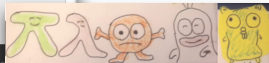


on Polymorphic  
and Sessions  
Functions

a Tale of Two  
Encodings

Bernardo Toninho @Nova

Nobuko Yoshida @Imperial



## NEWS

The paper *Multiparty asynchronous session types* by Kohei Honda, Nobuko Yoshida, and Marco Carbone, published in POPL 2008 has been awarded the ACM SIGPLAN Most Influential POPL Paper Award today at POPL 2018.

» more

10 Jan 2018

Estafet has published a page on their usage of the Scribble language developed in our group with RedHat and other industry partners.

» more

25 Sep 2017

Nick spoke at Golang UK 2017 on applying behavioural types to verify concurrent Go programs.

## SELECTED PUBLICATIONS

2018

Julien Lange , Nicholas Ng , Bernardo Toninho , Nobuko Yoshida : [A Static Verification Framework for Message Passing in Go using Behavioural Types](#). *To appear in ICSE 2018* .

Bernardo Toninho , Nobuko Yoshida : [Depending On Session Typed Process](#). *To appear in FoSSaCS 2018* .

Bernardo Toninho , Nobuko Yoshida : [On Polymorphic Sessions And Functions: A Talk of Two \(Fully Abstract\) Encodings](#). *To appear in ESOP 2018* .

Rumyana Neykova , Raymond Hu , Nobuko Yoshida , Fahd Abdeljallal : [Session Type Providers: Compile-time API Generation for Distributed Protocols with Interaction Refinements in F#](#). *To appear in CC 2018* .

### Post-docs:

Simon CASTELLAN

David CASTRO

Francisco FERREIRA

Raymond HU

Rumyana NEYKOVA

Nicholas NG

Alceste SCALAS

### PhD Students:

Assel ALTAYEVA

Juliana FRANCO

Eva GRAVERSEN

# 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.*



## 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



# End-to-End Switching Programme by DCC



**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



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

[www.estafet.com](http://www.estafet.com)

Estafet Managed Service



7. Generate exception report and send back to DCC

# End-to-End Switching Programme by DCC



**Estafet**

Innovate | Deliver | Transform

## Caveats:

1. Using earlier implementation of Scribble (CDL), because we already have those tools
2. Using earlier plugin to Eclipse - we'd want to improve this
3. We're not going via OpenTracing - this is part of the bid costs



7. Generate exception report and send back to DCC

Scope of the demo



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

CC'18

## A Session Type Provider

Compile-Time API Generation of Distributed Protocols with Refinements in F#

Rumyana Neykova  
Imperial College London  
United Kingdom

Raymond Hu  
Imperial College London  
United Kingdom

Nobuko Yoshida  
Imperial College London  
United Kingdom

Fahd Abdeljallal  
Imperial College London  
United Kingdom

### Abstract

We present a library for the specification and implementation of distributed protocols in native F# (and other .NET languages) based on multiparty session types (MPST). There are two main contributions. Our library is the first practical development of MPST to support what we refer to as *interaction refinements*: a collection of features related to the refinement of *protocols*, such as message-type refinements (value constraints) and message-value dependent control flow. A well-typed endpoint program using our library is guaranteed to perform only compliant session I/O actions on the refined protocol, up to premature termination. Our library is developed as a session *type provider*,

### 1 Introduction

*Type providers* [20, 27] are a .NET feature for a form of compile-time meta programming, designed to bridge between programming in statically typed languages such as F# and C#, and working with so-called *information spaces*—structured data sources such as SQL databases or XML data.

A type provider works as a compiler plugin that performs on-demand generation of *types*: it takes a schema for an external information space, and generates types that allow the data to be manipulated via a strongly-typed interface, with benefits such as static error detection and IDE auto-completion. For example, an instantiation of the in-built type provider for WSDL Web services [6] may look like



Graydon Hoare

@graydon\_pub

(This stuff is \_fantastic\_)

11:31 PM - 11 Mar 2018

32 Retweets 83 Likes



shots fired @zeeshanlakhani · Mar 12

Replying to @graydon\_pub @dsyme

Awesome!

Brendan Zabarauskas @brendanzab ·

Replying to @graydon\_pub

This stuff fills me with hope!

