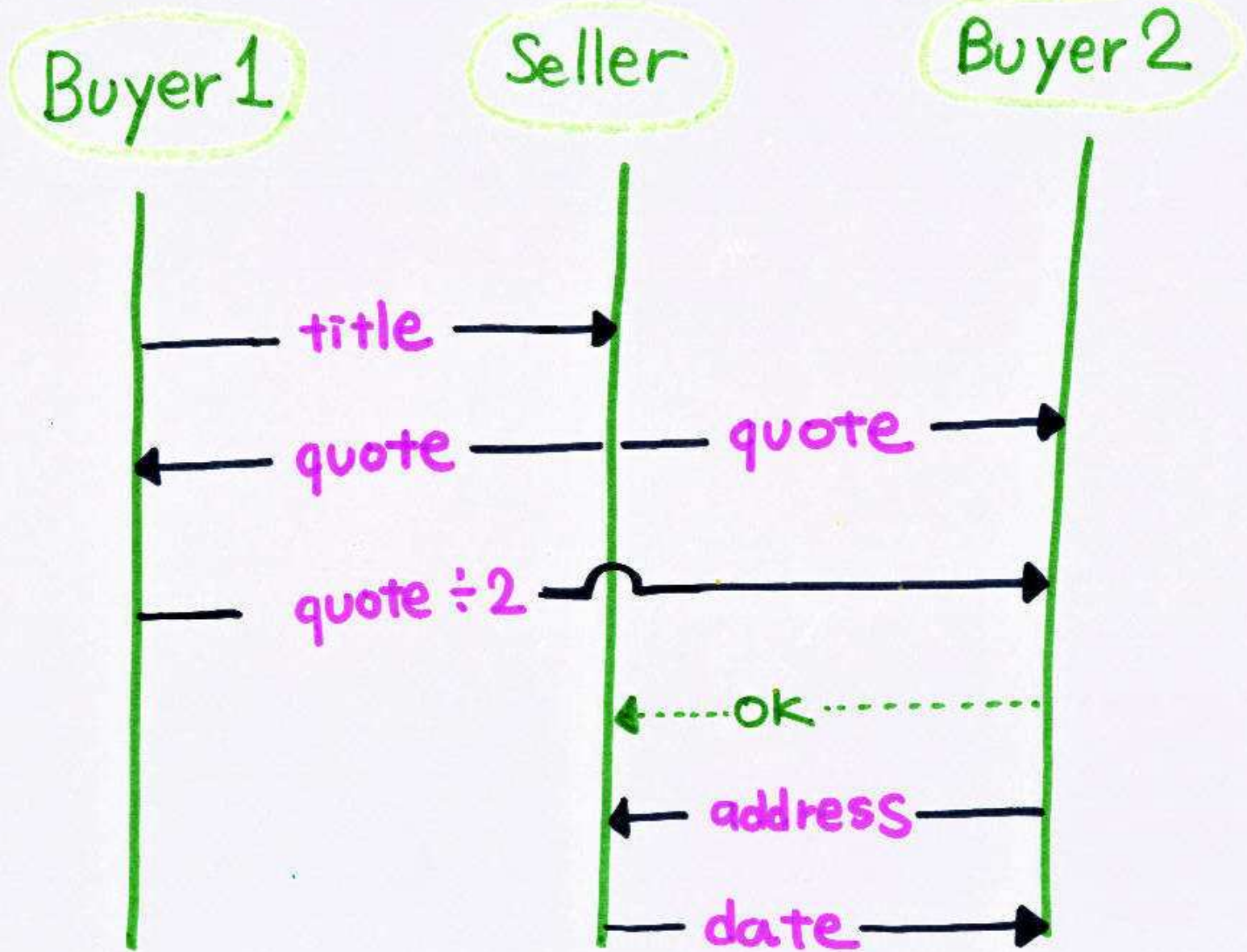
nt! Title ; ? Quote ; ! { ok: ! Add ; ? Date, retry: t }

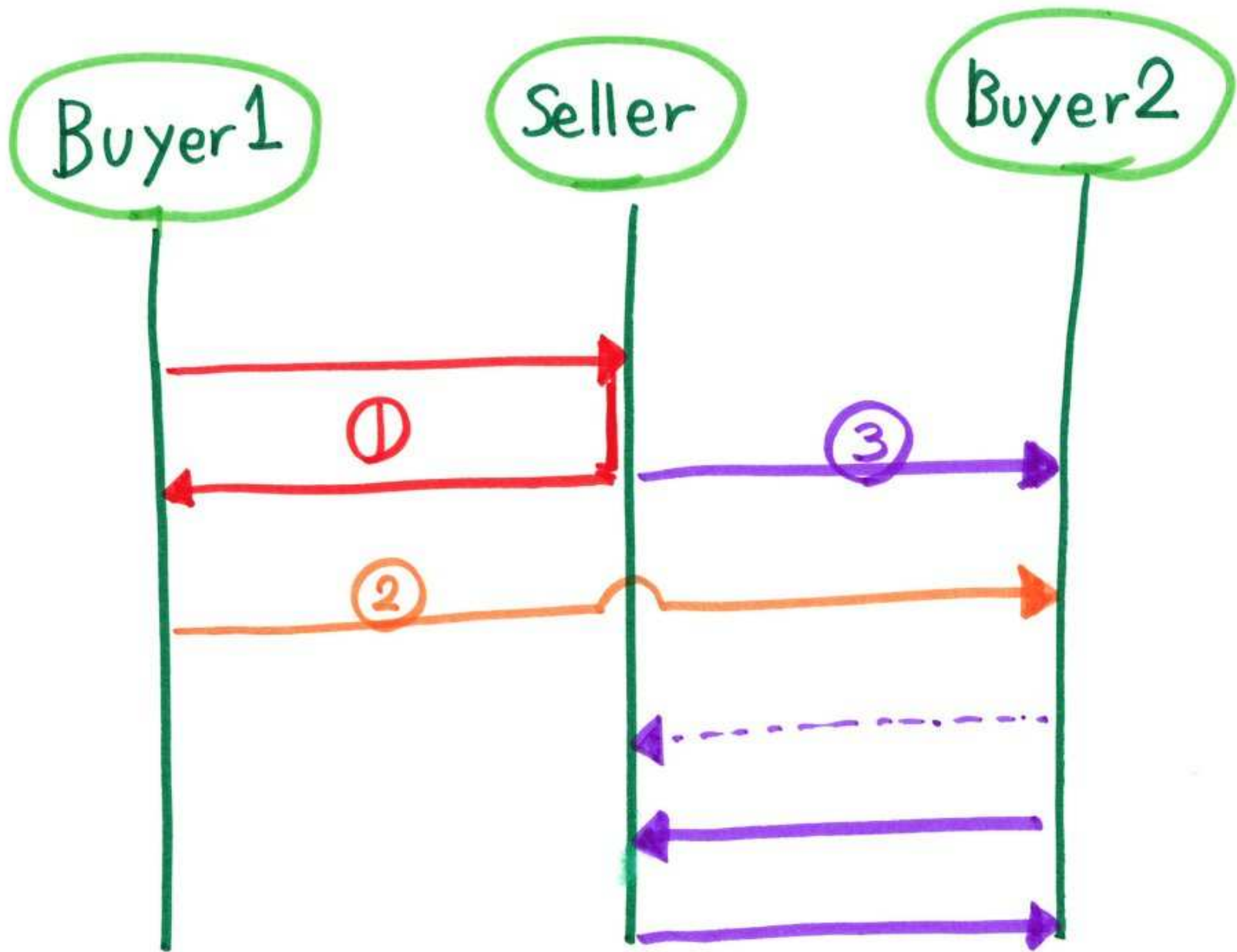


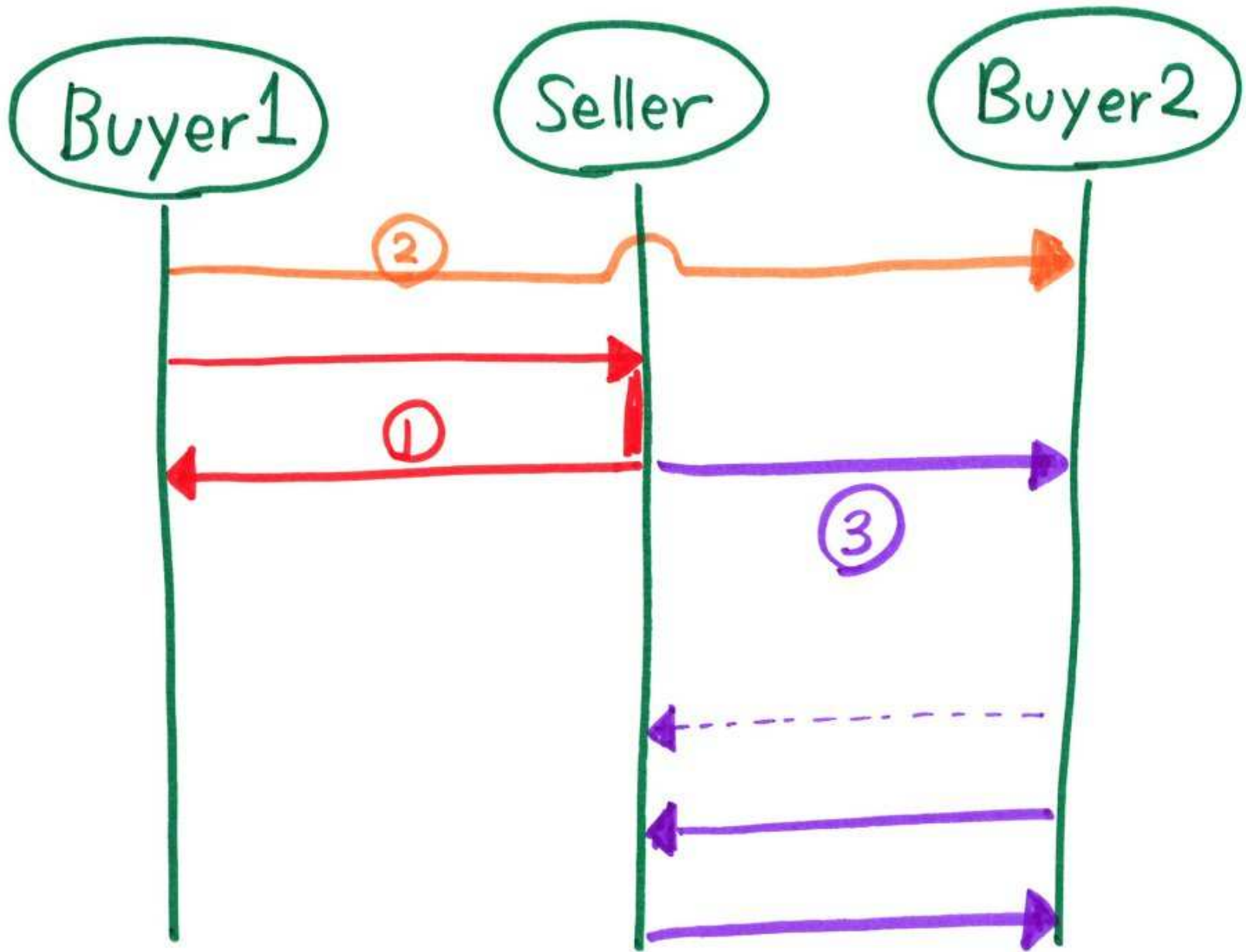
nt! Title ; ? Quote ; ! { ok: ! Add ; ? Date, retry: t }

