Multiparty Session Types and their Applications

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

Nobuko Yoshida, Rumyana Neykova
Imperial College London

The Kohei Honda Prize for Distributed Systems Queen Mary, University of London

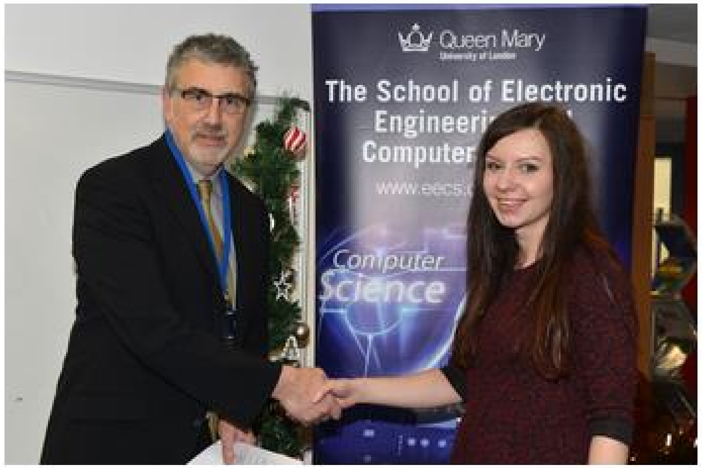
Posted with permission from QMUL on 17th Dec 2013. Original article written by Edmund Robinson.

This prize was instituted in 2013 and is awarded annually to one undergraduate student and one postgraduate student in recognition of their achievement in applying the highest quality scientific and engineering principles in the broad area of Distributed Systems. This is the area in which Dr Honda concentrated most of his teaching, and it is also the area in which he conducted his research. Its primary funding comes from a donation from his family, who wished to commemorate Dr Honda in this way. Additional funding has come from Dr Honda's own ETAPS Award. This prize is sponsored by Springer Verlag, and awarded annually by the ETAPS committee in recognition of an individual's research contribution. Dr Honda received the first such award posthumously, and the awarding panel expressed a wish that the funding be used to supplement this prize fund. The laudation for this award, written by Dr Honda's colleague, Prof Vladimiro Sassone is included later.

About Dr Honda

Kohei Honda was born and lived the first part of his life in Japan. Like many scientists he was fascinated by the idea of finding basic explanatory theories, like the physicists looking for grand unified theories of the universe. Kohei, though, was passionately interested in finding the right basic explanatory theory for the process of computation. Most academics agree that the basic theory

Winners 2013



Ms Anna Pawlicka
2013 winner (Undergraduate) source: QMUL

Mr. Valdmir Negacevshi
2013 winner (Postgraduate) source: QMUL

Outline

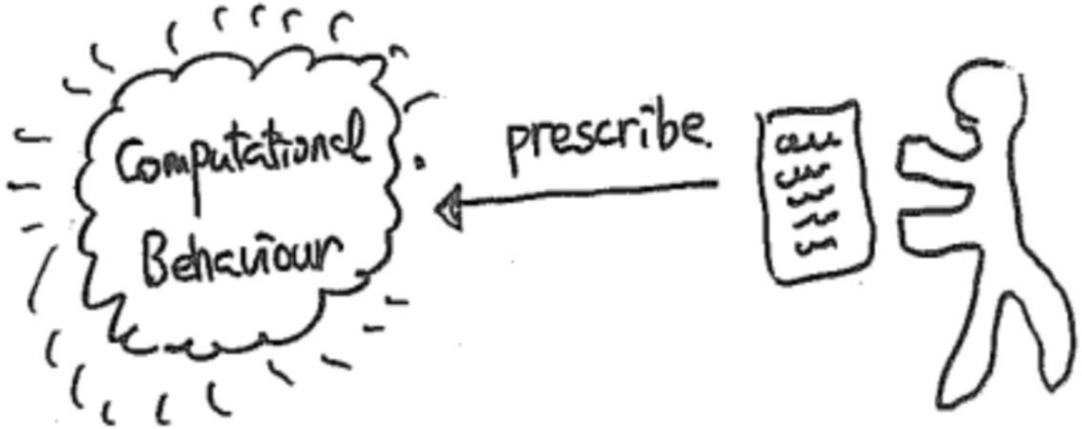
- Idioms for Interaction
- Multiparty Session Types
- Scribble and Applications to a Large-scale Cyberinfrastructure
- Recent Results on Multiparty Session Types

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, or 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        indices
data  _j,4
data  _l,4        left/right bounds
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      Initialization
mov $11,_stkr
                              Top Loop.
L1:  mov _s,rax
     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
                              Second Loop.
L2:  mov _l,rcx               i:=l.
     mov rcx,_i
     mov _r,rcx               j:=r.
     mov rcx,_j
     mov _l,rdx               rdx:=l+r
     add _r,rdx
     mov rdx,rax
     mov rax,rdx
     shr $31,rdx              rdx:=(l+r)/2
     add rdx,rax
     mov rax,rdx
     sar $1,rdx
     mov _a(,rdx,4),rcx       x:=a(rdx).
     mov rcx,_x
                              third Loop
L3:  mov _i,rax               rdx:=a(i).
     mov _a(,rax,4),rdx
     cmp rdx,_x
     jle L4                   if rdx≥x goto L4
     inc _i                   else i=i+1
     jmp L3                   goto L3. (loop).

L4:  mov _j,rax               rdx=a(j).
     mov _a(,rax,4),rdx
     cmpl rdx,_x              if rdx≤x goto L5
     jge L5                   else j:=j-1 and
     dec _j                   goto L4 (loop).
     jmp L4

L5:  mov _i,rax               if i>j goto L6.
     cmpl rax,_j
     jl L6
     mov _i,rax               w=a(i)
     mov _a(,rax,4),rcx
     mov rcx,_w
     mov _i,rax               a(i)=a(j).
     mov _j,rdx
     mov _a(,rdx,4),rcx
     mov rcx,_a(,rax,4)
     mov _j,rax               a(j)=w.
     mov _w,rcx
     mov rcx,_a(,rax,4)
     inc _i
     dec _j

L6:  mov _i,rax               if i≤j loop at L3.
     cmp rax,_j               else next.
     jl L7
     jmp L3

L7:  mov _i,rax               if i≥r then L8
     cmp rax,_r
     jle L8                   else s:=s+1
     inc _s
     mov _s,rax               stkl(s)=i;
     mov _i,rcx
     mov rcx,_stkl(,rax,4)
     mov _s,rax               stkr(s)=r.
     mov _r,rcx
     mov rcx,_stkr(,rax,4)

L8:
     mov _l,rax               if l≥r then
     cmp rax,_r               next else
     jle L9                   to the second loop.
     jmp L2

L9:  cmp $0,_s                if s=0 end
     jle L10                  else top loop.
     jmp L1

