session type

Session Types and Linear Logic

Bernardo Toninho and Nobuko Yoshida

University of Bath,
November 28, 2017

Us ∈ Mobility Research Group

MobilityReadingGroup

π-calculus, Session Types research at Imperial College

Home | People | Publications | Grants | Talks | Tools | Awards | Kohei 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 .

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

Academic Staff

Nobuko Yoshida

Research Associate

Raymond Hu

Julien Lange

Nicholas Ng

Xinyu Niu

Alceste Scalas

Bernardo Toninho

PhD Student

Assel Altayeva

Juliana Franco

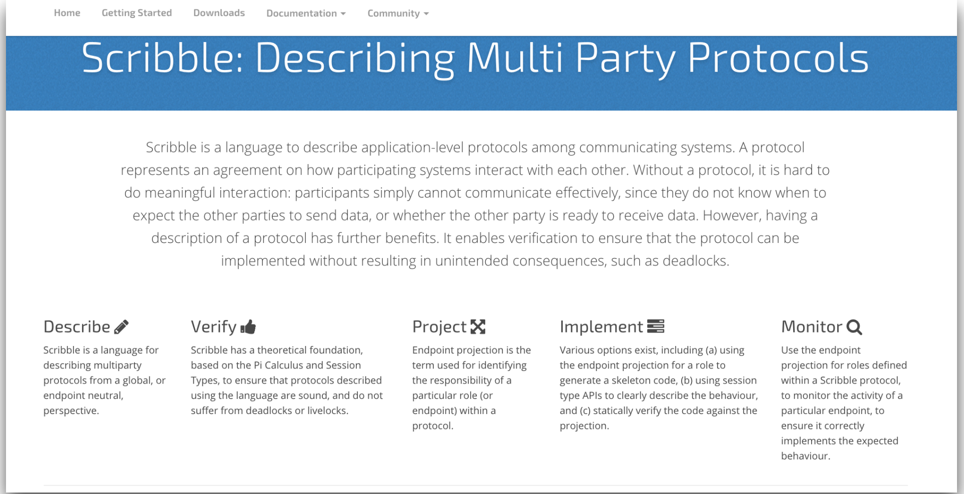
Rumyana Neykova

Weizhen Yang

OOI Collaboration

- **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

www.scribble.org



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

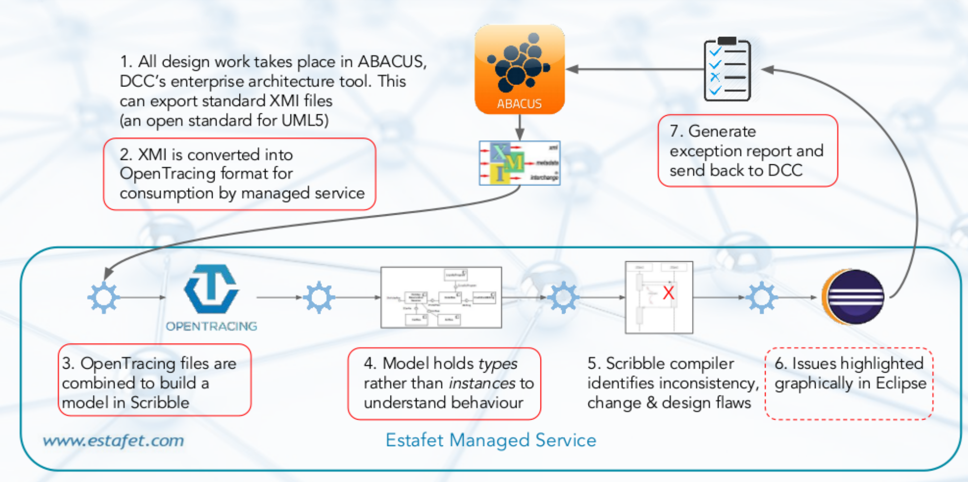
```
module examples;

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

Load a sample | Check | Protocol: examples.HelloWorld | Role: Me | Project | Generate Graph

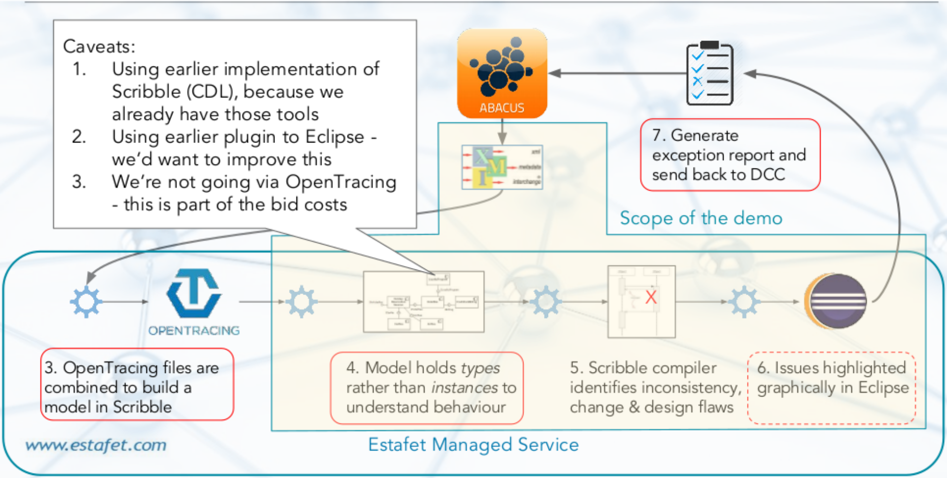
End-to-End Switching Programme by DCC

Estafet
Innovate | Deliver | Transform



End-to-End Switching Programme by DCC

Estafet
Innovate | Deliver | Transform



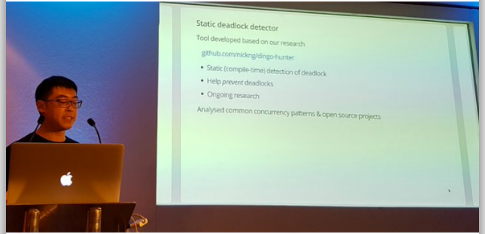
Interactions with Industries

Adam Bowen @adamnbowen · Sep 15
I didn't even know that session types existed an hour ago, but thanks to Nobuko Yoshida's great talk at #pwlconf, I want to learn more.

Nobuko Yoshida
Imperial College, London

DoC researcher to speak at Golang UK conference

by Vicky Kapogianni
20 July 2016



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

Click here to add content

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

Interactions with Industries

F#unctional Londoners Meetup Group

6 days ago · 6:30 PM

Session Types with Fahd Abdeljallal

43 Members

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

Distributed Systems
vs.
Compositionality

Dr. Roland Kuhn
@rolandkuhn — *CTO of Actyx*

actyx

Current State

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

Imperial College London

News

Home | College and Campus | Science | Engineering | Health | Business | Search here... Go

Go concurrency verification research at DoC grabs headline

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

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

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 .

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.

Session Types and Linear Logic

Bernardo Toninho and Nobuko Yoshida

University of Bath,
November 28, 2017

Introduction

Concurrent Processes

- Coordination of multiple simultaneously executing agents.
- Programming Models: Shared-Memory vs Message-Passing
- Hard to reason about:
  - Deadlocks
  - Data races
  - Concurrent interleavings make behaviour hard to predict
  - ...and also hard to replicate (e.g. testing).

Introduction

A Concurrency-Theoretic Approach: Session Types

Structuring Communication

- Communication without structure is hard to reason about.
- Structure communication around the concept of a session.
- Predetermined sequences of interactions along a (session) channel:
  - "Input a number, output a string and terminate."
  - "Either output or input a number."

Introduction

A Concurrency-Theoretic Approach: Session Types

Structuring Communication

- Communication without structure is hard to reason about.
- Structure communication around the concept of a session.
- Predetermined sequences of interactions along a (session) channel:
  - "Input a number, output a string and terminate."
  - "Either output or input a number."

Session-based Structuring

- Specify communication behaviour as sessions.
- Check that programs adhere to specification (session fidelity).

Introduction

A Concurrency-Theoretic Approach: Session Types

Session Types [Honda93]

- Types are descriptions of communication behaviour, assigned to channels.
- A way of guaranteeing communication discipline, statically.
- A linear typing discipline for π-calculus processes.
- Intrinsic notion of duality: Send/Receive, Offer choice/Select

Introduction

A Concurrency-Theoretic Approach: Session Types

Session Types [Honda93]

- Types are descriptions of communication behaviour, assigned to channels.
- A way of guaranteeing communication discipline, statically.
- A linear typing discipline for π-calculus processes.
- Intrinsic notion of duality: Send/Receive, Offer choice/Select

Syntax

$$
\begin{array}{lcl}
P, Q & ::= & x\langle y\rangle.P \mid x(y).P \mid x.\ell; P \mid x.\mathsf{case}\{\ell_i : P_i\}_{i\in I} \mid (\nu x)P \mid \mathbf{0} \mid (P \mid Q) \\
S, T & ::= & !S.T \mid ?S.T \mid \oplus\{\ell_i : T_i\}_{i\in I} \mid \&\{\ell_i : T_i\}_{i\in I}
\end{array}
$$

Introduction

A Concurrency-Theoretic Approach: Session Types

Session Types [Honda93]

- Types are descriptions of communication behaviour, assigned to channels.
- A way of guaranteeing communication discipline, statically.
- A linear typing discipline for π-calculus processes.
- Intrinsic notion of duality: Send/Receive, Offer choice/Select