nt? Title ; ! Quote ; ? { ok: ? Add ; ! Date, retry: t }

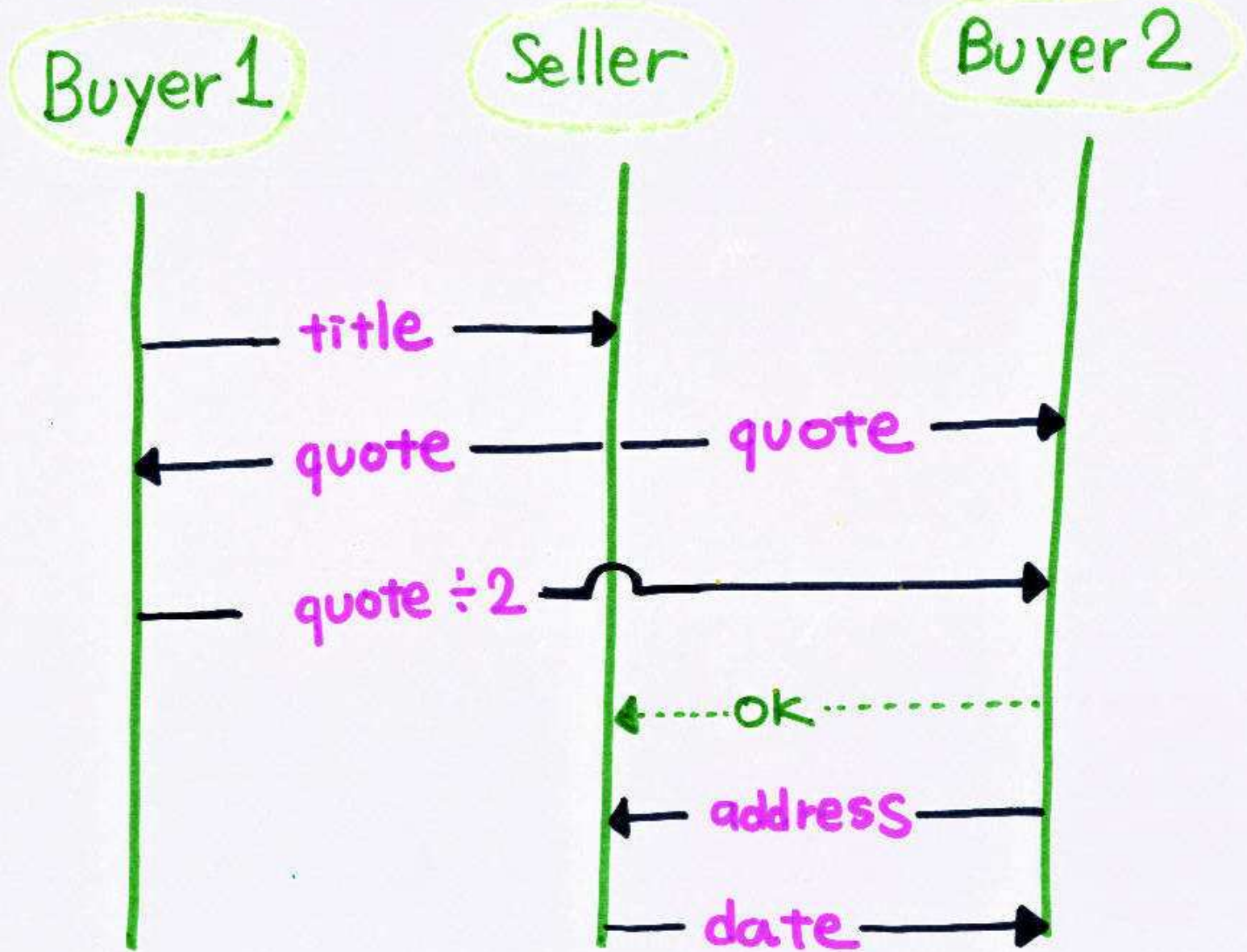
Multiparty Session Types







Multiparty Session Types



Alice

Bob

Carol

CA?c ; AB!a

AB?a ; BC!b

BC?b ; CA?c



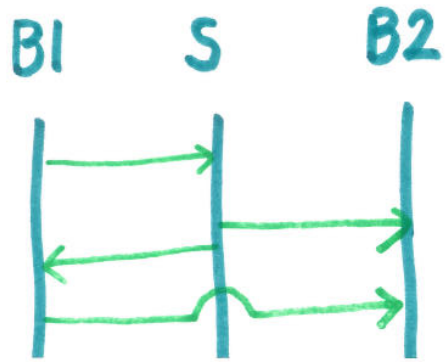
3 dual pairs

If you use
binary Session
Types ...

Deadlock!

Multi party Session Types

[Honda, Yoshida, Carbone 2008]



ⓐ

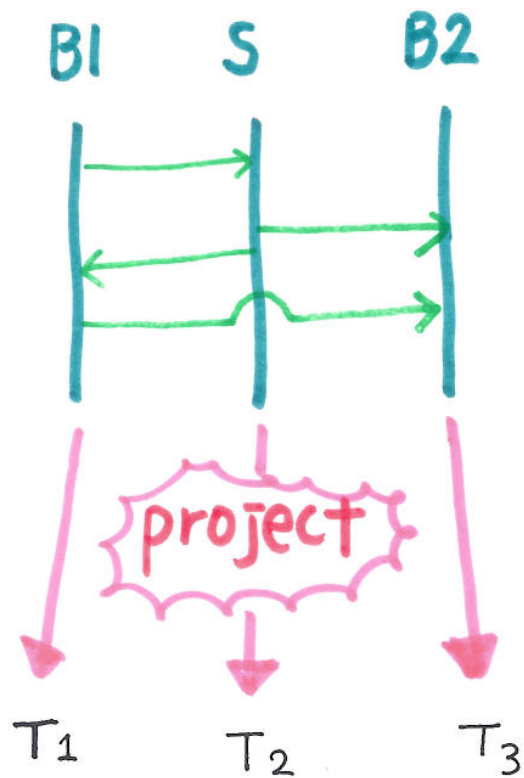
$BI \rightarrow S$ Int.

$S \rightarrow B2$ Char

STEP 1

Write Global Type

Multiparty Session Types [Honda, Yoshida, Carbone 2008]



(G)

$B1 \rightarrow S$ Int.

$S \rightarrow B2$ Char

(T)

$B1?Int. B2!Char$

STEP 1

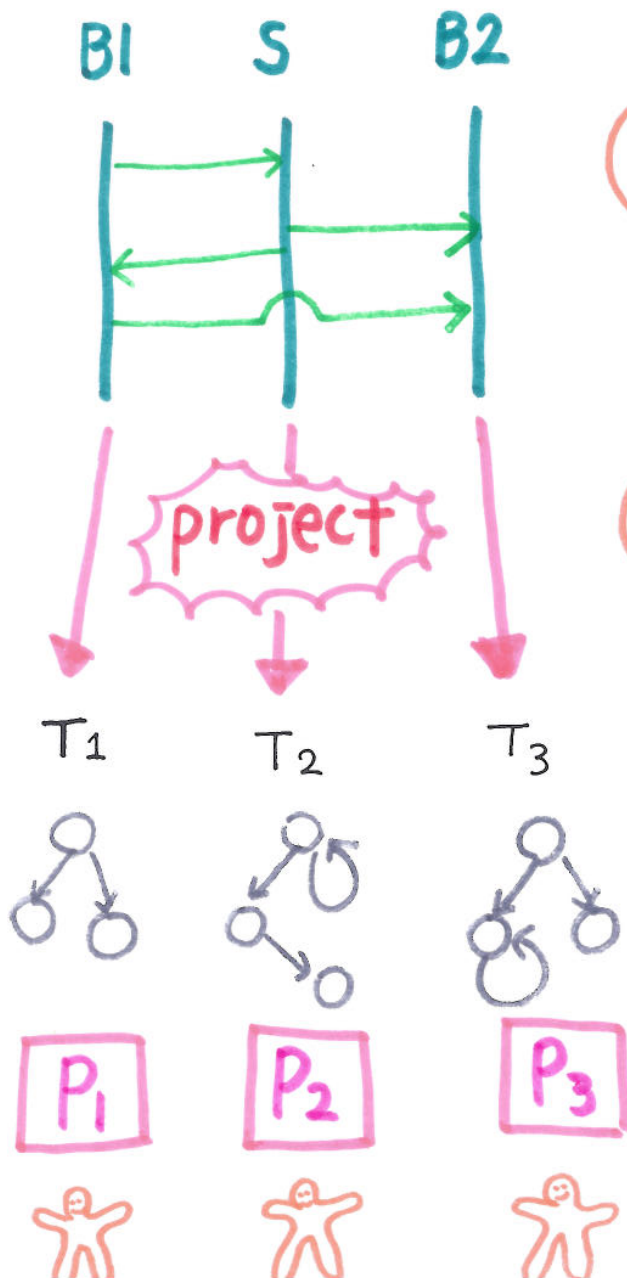
Write Global Type

STEP 2

Project to Local Types

Multiparty Session Types

[Honda, Yoshida, Carbone 2008]



(G)

$B1 \rightarrow S$ Int.

$S \rightarrow B2$ Char

(T)

$B1?Int. B2!Char$

(P)

$B1?(x). B2!<"apple">$

STEP 1

Write Global Type

STEP 2

Project to Local Type

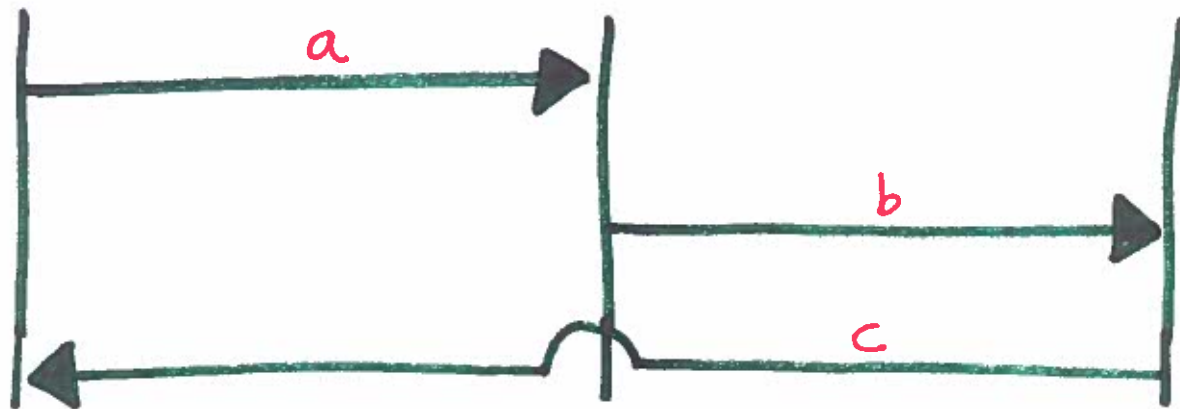
STEP 3

- Static Check
- Generate Code
- Run-time check

Alice

Bob

Carol



Global Type



Alice $AB!a; CA?c$

Bob $AB?a; BC!b$

Carol $BC?b; CA!c;$

LOCAL TYPES

NO Deadlock

Us ∈ **M**obility **R**esearch **G**roup



MobilityReadingGroup

π -calculus, Session Types research at Imperial College

[Home](#) [People](#) [Publications](#) [Grants](#) [Talks](#) [Tools](#) [Awards](#) [Kohel Honda](#)

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/>

Selected Publications 2016/2017



- **[ECOOP'17]** Alceste Scala, Raymond Hu, Ornella Darda, NY: A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming..
- **[COORDINATION'17]** Keigo Imai, NY and Shoji Yuen: Session-ocaml: a session-based library with polarities and lenses.
- **[FoSSaCS'17]** Julien Lange , NY: On the Undecidability of Asynchronous Session Subtyping.
- **[FASE'17]** Raymond Hu , NY: Explicit Connection Actions in Multiparty Session Types.
- **[CC'17]** Romyana Neykova , NY: Let It Recover: Multiparty Protocol-Induced Recovery.
- **[POPL'17]** Julien Lange , Nicholas Ng , Bernardo Toninho , NY: Fencing off Go: Liveness and Safety for Channel-based Programming.
- **[FPL'16]** Xinyu Niu , Nicholas Ng , Tomofumi Yuki , Shaojun Wang , NY, Wayne Luk : EURECA Compilation: Automatic Optimisation of Cycle-Reconfigurable Circuits.
- **[ECOOP'16]** Alceste Scala, NY: Lightweight Session Programming in Scala
- **[CC'16]** Nicholas Ng, NY: Static Deadlock Detection for Concurrent Go by Global Session Graph Synthesis.
- **[FASE'16]** Raymond Hu, NY: Hybrid Session Verification through Endpoint API Generation.
- **[TACAS'16]** Julien Lange, NY: Characteristic Formulae for Session Types.
- **[ESOP'16]** Dimitrios Kouzapas, Jorge A. Pérez, NY: On the Relative Expressiveness of Higher-Order Session Processes.
- **[POPL'16]** Dominic Orchard, NY: Effects as sessions, sessions as effects .