L10: ret
```

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    Partition into
     while  a[j]>x do  j:= j+1 end.   two parts.
     if  i≤j  then  swap(i,j);  i:=i+1; j:=j-1; end
   until  i>j;
    if  l<j  then  sort(l, j);        Recursively
    if  l<r  then  sort(i, r);        sort two parts.
  end

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

Quicksort in pure lambda:

$((\lambda xy.y(xxy))(\lambda xy.y(xxy)))\lambda q.\lambda l.$

$((\lambda x.x(\lambda xy.x))l)(\lambda x.x)$ } if l is nil then nil.

$(((\lambda xy.y(xxy))(\lambda xy.y(xxy))(\lambda c.\lambda xy.x((\lambda x.x(\lambda xy.x))x)y$

$((\lambda xy.\lambda z.z(\lambda xy.y)xy)((\lambda x.x(\lambda xyz.y))x)(c((\lambda x.x(\lambda xyz.z))x)y$ } concat.

$(q(\lambda xy.y(xxy))(\lambda xy.y(xxy))(\lambda f.\lambda px.((\lambda x.x(\lambda xy.x))x)(\lambda x.x)$ } sort and Filter

$(p((\lambda x.x(\lambda xyz.y))x))((\lambda xy.\lambda z.z(\lambda xy.y)xy)x$

$(f((\lambda x.x(\lambda xyz.z))x)))(f((\lambda x.x(\lambda xyz.z))x)))$

$(\lambda y.(((\lambda xy.y(xxy))(\lambda xy.y(xxy)))\lambda f'.\lambda xy.((\lambda x.x\lambda xy.x)y)$ } (λy. LTy. Ca…

$(\lambda xy.y)(((\lambda x.x\lambda xy.x)x)(\lambda xy.y)(f'((\lambda x.x\lambda xy.y)x)((\lambda x.x\lambda xy.y$

$y((\lambda x.x(\lambda xyz.y))l))((\lambda x.x(\lambda xyz.z))l))$ } cdr l

$((\lambda xy.\lambda z.z(\lambda xy.y)xy)((\lambda x.x(\lambda xyz.y))l)$ } Cons (Car l)

$(q((\lambda xy.y(xxy))(\lambda xy.y(xxy))(\lambda f.\lambda px.((\lambda x.x(\lambda xy.x))x)$ } sort and Filter

$(\lambda x.x)(p((\lambda x.x(\lambda xyz.y))x))((\lambda xy.\lambda z.z(\lambda xy.y)xy)x$

$(f((\lambda x.x(\lambda xyz.z))x)))(f((\lambda x.x(\lambda xyz.z))x)))$

$(\lambda y.((\lambda xy.y(xxy))(\lambda xy.y(xxy)))\lambda f''.\lambda xy.((\lambda x.x\lambda xy.x)x)$ } λy. Me y Car l

$(((\lambda x.x\lambda xy.x)y)(\lambda xy.x)(\lambda xy.y))$

$(((\lambda x.x\lambda xy.x)y)(\lambda xy.x)(f''((\lambda x.x\lambda xy.y)x)$

$((\lambda x.x\lambda xy.y)y)))y((\lambda x.x(\lambda xyz.y))l))((\lambda x.x(\lambda xyz.z))l))))).$ } Cdr l

Quicksort with combinators:

Y(λf. λl.

(Isnil l)

(Concat (f Filter (λy. LT y (Car l))

(Cdr l))

(Cons (Car l)

(f Filter (λy. Me y

(Car l)

(Cdr l)))))

$I = \lambda x.x$ $T = \lambda xy.x$ $F = \lambda xy.y$ $Y = (\lambda xy.y(xxy))(\lambda xy.y(xxy))$

Cons $= \lambda xy.\lambda z.zFxy$ Isnil $= \lambda x.xT$

Car $= \lambda x.x(\lambda xyz.y)$ Cdr $= \lambda x.x(\lambda xyz.z)$

Concat = Y (λc. λxy. x (Isnil x) y (Cons (Car x)(c (Cdr x) y)

Filter = Y(λf. λpx. (Isnil x) I (p (Car x)) (Cons x (Filter (Cdr x)))

(Filter (Cdr x))

Iszero = λx. xT   Pred = λx. xF

LT = Y (λf. λxy. (Iszero y) F ((Iszero x) T (f (Pred x)(Pred y

ME = Y (λf. λxy. (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

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

$\pi \quad P ::= \Sigma \pi_i.P_i \mid P|Q \mid (\nu x)P \mid !P \mid 0.$

with $\pi ::= x(\tilde{y}) \mid \bar{x}\langle \tilde{y} \rangle.$

λ in π

$[\![x]\!]_u \stackrel{def}{=} \bar{x}\langle u \rangle.$

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

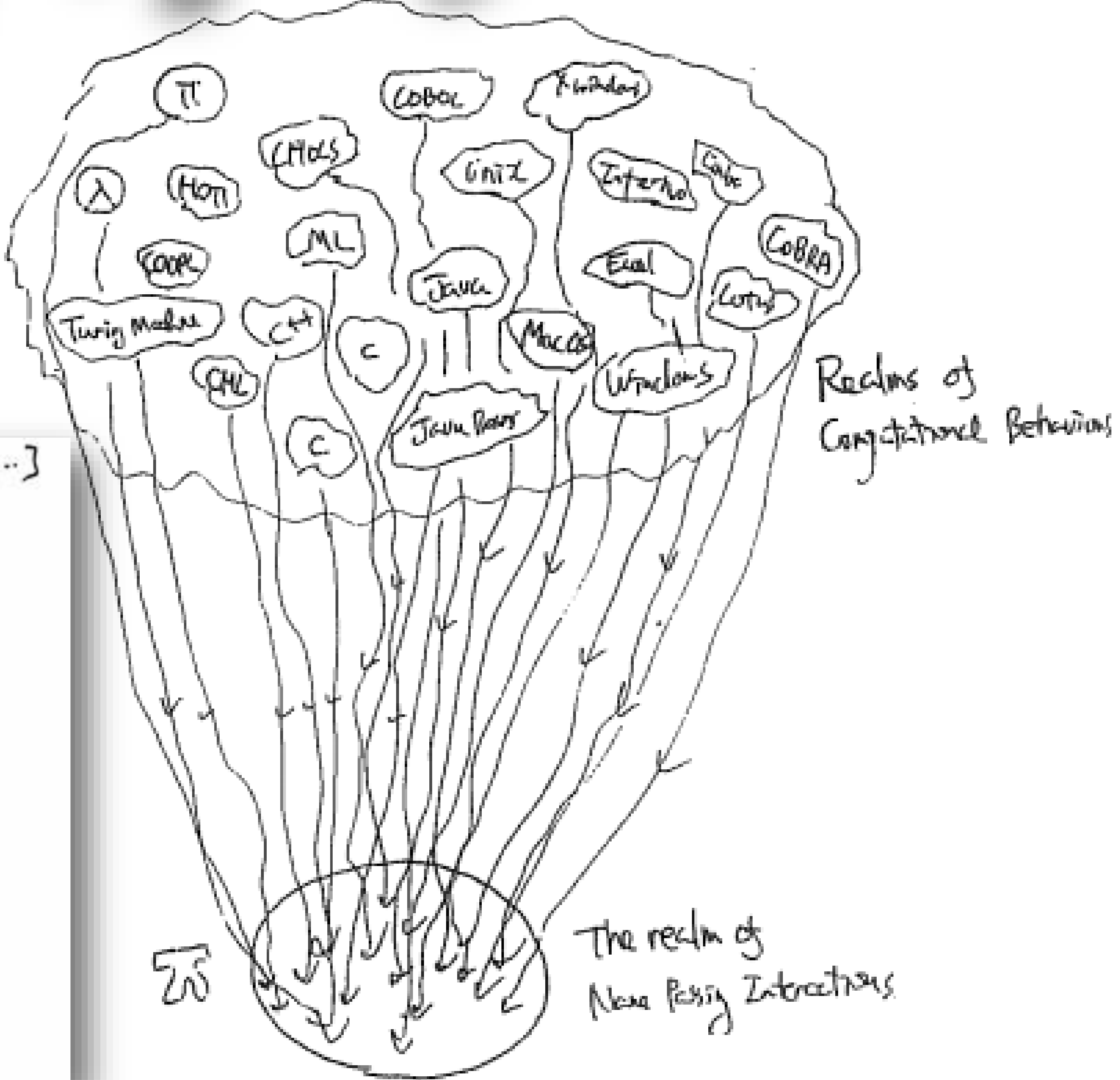
$[\![MN]\!]_u \stackrel{def}{=} (\nu f x)\left([\![M]\!]_f \mid \bar{f}\langle x u \rangle \mid [\![x = N]\!]\right)$

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

* 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 [Bellin and Scott 93]
- Arbitrary 'constant' interaction [HY94]
- strategies on Games [HO95]

⋮

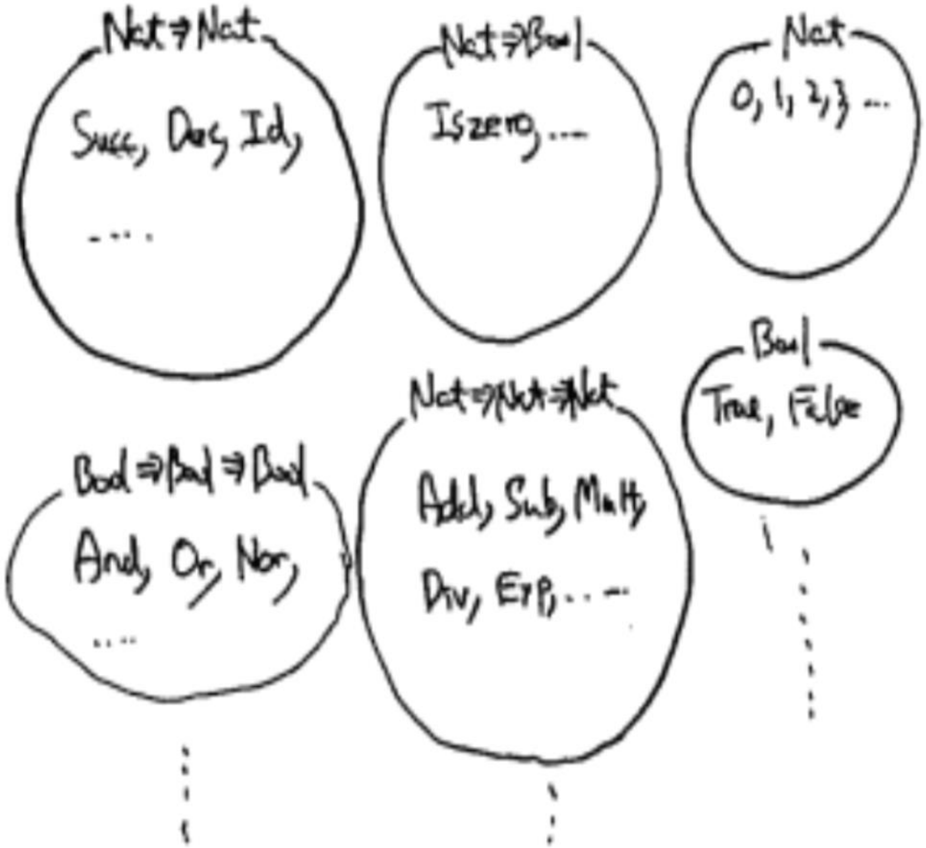


The Role of Types in π-calculus

- (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:

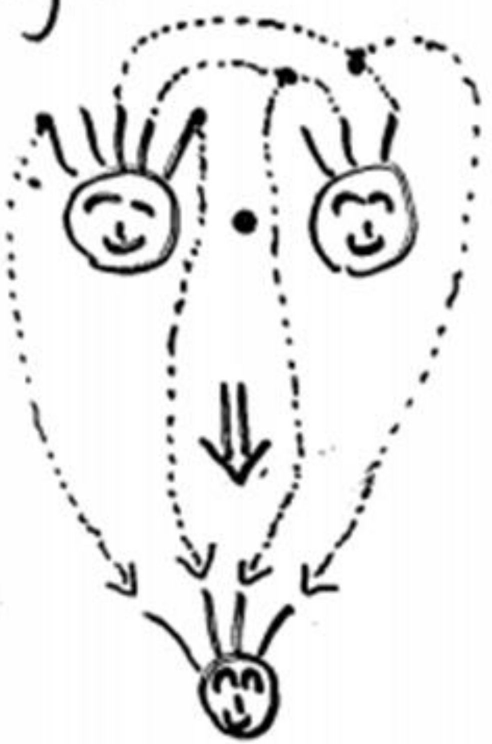
function application.

$$\begin{cases} f:\alpha\Rightarrow\beta \bullet e:\alpha = f \bullet e : \beta. \\ \text{else undefined.} \end{cases}$$

Process Types

• When it comes to processes,

composition becomes:



e.g.

3 + 5

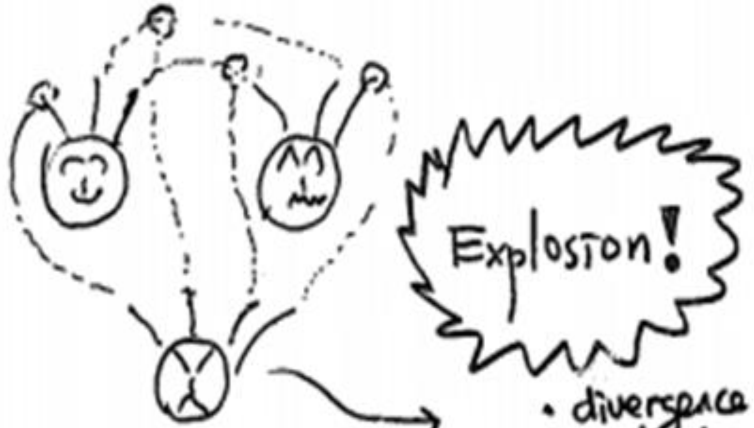
⇓

8

$P \bullet_\delta \delta \longmapsto_\tau \sigma$