Syntax

$$
\begin{array}{lcl}
P, Q & ::= & x\langle y\rangle.P \mid x(y).P \mid x.\ell; P \mid x.\mathsf{case}\{\ell_i : P_i\}_{i\in I} \mid (\nu x)P \mid \mathbf{0} \mid (P \mid Q) \\
S, T & ::= & !S.T \mid ?S.T \mid \oplus\{\ell_i : T_i\}_{i\in I} \mid \&\{\ell_i : T_i\}_{i\in I}
\end{array}
$$

Introduction

A Concurrency-Theoretic Approach: Session Types

Session Types [Honda93]

- Types are descriptions of communication behaviour, assigned to channels.
- A way of guaranteeing communication discipline, statically.
- A linear typing discipline for π-calculus processes.
- Intrinsic notion of duality: Send/Receive, Offer choice/Select

Syntax

$$
\begin{array}{lcl}
P, Q & ::= & x\langle y\rangle.P \mid x(y).P \mid x.\ell; P \mid x.\mathsf{case}\{\ell_i : P_i\}_{i\in I} \mid (\nu x)P \mid \mathbf{0} \mid (P \mid Q) \\
S, T & ::= & !S.T \mid ?S.T \mid \oplus\{\ell_i : T_i\}_{i\in I} \mid \&\{\ell_i : T_i\}_{i\in I}
\end{array}
$$

Introduction

A Concurrency-Theoretic Approach: Session Types

Session Types [Honda93]

- Types are descriptions of communication behaviour, assigned to channels.
- A way of guaranteeing communication discipline, statically.
- A linear typing discipline for π-calculus processes.
- Intrinsic notion of duality: Send/Receive, Offer choice/Select

Syntax

$$
\begin{array}{rcl}
P, Q & ::= & x\langle y\rangle.P \mid x(y).P \mid x.\ell; P \mid x.\mathsf{case}\{\ell_i : P_i\}_{i\in I} \mid (\nu x)P \mid \mathbf{0} \mid (P \mid Q) \\
S, T & ::= & !S.T \mid ?S.T \mid \oplus\{\ell_i : T_i\}_{i\in I} \mid \&\{\ell_i : T_i\}_{i\in I}
\end{array}
$$

Session Types [Honda93]

- Types are descriptions of communication behaviour, assigned to channels.
- A way of guaranteeing communication discipline, statically.
- A linear typing discipline for π-calculus processes.
- Intrinsic notion of duality: Send/Receive, Offer choice/Select

Syntax

$$
\begin{array}{lcl}
P, Q & ::= & x\langle y\rangle.P \mid x(y).P \mid x.\ell; P \mid x.\mathsf{case}\{\ell_i : P_i\}_{i\in I} \mid (\nu x)P \mid \mathbf{0} \mid (P \mid Q) \\
S, T & ::= & !S.T \mid ?S.T \mid \oplus\{\ell_i : T_i\}_{i\in I} \mid \&\{\ell_i : T_i\}_{i\in I}
\end{array}
$$

A pair of processes interacting on dual sessions is deadlock-free!

Introduction

Session Types and Linear Logic

Linear Logic [Girard87]

- A marriage of classical dualities and constructivism.
- A logic of resources and interaction.
- Resource independence captures non-determinism/concurrency.
- Connected to concurrency early on [Abramsky93,BellinScott94]

Introduction

Session Types and Linear Logic

Linear Logic [Girard87]

- A marriage of classical dualities and constructivism.
- A logic of resources and interaction.
- Resource independence captures non-determinism/concurrency.
- Connected to concurrency early on [Abramsky93,BellinScott94]

Session Types and ILL [CairesPfenning01]

- Its possible to interpret session types as linear logic propositions.
- Linear logic proofs as process typing derivations.
- Proof dynamics as process dynamics.

Why does it matter?

- Good metalogical properties map to good program properties.
- Progress, session fidelity, type preservation “for free”.
- Reasoning built-in.
- Much more compositional than traditional approaches (i.e. extensibility).

Introduction

Roadmap

1. Basic interpretation
2. Enriching the type structure (dependent types and polymorphism)
3. Reasoning Techniques: Logical equivalence, type isomorphisms and proof conversions.

ILL Interpretation

Key Ideas

- Session Types as Intuitionistic Linear Propositions.
- Sequent calculus rules as π-calculus typing rules.
- Cut reduction as process reduction.

ILL Interpretation

Key Ideas

- Session Types as Intuitionistic Linear Propositions.
- Sequent calculus rules as π-calculus typing rules.
- Cut reduction as process reduction.

Propositions

$$A, B \quad ::= \quad A \otimes B \mid A \multimap B \mid \mathbf{1} \mid A \,\&\, B \mid A \oplus B \mid\, !A$$

ILL Interpretation

Key Ideas

- Session Types as Intuitionistic Linear Propositions.
- Sequent calculus rules as π-calculus typing rules.
- Cut reduction as process reduction.

Propositions

$$A, B \quad ::= \quad A \otimes B \mid A \multimap B \mid \mathbf{1} \mid A \mathbin{\&} B \mid A \oplus B \mid !A$$

Sequents

- Duality of offering (right rules) and using (left rules) a session.
- Proof composition (cut) as process composition.
- Identity as forwarding/renaming.

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{A_1, \ldots, \quad A_m}^{\Gamma} ; \overbrace{A_1, \ldots, \quad A_n}^{\Delta} \Longrightarrow \quad A$$

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

Cut as Composition

$$\frac{\Gamma; \Delta \Longrightarrow \qquad A \quad \Gamma; \Delta', \quad A \Longrightarrow \qquad C}{\Gamma; \Delta, \Delta' \Longrightarrow \qquad C}\ \text{cut}$$

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

Cut as Composition

$$\frac{\Gamma; \Delta \Longrightarrow {\color{red}P :: x}{:}A \quad \Gamma; \Delta', \quad A \Longrightarrow \quad C}{\Gamma; \Delta, \Delta' \Longrightarrow \quad C}\ \text{cut}$$