Selected Publications 2016/2017



- [ECOOP'17] Alceste Scala, Raymond Hu, Ornela Darda, NY :A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming.
- [COORDINATION'17] Keigo Imai, NY and Shoji Yuen: Session-ocaml: a session-based library with polarities and lenses.
- [FoSSaCS'17] Julien Lange , NY : On the Undecidability of Asynchronous Session Subtyping.
- [FASE'17] Raymond Hu , NY : Explicit Connection Actions in Multiparty Session Types.
- [CC'17] Rumyana Neykova , NY: Let It Recover: Multiparty Protocol-Induced Recovery.
- [POPL'17] Julien Lange , Nicholas Ng , Bernardo Toninho , NY: Fencing off Go: Liveness and Safety for Channel-based Programming.
- [FPL'16] Xinyu Niu , Nicholas Ng , Tomofumi Yuki , Shaojun Wang , NY, Wayne Luk: EURECA Compilation: Automatic Optimisation of Cycle-Reconfigurable Circuits.
- [ECOOP'16] Alceste Scala, NY: Lightweight Session Programming in Scala
- [CC'16] Nicholas Ng, NY: Static Deadlock Detection for Concurrent Go by Global Session Graph Synthesis.
- [FASE'16] Raymond Hu, NY: Hybrid Session Verification through Endpoint API Generation.
- [TACAS'16] Julien Lange, NY: Characteristic Formulae for Session Types.
- [ESOP'16] Dimitrios Kouzapas, Jorge A. Pérez, NY: On the Relative Expressiveness of Higher-Order Session Processes.
- [POPL'16] Dominic Orchard, NY: Effects as Sessions, Sessions as Effects.