• But some composition is

dangerous!

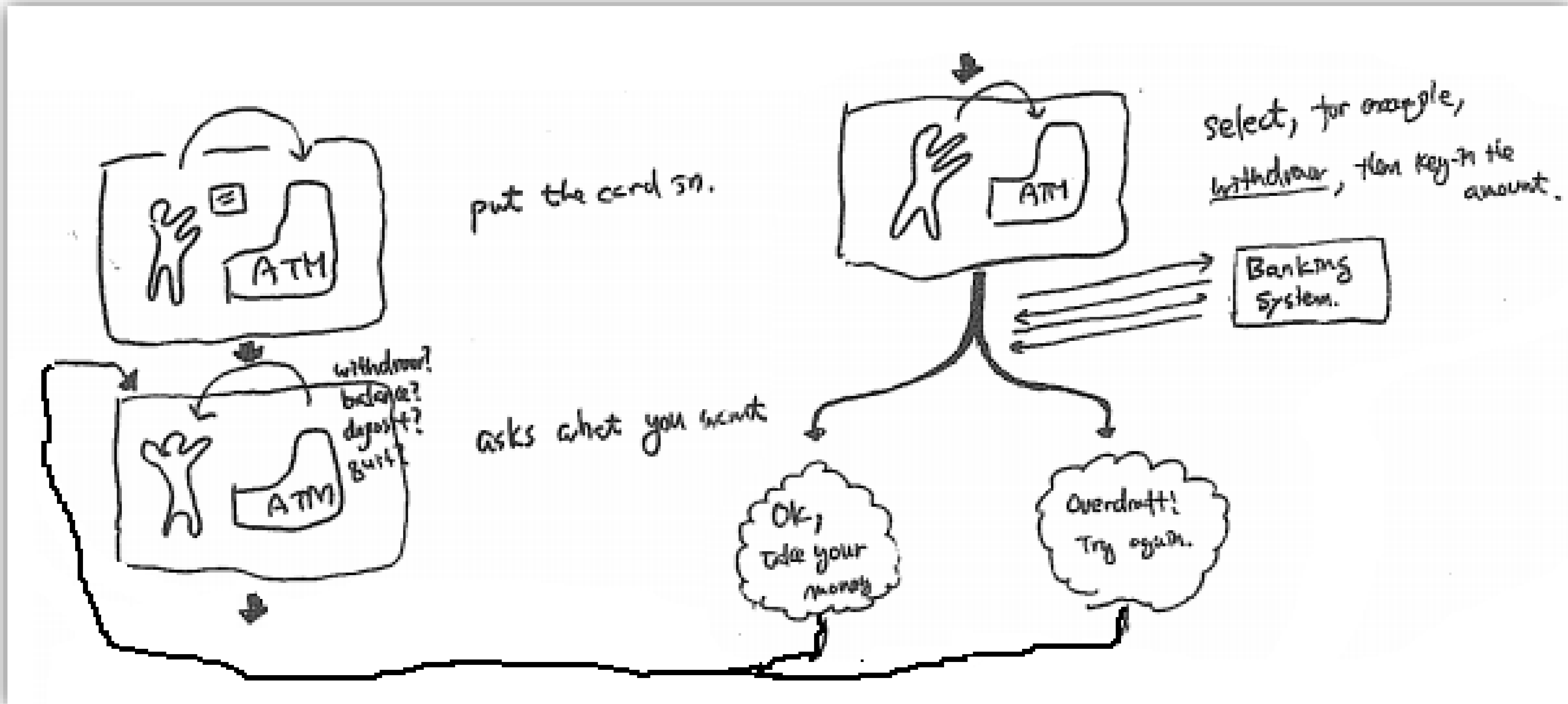


• divergence
• deadlock
• run-time error
⋮

• Therefore we type processes,

The connection is prohibited

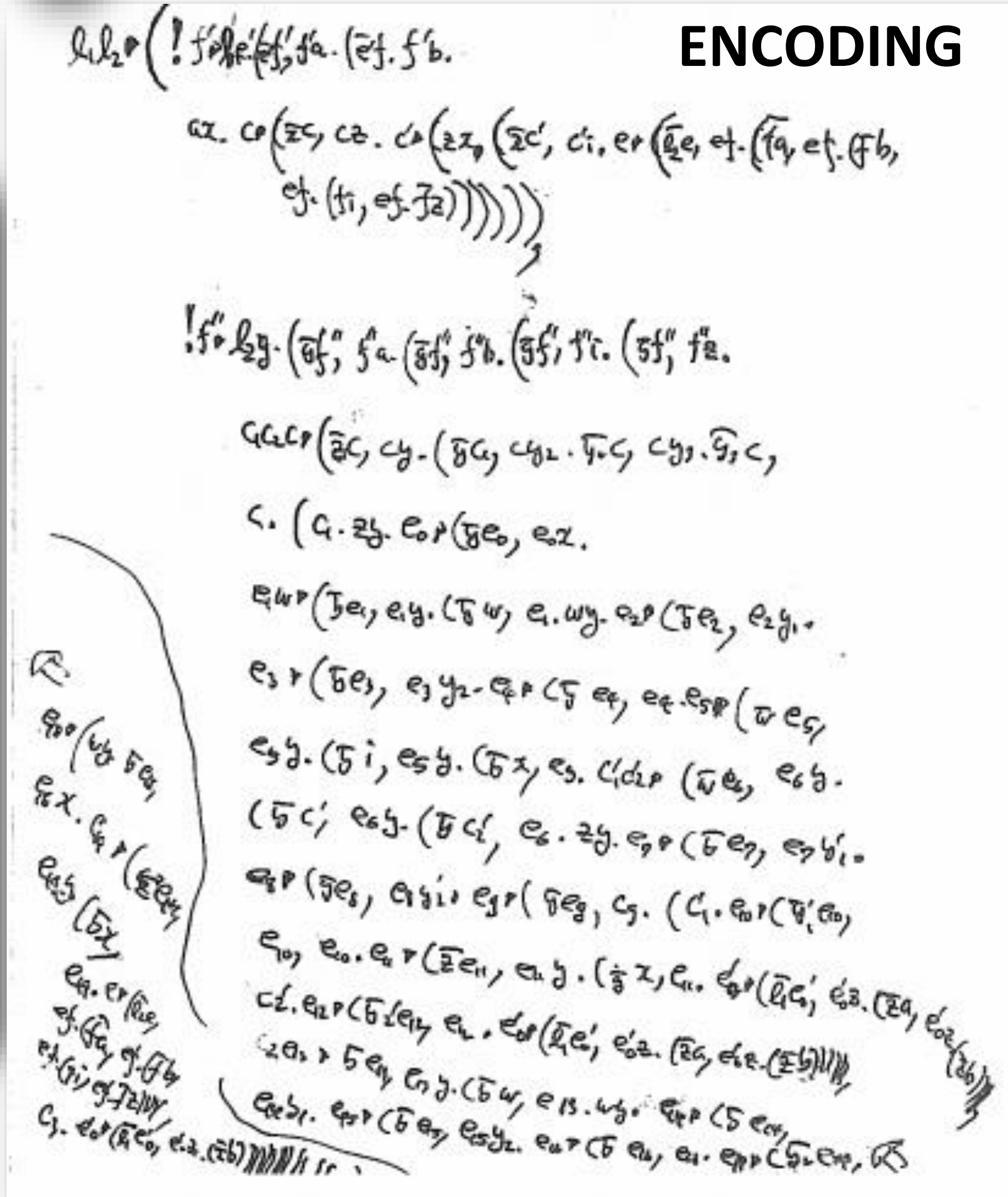
Implementing ATM



Implementing ATM

ATM(cb) $\stackrel{\text{def}}{=}$

a⟨z⟩:: z?ID; [?wd; ?X;   ∦ ↔ user.
    b⟨w⟩:: w!wt; !ID; !X;   ∦ ↔ bank.
        [?ok;   ∦ :
            z!ok; !X; ATM(cb),   ∦ ↔ user
         ?overdraft: ∦ ↔ bank
            z!over; ATM(cb)],   ∦ ↔ user.
 ?bal;   ∦ ↔ user
    b⟨w⟩:: w!bal; ?X;   ∦ ↔ bank
        z!X; ATM(cb)]



Current: Communication is Ubiquitous

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

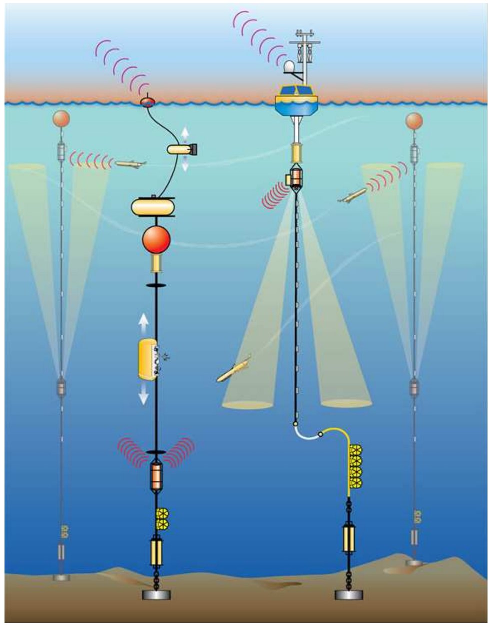
Current: Communication is Ubiquitous

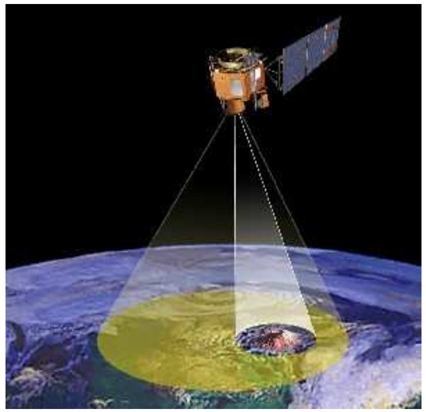
- The way to organise software is increasingly based on communications (Cloud Computing, many cores, message-passing parallel computation, ...)
- Question $\Longrightarrow$ Multiparty session type theory
  - How to formally abstract/specify/implement/control communications?
  - How to apply mobile processes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

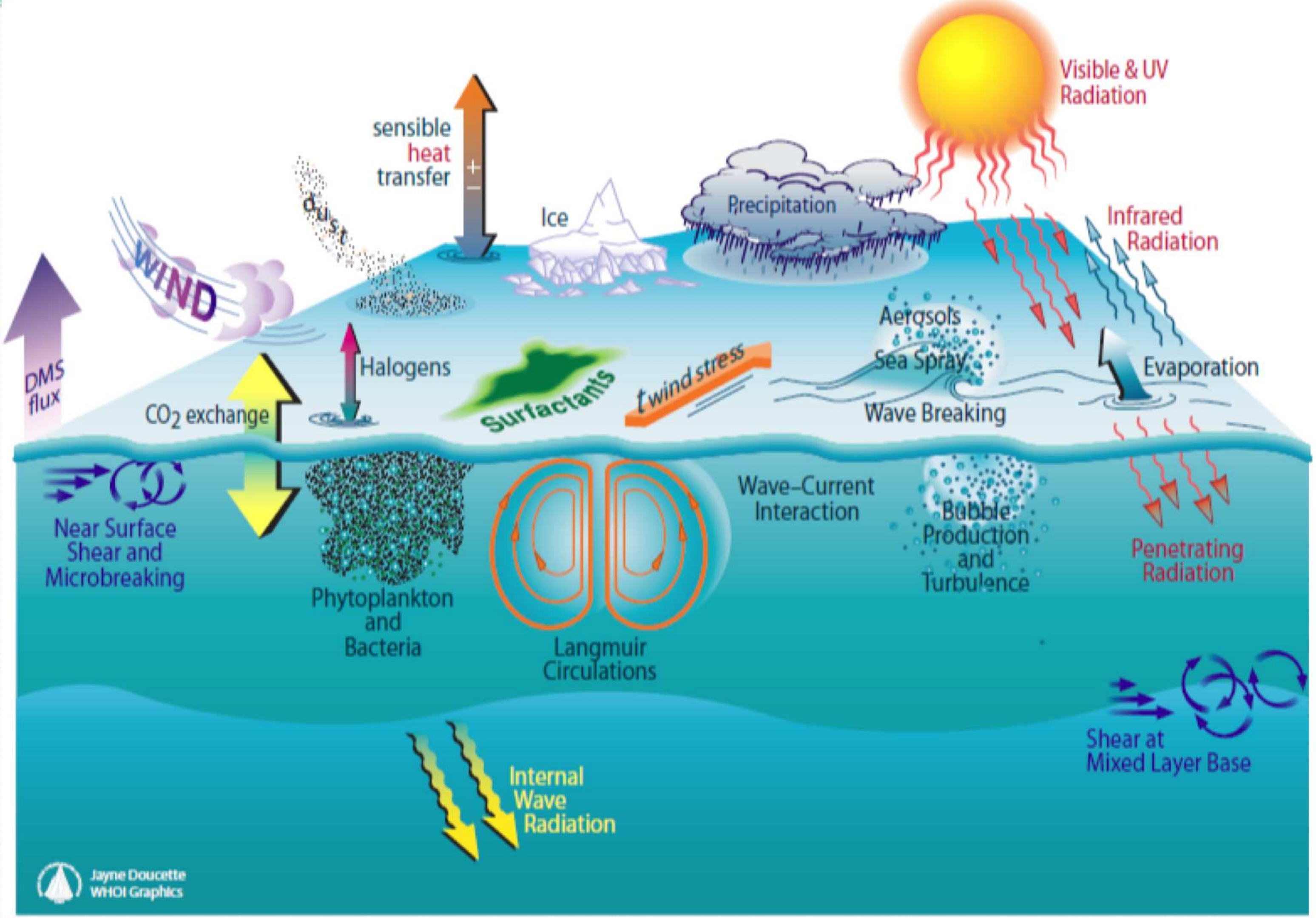
    $\Longrightarrow$ large-scale cyberinfrastructure for e-Science

Ocean Observatories Initiative

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

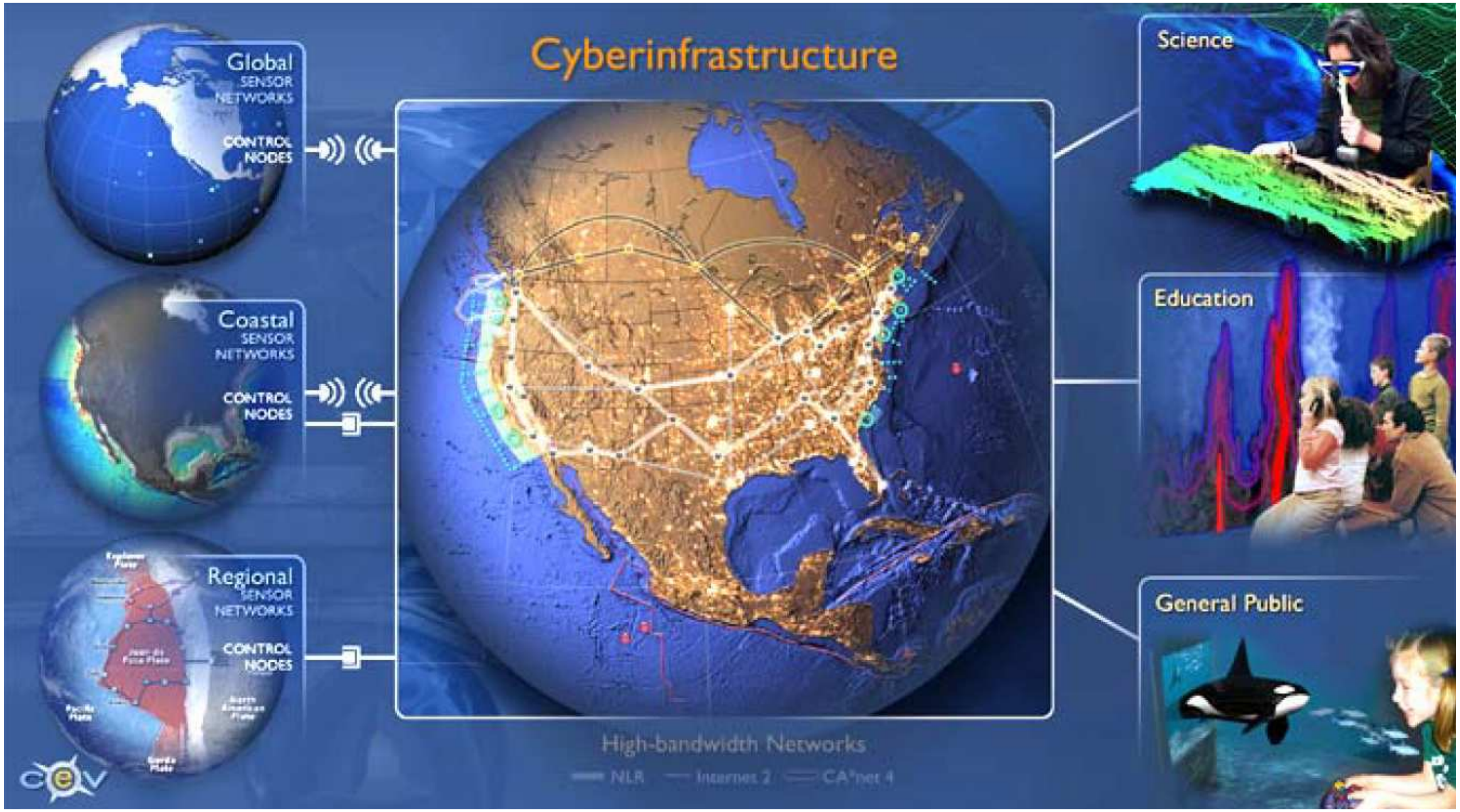


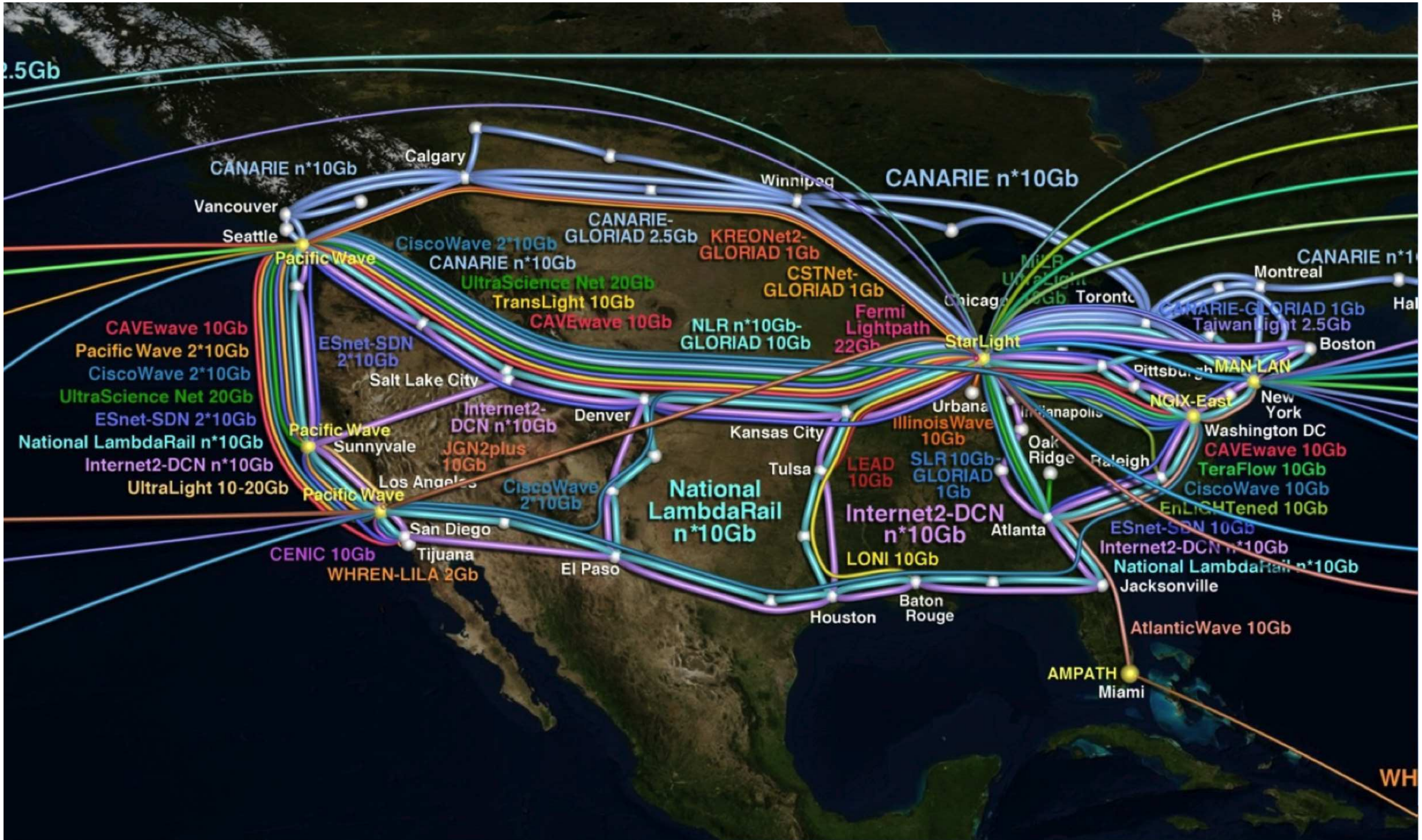




Visible & UV
Radiation
sensible
heat
transfer
dust
Ice
Precipitation
Infrared
Radiation
WIND
DMS
flux
Halogens
Aerosols
Sea Spray
Evaporation
Surfactants
wind stress
Wave Breaking
CO2 exchange
Near Surface
Shear and
Microbreaking
Phytoplankton
and
Bacteria
Langmuir
Circulations
Wave–Current
Interaction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

Ocean Observatories Initiative





Ocean Observatories Initiative

Challenges

- The need to specify, catalogue, program, implement and manage *multiparty message passing protocols*.
- Communication assurance
  - Correct message ordering and synchronisation
  - Deadlock-freedom, progress and liveness
  - Dynamic message monitoring and recovery
  - Logical constraints on message values
- Shared and used over a long-term period (e.g. 30 years in OOI).

Why Multiparty Session Types?

- Robin Milner (2002): *Types are the leaven of computer programming; they make it digestible*.

  $\Longrightarrow$ 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 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]

⇓

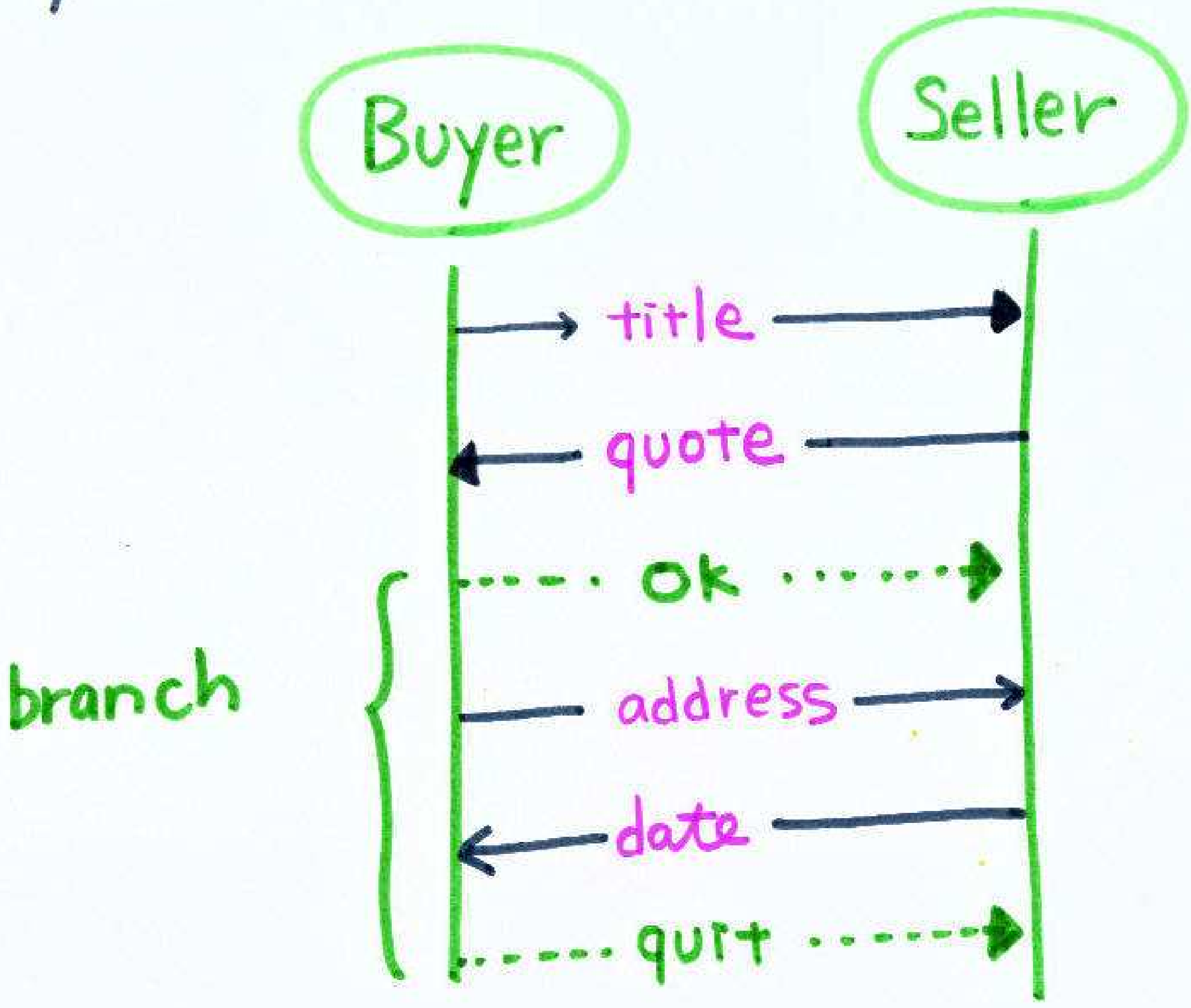
Scribble at π4 Technology

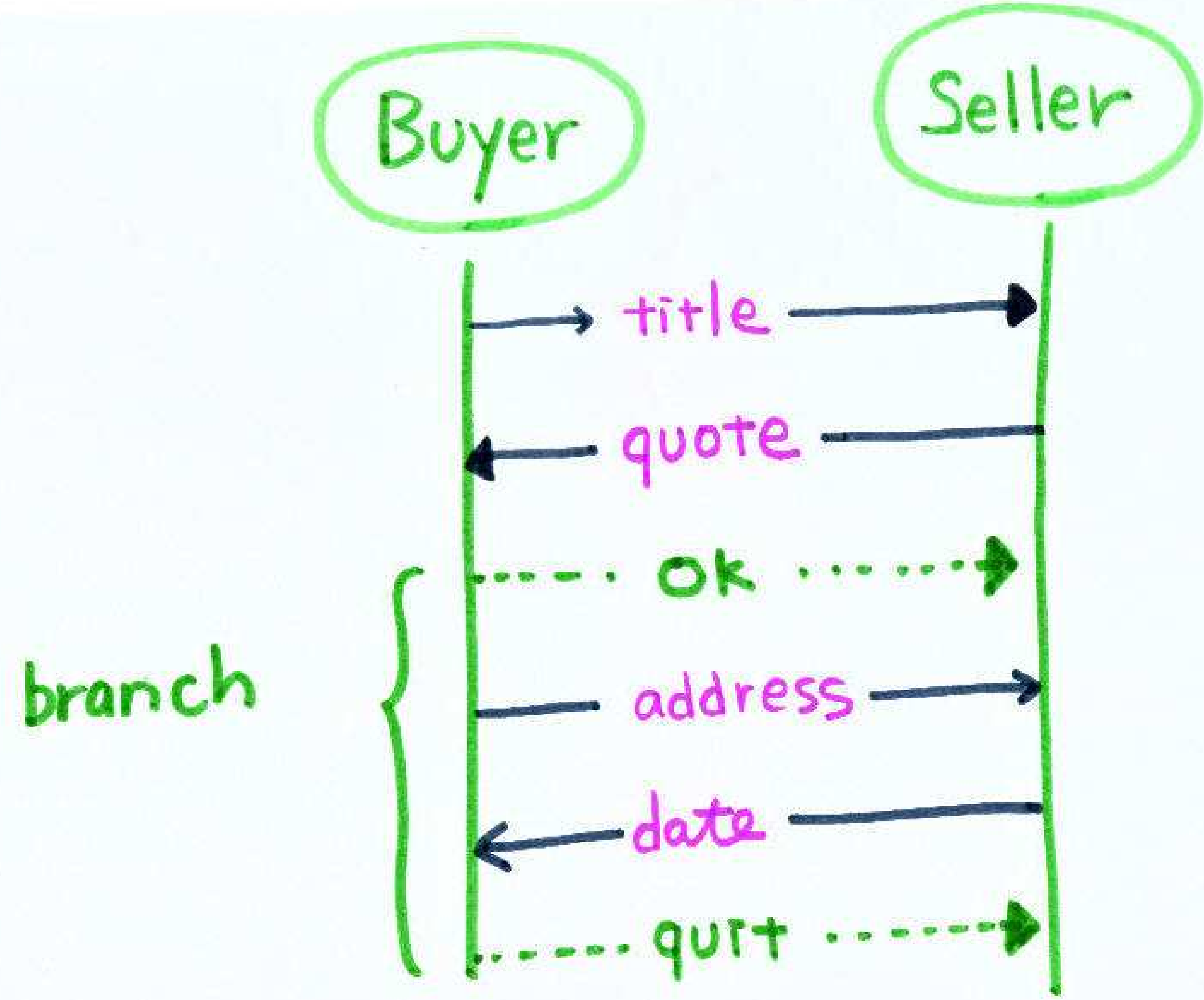
⇓

Multiparty Session Types [POPL'08]

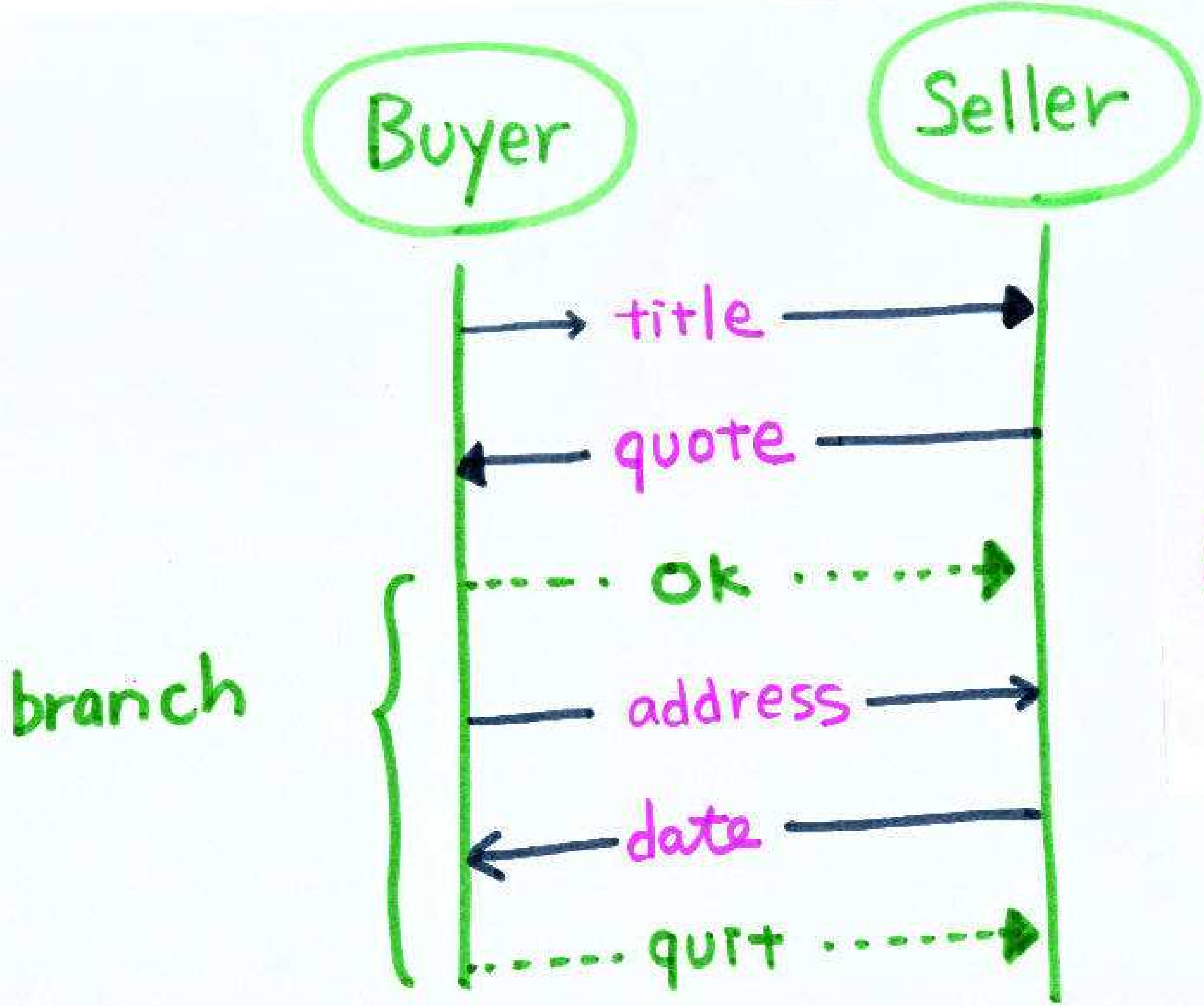
⇓

Binary Session Types : Buyer - Seller Protocol





$$!\,\mathrm{String}\,;\,?\,\mathrm{Int}\,;\,\oplus\{\mathrm{ok}:\,!\mathrm{String}\,;\,?\mathrm{Date}\,;\mathrm{end},\ \mathrm{quit}:\mathrm{end}\}$$



P has T

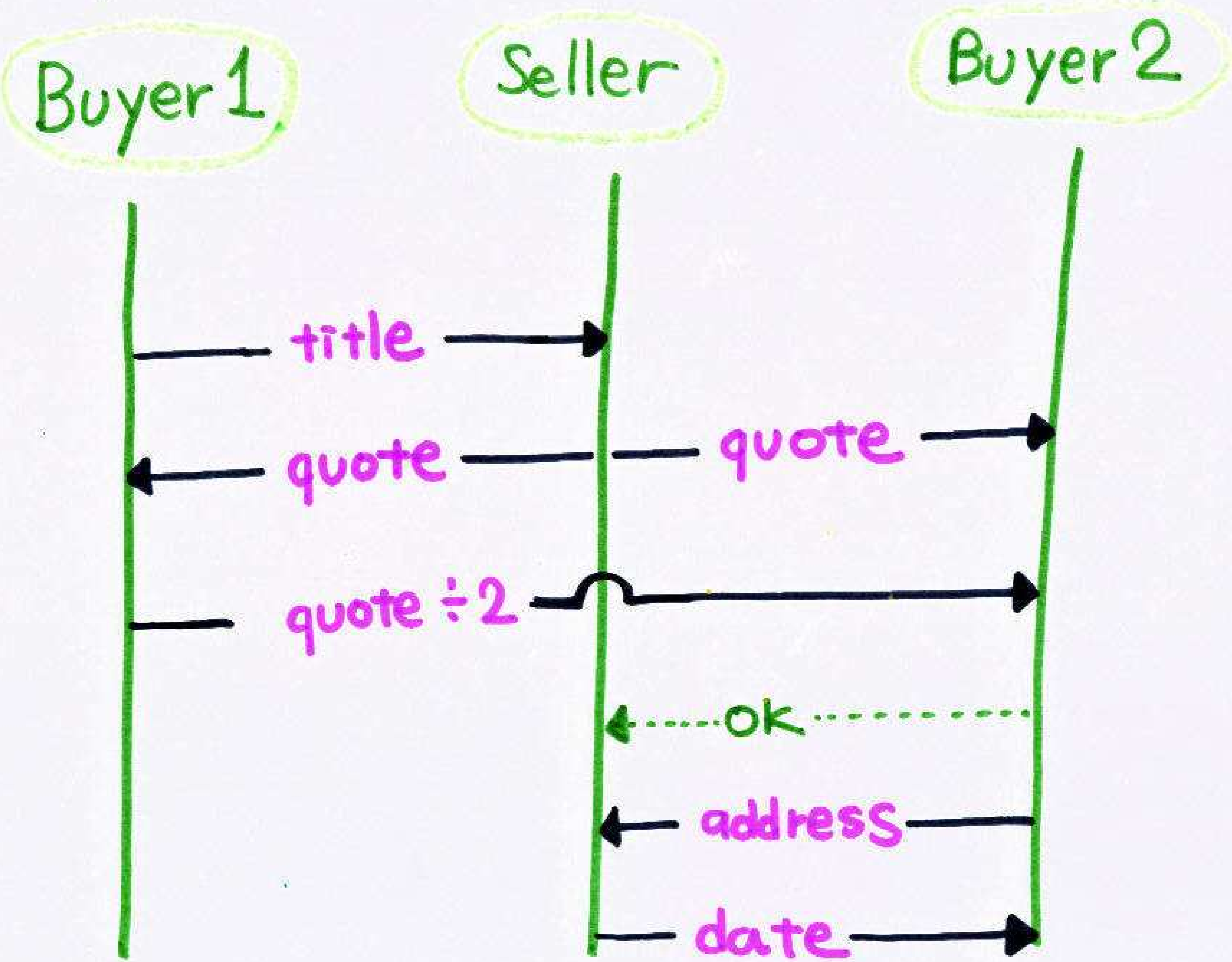
Q has $\overline{T}$ dual

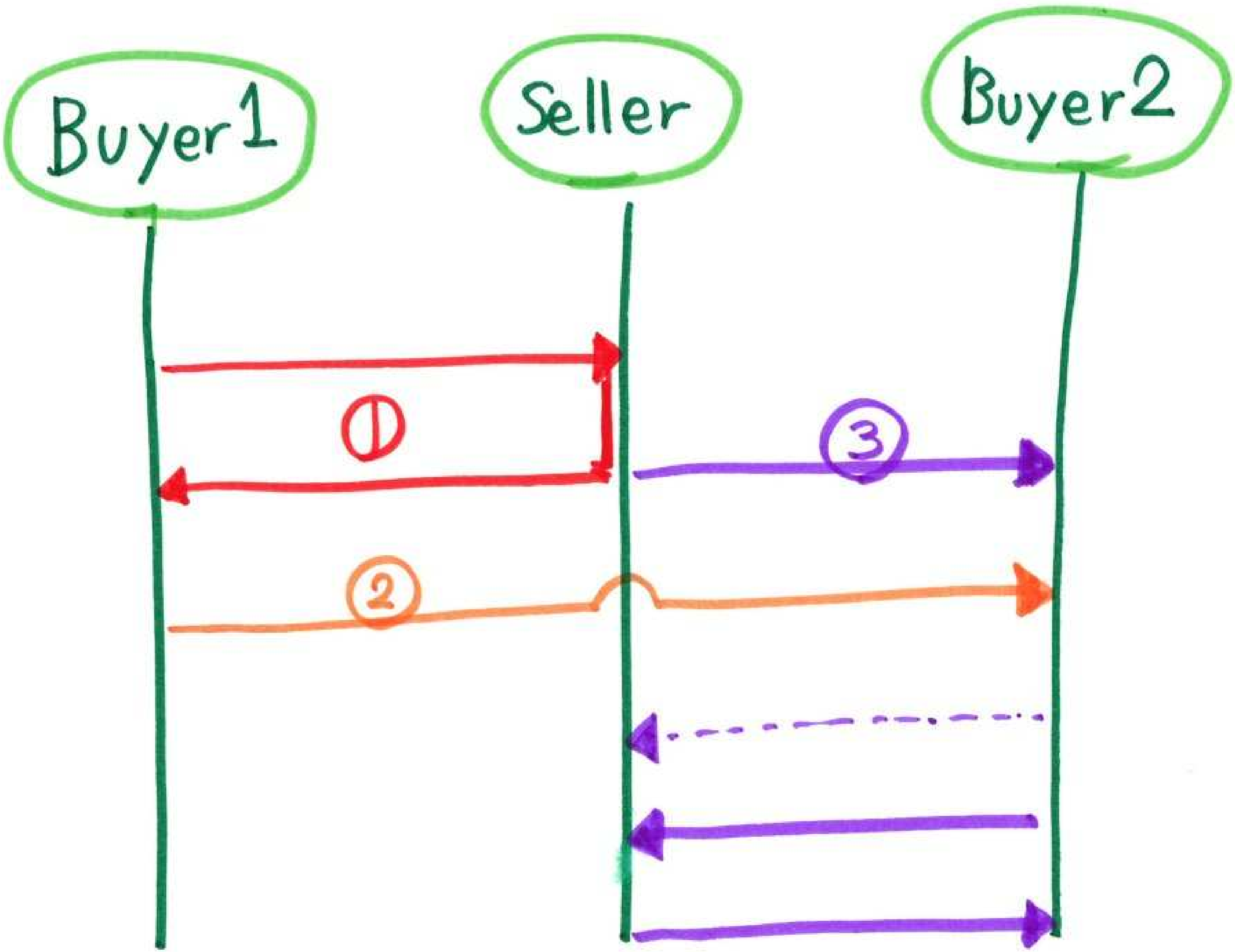
P | Q typable

$$!\,\mathrm{String}\,;\,?\,\mathrm{Int}\,;\,\oplus\{\mathrm{ok}: !\,\mathrm{String}\,;\,?\,\mathrm{Date}\,;\mathrm{end},\ \mathrm{quit}:\mathrm{end}\}$$

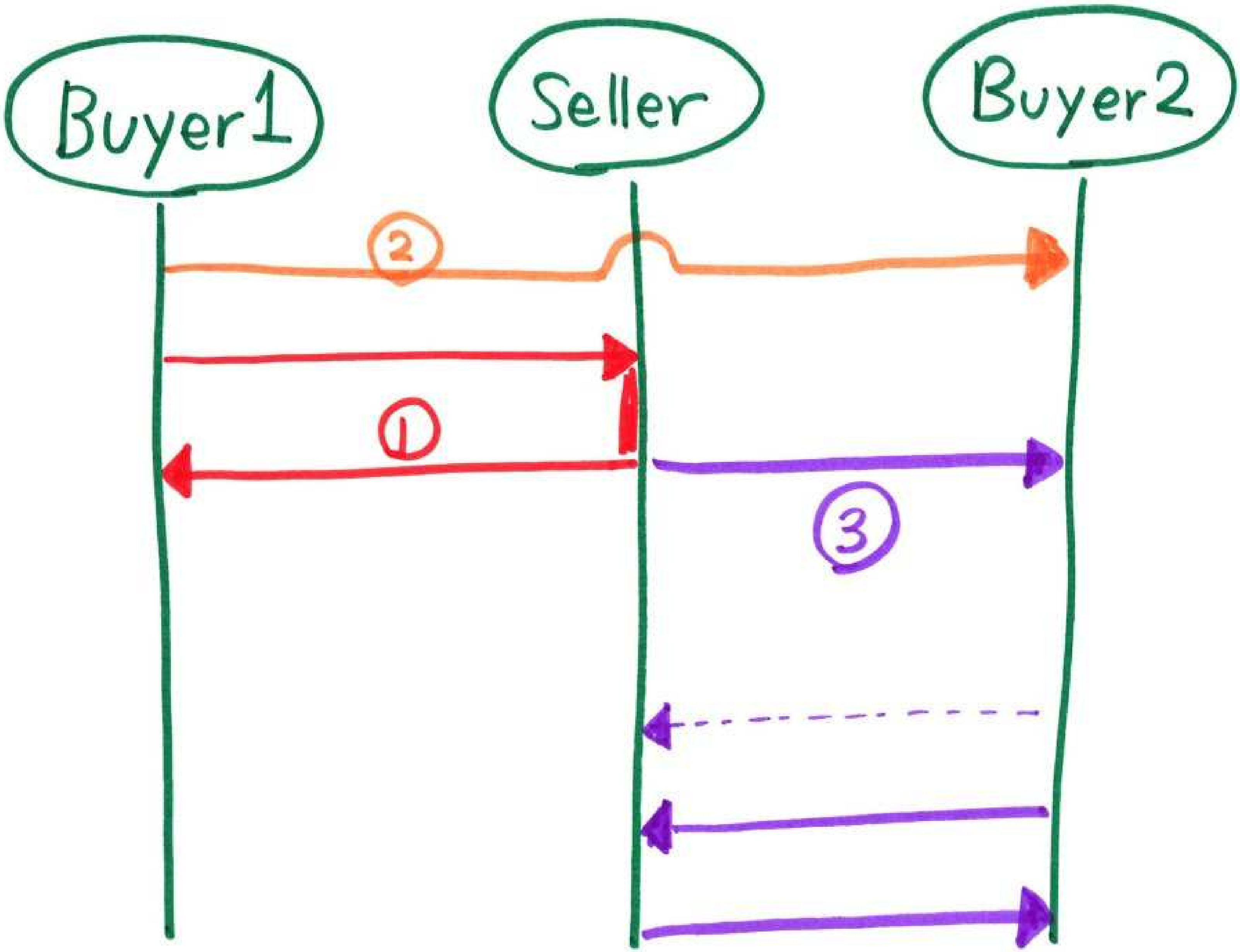
dual $$?\,\mathrm{String}\,;\,!\,\mathrm{Int}\,;\,\&\{\mathrm{ok}: ?\,\mathrm{String}\,;\,!\,\mathrm{Date}\,;\mathrm{end},\ \mathrm{quit}:\mathrm{end}\}$$

Multiparty Session Types

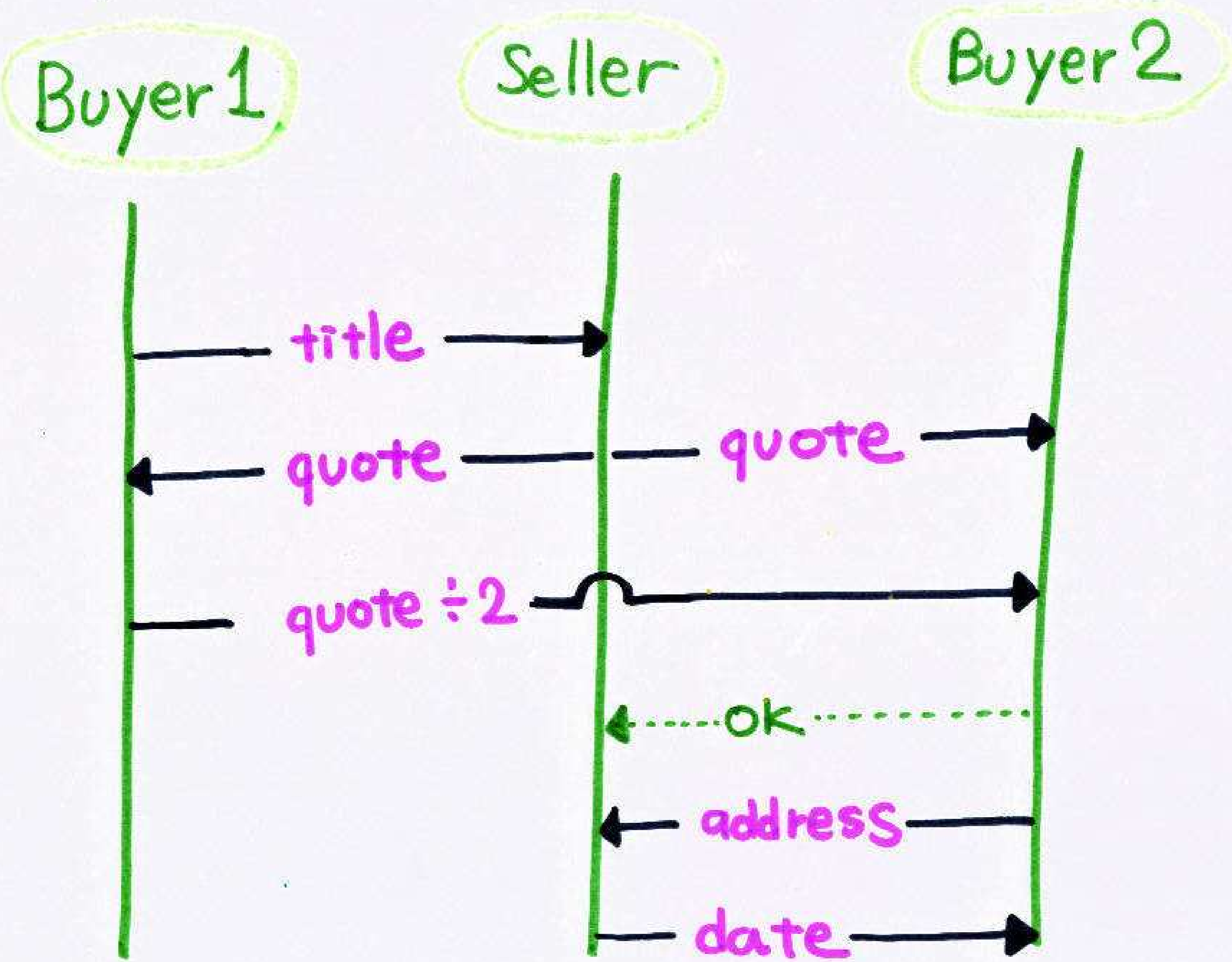




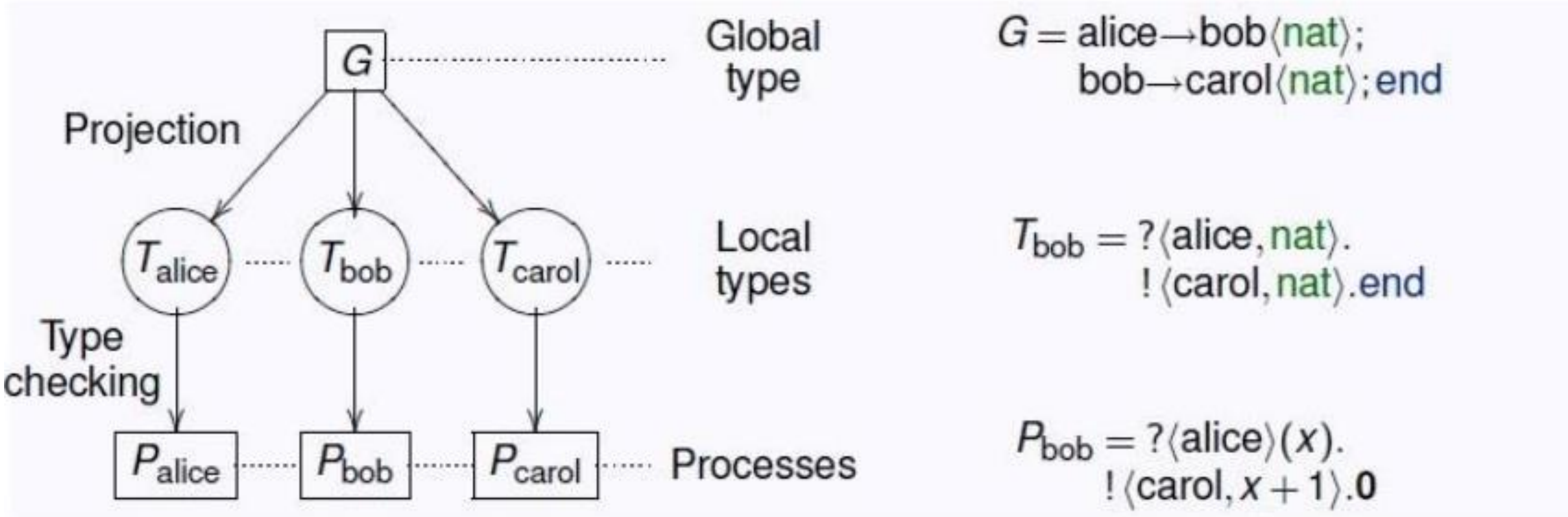
Buyer1
Seller
Buyer2
1
2
3


Buyer1
Seller
Buyer2
2
1
3

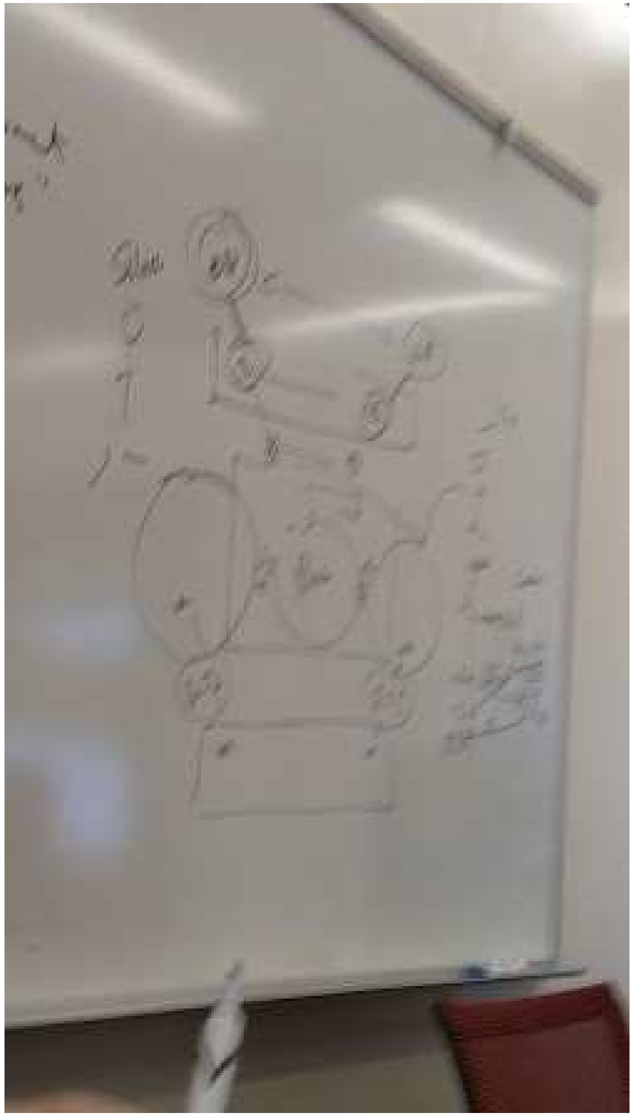
Multiparty Session Types



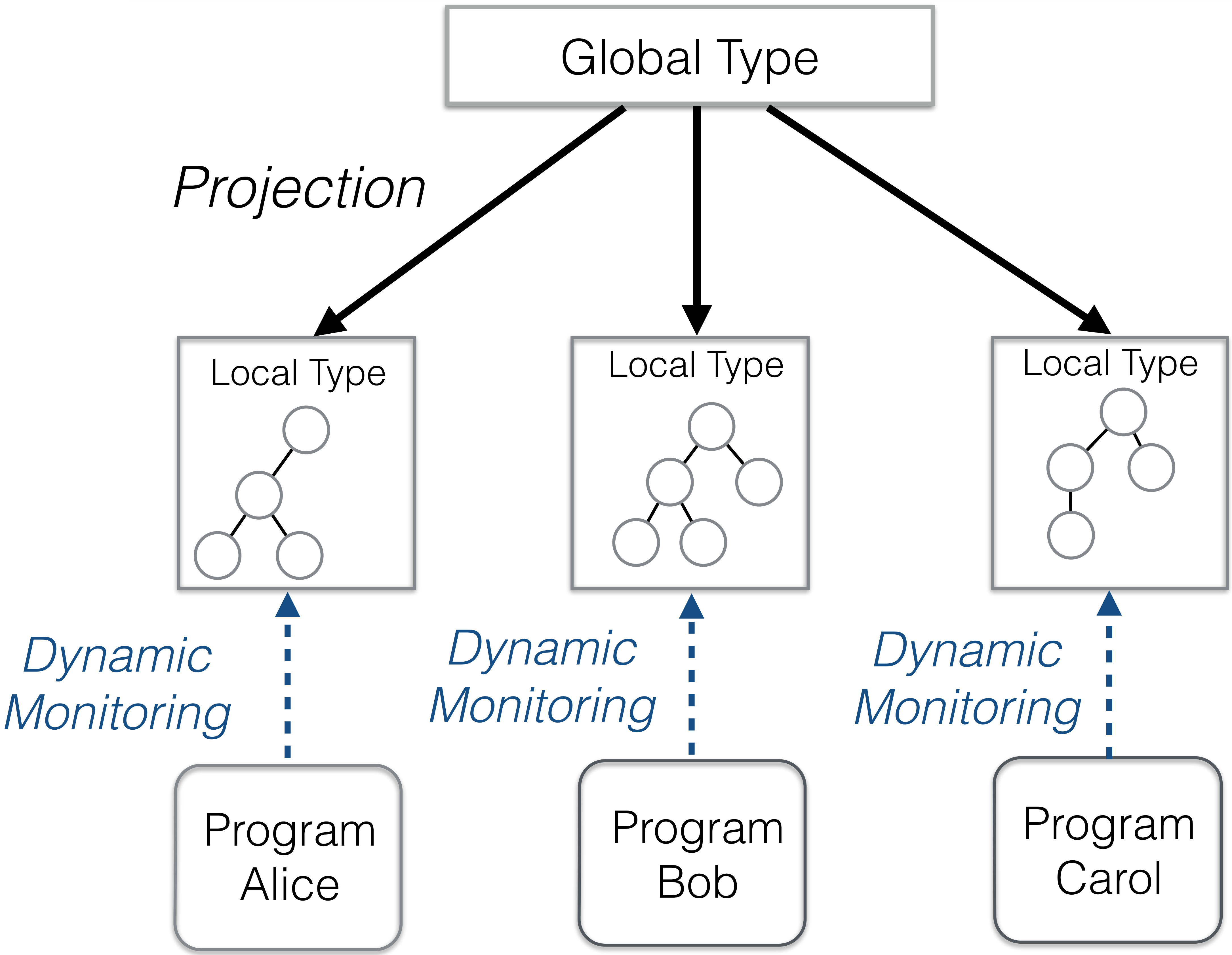
Session Types Overview



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

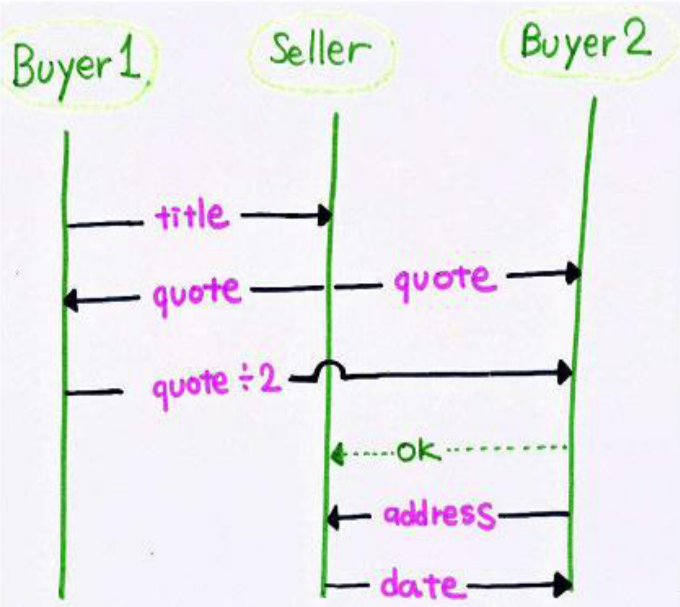


Dynamic Monitoring
[RV'13, COORDINATION'14, FMSD'15]



Two Buyer Protocol in Scribble

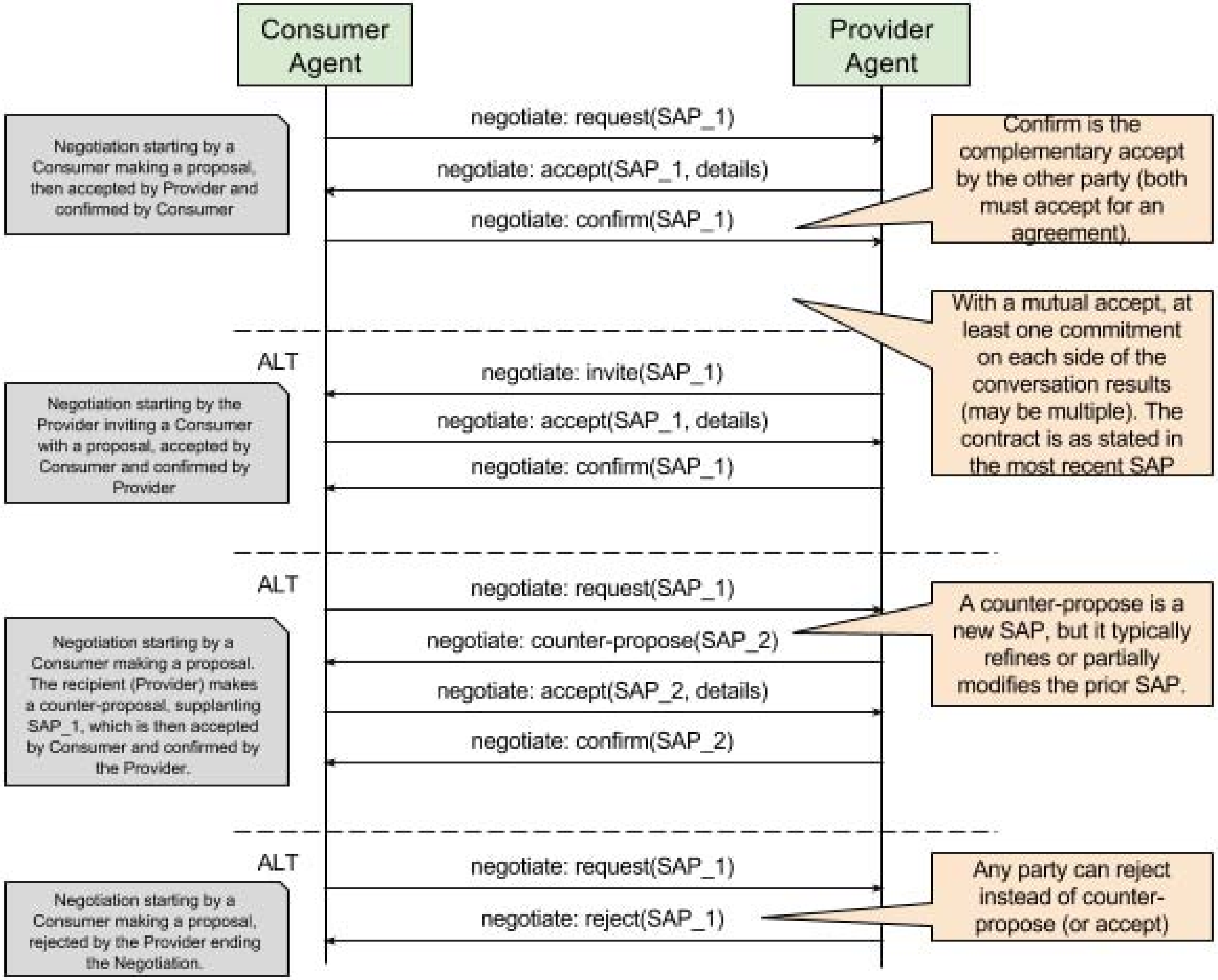
```
type <java> "java.lang.String" from "rt.jar" as String