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

Cut as Composition

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A \quad \Gamma; \Delta', x{:}A \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, \Delta' \Longrightarrow \qquad C}\ \text{cut}$$

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

Cut as Composition

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A \quad \Gamma; \Delta', x{:}A \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu x)(P \mid Q) :: z{:}C}\ \text{cut}$$

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

Cut as Composition

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A \quad \Gamma; \Delta', x{:}A \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu x)(P \mid Q) :: z{:}C}\ \text{cut}$$

Parallel composition of P, offering $x{:}A$ and Q, using $x{:}A$.

Identity as Renaming

$$\frac{}{\Gamma; \quad A \Longrightarrow \qquad\qquad A}$$

ILL Interpretation

Judgmental Principles

Typing Judgment

$$\overbrace{u_1{:}A_1, \ldots, u_m{:}A_m}^{\Gamma}; \overbrace{x_1{:}A_1, \ldots, x_n{:}A_n}^{\Delta} \Longrightarrow P :: x{:}A$$

Process P provides A along x if composed with sessions in Δ and Γ.

Identity as Renaming

$$\frac{}{\Gamma; x{:}A \Longrightarrow [x \leftrightarrow z] :: z{:}A}$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow A \quad \Gamma; \Delta' \Longrightarrow B}{\Gamma; \Delta, \Delta' \Longrightarrow A \otimes B} \otimes R$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow \qquad\qquad A \otimes B} \otimes R$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: x{:}A \otimes B} \otimes R$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: x{:}A \otimes B} \otimes R$$

$$\frac{\Gamma; \Delta, \quad A, \quad B \Longrightarrow \quad C}{\Gamma; \Delta, \quad A \otimes B \Longrightarrow \quad C} \otimes L$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: x{:}A \otimes B} \otimes R$$

$$\frac{\Gamma; \Delta, y : A, x : B \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, \quad A \otimes B \Longrightarrow \qquad\qquad C} \otimes L$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: x{:}A \otimes B} \otimes R$$

$$\frac{\Gamma; \Delta, y : A, x : B \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}A \otimes B \Longrightarrow x(y).Q :: z{:}C} \otimes L$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: x{:}A \otimes B} \otimes R$$

$$\frac{\Gamma; \Delta, y : A, x : B \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}A \otimes B \Longrightarrow x(y).Q :: z{:}C} \otimes L$$

Proof Reduction

$$\Gamma; \Delta, \Delta' \Longrightarrow (\nu x)((\nu y)x\langle y\rangle.(P_1 \mid P_2) \mid x(y).Q) :: z{:}C$$

ILL Interpretation

Propositions

Multiplicative Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: y{:}A \quad \Gamma; \Delta' \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta, \Delta' \Longrightarrow (\nu y)x\langle y\rangle.(P_1 \mid P_2) :: x{:}A \otimes B} \otimes R$$

$$\frac{\Gamma; \Delta, y : A, x : B \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}A \otimes B \Longrightarrow x(y).Q :: z{:}C} \otimes L$$

Proof Reduction

$$\Gamma; \Delta, \Delta' \Longrightarrow (\nu x)((\nu y)x\langle y\rangle.(P_1 \mid P_2) \mid x(y).Q) :: z{:}C$$

$$\longrightarrow \quad \Gamma; \Delta, \Delta' \Longrightarrow (\nu x)(\nu y)(P_1 \mid P_2 \mid Q) :: z{:}C$$

ILL Interpretation

Propositions

Linear Implication

$$\frac{\Gamma; \Delta, \quad A \Longrightarrow \qquad B}{\Gamma; \Delta \Longrightarrow \qquad A \multimap B} \multimap R$$

ILL Interpretation

Propositions

Linear Implication

$$\frac{\Gamma; \Delta, y : A \Longrightarrow P :: x{:}B}{\Gamma; \Delta \Longrightarrow \qquad\qquad A \multimap B} \multimap R$$

ILL Interpretation

Propositions

Linear Implication

$$\frac{\Gamma; \Delta, y : A \Longrightarrow P :: x{:}B}{\Gamma; \Delta \Longrightarrow x(y).P :: x{:}A \multimap B} \multimap R$$

ILL Interpretation

Propositions

Linear Implication

$$\frac{\Gamma; \Delta, y : A \Longrightarrow P :: x{:}B}{\Gamma; \Delta \Longrightarrow x(y).P :: x{:}A \multimap B} \multimap R$$

$$\frac{\Gamma; \Delta \Longrightarrow \qquad A \quad \Gamma; \Delta', \quad B \Longrightarrow \qquad C}{\Gamma; \Delta, \Delta', \quad A \multimap B \Longrightarrow \qquad C} \multimap L$$

ILL Interpretation
Propositions

Linear Implication

$$\frac{\Gamma; \Delta, y : A \Longrightarrow P :: x{:}B}{\Gamma; \Delta \Longrightarrow x(y).P :: x{:}A \multimap B} \multimap R$$

$$\frac{\Gamma; \Delta \Longrightarrow Q_1 :: y{:}A \quad \Gamma; \Delta', x{:}B \Longrightarrow Q_2 :: z{:}C}{\Gamma; \Delta, \Delta', \quad A \multimap B \Longrightarrow \qquad C} \multimap L$$

ILL Interpretation

Propositions

Linear Implication

$$\frac{\Gamma; \Delta, y : A \Longrightarrow P :: x{:}B}{\Gamma; \Delta \Longrightarrow x(y).P :: x{:}A \multimap B} \multimap R$$

$$\frac{\Gamma; \Delta \Longrightarrow Q_1 :: y{:}A \quad \Gamma; \Delta', x{:}B \Longrightarrow Q_2 :: z{:}C}{\Gamma; \Delta, \Delta', x{:}A \multimap B \Longrightarrow (\nu y)x\langle y\rangle.(Q_1 \mid Q_2) :: z{:}C} \multimap L$$

ILL Interpretation
Propositions

Linear Implication