Relationship with Communicating AUTOMATA

ESOP 12

Characterisation

ICALP 13

Synthesis

CONCUR 15

Timed

Automata

POPL 15

Graphical

Synthesis

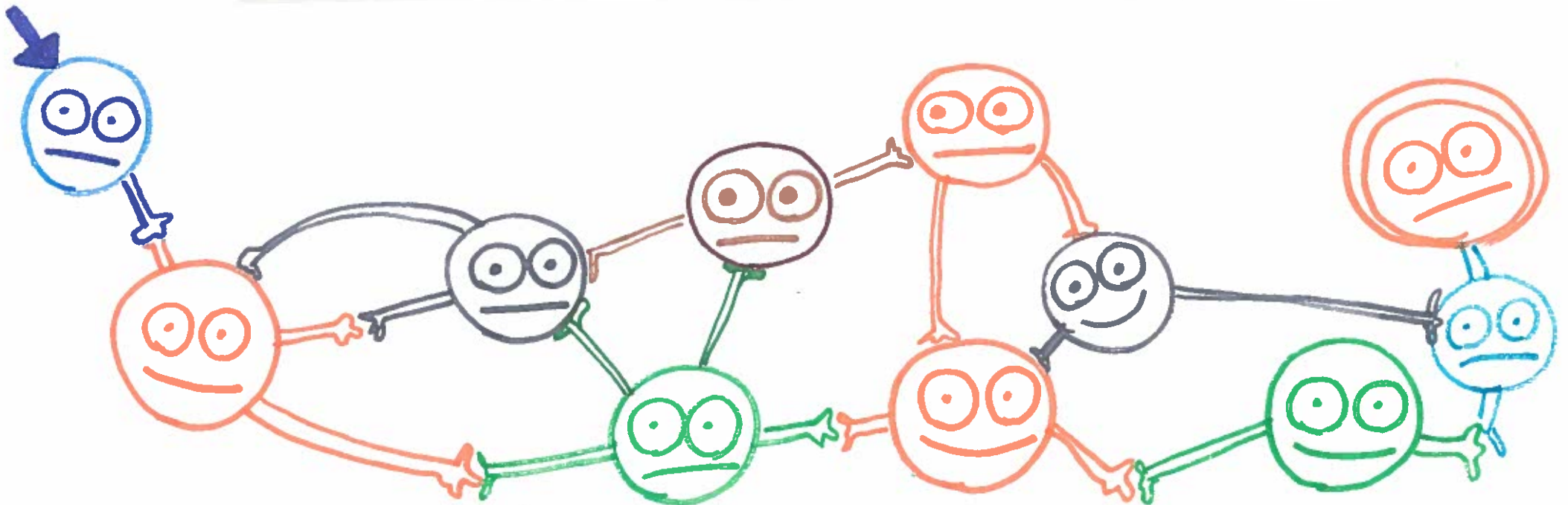
TACAS 16

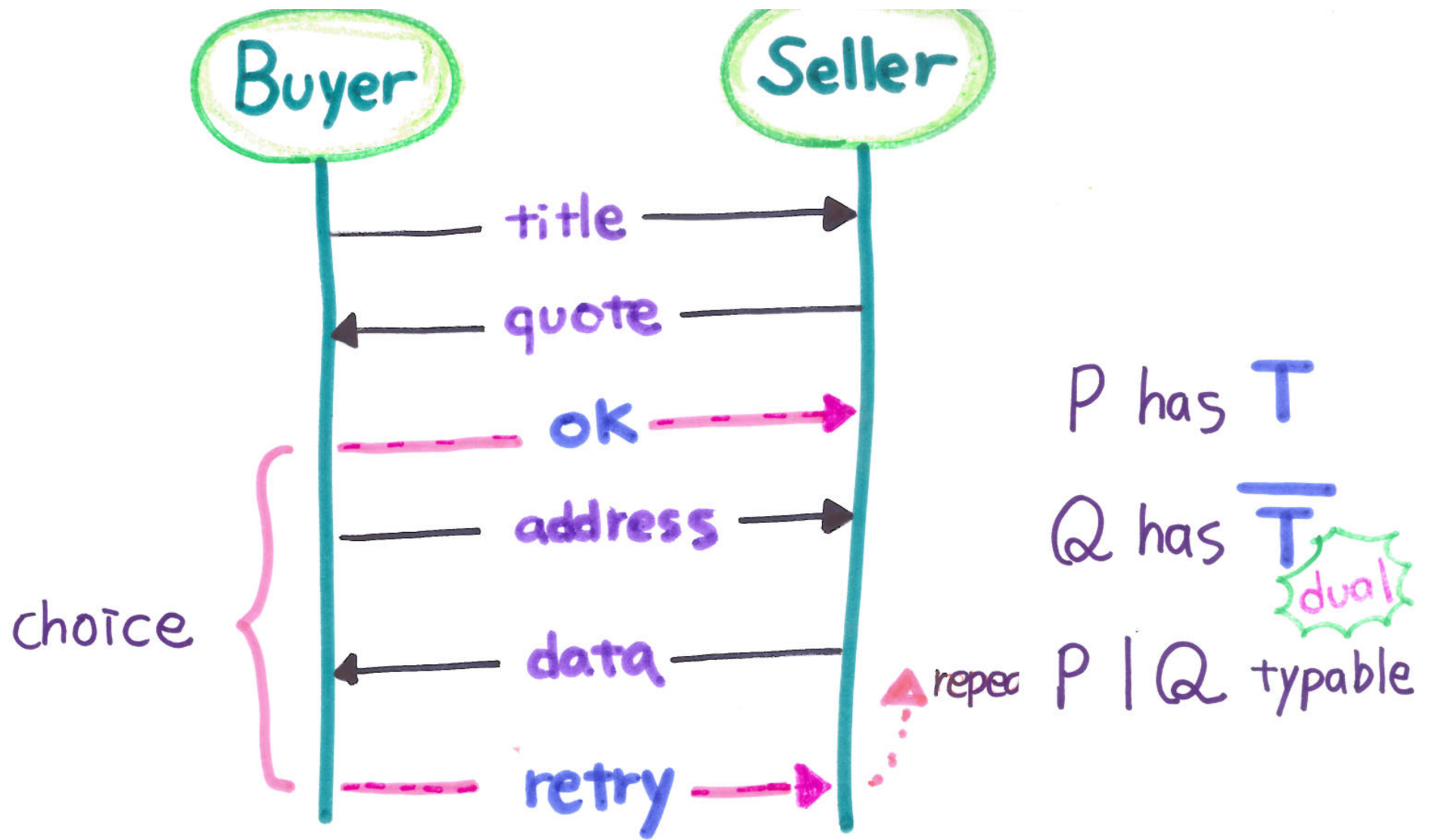
Subtyping

+ Model-checking

FOSSACS 17

Undecidability

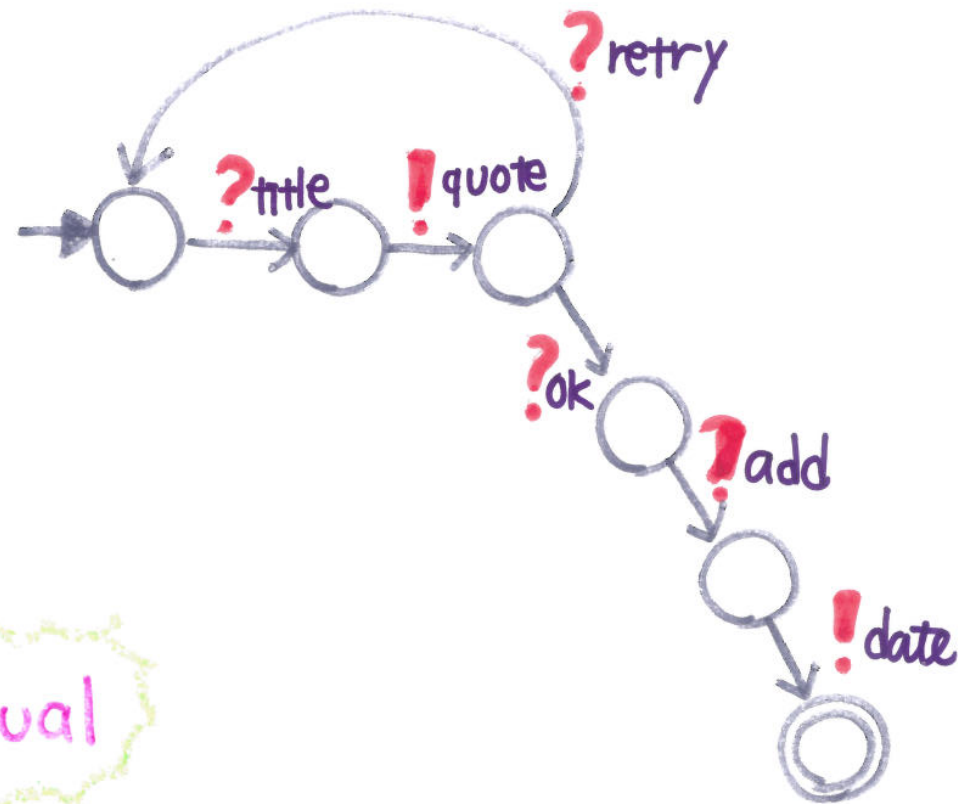
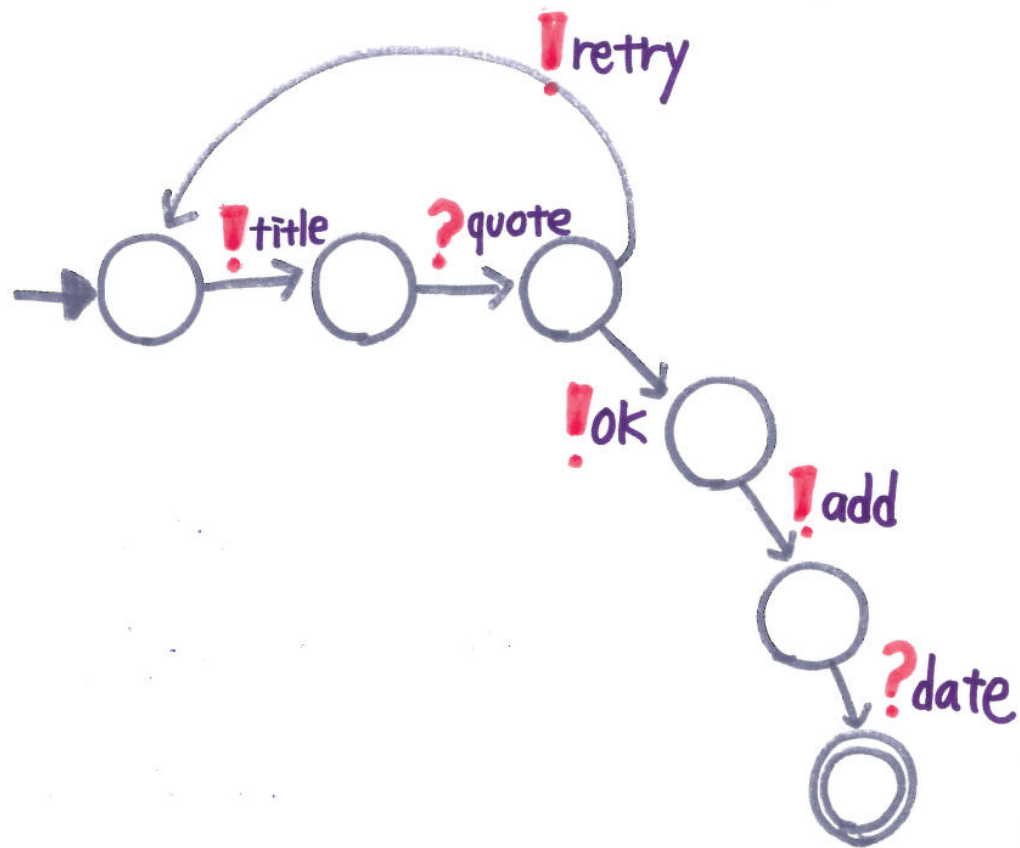




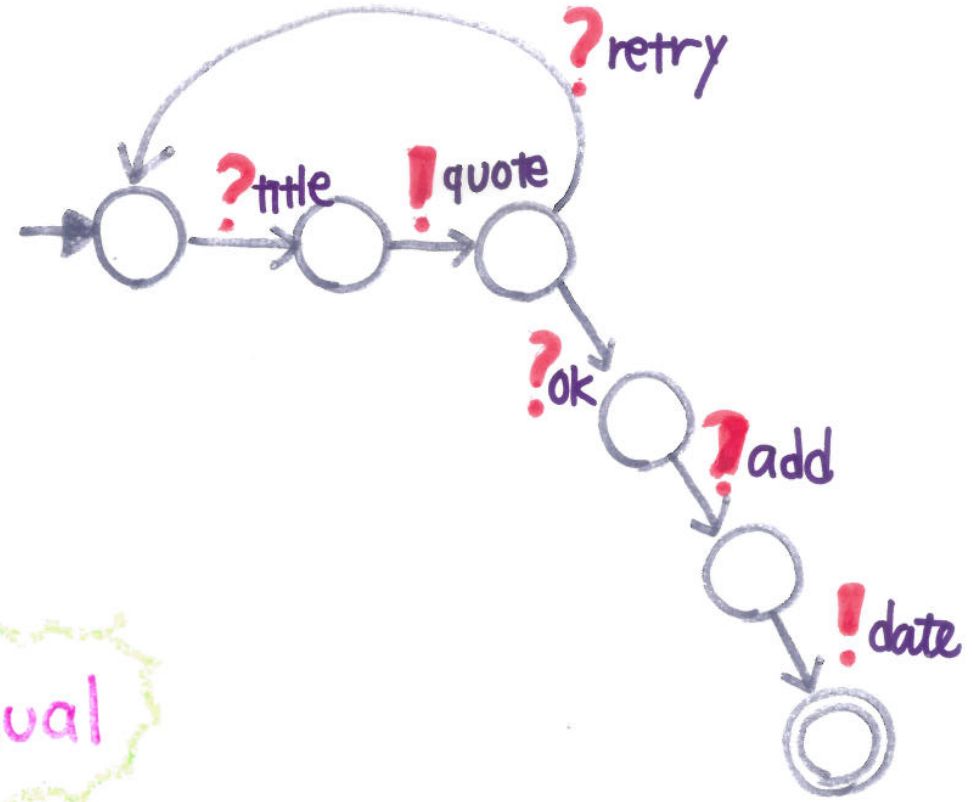
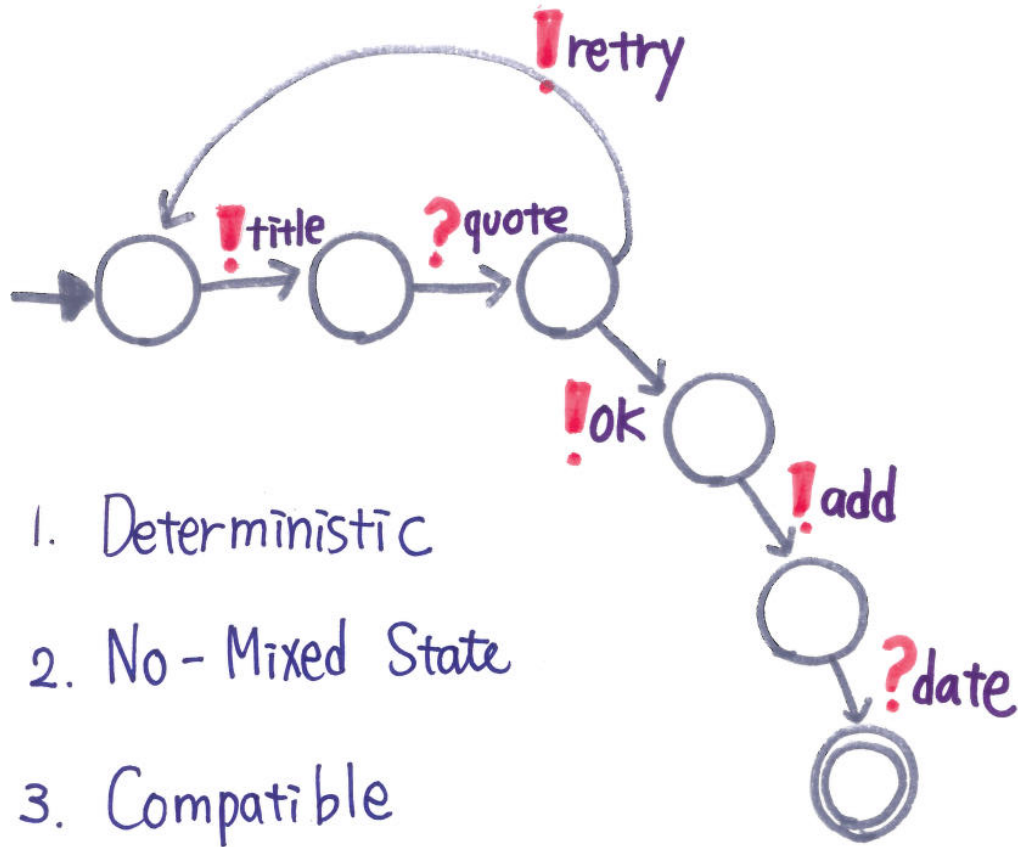
nt! Title ; ? Quote ; ! { ok: ! Add ; ? Date, retry: t }

nt? Title ; ! Quote ; ? { ok: ? Add ; ! Date, retry: t }

Communicating Automata [1980s]



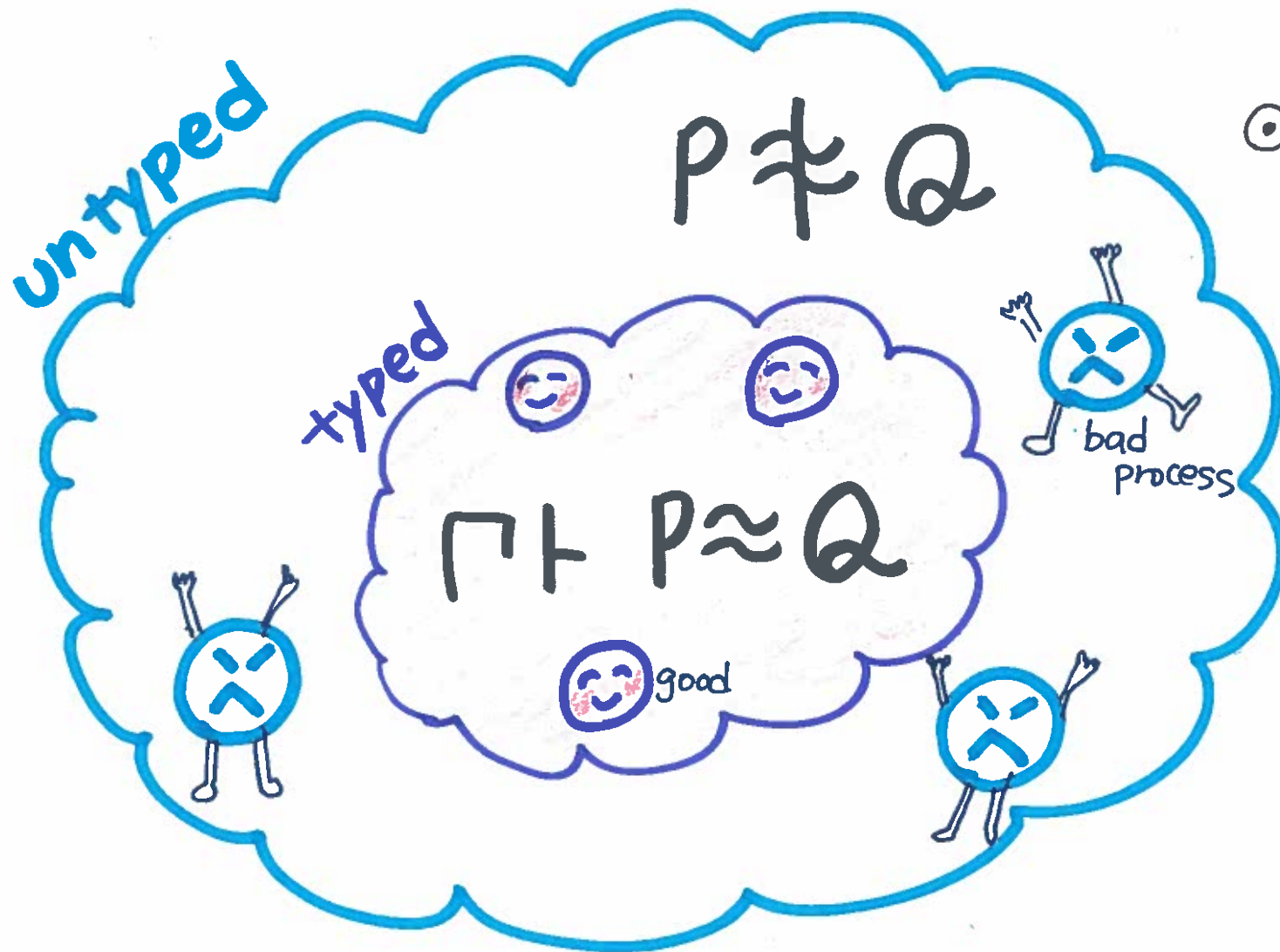
dual



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

Typed Semantics in π 1991 \rightarrow

IO-subtyping, Linear types, Secure Information Flow, ...