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;
} } }
```

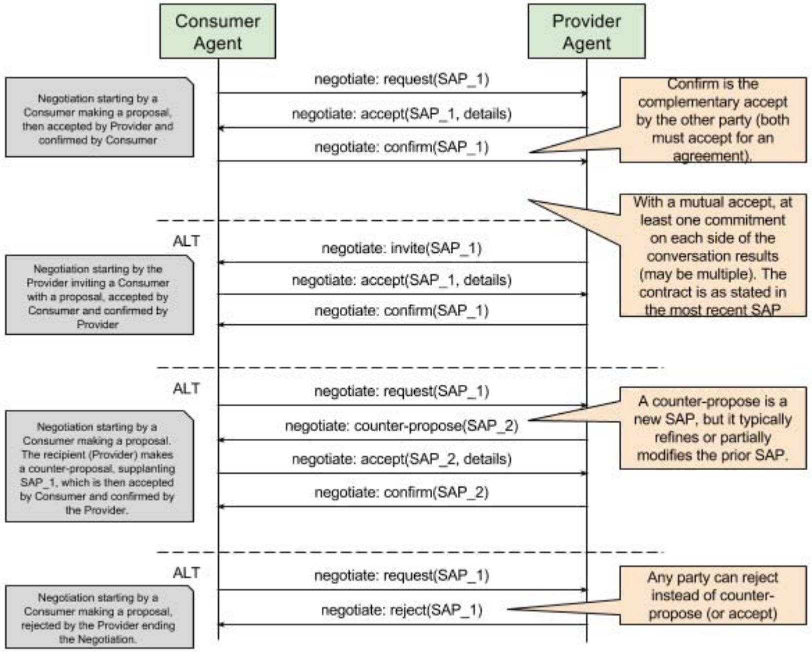
OOI agent negotiation 1/5



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

OOI agent negotiation 2/5

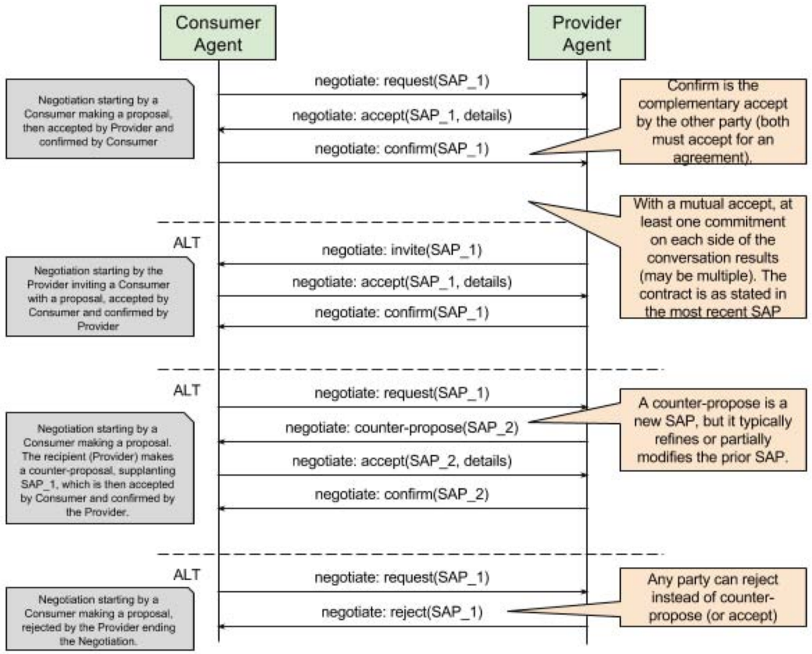
```
type <yml> "SAPDoc1" from "SAPDoc1.yml" as SAP;