$$\frac{\Gamma; \Delta, y : A \Longrightarrow P :: x{:}B}{\Gamma; \Delta \Longrightarrow x(y).P :: x{:}A \multimap B} \multimap R$$

$$\frac{\Gamma; \Delta \Longrightarrow Q_1 :: y{:}A \quad \Gamma; \Delta', x{:}B \Longrightarrow Q_2 :: z{:}C}{\Gamma; \Delta, \Delta', x{:}A \multimap B \Longrightarrow (\nu y)x\langle y\rangle.(Q_1 \mid Q_2) :: z{:}C} \multimap L$$

Linear Implication as input. Reduction is the same as for $\otimes$.

ILL Interpretation

Propositions

Multiplicative Unit

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{1}}\ \mathbf{1}R$$

ILL Interpretation

Propositions

Multiplicative Unit

$$\dfrac{}{\Gamma; \cdot \Longrightarrow \mathbf{0} :: x{:}\mathbf{1}}\ \mathbf{1}R$$

ILL Interpretation

Propositions

Multiplicative Unit

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{0} :: x{:}\mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma; \Delta \Longrightarrow \qquad C}{\Gamma; \Delta, \quad \mathbf{1} \Longrightarrow \qquad C}\ \mathbf{1}L$$

ILL Interpretation

Propositions

Multiplicative Unit

$$\frac{}{\Gamma ; \cdot \Longrightarrow \mathbf{0} :: x{:}\mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma ; \Delta \Longrightarrow Q :: z{:}C}{\Gamma ; \Delta, \quad \mathbf{1} \Longrightarrow \qquad C}\ \mathbf{1}L$$

ILL Interpretation

Propositions

Multiplicative Unit

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{0} :: x{:}\mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}\mathbf{1} \Longrightarrow Q :: z{:}C}\ \mathbf{1}L$$

Proof Reduction

$$\Gamma; \Delta \Longrightarrow (\nu x)(\mathbf{0} \mid Q) :: z{:}C$$

ILL Interpretation

Propositions

Multiplicative Unit

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{0} :: x{:}\mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}\mathbf{1} \Longrightarrow Q :: z{:}C}\ \mathbf{1}L$$

Proof Reduction

$$\begin{array}{rl} & \Gamma; \Delta \Longrightarrow (\nu x)(\mathbf{0} \mid Q) :: z{:}C \\ \equiv & \Gamma; \Delta \Longrightarrow Q :: z{:}C \end{array}$$

ILL Interpretation

Propositions

Multiplicative Unit

$$\frac{}{\Gamma; \cdot \Longrightarrow \mathbf{0} :: x{:}\mathbf{1}}\ \mathbf{1}R \qquad \frac{\Gamma; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}\mathbf{1} \Longrightarrow Q :: z{:}C}\ \mathbf{1}L$$

Proof Reduction

$$\begin{array}{rl} & \Gamma; \Delta \Longrightarrow (\nu x)(\mathbf{0} \mid Q) :: z{:}C \\ \equiv & \Gamma; \Delta \Longrightarrow Q :: z{:}C \end{array}$$

Multiplicative Unit as Termination

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow A \quad \Gamma; \Delta \Longrightarrow B}{\Gamma; \Delta \Longrightarrow A \mathbin{\&} B} \mathbin{\&}R$$

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: x{:}A \quad \Gamma; \Delta \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta \Longrightarrow \qquad\qquad\qquad\qquad A \,\&\, B} \;\&R$$

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: x{:}A \quad \Gamma; \Delta \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta \Longrightarrow x.\mathsf{case}(P_1, P_2) :: x{:}A \mathbin{\&} B} \;\& R$$

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: x{:}A \quad \Gamma; \Delta \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta \Longrightarrow x.\mathsf{case}(P_1, P_2) :: x{:}A \mathbin{\&} B} \;\&R$$

$$\frac{\Gamma; \Delta, \quad A \Longrightarrow \qquad C}{\Gamma; \Delta, \quad A \mathbin{\&} B \Longrightarrow \qquad C} \;\&L_1$$

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: x{:}A \quad \Gamma; \Delta \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta \Longrightarrow x.\mathsf{case}(P_1, P_2) :: x{:}A \mathbin{\&} B} \;\&R$$

$$\frac{\Gamma; \Delta, x{:}A \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, \quad A \mathbin{\&} B \Longrightarrow \qquad C} \;\&L_1$$

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: x{:}A \quad \Gamma; \Delta \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta \Longrightarrow x.\mathsf{case}(P_1, P_2) :: x{:}A \mathbin{\&} B} \ \&R$$

$$\frac{\Gamma; \Delta, x{:}A \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}A \mathbin{\&} B \Longrightarrow x.\mathsf{inl}; Q :: z{:}C} \ \&L_1$$

ILL Interpretation

Propositions

Additive Conjunction

$$\frac{\Gamma; \Delta \Longrightarrow P_1 :: x{:}A \quad \Gamma; \Delta \Longrightarrow P_2 :: x{:}B}{\Gamma; \Delta \Longrightarrow x.\mathsf{case}(P_1, P_2) :: x{:}A \mathbin{\&} B} \;\&R$$

$$\frac{\Gamma; \Delta, x{:}A \Longrightarrow Q :: z{:}C}{\Gamma; \Delta, x{:}A \mathbin{\&} B \Longrightarrow x.\mathsf{inl}; Q :: z{:}C} \;\&L_1$$

Proof Reduction

$$\Gamma; \Delta, \Delta' \Longrightarrow (\nu x)(x.\mathsf{case}(P_1, P_2) \mid x.\mathsf{inl}; Q) :: z{:}C$$

$$\longrightarrow \Gamma; \Delta, \Delta' \Longrightarrow (\nu x)(P_1 \mid Q) :: z{:}C$$

ILL Interpretation

Propositions

Additive Disjunction

$$\frac{\Gamma; \Delta \Longrightarrow A}{\Gamma; \Delta \Longrightarrow A \oplus B} \oplus R_1$$

ILL Interpretation
Propositions

Additive Disjunction

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A}{\Gamma; \Delta \Longrightarrow \qquad A \oplus B} \oplus R_1$$

ILL Interpretation

Propositions

Additive Disjunction