- ⊙ Correctness of Encoding
- ⊙ Limit environment \sqcap
 \Rightarrow Equate more processes
- ⊙ Compositional

GLOBALLY GOVERNED SESSION SEMANTICS

CONCUR 15
LMCS



Dimitrios
Kouzapas
Glasgow



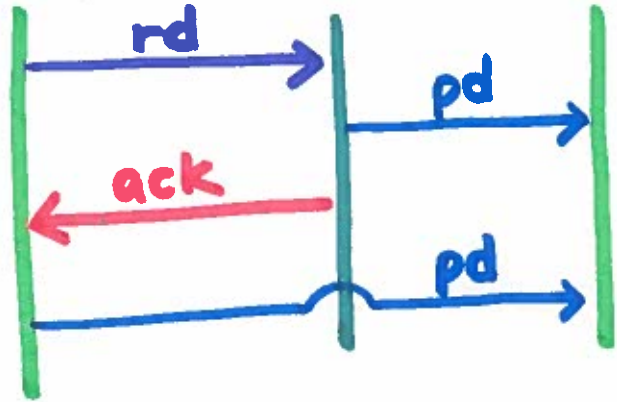
Nobuko
Yoshida

Imperial
College London



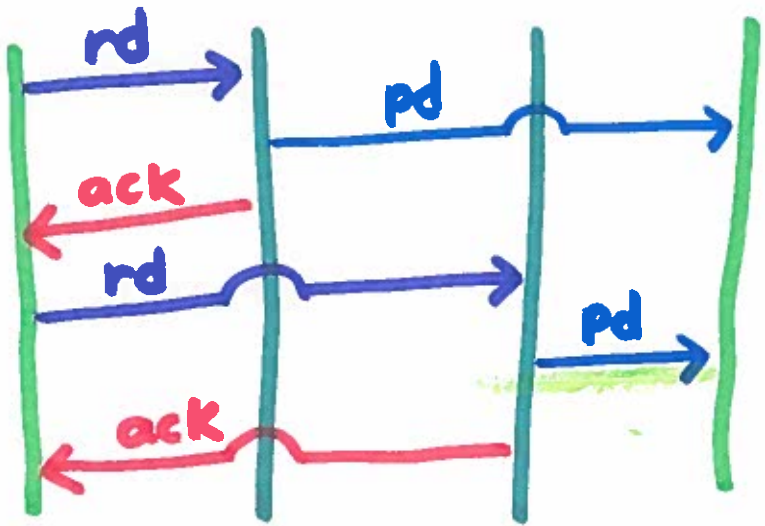
Usecase : UC.R2.13 "Acquire Data from Instrument" from Ocean Observatories Initiative (OOI)

Instrument Agent User

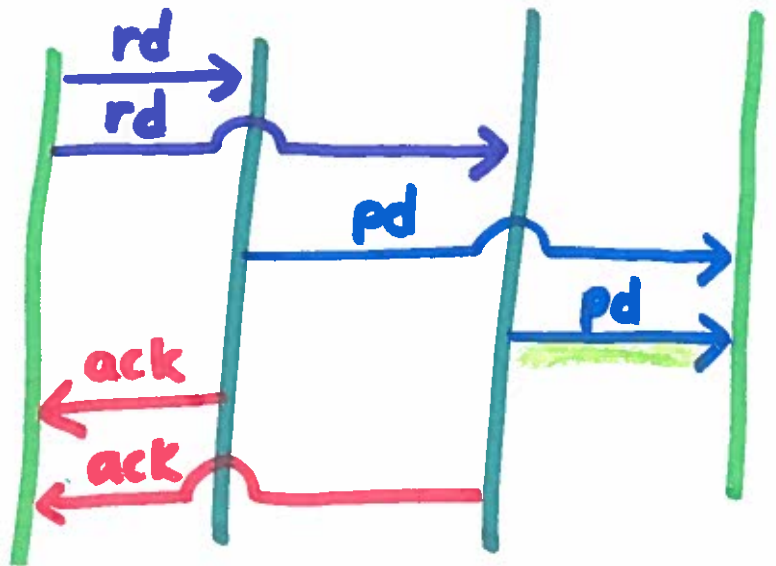


A1 → U: <pd>.
 A2 → U: <pd>.
 end

I A1 A2 U

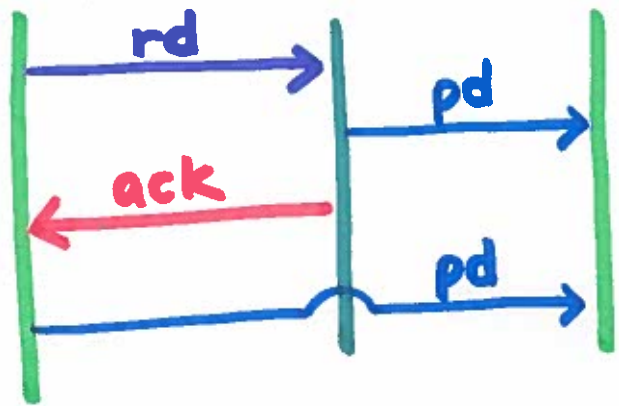


I A1 A2 U



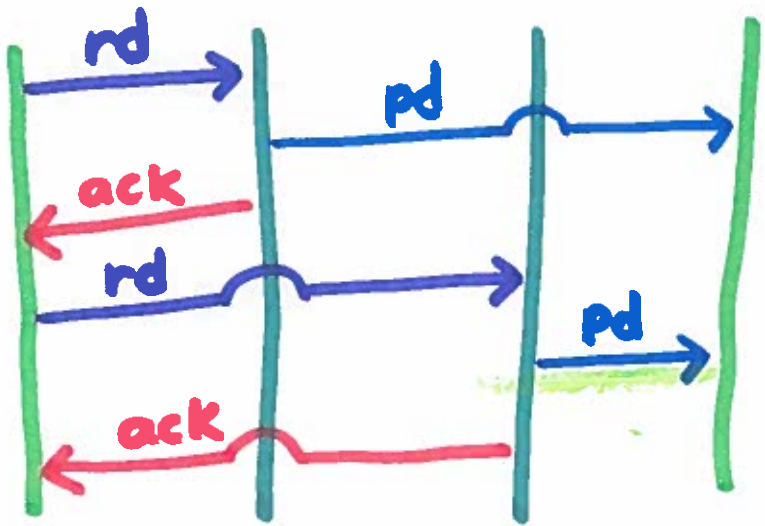
Usecase : UC.R2.13 "Acquire Data from Instrument" from Ocean Observatories Initiative (OOI)

Instrument Agent User



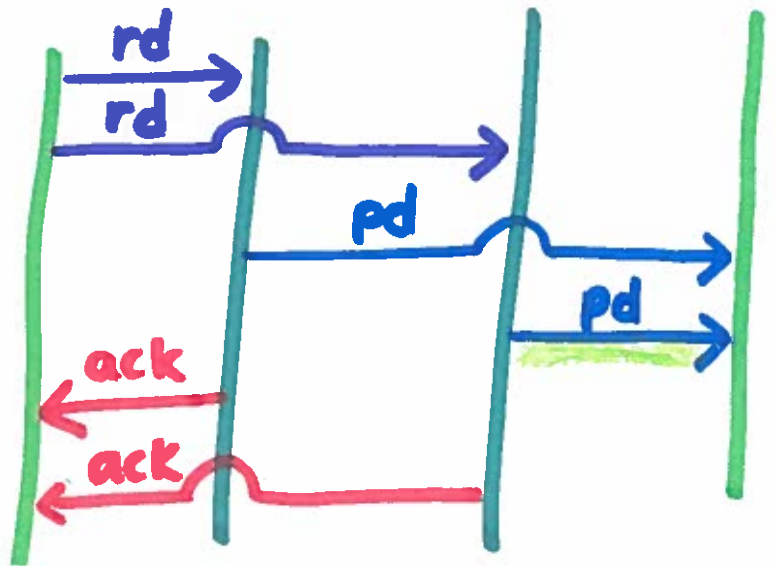
AI → U: <pd>.
 A2 → U: <pd>.
 end

I A1 A2 U



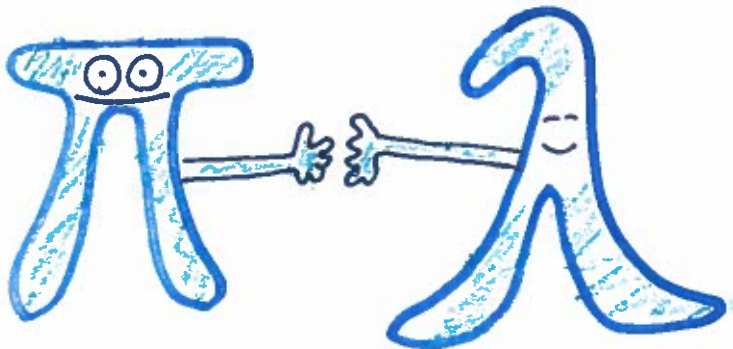
≈ g

I A1 A2 U



HOT and

TYPES

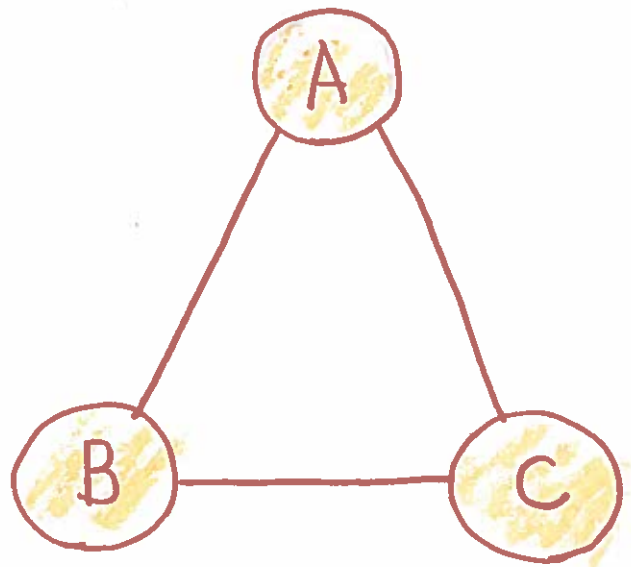


CONCUR 16 Characteristic
Bisimulations

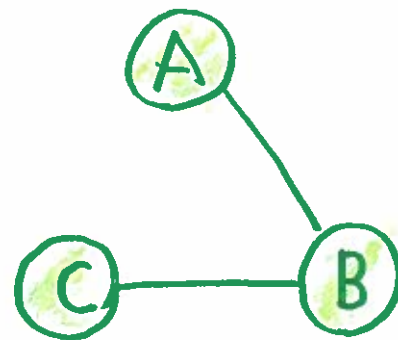
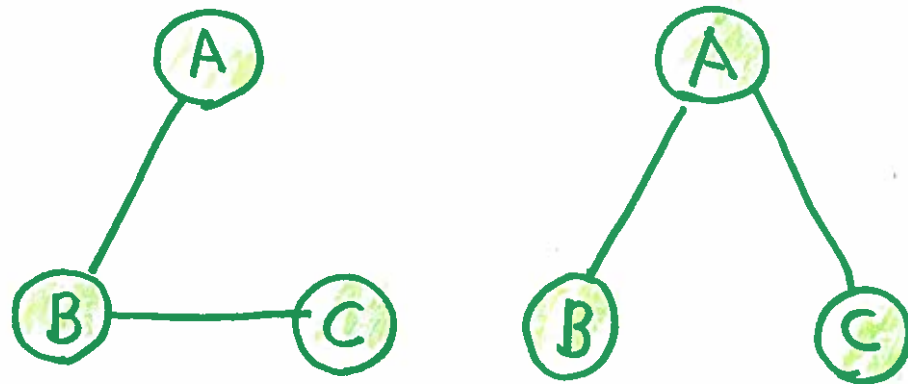
ESOP 16 Expressiveness