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




```
}
```

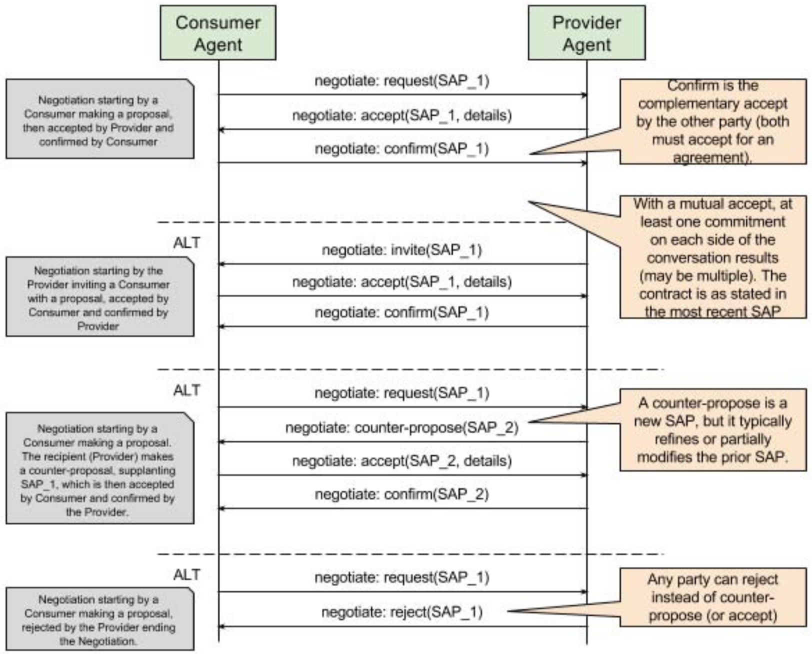
OOI agent negotiation 3/5 (choice)

```
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;



} }
```



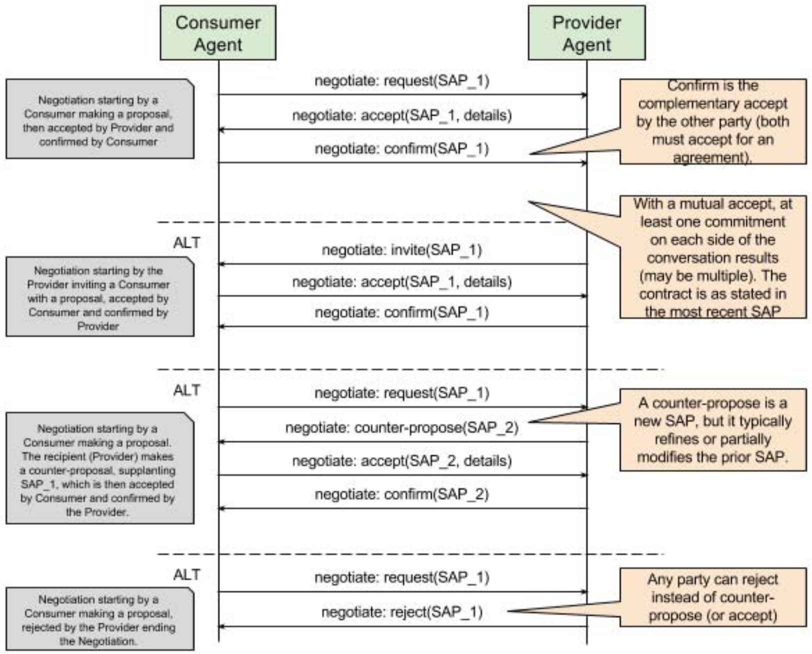
OOI agent negotiation 4/5

```
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;
      }
    }
```