Ryan Riley @panesofglass · Mar 12

Replying to @graydon\_pub

This is amazing! I guess I need to switch



# Behavioural Type-Based Static Verification Framework

for

# GO



Juhen Lange



Nicholas Ng



Bernardo Toninho



Nobuko Yoshida



# GOLANG UK CONFERENCE

16<sup>th</sup>\*, 17<sup>th</sup> & 18<sup>th</sup> AUGUST 2017

The Brewery, London



Imperial College  
London

Home

College and Campus

Science

Engineering

Health

Business

Search here...

Go

## Go concurrency verification research at DoC grabs headline

POPL'17

## the morning paper

ICSE'18

an interesting/influential/important paper from the world of CS every weekday morning, as selected by Adrian Colyer

Home About Info QR Editions Subscribe

### A static verification framework for message passing in Go using behavioural types

JANUARY 25, 2018

tags: Concurrency, Programming Languages

[A static verification framework for message passing in Go using behavioural types](#) Lange et al., *ICSE 18*

*With thanks to Alexis Richardson who first forwarded this paper to me.*

We're jumping ahead to ICSE 18 now, and a paper that has been accepted for publication there later this year. It fits with the theme we've been exploring this week though, so I thought I'd cover it now. We've seen verification techniques applied in the context of [Rust](#) and [JavaScript](#), looked at the integration of [linear types in Haskell](#), and today it is the turn of Go!

SUBSCRIBE



never miss an issue! The Morning Paper delivered straight to your inbox.

SEARCH

type and press enter

ARCHIVES

Select Month

MOST READ IN THE  
LAST FEW DAYS

currency  
rates a

tured in the  
'high  
interesting  
easily  
le of the  
L (Principles

# Selected Publications 2017/2018

- ▶ **[LICS'18]** Romain Demangeon, NY: Casual Computational Complexity of Distributed Processes.
- ▶ **[CC'18]** Romyana Neykova , Raymond Hu, NY, Fahd Abdeljallal: Session Type Providers: Compile-time API Generation for Distributed Protocols with Interaction Refinements in F#.
- ▶ **[FoSSaCS'18]** Bernardo Toninho, NY: Depending On Session Typed Process.
- ▶ **[ESOP'18]** Bernardo Toninho, NY: On Polymorphic Sessions And Functions: A Talk of Two (Fully Abstract) Encodings.
- ▶ **[ESOP'18]** Malte Viering, Tzu-Chun Chen, Patrick Eugster, Raymond Hu , Lukasz Ziarek: A Typing Discipline for Statically Verified Crash Failure Handling in Distributed Systems.
- ▶ **[ICSE'18]** Julien Lange, Nicholas Ng, Bernardo Toninho, NY : A Static Verification Framework for Message Passing in Go using Behavioural Types
- ▶ **[ECOOP'17]** Alceste Scala, Raymond Hu, Ornela Darda, NY: A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming..
- ▶ **[COORDINATION'17]** Keigo Imai, NY, 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.

# Selected Publications 2017/2018


- ▶ **[LICS'18]** Romain Demangeon, NY: Casual Computational Complexity of Distributed Processes.
- ▶ **[CC'18]** Romyana Neykova , Raymond Hu, NY, Fahd Abdeljallal: Session Type Providers: Compile-time API Generation for Distributed Protocols with Interaction Refinements in F#.
- ▶ **[FoSSaCS'18]** Bernardo Toninho, NY: Depending On Session Typed Process.
- ▶ **[ESOP'18]** Bernardo Toninho, NY: On Polymorphic Sessions And Functions: A Talk of Two (Fully Abstract) Encodings.
- ▶ **[ESOP'18]** Malte Viering, Tzu-Chun Chen, Patrick Eugster, Raymond Hu , Lukasz Ziarek: A Typing Discipline for Statically Verified Crash Failure Handling in Distributed Systems.
- ▶ **[ICSE'18]** Julien Lange, Nicholas Ng, Bernardo Toninho, NY : A Static Verification Framework for Message Passing in Go using Behavioural Types.
- ▶ **[ECOOP'17]** Alceste Scala, Raymond Hu, Ornella Darda, NY: A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming.
- ▶ **[COORDINATION'17]** Keigo Imai, NY, 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.