ACTA INFORMATICA

Interconnectability of Session Logical Processes



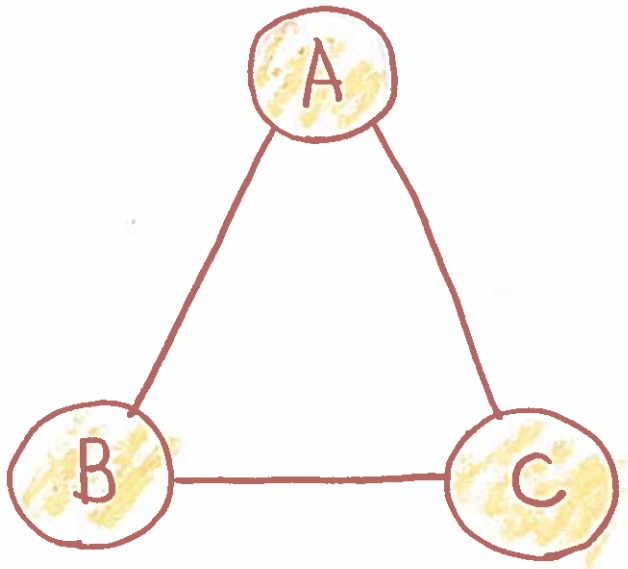
Multi Party



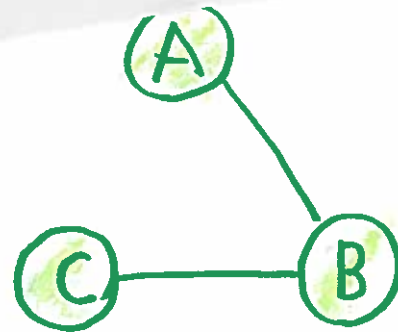
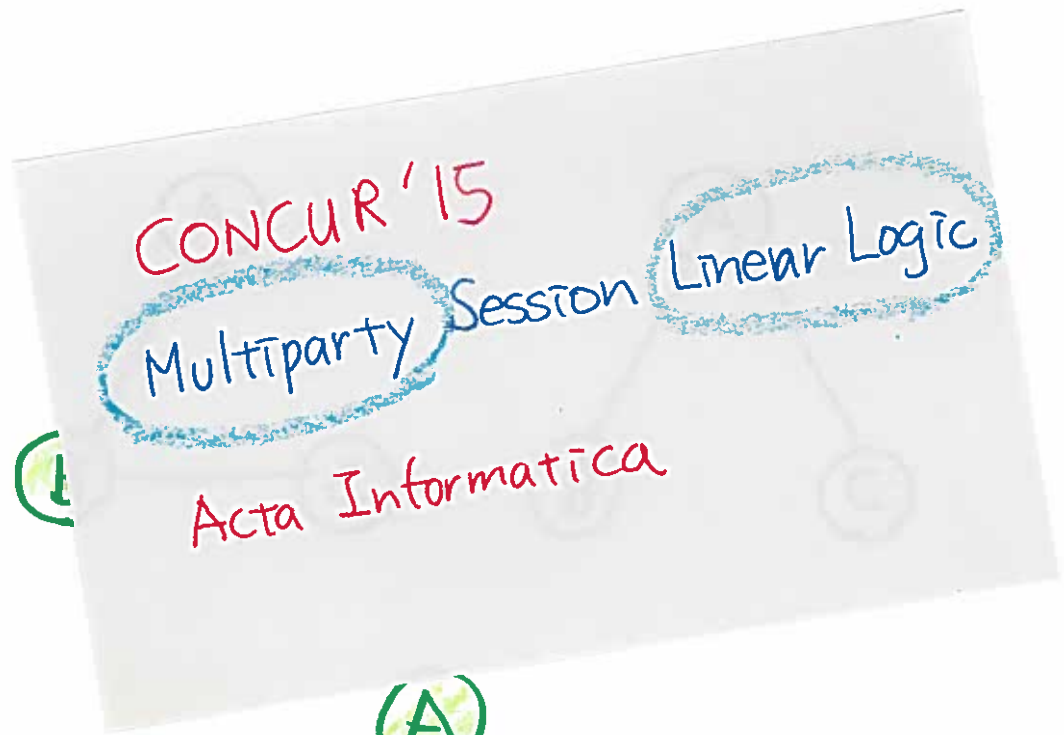
Linear Logical Session

cf. [1995] Specification Structures and Proposition
ac TVBDC

Interconnectability of Session Logical Processes



Multi Party



Linear Logical Session

cf. [1995] Specification Structures and Proposition
as TVBAC



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/>

```
1  module examples;
2
3  global protocol HelloWorld(role Me, role World) {
4    hello() from Me to World;
5    choice at World {
6      goodMorning1() from World to Me;
7    } or {
8      goodMorning1() from World to Me;
9    }
10 }
11
```

Load a sample 

Check

Protocol:

Role:

Project

Generate Graph

OOI Collaboration

OOI OCEAN OBSERVATORY INITIATIVE

Location
CURRENT LOCATION

SEARCH

RESOURCES

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

Welcome to Release 2 of the Ocean Observing Initiative (OOI). You already have access to many OOI features and real-time data. You can also customize the functionality on this page for your use of the OOI as our Guest.

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

RECENT UPDATES

ID	Name	Date	Type	Observatory	Location
01	01 - Oregon Coast Wave Hinge	2012-01-01 22:00:00	Wave	Wave	North Pacific Ocean
02	02 - California Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
03	03 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
04	04 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
05	05 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
06	06 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
07	07 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
08	08 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
09	09 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean
10	10 - Oregon Seafloor Hydrography	2012-01-01 22:00:00	Hydrography	Hydrography	North Pacific Ocean

DASHBOARD
RECENT IMAGES

- Clitter
- Gorgonian Coral
- Acoustic Release
- Seafloor COT
- Marine Capital
- Surface Buoy

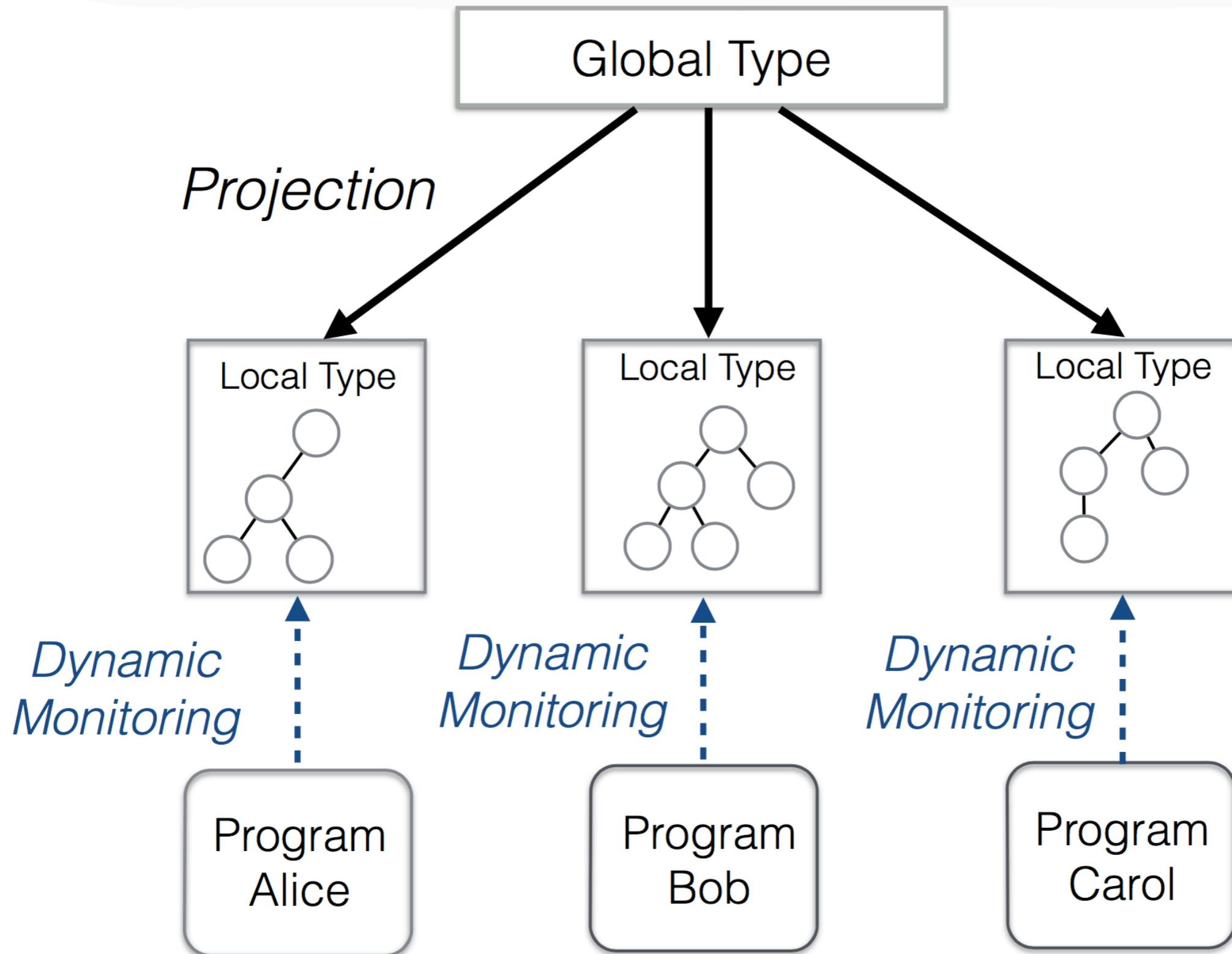
UPCOMING EVENTS

- Oregon Coast Wave Hinge
- Water Surface Elevation

- **TCS'16:** Monitoring Networks through Multiparty Session Types. Laura Bocchi , Tzu-Chun Chen , Romain Demangeon , Kohei Honda , Nobuko Yoshida
- **LMCS'16:** Multiparty Session Actors. Rumyana Neykova, Nobuko Yoshida
- **FMSD'15:** Practical interruptible conversations: Distributed dynamic verification with multiparty session types and Python. Romain Demangeon , Kohei Honda , Raymond Hu , Rumyana Neykova , Nobuko Yoshida
- **TGC'13:** The Scribble Protocol Language. Nobuko Yoshida , Raymond Hu , Rumyana Neykova , Nicholas Ng