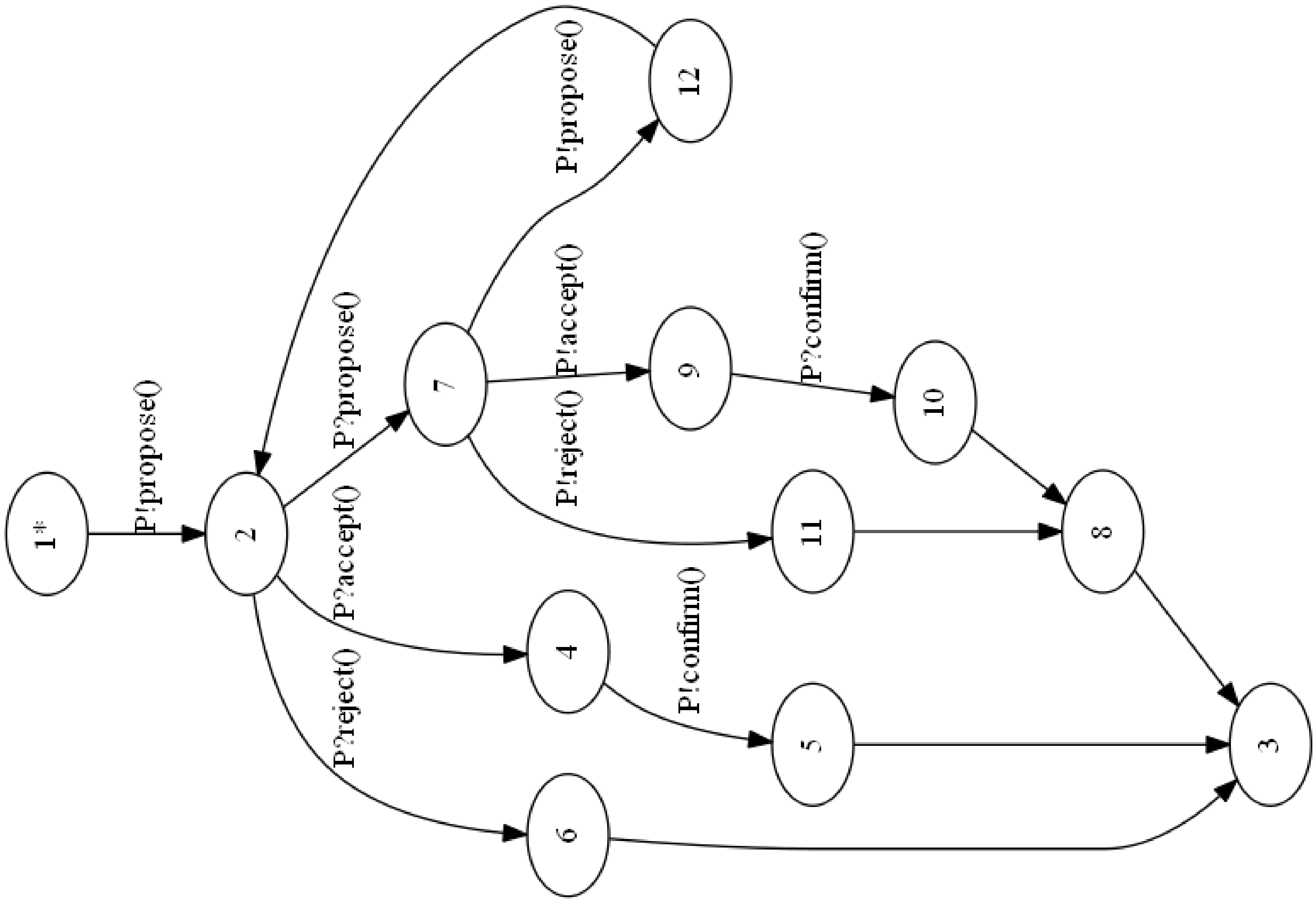

```
  }
```

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

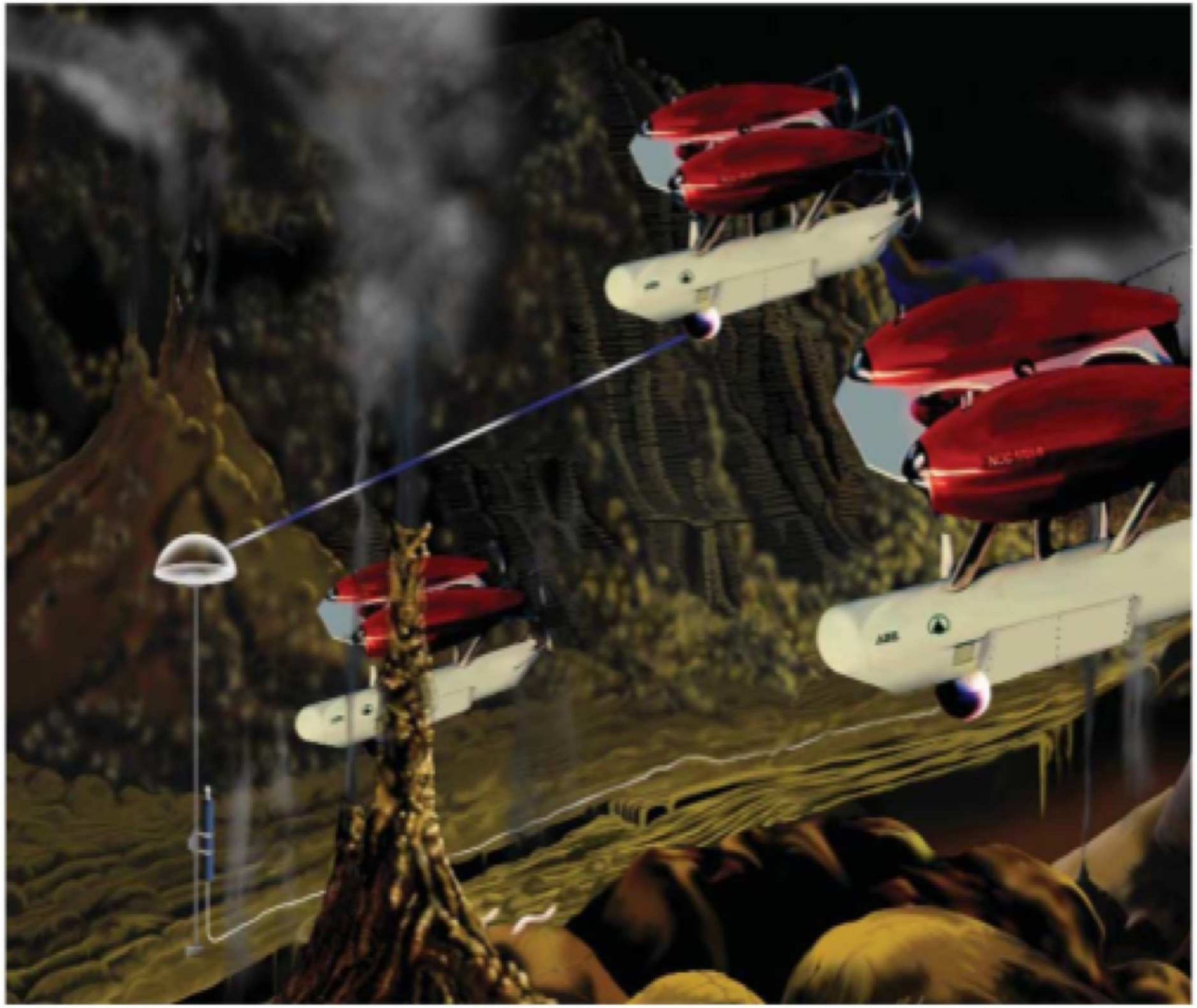


Figure 5: A coordinated set of autonomous underwater vehicles

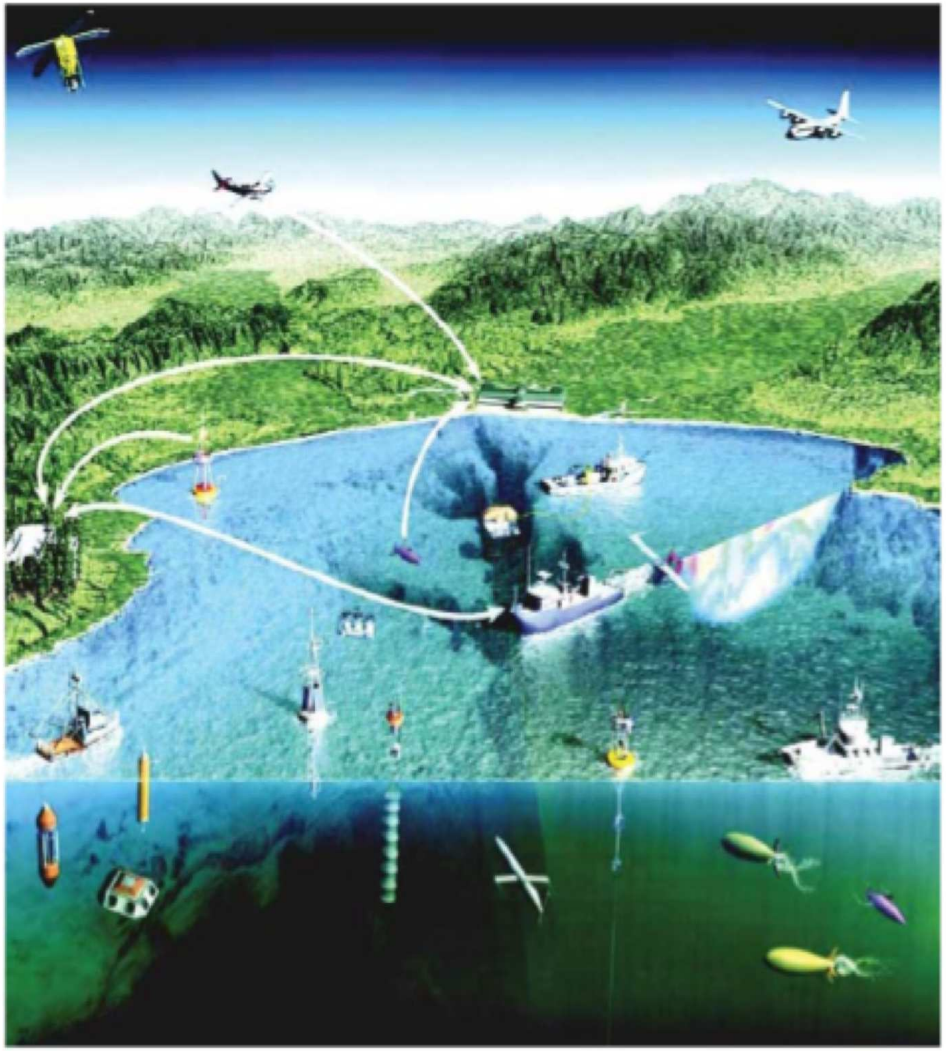
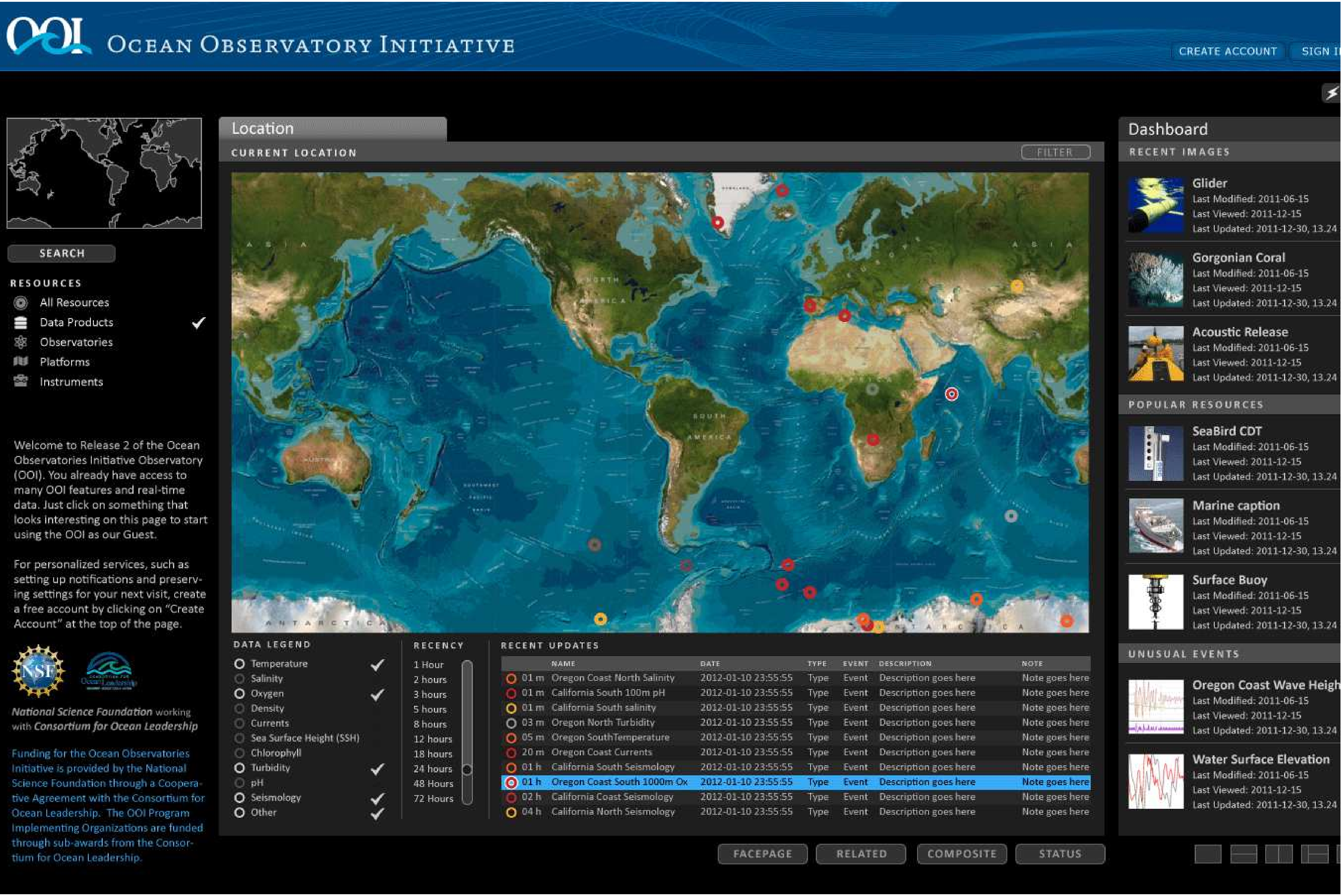
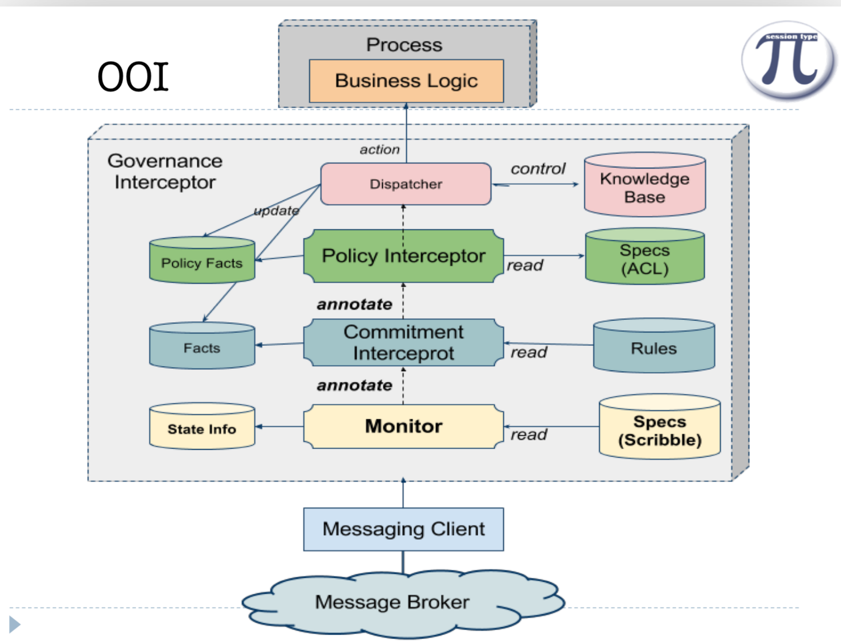


Figure 3: Observatory comprised of ships, aircraft and autonomous vehicles linked to assimilation modeling capabilities on shore

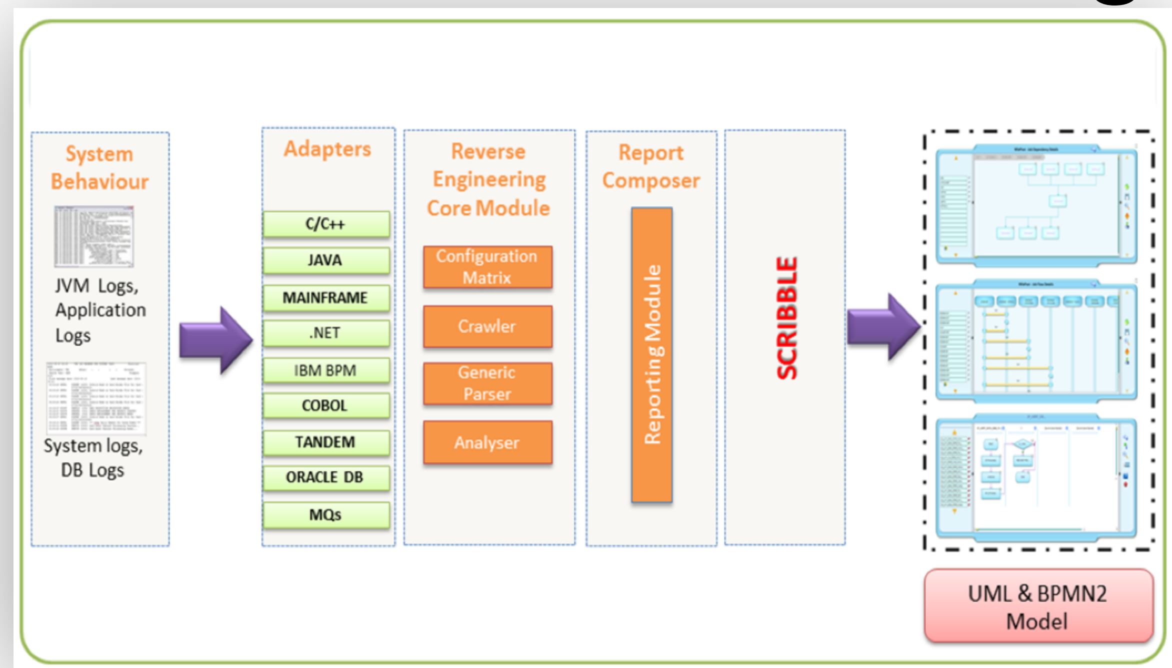

OCEAN OBSERVATORY INITIATIVE
CREATE ACCOUNT
SIGN I
SEARCH
RESOURCES
All Resources
Data Products
Observatories
Platforms
Instruments
Welcome to Release 2 of the Ocean Observatories Initiative Observatory (OOI). You already have access to many OOI features and real-time data. Just click on something that looks interesting on this page to start using the OOI as our Guest.
For personalized services, such as setting up notifications and preserving settings for your next visit, create a free account by clicking on "Create Account" at the top of the page.
National Science Foundation working with Consortium for Ocean Leadership
Funding for the Ocean Observatories Initiative is provided by the National Science Foundation through a Cooperative Agreement with the Consortium for Ocean Leadership. The OOI Program Implementing Organizations are funded through sub-awards from the Consortium for Ocean Leadership.
Location
CURRENT LOCATION
FILTER
DATA LEGEND
Temperature
Salinity
Oxygen
Density
Currents
Sea Surface Height (SSH)
Chlorophyll
Turbidity
pH
Seismology
Other
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
FACEPAGE
RELATED
COMPOSITE
STATUS
Dashboard
RECENT IMAGES
Glider
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24
Gorgonian Coral
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24
Acoustic Release
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24
POPULAR RESOURCES
SeaBird CDT
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24
Marine caption
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24
Surface Buoy
Last Modified: 2011-06-15
Last Viewed: 2011-12-15
Last Updated: 2011-12-30, 13.24
UNUSUAL EVENTS
Oregon Coast Wave 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