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A}{\Gamma; \Delta \Longrightarrow x.\mathsf{inl}; P :: x{:}A \oplus B} \oplus R_1$$

ILL Interpretation

Propositions

Additive Disjunction

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A}{\Gamma; \Delta \Longrightarrow x.\mathsf{inl}; P :: x{:}A \oplus B} \oplus R_1$$

$$\frac{\Gamma; \Delta, \quad A \Longrightarrow \qquad C \quad \Gamma; \Delta, \quad B \Longrightarrow \qquad C}{\Gamma; \Delta, \quad A \oplus B \Longrightarrow \qquad C} \oplus L$$

ILL Interpretation

Propositions

Additive Disjunction

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A}{\Gamma; \Delta \Longrightarrow x.\mathsf{inl}; P :: x{:}A \oplus B} \oplus R_1$$

$$\frac{\Gamma; \Delta, x{:}A \Longrightarrow Q_1 :: z{:}C \quad \Gamma; \Delta, x{:}B \Longrightarrow Q_2 :: z{:}C}{\Gamma; \Delta, \quad A \oplus B \Longrightarrow \qquad C} \oplus L$$

ILL Interpretation

Propositions

Additive Disjunction

$$\frac{\Gamma; \Delta \Longrightarrow P :: x{:}A}{\Gamma; \Delta \Longrightarrow x.\mathsf{inl}; P :: x{:}A \oplus B} \oplus R_1$$

$$\frac{\Gamma; \Delta, x{:}A \Longrightarrow Q_1 :: z{:}C \quad \Gamma; \Delta, x{:}B \Longrightarrow Q_2 :: z{:}C}{\Gamma; \Delta, x{:}A \oplus B \Longrightarrow x.\mathsf{case}(Q_1, Q_2) :: z{:}C} \oplus L$$

Same proof reductions as $\&$.

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow A \qquad \Gamma, A; \Delta \Longrightarrow C}{\Gamma; \Delta \Longrightarrow C}\ \text{cut}^!$$

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow \qquad\qquad C}\ \mathrm{cut}^{!}$$

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid Q) :: z{:}C}\ \mathsf{cut}^{!}$$

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid Q) :: z{:}C}\ \mathsf{cut}^{!}$$

Parallel composition of P, offering $x{:}A$ and Q, using $u{:}A$ persistently.

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid Q) :: z{:}C}\ \mathsf{cut}^!$$

Parallel composition of P, offering $x{:}A$ and Q, using $u{:}A$ persistently.

Copy

$$\frac{\Gamma, \quad A; \Delta, \quad A \Longrightarrow \qquad C}{\Gamma, \quad A; \Delta \Longrightarrow \qquad C}\ \mathsf{copy}$$

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid Q) :: z{:}C}\ \text{cut}^!$$

Parallel composition of P, offering $x{:}A$ and Q, using $u{:}A$ persistently.

Copy

$$\frac{\Gamma, {\color{red}u}{:}A; \Delta, {\color{red}x}{:}A \Longrightarrow {\color{red}P} :: {\color{red}z}{:}C}{\Gamma, \quad A; \Delta \Longrightarrow \qquad\qquad\qquad\qquad C}\ \text{copy}$$

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid Q) :: z{:}C}\ \mathsf{cut}^!$$

Parallel composition of P, offering $x{:}A$ and Q, using $u{:}A$ persistently.

Copy

$$\frac{\Gamma, u{:}A; \Delta, x{:}A \Longrightarrow P :: z{:}C}{\Gamma, u{:}A; \Delta \Longrightarrow (\nu x)u\langle x\rangle.Q :: z{:}C}\ \mathsf{copy}$$

ILL Interpretation

Judgmental Principles for Exponential

Persistent Cut

$$\frac{\Gamma; \cdot \Longrightarrow P :: x{:}A \qquad \Gamma, u{:}A; \Delta \Longrightarrow Q :: z{:}C}{\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid Q) :: z{:}C}\ \text{cut}^!$$

Parallel composition of P, offering $x{:}A$ and Q, using $u{:}A$ persistently.

Copy

$$\frac{\Gamma, u{:}A; \Delta, x{:}A \Longrightarrow P :: z{:}C}{\Gamma, u{:}A; \Delta \Longrightarrow (\nu x)u\langle x\rangle.Q :: z{:}C}\ \text{copy}$$

Proof Reduction

$$\Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid (\nu x)u\langle x\rangle.Q) :: z{:}C$$

$$\longrightarrow \Gamma; \Delta \Longrightarrow (\nu u)(!u(x).P \mid (\nu x)(P \mid Q)) :: z{:}C$$

ILL Interpretation

Propositions

Exponential

$$\frac{\Gamma ; \cdot \Longrightarrow A}{\Gamma ; \cdot \Longrightarrow\ !A}\ !R$$

ILL Interpretation

Propositions

Exponential

$$\frac{\Gamma; \cdot \Longrightarrow P :: y{:}A}{\Gamma; \cdot \Longrightarrow \qquad\qquad !A}\ !R$$

ILL Interpretation

Propositions

Exponential

$$\frac{\Gamma; \cdot \Longrightarrow P :: y{:}A}{\Gamma; \cdot \Longrightarrow !x(y).P :: x{:}!A}\ !R$$

ILL Interpretation

Propositions

Exponential

$$\frac{\Gamma; \cdot \Longrightarrow P :: y{:}A}{\Gamma; \cdot \Longrightarrow !x(y).P :: x{:}!A}\ !R \qquad \frac{\Gamma, \quad A; \Delta \Longrightarrow \qquad C}{\Gamma; \Delta, \quad !A \Longrightarrow \qquad C}\ !L$$

ILL Interpretation

Propositions

Exponential

$$\frac{\Gamma; \cdot \Longrightarrow P :: y{:}A}{\Gamma; \cdot \Longrightarrow !x(y).P :: x{:}!A}\ !R \qquad \frac{\Gamma, u{:}A; \Delta \Longrightarrow P :: z{:}C}{\Gamma; \Delta, \quad !A \Longrightarrow \qquad C}\ !L$$

ILL Interpretation

Propositions

Exponential

$$\frac{\Gamma; \cdot \Longrightarrow P :: y{:}A}{\Gamma; \cdot \Longrightarrow\ !x(y).P :: x{:}!A}\ !R \qquad \frac{\Gamma, u{:}A; \Delta \Longrightarrow P :: z{:}C}{\Gamma; \Delta, x{:}!A \Longrightarrow P\{x/u\} :: z{:}C}\ !L$$

Proof reduction transforms a cut into a cut$^!$ (struct. equivalence).

ILL Interpretation

Metatheory

Operational Correspondence and Subject Reduction

If $\Gamma; \Delta \Longrightarrow P :: z{:}A$ and $P \rightarrow P'$ then $\exists Q$ such that $\Gamma; \Delta \Longrightarrow Q :: z{:}A$ and $P' \equiv Q$.