# POPL 2019 Research Papers

[Write a Blog >>](#)



## 15:21 - 16:27: [Research Papers - Capabilities and Session Types I at POPL Track 2](#)


15:21 - 15:43  [StkTokens: Enforcing Well-Bracketed Control Flow and Stack Encapsulation Using Linear Capabilities](#)  
*Talk*  
Lau Skorstengaard , Dominique Devriese [Vrije Universiteit Brussel, Belgium](#), Lars Birkedal [Aarhus University](#)

15:43 - 16:05  [Two sides of the same coin: Session Types and Game Semantics](#)  
*Talk*  
Simon Castelan [Imperial College London, UK](#), Nobuko Yoshida [Imperial College London](#)  
[DOI](#) [Pre-print](#)

16:05 - 16:27  [Exceptional Asynchronous Session Types: Session Types without Tiers](#)  
*Talk*  
Simon Fowler [The University of Edinburgh](#), Sam Lindley [University of Edinburgh, UK](#), J. Garrett Morris [University of Kansas, USA](#), Sara Décova  
[Pre-print](#)

## 16:37 - 17:43: [Research Papers - Session Types II at POPL Track 2](#)

16:37 - 16:59  [Interconnectability of Session-Based Logical Processes](#)   
*Talk*  
Bernardo Toninho [NOVA-LINCS, FCT/UNL](#), Nobuko Yoshida [Imperial College London](#)  
[DOI](#) [Pre-print](#)

16:59 - 17:21  [Distributed Programming using Role-Parametric Session Types in Go](#)  
*Talk*  
David Castro [Imperial College London](#), Raymond Hu [Imperial College London](#), Sung-Shik Jongmans [Open University of the Netherlands](#), Nicholas Ng [Imperial College London](#), Nobuko Yoshida [Imperial College London](#)  
[DOI](#) [Pre-print](#)

17:21 - 17:43  [Less is More: Multiparty Session Types Revisited](#)  
*Talk*  
Alceste Scalas [Imperial College London](#), Nobuko Yoshida [Imperial College London](#)  
[DOI](#) [Pre-print](#)



on Polymorphic  
and Sessions  
Functions

a Tale of Two  
Encodings

Bernardo Toninho @Nova

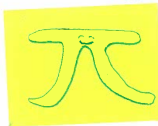
Nobuko Yoshida @Imperial

on Polymorphic

Sessions

and

Function



a Tale of Two

Fully  
Abstract

Encodings

Bernardo Toninho @ Nova

Nobuko Yoshida @ Imperial

# on Polymorphic Sessions and Functions



a Tale of Two Fully Abstract Encodings

Bernardo Toninho @ Nova

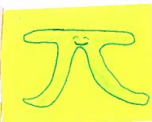
Nobuko Yoshida @ Imperial

on Polymorphic

Sessions

and

Function



a Tale of Two



Fully  
Abstract

Encodings

Bernardo Toninho @Nova

Nobuko Yoshida @Imperial

# The $\pi$ -calculus as a Descriptive Tool

by Kohei  
Honda  
1995

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

$$\pi \quad P ::= \sum \pi_i.P_i \mid P_1Q \mid \langle \nu a \rangle P \mid !P \mid \emptyset.$$

$$\text{with } \pi ::= x \langle \bar{a} \rangle \mid \bar{x} \langle a \rangle.$$

Milner's  
Encoding  
1991

$\lambda$  in  $\pi$

$$[x]_u \triangleq \bar{x}(u).$$

$$[\lambda x.M]_u \triangleq u(x.u). [M]_u.$$

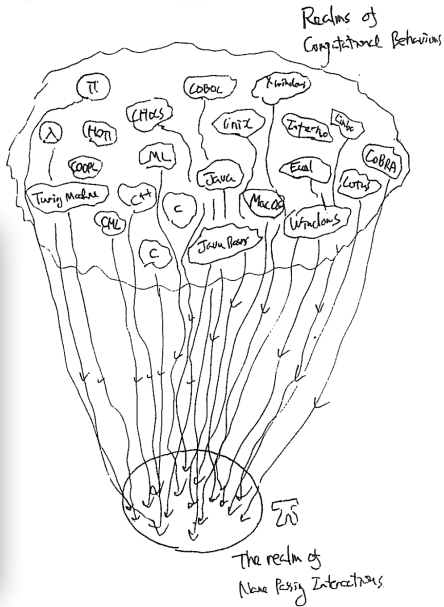
$$[MN]_u \triangleq (vz) ([M]_y \mid \bar{z}(u) \mid [z=N]_y)$$