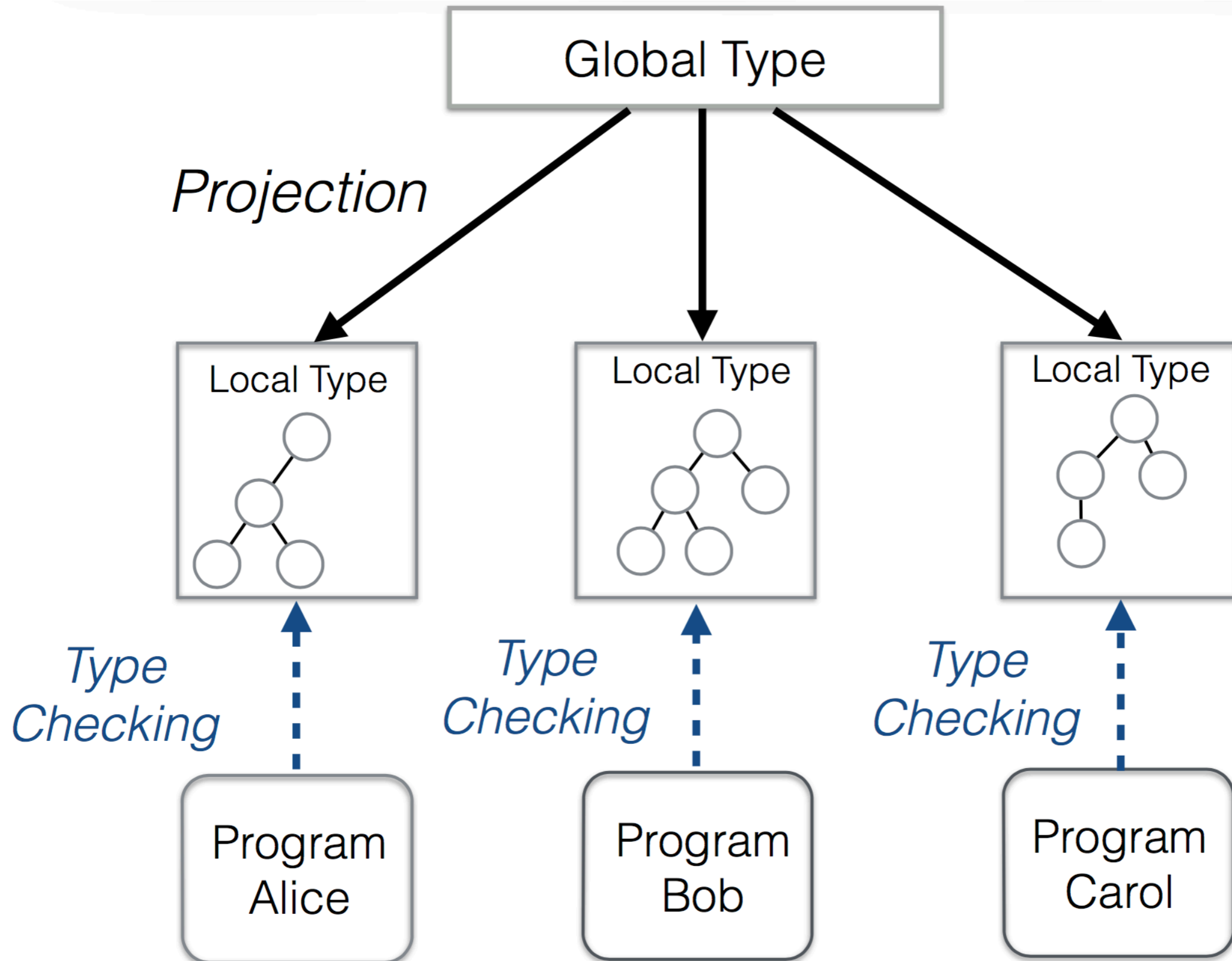
Dynamic Monitoring

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



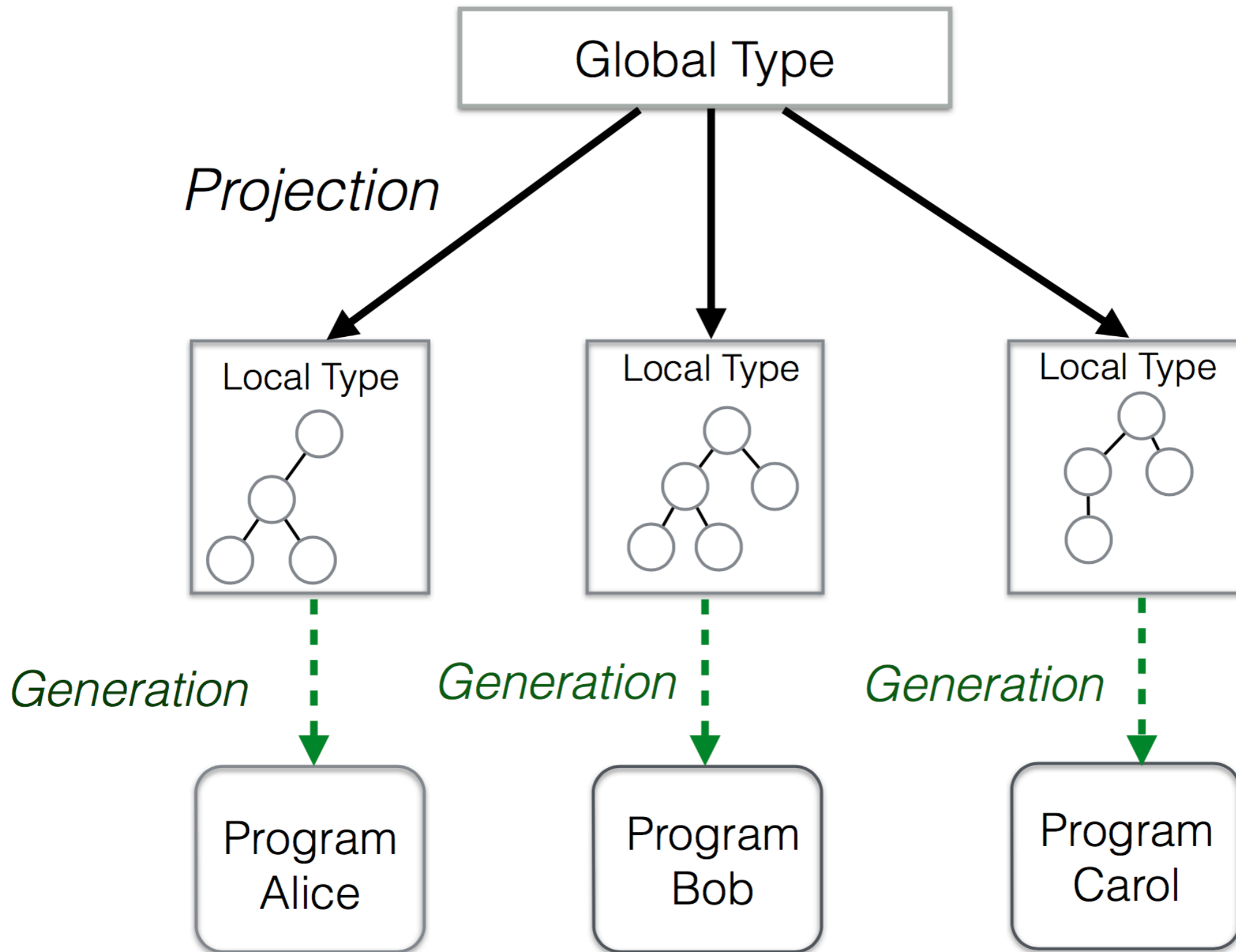
Type Checking

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



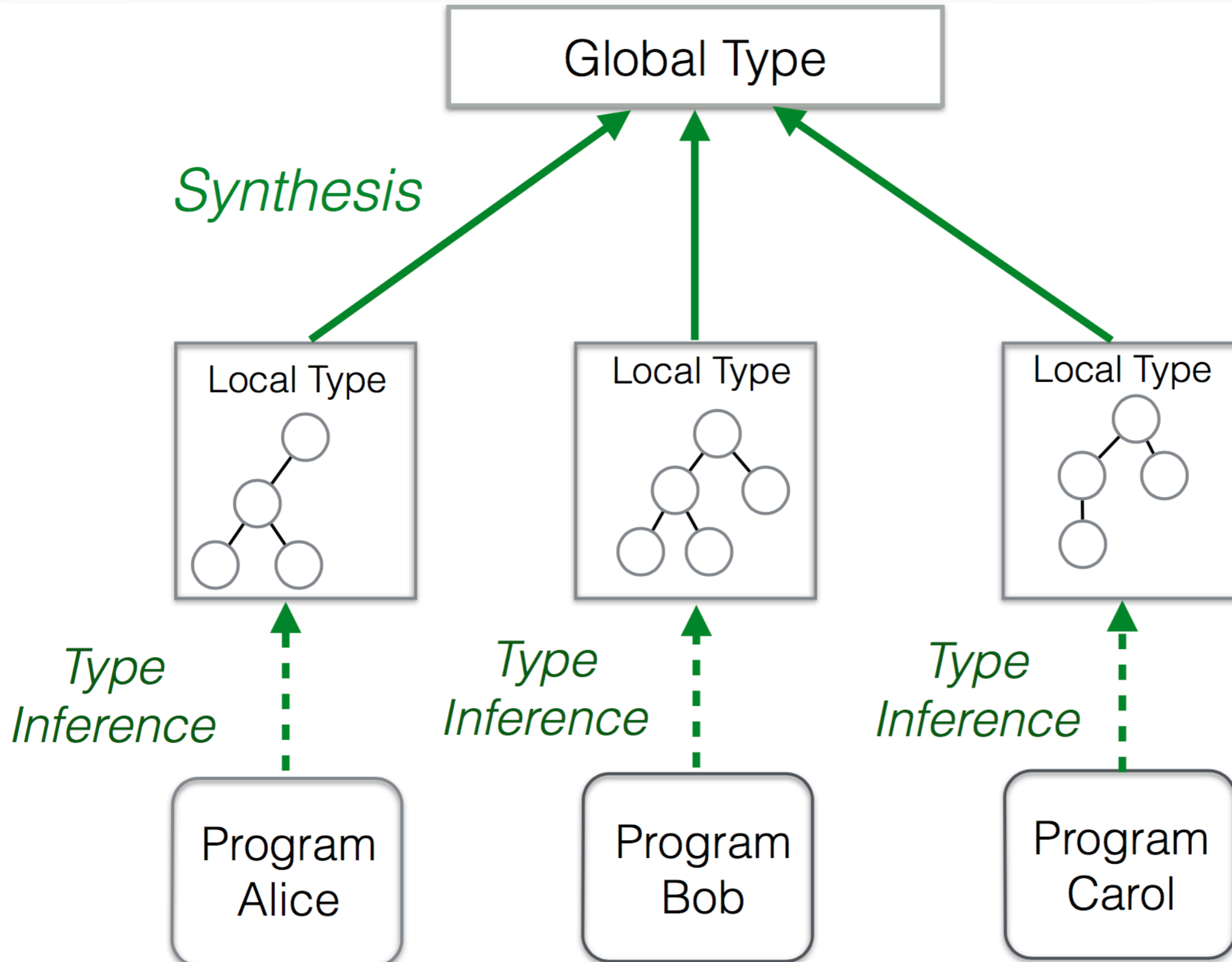
Code Generation

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



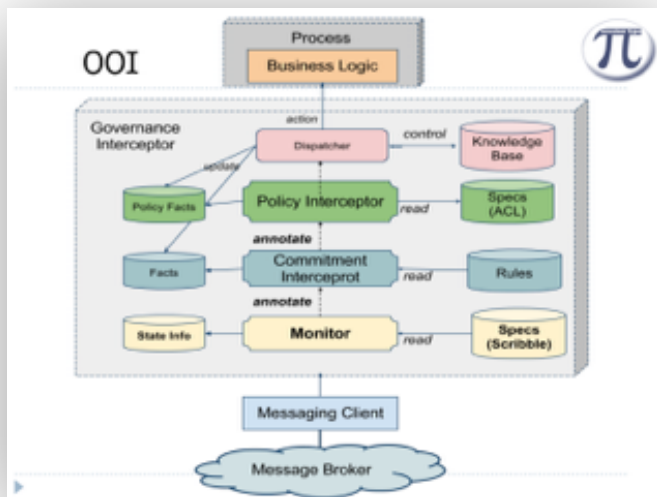
Synthesis

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

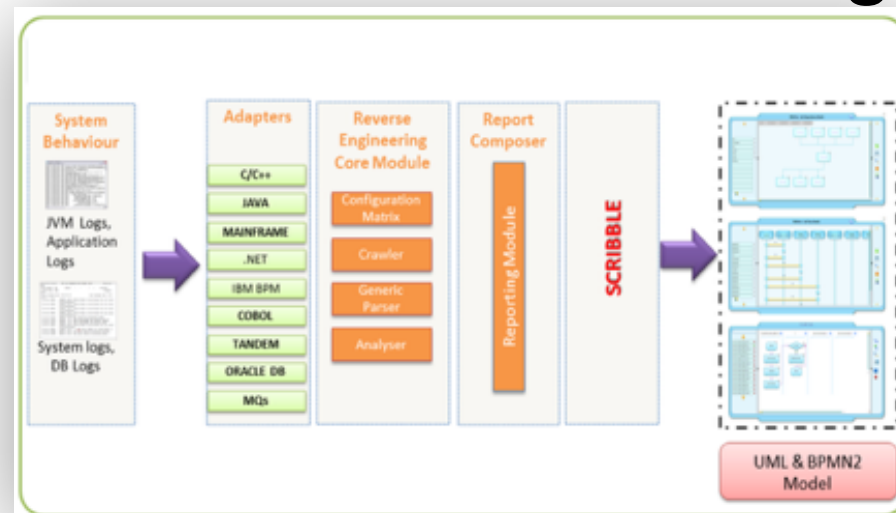


Session Type based Tools

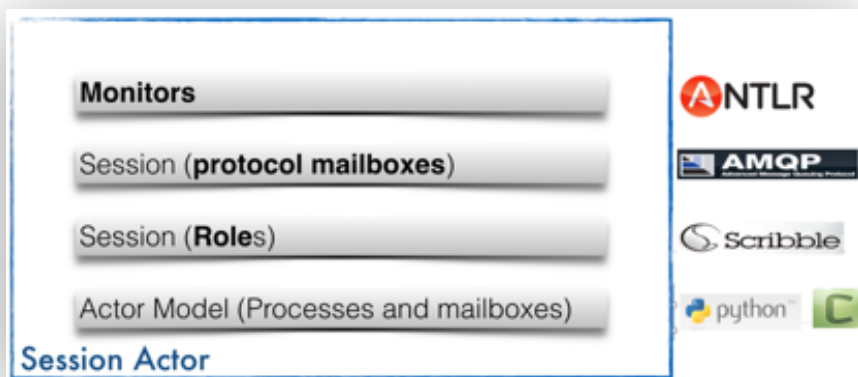
OOI Governance



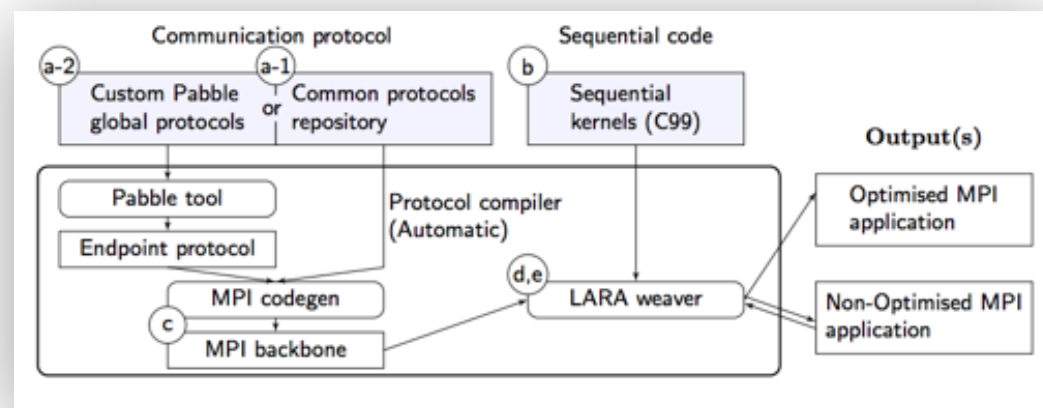
ZDLC: Process Modeling



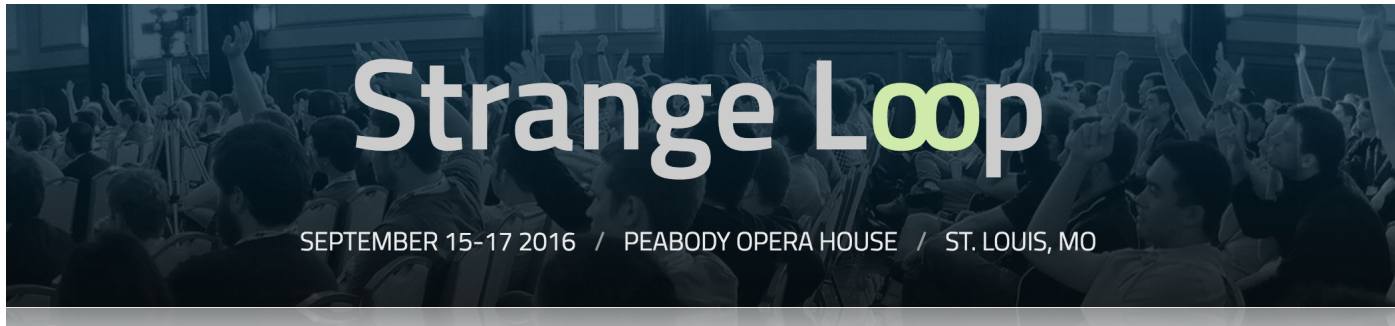
Actor Verification



MPI code generations



Interactions with Industries



Nobuko Yoshida
Imperial College, London

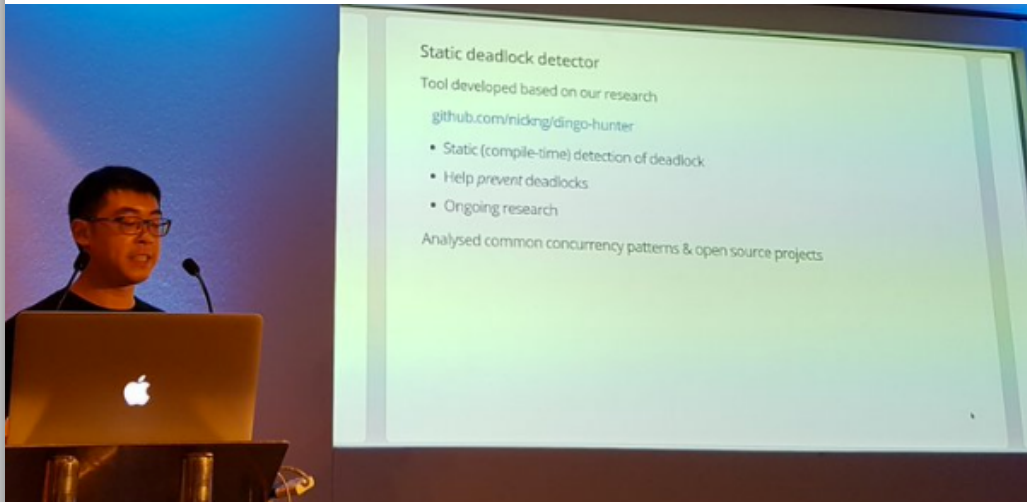


Adam Bowen @adamnbowen · Sep 15

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.

DoC researcher to speak at Golang UK conference

by Vicky Kapogianni
20 July 2016



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



The Golang UK Conference

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)

Java API Generation [FASE'16]

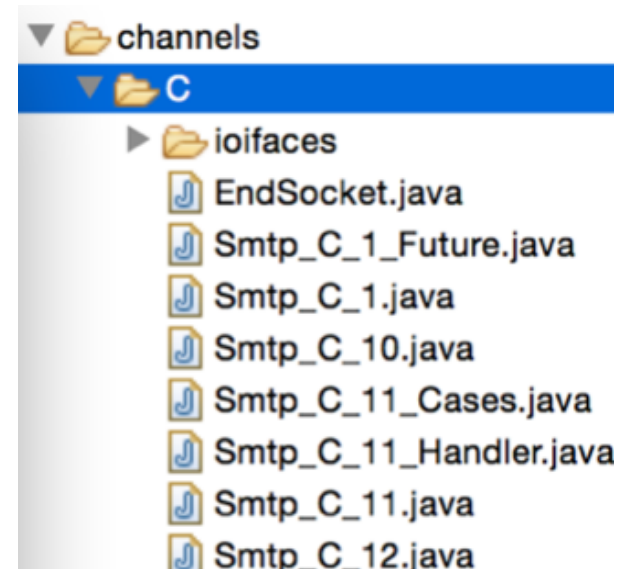