Applications

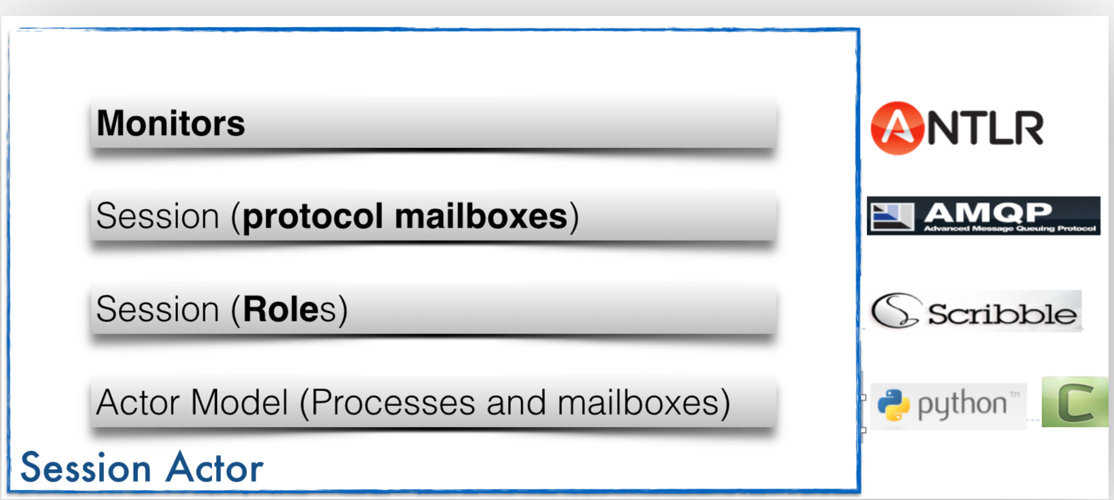
OOI Governance



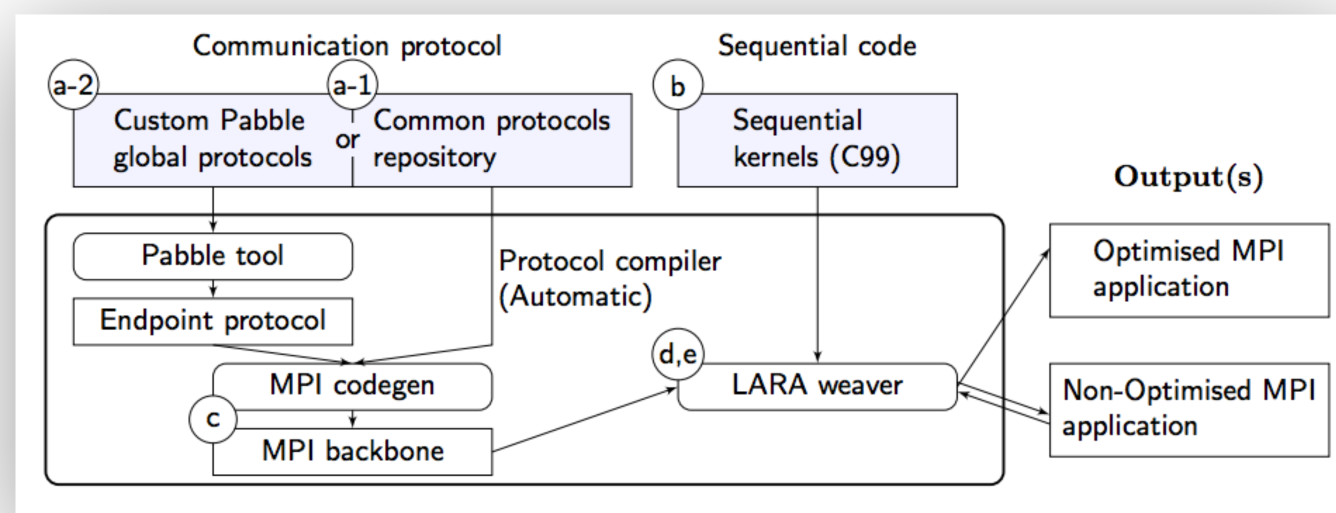
ZDLC: Process Modeling



Protocol Verification

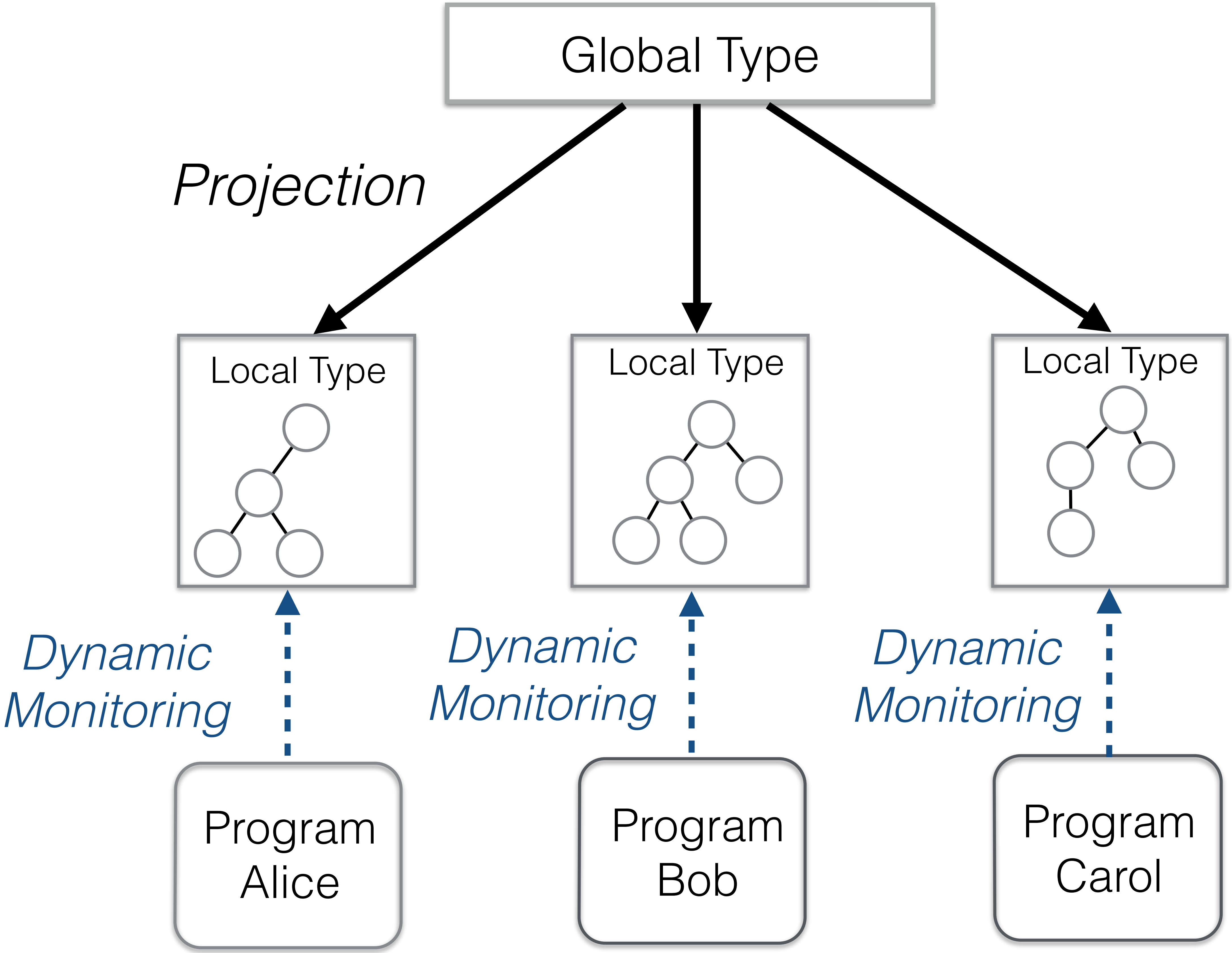


MPI code generations

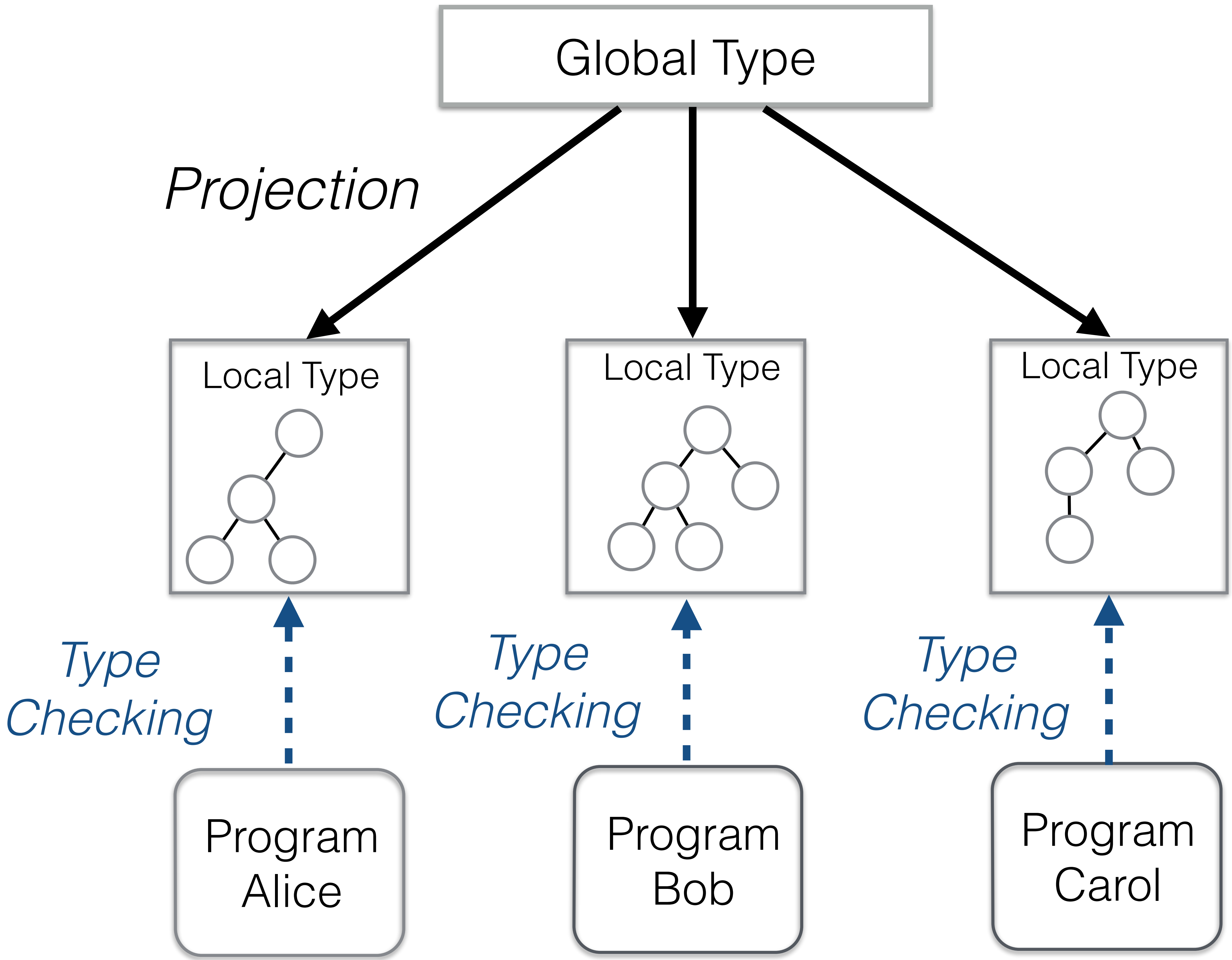


Dynamic Monitoring

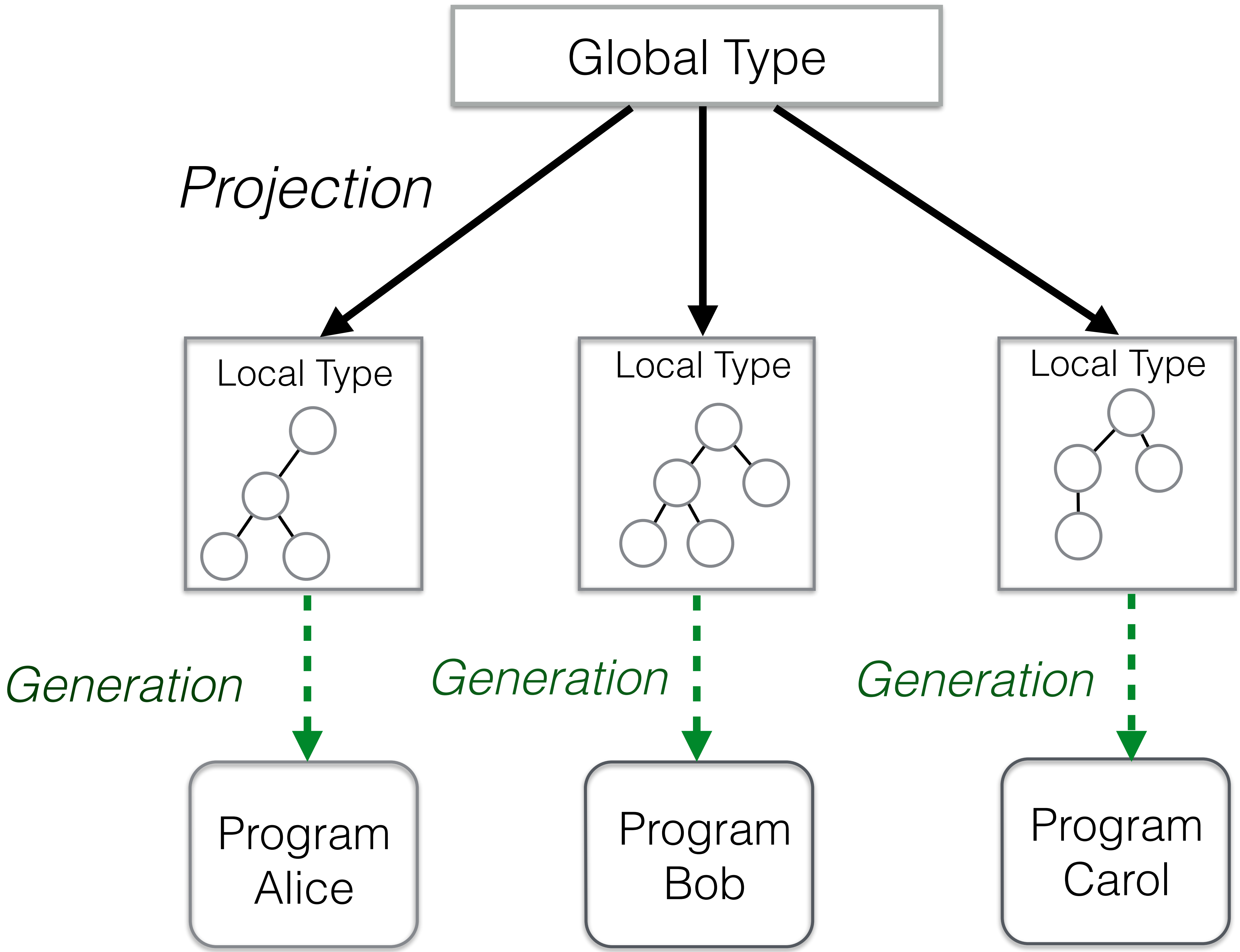
[RV'13, COORDINATION'14, FMSD'15]



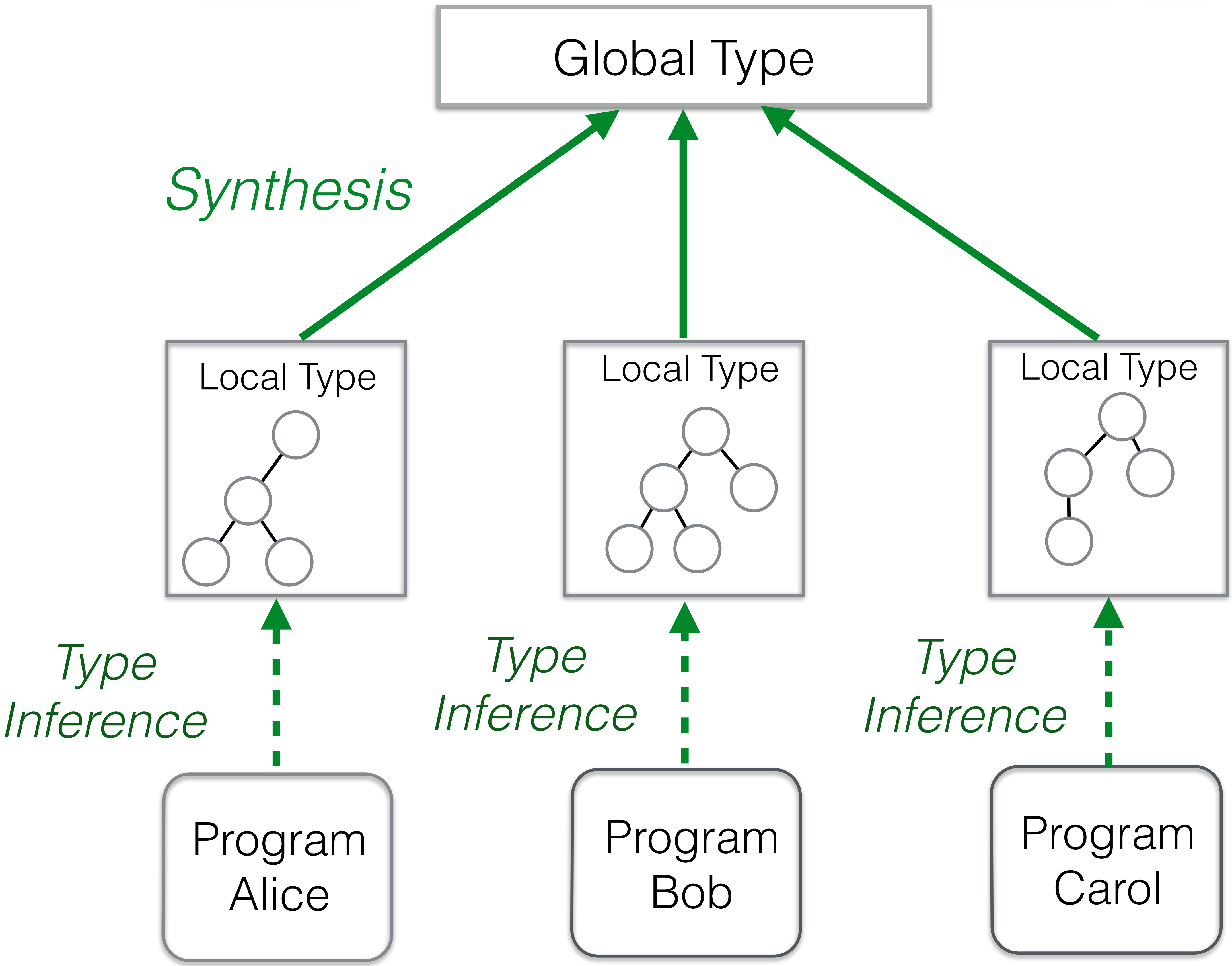
Type Checking [OOPSLA'15, POPL'16]



Code Generation [CC'15]



Synthesis
[ICALP'13, POPL'15, CONCUR'15]



Scribble SMTP

RFC 821 August 1982
Simple Mail Transfer Protocol

TABLE OF CONTENTS

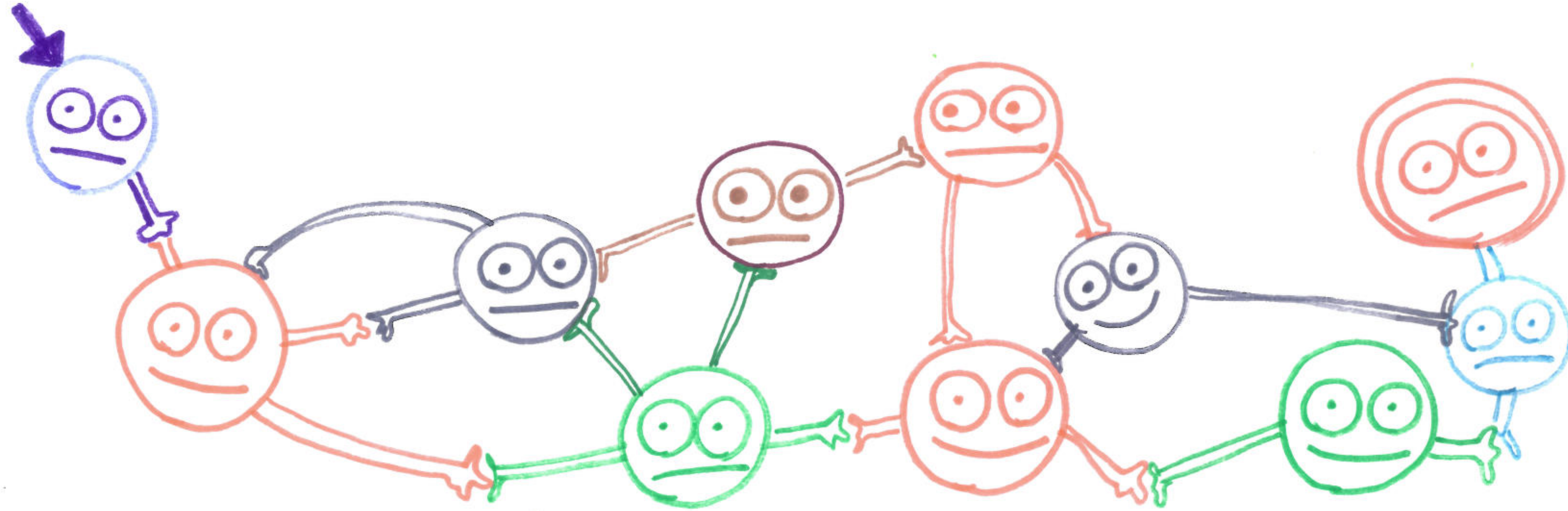
Multiparty Compatibility in Communicating Automata