$$\text{with } [z=N]_y \triangleq !z(y). [N]_y.$$

# \* Examples of Representable Computation.

- $\lambda$ -calculus [MPW83, Milner 90, Milner 92, ...]
- Concurrent Object [Walker 81]
- $\omega$ -order term passing [Sangiorgi 92]
- Various data structures [Milner 92, ...]
- Proof Nets [Bellare and Scott 93]
- Abstract 'constant' interaction [HY84]
- Strategies on Games [HO85]

⋮



# \* Examples of Representable Computation.

Operationally

Sound

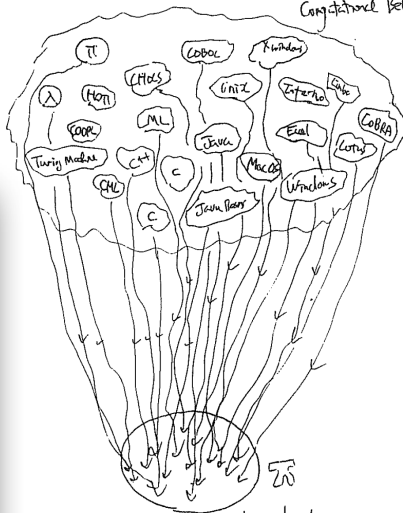
but NOT

Fully Abstract

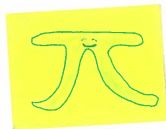
- $\lambda$ -calculus [MPW89, Milner 90, Milner 92, ...]
- Concurrent Object [Walker 81]
- $\omega$ -order term passing [Sangiorgi 92]
- Various data structures [Milner 92, ...]
- Proof Nets [Bellare and Scott 93]
- Abstract 'constant' interaction [HY94]
- Strategies on Games [HO95]

⋮

Realms of Computational Behaviors



The realm of Name Passing Interactions



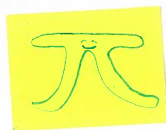
$M \approx N$



$\llbracket M \rrbracket \approx \llbracket N \rrbracket$

Contextual  
Congruence

Contextual  
Congruence



$M \approx N$



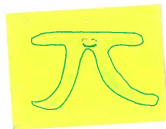
$\llbracket M \rrbracket \approx \llbracket N \rrbracket$

Contextual  
Congruence

Contextual  
Congruence

$C[P] \Downarrow_a \text{ iff } C[Q] \Downarrow_a$

$C[\llbracket M \rrbracket] \Downarrow_a \text{ iff } C[\llbracket N \rrbracket] \Downarrow_a$



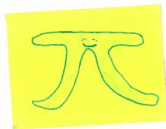
$M \approx N$



$\llbracket M \rrbracket \approx \llbracket N \rrbracket$

$C[P] \Downarrow_a \text{ iff } C[Q] \Downarrow_a$

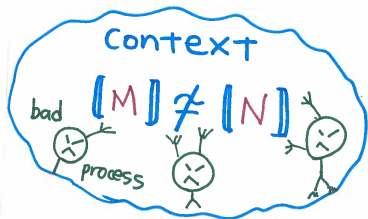
$C[\llbracket M \rrbracket] \Downarrow_a \text{ iff } C[\llbracket N \rrbracket] \Downarrow_a$