ILL Interpretation

Metatheory

Operational Correspondence and Subject Reduction

If $\Gamma; \Delta \Longrightarrow P :: z{:}A$ and $P \rightarrow P'$ then $\exists Q$ such that $\Gamma; \Delta \Longrightarrow Q :: z{:}A$ and $P' \equiv Q$.

Global Progress

$$live(P) \triangleq (\nu \overline{x})(Q \mid R) \text{ with } Q \equiv \pi.Q' \text{ or } Q \equiv [x \leftrightarrow y]$$

If $\emptyset \Longrightarrow P :: x{:}\mathbf{1}$ and $live(P)$ then $\exists Q$ such that $P \rightarrow Q$.

Much stronger property than in classical session types, “for free”!

ILL Interpretation

Summary

- Linear Propositions as Session Types.
- Intuitionistic proofs as session-typed processes.
- Process reduction maps to proof conversion.

ILL Interpretation

Summary

- Linear Propositions as Session Types.
- Intuitionistic proofs as session-typed processes.
- Process reduction maps to proof conversion.
- . . . but not all proof conversions are process reductions!

Richer Type Theories

Session Types

- Only express simple communication patterns.
- No interesting properties of exchanged data.
- No sophisticated properties of processes.

Richer Type Theories

Motivation

Session Types

- Only express simple communication patterns.
- No interesting properties of exchanged data.
- No sophisticated properties of processes.

Answers from Logic

- Enrich the logic/types: Quantifiers, Modalities
- Dependent Session Types [Toninho et al.11]
- Polymorphic Session Types [Pérez et al.11, Wadler11]
- Internalisation in a linear monad [Toninho et al.12]

Richer Type Theories

Where are we?

Dependent Session Types [Toninho et al.11, Pfenning et al.11]

- Two new types: $\forall x{:}\tau.A$ and $\exists x{:}\tau.A$
- Parametric in the language of types τ.

Richer Type Theories

Where are we?

Dependent Session Types [Toninho et al.11, Pfenning et al.11]

- Two new types: $\forall x{:}\tau.A$ and $\exists x{:}\tau.A$
- Parametric in the language of types τ.
- $\forall x{:}\tau.A$ - Input a term $M : \tau$, continue as $A(M)$.
- $\exists x{:}\tau.A$ - Output a term $M : \tau$, continue as $A(M)$.

Richer Type Theories

Where are we?

Dependent Session Types [Toninho et al.11, Pfenning et al.11]

- Two new types: $\forall x{:}\tau.A$ and $\exists x{:}\tau.A$
- Parametric in the language of types τ.
- $\forall x{:}\tau.A$ - Input a term $M : \tau$, continue as $A(M)$.
- $\exists x{:}\tau.A$ - Output a term $M : \tau$, continue as $A(M)$.
- If τs are dependent: proof communication.
- With affirmation and proof irrelevance: proof certificates.

Richer Type Theories

Where are we?

Dependent Session Types [Toninho et al.11, Pfenning et al.11]

- Two new types: $\forall x{:}\tau.A$ and $\exists x{:}\tau.A$
- Parametric in the language of types τ.
- $\forall x{:}\tau.A$ - Input a term $M : \tau$, continue as $A(M)$.
- $\exists x{:}\tau.A$ - Output a term $M : \tau$, continue as $A(M)$.
- If τs are dependent: proof communication.
- With affirmation and proof irrelevance: proof certificates.

$$\mathsf{indexer}_{simple} \triangleq \; !(\mathsf{file} \multimap \mathsf{file} \otimes \mathbf{1})$$

Richer Type Theories

Where are we?

Dependent Session Types [Toninho et al.11, Pfenning et al.11]

- Two new types: $\forall x{:}\tau.A$ and $\exists x{:}\tau.A$
- Parametric in the language of types τ.
- $\forall x{:}\tau.A$ - Input a term $M : \tau$, continue as $A(M)$.
- $\exists x{:}\tau.A$ - Output a term $M : \tau$, continue as $A(M)$.
- If τs are dependent: proof communication.
- With affirmation and proof irrelevance: proof certificates.

$$\begin{array}{lcl} \mathsf{indexer}_{simple} & \triangleq & !(\mathsf{file} \multimap \mathsf{file} \otimes \mathbf{1}) \\ \mathsf{indexer}_{dspec} & \triangleq & !(\forall f{:}\mathsf{file}.\mathsf{pdf}(f) \multimap \exists g{:}\mathsf{file}.\mathsf{pdf}(g) \otimes \mathsf{agree}(f,g) \otimes \mathbf{1}) \end{array}$$

Richer Type Theories

Where are we?

Dependent Session Types [Toninho et al.11, Pfenning et al.11]

- Two new types: $\forall x{:}\tau.A$ and $\exists x{:}\tau.A$
- Parametric in the language of types τ.
- $\forall x{:}\tau.A$ - Input a term $M : \tau$, continue as $A(M)$.
- $\exists x{:}\tau.A$ - Output a term $M : \tau$, continue as $A(M)$.
- If τs are dependent: proof communication.
- With affirmation and proof irrelevance: proof certificates.

$$
\begin{array}{lcl}
\mathrm{indexer}_{simple} & \triangleq & !(\mathrm{file} \multimap \mathrm{file} \otimes \mathbf{1}) \\
\mathrm{indexer}_{dspec} & \triangleq & !(\forall f{:}\mathrm{file}.\mathrm{pdf}(f) \multimap \exists g{:}\mathrm{file}.\mathrm{pdf}(g) \otimes \mathrm{agree}(f,g) \otimes \mathbf{1}) \\
\mathrm{indexer}_{irrev} & \triangleq & !(\forall f{:}\mathrm{file}.[\mathrm{pdf}(f)] \multimap \exists g{:}\mathrm{file}.[\mathrm{pdf}(g)] \otimes [\mathrm{agree}(f,g)] \otimes \mathbf{1})
\end{array}
$$

Richer Type Theories

Where are we?

Polymorphism and Parametricity [Pérez et al.11, Wadler11]

- Second-order quantification ($\forall X.A$ and $\exists X.A$).
- Communication of session types / abstract protocols.
- Relational parametricity results in the style of System F.