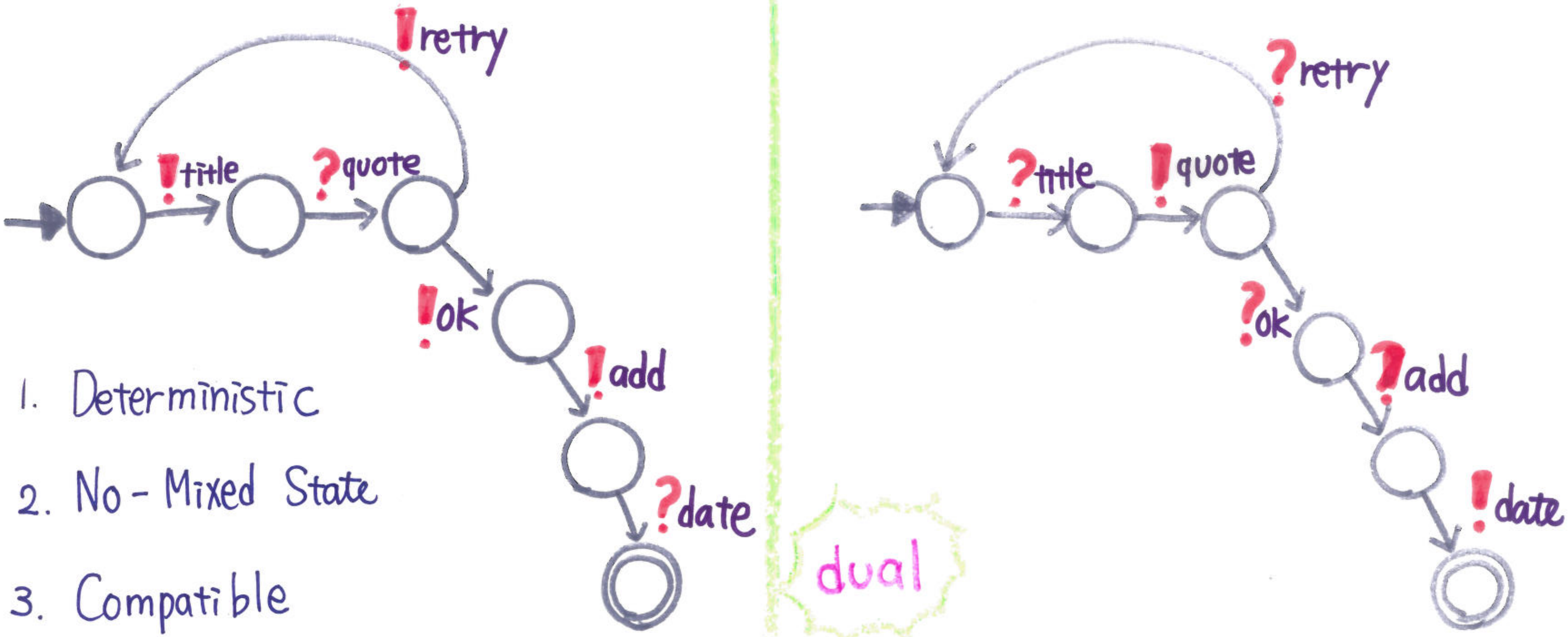
Synthesis and Characterisation of Multiparty Session Types

ICALP'13

Nobuko Yoshida

Pierre-Malo Deniélou



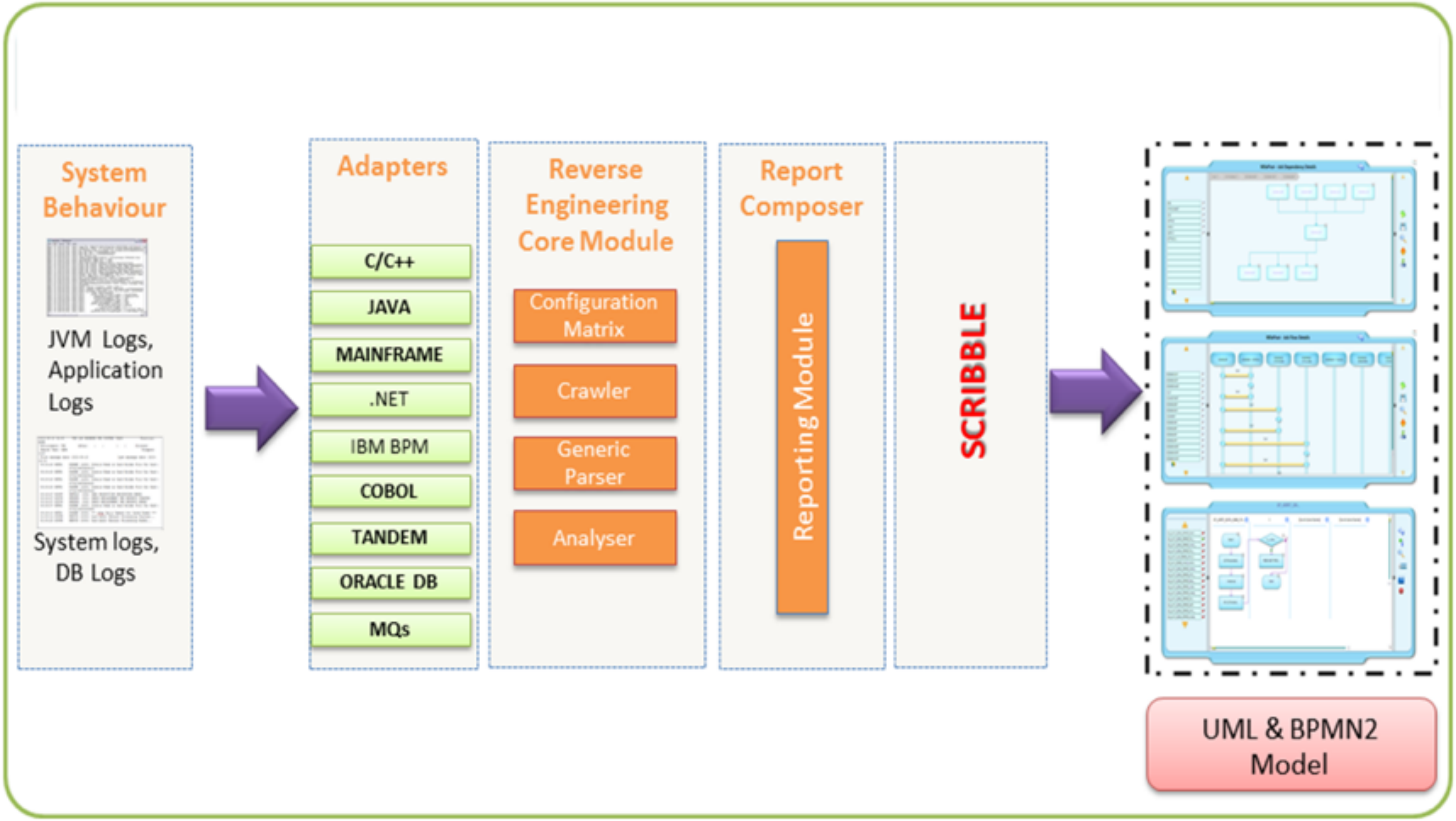


1. Deterministic
2. No-Mixed State
3. Compatible

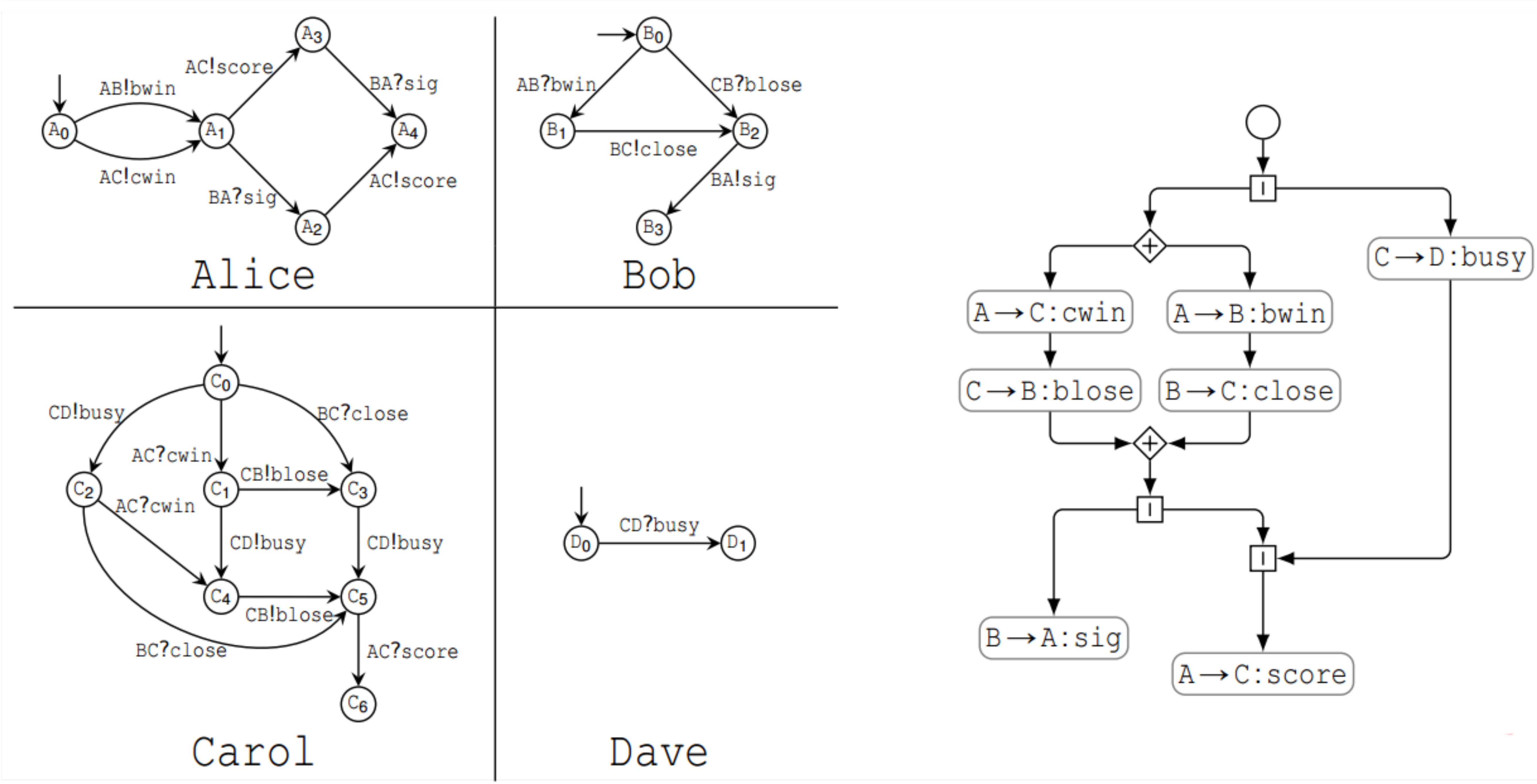
[Gouda et al 1986] Two compatible machines without mixed states which are deterministic satisfy deadlock-freedom.

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

Zero Deviation Life Cycle Platform



From Communicating Machines to Graphical Choreographies [POPL'15, CONCUR'15]

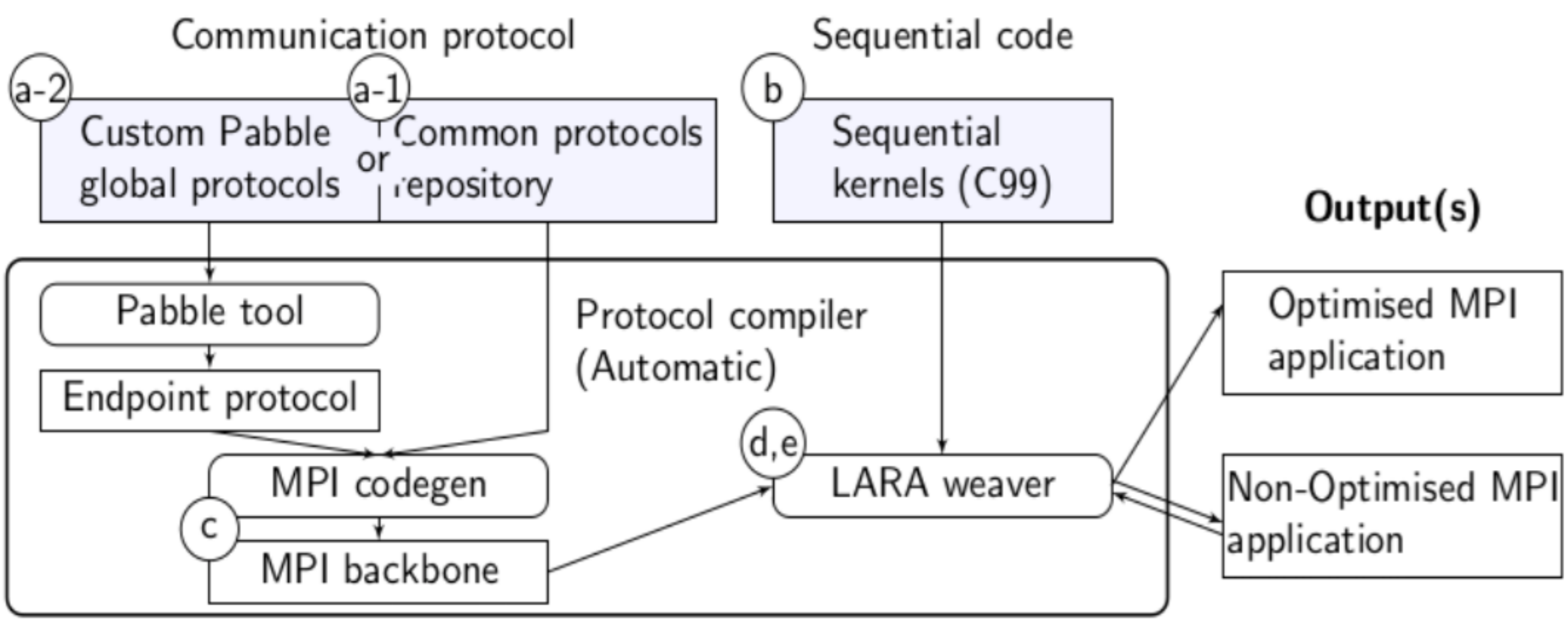


[ESOP'10,ESOP'12,CONCUR'12,CONCUR'14]

Message Passing Programming [CC'15, OOPSLA'15]

A complete parallel programming workflow

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

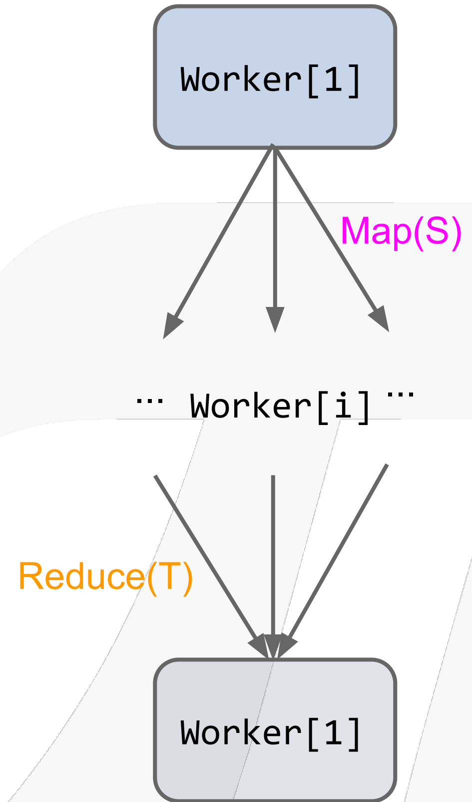


Example: Simple search engine

Scatter-Gather protocol

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

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

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

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



Scatter-Gather protocol

Example: search engine

Merging backbone with kernels

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

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

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

Evaluation

Productivity: Flexibility

Reusable protocols

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

Berkeley Dwarfs [CACM'09]

- Representative parallel computing patterns
- 4 of 5 HPC patterns

		Repository	Berkeley HPC Dwarfs
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

Language and Implementations

- Carrying out large-scale experiences with OOI, VMWare, Red Hat, Congnizant, Pivotal, Amazon, AMQP, RabbitMQ
  - JBoss SCRIBBLE [ICDCIT'10,COB'12,TGC'13] and ZDLC projects
- High-performance computing
  Session Java [ECOOP'08,ECOOP'10,Coordination'11]
  $\Longrightarrow$ Multiparty Session C and MPI
  [TOOLS'12,Hearts'12,EuroMPI'12,PDP'14,CC'15,OOPSLA'15]
- Multiparty session languages Ocaml, Java, C, MPI, Python, Scala, Jolie, Haskell, Erlang
  - Effect and Concurrent Haskell [POPL'16]
  - Practical interruptible conversations: Distributed dynamic verification with session types and Python [RV'13,FMCD'15]
  - Multiparty Session Actors [COORDINATION'14]

Multiparty Session Type Theory

- Multiparty Asynchronous Session Types [POPL'08,JACM]
- Progress
  - Global Progress in Dynamically Interleaved Multiparty Sessions [CONCUR'08], [Math. Struct. Comp. Sci.]
  - Inference of Progress Typing [Coordination'13]
- Asynchronous Optimisations and Resource Analysis
  - Global Principal Typing in Partially Commutative Asynchronous Sessions [ESOP'09]
  - Higher-Order Pi-Calculus [TLCA'07,TLCA'09,Info.&Comp]
  - Buffered Communication Analysis in Distributed Multiparty Sessions [CONCUR'10]

- Extensions of Multiparty Session Types
  - Multiparty Symmetric Sum Types [Express'10]
  - Trustworthy Pervasive Healthcare Services via Multi-party Session Types [FHIES'12]
  - Parameterised Multiparty Session Types [FoSSaCs'10, LMCS, SPLASH'15]
  - Global Escape in Multiparty Sessions [FSTTCS'10] [Math. Struct. Comp. Sci.]
  - Dynamic Multirole Session Types [POPL'11]
  - Nested Multiparty Sessions [CONCUR'12]
  - Timed Multiparty Session Types [CONCUR'14]
- Dynamic Monitoring
  - Monitoring Networks through Multiparty Sessions [TGC'11] [FORTE'13]

- Automata Theories
  - Multiparty Session Automata [ESOP'12]
  - Synthesis in Communicating Automata [ICALP'13]
  - From communicating machines to graphical choreographies [POPL'15]
  - Meeting Deadlines Together [CONCUR'15]
- Denotational and Trace Semantics
  - Expressiveness of Multiparty Session Types [FSTTCS'15]
- Petri Nets
  - Multiparty Session Nets [TGC'14]
- Typed Behavioural Theories
  - On Asynchronous Eventful Session Semantics [FORTE'11] [Math. Struct. Comp. Sci.]
  - Governed Session Semantics [CONCUR'13]
  - Characteristic Bisimulations for Higher-Order Session Processes [CONCUR'15]

- Choreography Languages
  - Compositional Choreographies [CONCUR'13]
- Logics
  - Design-by-Contract for Distributed Multiparty Interactions [CONCUR'10]
  - Specifying Stateful Asynchronous Properties [CONCUR'12]
  - Multiparty, Multi-session Logic [TGC'12]
  - Multiparty Session Types as Coherence Proofs [CONCUR'15]

Session Type Reading List

- Home Page http://mrg.doc.ic.ac.uk/
- [ESOP'98] Language Primitives and Type Disciplines for Structured Communication-based Programming, Honda, Vasconcelos and Kubo
- [SecRet'06] Language Primitives and Type Disciplines for Structured Communication-based Programming Revisited, Yoshida and Vasconcelos, ENTCS.
- [SFM'15] Gentle Introduction to Multiparty Asynchronous Session Types, Coppo et al.
- [POPL'15] From communicating machines to graphical choreographies, Lange, Tuosto and Yoshida.

- [COB'14,TGC'13] The Scribble Protocol Language, Honda et al.
- [ECOOP'08] Session-Based Distributed Programming in Java, Hu, Yoshida and Honda.
- [FMSD'15] Practical interruptible conversations: Distributed dynamic verification with multiparty session types and Python, Demangeon, Honda, Hu, Neykova and Yoshida.
- [CC'15] Protocols by Default: Safe MPI Code Generation based on Session Types, Ng, Coutinho and Yoshida.

A rare cluster of qualities

From the team of OOI CI:

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