$$M \approx N$$

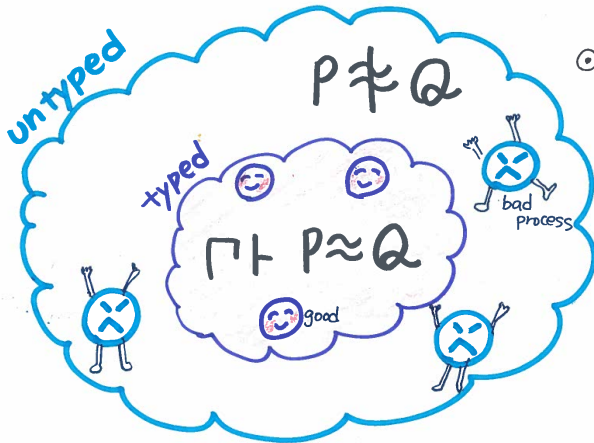


$$[M] \approx [N]$$



# Typed Semantics in $\pi$ 1991 $\rightarrow$

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



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

# \* Examples of Representable Computation.

- $\lambda$ -calculus [MPW89, Milner 90, Milner 92, ...]
- Concurrent Object [Walker 81]
- $\omega$ -order term passing [Sangiorgi 92]
- Various data structures [Milner 92, ...]
- Proof Nets [Bellare and Scott 93]
- Abstract 'constant' interaction [HY94]
- Strategies on Games [HO95]
- ...

Game Semantics



Fully Abstract

The realm of Name Passing Interactions

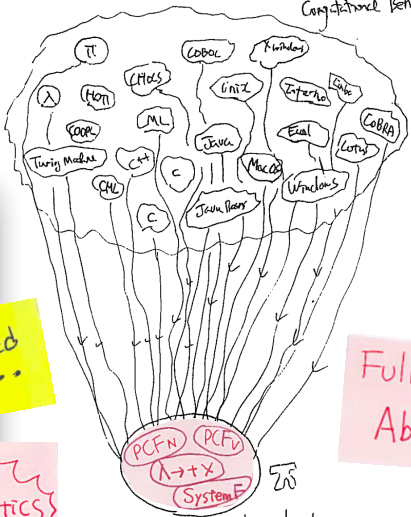
# \* Examples of Representable Computation.

- $\lambda$ -calculus [MPW89, Milner90, Milner92, ...]
- Concurrent Object [Walker81]
- $\omega$ -order term passing [Sangiorgi 92]
- Various data structures [Milner90]
- Proof Nets [Bellare and Scott 93]
- Abstract 'constant' interaction [ ... ]
- Strategies on Games [HOBS]

Complicated  
...

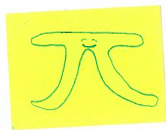
Game Semantics

Realms of Computational Behaviors



The realm of Name Passing Interactions

Fully Abstract



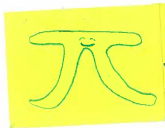
$M \approx N$



$[M] \approx [N]$



System



Session



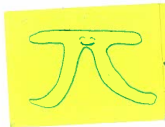
$M \approx N$



$\llbracket M \rrbracket \approx \llbracket N \rrbracket$



System



Session



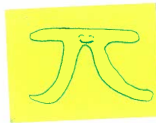
$M \approx N$



$[[M]] \approx [[N]]$



System



Session



$$M \approx N$$

$$[[M]] \approx [[N]]$$

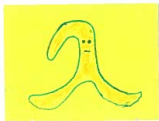
$$[[P]] \approx [[Q]]$$



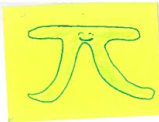
$$P \approx Q$$

Reverse

New



System  
F



Session



$$M \approx N$$

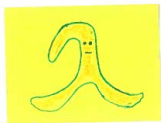
$$\llbracket M \rrbracket \approx \llbracket N \rrbracket$$

$$\llbracket P \rrbracket \approx \llbracket Q \rrbracket$$

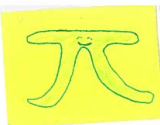


$$P \approx Q$$

Reverse  
New



System  
 $\models$



Session



$$M \approx N$$

$$\llbracket M \rrbracket \approx \llbracket N \rrbracket$$

$$\llbracket P \rrbracket \approx \llbracket Q \rrbracket$$



$$P \approx Q$$

TWO APPLICATIONS

# Session $\pi$ Caires, Pérez, Pfenning, Toninho 13

## Types

$$A ::= \underline{A \otimes B} \mid \underline{A \multimap B} \mid \underline{1} \mid \underline{!A}$$
$$\mid \underline{\forall x. A} \mid \underline{\exists x. A} \mid X$$