Richer Type Theories

Where are we?

Polymorphism and Parametricity [Pérez et al.11, Wadler11]

- Second-order quantification ($\forall X.A$ and $\exists X.A$).
- Communication of session types / abstract protocols.
- Relational parametricity results in the style of System F.

Monadic Integration [Toninho et al.12]

- $M : \{\Gamma; \Delta \vdash z{:}A\}$, functional type of an open process P (construct $\{P\}$).
- Process construct bind$(M, z.Q)$:
  - Evaluate M to a (suspended) process $\{P\}$.
  - Run underlying process P in parallel with Q.

Richer Type Theories

Where are we?

Polymorphism and Parametricity [Pérez et al.11, Wadler11]

- Second-order quantification ($\forall X.A$ and $\exists X.A$).
- Communication of session types / abstract protocols.
- Relational parametricity results in the style of System F.

Monadic Integration [Toninho et al.12]

- $M : \{\Gamma; \Delta \vdash z{:}A\}$, functional type of an open process P (construct $\{P\}$).
- Process construct bind$(M, z.Q)$:
  - Evaluate M to a (suspended) process $\{P\}$.
  - Run underlying process P in parallel with Q.
- Higher-Order Sessions, e.g.: $\forall x{:}\{z{:}A\}.\forall y{:}\{z{:}B\}.A \otimes B$

Richer Type Theories

Where are we?

Monadic Dependencies (Ongoing work)

By adding the (contextual) monad to a dependent type theory we get:

- Relations on processes as type families (e.g. $\Pi x{:}\{z{:}A\}.\Pi y{:}\{z{:}B\}.\mathsf{type}$)

Richer Type Theories

Where are we?

Monadic Dependencies (Ongoing work)

By adding the (contextual) monad to a dependent type theory we get:

- Relations on processes as type families (e.g. $\Pi x{:}\{z{:}A\}.\Pi y{:}\{z{:}B\}.\mathsf{type}$)
- Value dependent communication (e.g. $\forall x{:}\mathsf{Nat}.\mathsf{if}\ x = 5\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2$)

Richer Type Theories

Where are we?

Monadic Dependencies (Ongoing work)

By adding the (contextual) monad to a dependent type theory we get:

- Relations on processes as type families (e.g. $\Pi x{:}\{z{:}A\}.\Pi y{:}\{z{:}B\}.\mathsf{type}$)
- Value dependent communication (e.g. $\forall x{:}\mathsf{Nat}.\mathsf{if}\ x = 5\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2$)
- Indexed session types: $\mathsf{countDown} :: \Pi x{:}\mathsf{Nat}.\mathsf{stype}$

Richer Type Theories

Where are we?

Monadic Dependencies (Ongoing work)

By adding the (contextual) monad to a dependent type theory we get:

- Relations on processes as type families (e.g. $\Pi x{:}\{z{:}A\}.\Pi y{:}\{z{:}B\}.\mathsf{type}$)
- Value dependent communication (e.g. $\forall x{:}\mathsf{Nat}.\mathsf{if}\ x = 5\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2$)
- Indexed session types: countDown :: $\Pi x{:}\mathsf{Nat}.\mathsf{stype}$
- Ability to specify and prove properties of processes within the theory:

$$
\begin{array}{rcl}
\mathsf{simpleCounter} & :: & \mathsf{Nat} \rightarrow \{\mathsf{simpleCounter}\} \\
\mathsf{simpleCounter} & = & \lambda x{:}\mathsf{Nat}.\{\ldots\}
\end{array}
$$

Richer Type Theories

Where are we?

Monadic Dependencies (Ongoing work)

By adding the (contextual) monad to a dependent type theory we get:

- Relations on processes as type families (e.g. $\Pi x{:}\{z{:}A\}.\Pi y{:}\{z{:}B\}.\mathrm{type}$)
- Value dependent communication (e.g. $\forall x{:}\mathrm{Nat}.\mathrm{if}\ x = 5\ \mathrm{then}\ T_1\ \mathrm{else}\ T_2$)
- Indexed session types: $\mathrm{countDown} :: \Pi x{:}\mathrm{Nat}.\mathrm{stype}$
- Ability to specify and prove properties of processes within the theory:

$$
\begin{array}{lcl}
\mathrm{simpleCounter} & :: & \mathrm{Nat} \rightarrow \{\mathrm{simpleCounter}\} \\
\mathrm{simpleCounter} & = & \lambda x{:}\mathrm{Nat}.\{\ldots\} \\
\\
\mathrm{corrCount} & :: & \Pi x{:}\mathrm{Nat}.\Pi y{:}\{\mathrm{simpleCounter}\}.\mathrm{type} \\
\ldots & &
\end{array}
$$

Richer Type Theories

Where are we?

Monadic Dependencies (Ongoing work)

By adding the (contextual) monad to a dependent type theory we get:

- Relations on processes as type families (e.g. $\Pi x{:}\{z{:}A\}.\Pi y{:}\{z{:}B\}.\mathsf{type}$)
- Value dependent communication (e.g. $\forall x{:}\mathsf{Nat}.\mathsf{if}\ x = 5\ \mathsf{then}\ T_1\ \mathsf{else}\ T_2$)
- Indexed session types: $\mathsf{countDown} :: \Pi x{:}\mathsf{Nat}.\mathsf{stype}$
- Ability to specify and prove properties of processes within the theory:

$$
\begin{array}{lcl}
\mathsf{simpleCounter} & :: & \mathsf{Nat} \rightarrow \{\mathsf{simpleCounter}\} \\
\mathsf{simpleCounter} & = & \lambda x{:}\mathsf{Nat}.\{\ldots\} \\
\\
\mathsf{corrCount} & :: & \Pi x{:}\mathsf{Nat}.\Pi y{:}\{\mathsf{simpleCounter}\}.\mathsf{type} \\
\ldots & & \\
\mathsf{correctness} & :: & \Pi n{:}\mathsf{Nat}.\mathsf{corrCount}\ n\ (\mathsf{simpleCounter}(n))
\end{array}
$$

Proof Conversions and Type Isomorphisms

Proof Conversions

- Process reductions map to principal cut reductions.
- What about the remaining proof conversions?
- Can we understand them in concurrency theoretic terms?

Proof Conversions and Type Isomorphisms

Introduction

Proof Conversions

- Process reductions map to principal cut reductions.