RFC 821 August 1982
Simple Mail Transfer Protocol

TABLE OF CONTENTS

1.	INTRODUCTION	14
2.	THE SMTP MODEL	14
3.	THE SMTP PROCEDURE	14
3.1.	Mail	14
3.2.	Forwarding	14
3.3.	Verifying and Expanding	14
3.4.	Sending and Mailing	14
3.5.	Opening and Closing	14
3.6.	Relaying	14
3.7.	Domains	14
3.8.	Changing Roles	14
4.	THE SMTP SPECIFICATIONS	14
4.1.	SMTP Commands	14
4.1.1.	Command Semantics	14
4.1.2.	Command Syntax	14
4.2.	SMTP Replies	14
4.2.1.	Reply Codes by Function Group	14
4.2.2.	Reply Codes in Numeric Order	14
4.3.	Sequencing of Commands and Replies	14
4.4.	State Diagrams	14
4.5.	Details	14
4.5.1.	Minimum Implementation	14
4.5.2.	Transparency	14
4.5.3.	Sizes	14

□



```
.send(Smtplib.S, new DataLine("Session  
.send(Smtplib.S, new EndOfData())  
.receive(Smtplib.S, Smtplib._250, new Buf  
.S  
● send(S role, Mail m) : Smtplib_C_11 - Smtplib_C_10  
● send(S role, Quit m) : EndSocket - Smtplib_C_10
```

Scribble – Proving a distributed design



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

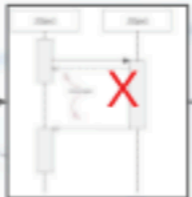
2. XMI is converted into OpenTracing format for consumption by managed service



7. Generate exception report and send back to DCC



OPENTRACING



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