## Processes

$$P ::= \underline{x \langle y \rangle. P} \mid \underline{x(y). P} \mid \underline{0} \mid \underline{!x(y). P}$$
$$\mid \underline{x(Y). P} \mid \underline{x \langle B \rangle. P} \mid [x \leftrightarrow y]$$
$$\mid (\nu x) P \mid (P \mid Q)$$

# Judgement

$$X_1, \dots, X_k; a_1: A_1, \dots, a_n: A_n \vdash P :: b: A$$

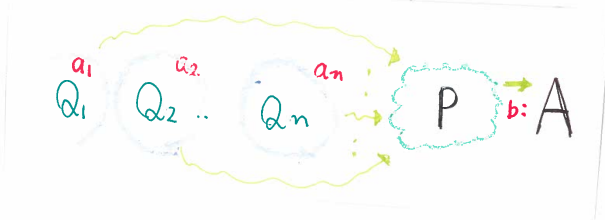
Process

Poly Vars

name Type

name type

process P provides A along b if composed with sessions  $\vec{a}: \vec{A}$



# Judgement

$$\underbrace{X_1, \dots, X_k}_{\text{Poly Vars}} ; \underbrace{a_1: A_1, \dots, a_n: A_n}_{\substack{\uparrow \\ \text{name} \quad \text{Type}}} \vdash \underbrace{P}_{\text{Process}} :: \underbrace{b: A}_{\substack{\uparrow \\ \text{name} \quad \text{type}}}$$

process  $P$  provides  $A$  along  $b$  if composed with sessions  $\vec{a}: \vec{A}$



# Judgement

$$\underbrace{X_1, \dots, X_k}_{\text{Poly Vars}} ; \underbrace{a_1: A_1, \dots, a_m: A_m}_{\substack{\uparrow \text{ name} \\ \swarrow \text{ Type}}} \vdash \underbrace{P}_{\text{Process}} \text{ :: } \underbrace{b: A}_{\substack{\uparrow \text{ name} \\ \swarrow \text{ type}}}$$

## Cut Elimination

$$\frac{\Delta_1 \vdash P_1 \text{ :: } a: A \quad \Delta_2, a: A \vdash P_2 \text{ :: } b: B}{\Delta_1, \Delta_2 \vdash (v a) (P_1 | P_2) \text{ :: } b: B}$$

## Identity

$$a: A \vdash [a \leftrightarrow b] \text{ :: } b: A$$

# Polymorphic Session

$$\forall R \quad \frac{x; \Delta \vdash P :: a:A}{\Delta \vdash a(X).P :: a:\forall X.A} \quad \text{Input}$$

$$\exists R \quad \frac{\Delta \vdash P :: a:A\{B/X\}}{\Delta \vdash a\langle B \rangle.P :: a:\exists X.A} \quad \text{output}$$

Left rules define how to **use** a session of a given type

# Polymorphic Session

Deadlock  
Free  
Live

Strong  
Normalising

$$\forall R \quad \frac{x; \Delta \vdash P :: a:A}{\Delta \vdash a(X).P :: a:\forall X.A}$$

Input

$$\Delta \vdash P :: a:A\{B/X\}$$

$$\Delta \vdash a\langle B \rangle.P :: a:\exists X.A$$

output

$\approx \pi$

Barbed  
Congruence

Left rules define how to **use** a session of a given type

# Linear F

zhao, Zhang, Zdancewicz 2010



Types

$$A ::= A \otimes B \mid A \multimap B \mid !A \mid 1 \mid 2 \\ \mid \forall x. A \mid \exists x. A \mid X$$

Terms

$$M, N ::= \lambda x. M \mid MN \mid \langle M \otimes N \rangle \mid \text{let } x \otimes y = M \text{ in } N \\ \mid !M \mid \text{let } !u = M \text{ in } N \\ \mid \Lambda X. M \mid M[A] \mid \text{pack } A \text{ with } M \mid \text{let } (X, y) = M \\ \text{in } N \\ \mid \text{let } 1 = M \text{ in } N \mid \langle \rangle \mid T \mid F$$

Encoding: from  to  Milner 90 + CPPT'12

From Natural Deduction to Sequent Calculus

Intro  $\Rightarrow$  Right / Elim  $\Rightarrow$  Left + Cut + Identity

Encoding is FUN 

Encoding: from  to  Milner 90 + CPPT'12

From Natural Deduction to Sequent Calculus

Intro  $\Rightarrow$  Right / Elim  $\Rightarrow$  Left + Cut + Identity

$$\boxed{[M]_a}$$

$\uparrow$   
name

$$[x]_a = [x \leftrightarrow a]$$

$$[\langle \rangle]_a = 0$$

$$[\lambda x. M]_a = a(x). [M]_a$$

$$[MN]_a = [M]_x \mid \bar{x}(y). [N]_y \mid [x \leftrightarrow a]$$

# The $\pi$ -calculus as a Descriptive Tool

$\lambda$  in  $\pi$

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

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

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

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

Milner's  
Encoding  
1991

Encoding: from  $\lambda$  to  $\pi$  Milner 90 + CPPT'12

From Natural Deduction to Sequent Calculus

Intro  $\Rightarrow$  Right / Elim  $\Rightarrow$  Left + Cut + Identity

$[M]_a$   
 $\uparrow$   
 name

$$[x]_a = [x \leftrightarrow a]$$

$$[\langle \rangle]_a = 0$$

$$[MN]_a = [M]_x \mid \bar{x}(y). [N]_y \mid [x \leftrightarrow a]$$

$\lambda$  in  $\pi$

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

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

$$[MN]_u \stackrel{\text{def}}{=} (\nu z) ([M]_z \mid \bar{z}(uw) \mid [x=N])$$

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

$$[\lambda x.M]_a = a(x). [M]_a$$

From  to 

From Sequent Calculus to Natural Deduction

$\llbracket P \rrbracket \Delta \vdash a:A$  with  $\Delta \vdash P :: \underline{a:A}$

$\llbracket 0 \rrbracket = \langle \rangle$

$\llbracket [x \leftrightarrow a] \rrbracket = x$

$\llbracket a(x). P \rrbracket = \lambda x. \llbracket P \rrbracket$

$\llbracket a(x). P \rrbracket = \Lambda X. \llbracket P \rrbracket$

Parallel

$$\left[ \frac{\Delta_1 \vdash P :: a:A \quad \Delta_2, a:A \vdash Q :: b:C}{\Delta_1, \Delta_2 \vdash (\nu a) (P \mid Q) :: b:C} \right]$$

Substitute P into a in Q

$$= \frac{\Delta_2, a:A \vdash [Q]_b :: C \quad \Delta_1 \vdash [P]_a :: A}{\Delta_1, \Delta_2 \vdash [Q] \{ [P] / a \} :: C}$$

Poly

$$\left[ x \langle B \rangle. P \right] = \left[ P \right] \{ x[B] / x \}$$

↙ Type

Application of B to x

↑  
replace x by x[B]

$$\text{cf. } \left[ x \langle b \rangle. (P \mid Q) \right] = \left[ Q \right] \{ (x \langle P \rangle) / x \}$$

Application of P to x

# Theorems

Operational Correspondence / Type Preserving

Inverse

$$(\llbracket M \rrbracket_z) \cong M$$

$$\llbracket (P) \rrbracket_z \cong P$$

Full Abstraction

$$\llbracket M \rrbracket_z \cong \llbracket N \rrbracket_z \text{ iff } M \cong N$$

$$(P) \cong (Q) \text{ iff } P \cong Q$$

# Theorems

Operational Correspondence / Type Preserving

Inverse

$$(\llbracket M \rrbracket_z) \cong M$$

$$\llbracket (P) \rrbracket_z \cong P$$

Definability

$$\forall P \exists M \llbracket M \rrbracket_z \cong P$$

$$\forall M \exists P (P) \cong M$$

$$\llbracket M \rrbracket_z \cong P$$

$$(P) \cong M$$

Full Abstraction

$$\llbracket M \rrbracket_z \cong \llbracket N \rrbracket_z \text{ iff } M \cong N$$

$$(P) \cong (Q) \text{ iff } P \cong Q$$

# Theorems

## Operational Correspondence / Type Preserving

### Inverse

$$(\llbracket M \rrbracket_z) \cong M$$



$$\llbracket (P) \rrbracket_z \cong P$$



### Full Abstraction

$$\llbracket M \rrbracket_z \cong \llbracket N \rrbracket_z \text{ iff } M \cong N$$



$$(P) \cong (Q) \text{ iff } P \cong Q$$

Derived

# Theorems

Operational Correspondence / Type Preserving

Inverse

$$(\llbracket M \rrbracket_z) \cong M$$



$$\llbracket (PD) \rrbracket_z \cong P$$



Full Abstraction

$$\llbracket M \rrbracket_z \cong \llbracket N \rrbracket_z$$



$$M \cong N$$

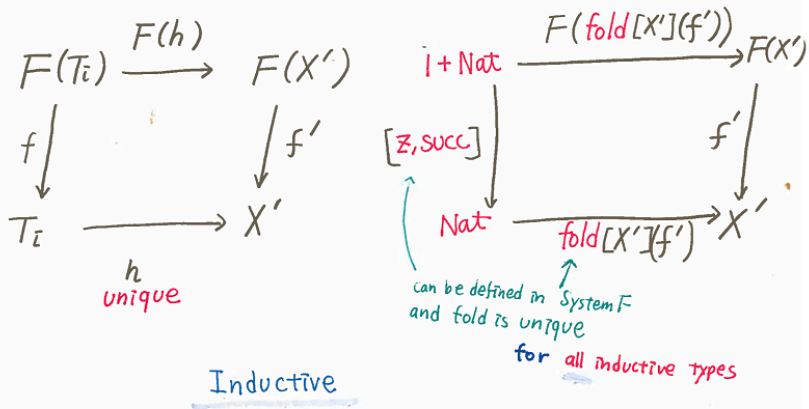
$$(PD) \cong (QD)$$



$$P \cong Q$$

# Application 1 Inductive and Coinductive Types

Parametric Poly is Expressive Enough to Encode Inductive/Co-Inductive Types as Initial / Final (Co)Algebra



## Coinductive Types

$$\begin{array}{ccc} X' & \xrightarrow{h} & T_f \\ f' \downarrow & & \downarrow f \\ F(X') & \xrightarrow{F(h)} & F(T_f) \end{array} \qquad \begin{array}{ccc} X' & \xrightarrow{\text{unfold } [X](f')} & \text{NatStream} \\ f' \downarrow & & \downarrow [\text{hd}, \text{tl}] \\ F(X') & \xrightarrow{F(\text{unfold } [X](f'))} & \text{Nat} \times \text{NatStream} \end{array}$$

Question Sess Poly  $\pi$  is Expressive Enough?

## Theorems

$\forall Q$  s.t.  $u: F(X) \rightarrow X, y_1: T_z \vdash Q :: y_2: X. \text{Fold}(X) \cong Q$

$\forall Q$  s.t.  $u: A \rightarrow F(A), y_1: A \vdash Q :: y_2: T, Q \cong \text{Unfold}(A)$   
A

## Application (2) $HO\pi + PROcess\ Monad$

- Full Abstraction / Inverse  $x \langle M \rangle, P$
- Strong Normalisation via Strong Normalisation

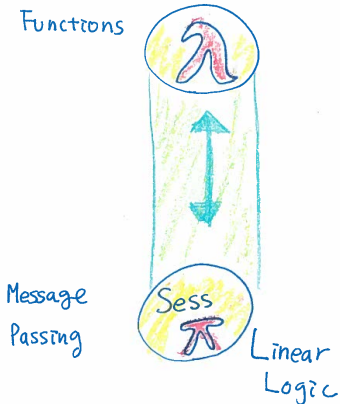
$HO\pi$

$\lambda$

$$P \rightarrow Q \quad \Rightarrow \quad (P) \dashv\dashv (Q)$$

cf. [CPPT'13] logical relation

# Summary



- <sup>LL-based</sup> Use Sess  $\pi$  to articulate boarder computations
- Algebraic Programming (F-algebra)

# Summary

SAD?



Functions



Linear  
Logic

Message  
Passing

LL-based

- Use Sess  $\pi$  to articulate boarder computations
- Algebraic Programming (F-algebra)

# Summary

SAD?



Functions



Message Passing

Linear Logic

NO!



LL-based

- Use Sess  $\pi$  to articulate boarder computations
- Algebraic Programming (F-algebra)