- What about the remaining proof conversions?
- Can we understand them in concurrency theoretic terms?

Approach

We decompose proof conversions into three classes:

- Computational Conversions (i.e. principal cut conversions).
- Cut Conversions (i.e. permutting two cuts in a proof).
- Commuting Conversions (i.e. commutting inference rules).

First two correspond to reductions and structural equivalences.

Proof Conversions

Commuting Conversions

Commuting Conversions induce a congruence $\simeq_c$ on typed processes

$\otimes L / \otimes L$ Commuting Conversion

$$x{:}A \otimes B, z{:}C \otimes D \Longrightarrow x(y).z(w).P \simeq_c z(w).x(y).P :: v{:}E$$

Commuting (input) prefixes appears, at first, counterintuitive.

Proof Conversions

Commuting Conversions

Commuting Conversions induce a congruence $\simeq_c$ on typed processes

$\otimes L / \otimes L$ Commuting Conversion

$$x{:}A \otimes B, z{:}C \otimes D \Longrightarrow x(y).z(w).P \simeq_c z(w).x(y).P :: v{:}E$$

Commuting (input) prefixes appears, at first, counterintuitive.

Typed Contextual Equivalence

In any well-typed context, we cannot distinguish the two processes:

$$(\nu x)(\nu z)(x(y).z(w).P \mid R_x \mid S_z) \cong (\nu x)(\nu z)(z(w).x(y).P \mid R_x \mid S_z) :: v{:}E$$

Actions along x and z are not observable.

Proof Conversions

Typed Contextual Equivalence

Typed Contextual Equivalence

- How to define this equivalence in a tractable way?
- Typed Contextual Bisimilarity.

Proof Conversions

Typed Contextual Equivalence

Typed Contextual Equivalence

- How to define this equivalence in a tractable way?
- Typed Contextual Bisimilarity.

Contextual Equivalence

- Contextual: For all typed contexts...
- Typed bisimilarity on closed processes:
  - $P \sim Q :: x{:}A \multimap B$ iff $P \xrightarrow{x(y)} P'$ implies $Q \overset{x(y)}{\Rightarrow} Q'$ and $\forall R. \cdot \Longrightarrow R :: y{:}A$ we have $(\nu y)(P \mid R) \sim (\nu y)(Q \mid R) :: x{:}B$
  - $P \sim Q :: x{:}C$ iff $P \xrightarrow{\tau} P'$ implies $Q \Rightarrow Q'$ and $P' \sim Q' :: x{:}C$.
  - ...

Proof Conversions

Issue

$\otimes L / \otimes L$ Conversion Revisited

$$(\nu x)(\nu z)(x(y).z(w).P \mid R_x \mid S_z) \sim (\nu x)(\nu z)(z(w).x(y).P \mid R_x \mid S_z) :: v{:}E?$$

- Suppose input along x matches an output in R_x on the left proc.

Proof Conversions

Issue

$\otimes L / \otimes L$ Conversion Revisited

$$(\nu x)(\nu z)(x(y).z(w).P \mid R_x \mid S_z) \sim (\nu x)(\nu z)(z(w).x(y).P \mid R_x \mid S_z) :: v{:}E?$$

- Suppose input along x matches an output in R_x on the left proc.
- How do we know the right side process can match it?
- What if S_z never outputs to z?

Proof Conversions

Issue

$\otimes L / \otimes L$ Conversion Revisited

$$(\nu x)(\nu z)(x(y).z(w).P \mid R_x \mid S_z) \sim (\nu x)(\nu z)(z(w).x(y).P \mid R_x \mid S_z) :: v{:}E?$$

- Suppose input along x matches an output in R_x on the left proc.
- How do we know the right side process can match it?
- What if S_z never outputs to z?
- Requires termination!

Proof Conversions

Issue

$\otimes L/\otimes L$ Conversion Revisited

$$(\nu x)(\nu z)(x(y).z(w).P \mid R_x \mid S_z) \sim (\nu x)(\nu z)(z(w).x(y).P \mid R_x \mid S_z) :: v{:}E?$$

- Suppose input along x matches an output in R_x on the left proc.
- How do we know the right side process can match it?
- What if S_z never outputs to z?
- Requires termination!

Termination and Equivalence

- Can we develop a uniform solution?
- Inspiration from functional “world”: Linear Logical Relations!

Proof Conversions

Termination and Equivalence

Linear Logical Relations [Pérez et al. 12]

- Termination: Inductively defined unary predicate.
- Contextual Equivalence: Extension to relational setting.

Proof Conversions

Termination and Equivalence

Linear Logical Relations [Pérez et al. 12]

- Termination: Inductively defined unary predicate.
- Contextual Equivalence: Extension to relational setting.

Logical Predicate

- Terminating by construction.
- Inductive on typing derivations: $\mathcal{L}[\Gamma; \Delta \vdash T]$
  - $P \in \mathcal{L}[\Gamma; y{:}A, \Delta \vdash T]$ if $\forall R \in \mathcal{L}[y : A].(\nu y)(R \mid P) \in \mathcal{L}[\Gamma; \Delta \vdash T]$.

Proof Conversions

Termination and Equivalence

Linear Logical Relations [Pérez et al. 12]

- Termination: Inductively defined unary predicate.
- Contextual Equivalence: Extension to relational setting.

Logical Predicate

- Terminating by construction.
- Inductive on typing derivations: $\mathcal{L}[\Gamma; \Delta \vdash T]$
  - $P \in \mathcal{L}[\Gamma; y{:}A, \Delta \vdash T]$ if $\forall R \in \mathcal{L}[y : A].(\nu y)(R \mid P) \in \mathcal{L}[\Gamma; \Delta \vdash T]$.
- Base case is inductive on types:
  - $P \in \mathcal{L}[z{:}A \multimap B] \triangleq$ if $P \xrightarrow{z(y)} P'$ then $\forall Q \in \mathcal{L}[y{:}A].(\nu y)(P' \mid Q) \in \mathcal{L}[z{:}B]$
  - $\ldots$

Proof Conversions

Termination and Equivalence

Linear Logical Relations [Pérez et al. 12]

- Termination: Inductively defined unary predicate.
- Contextual Equivalence: Extension to relational setting.

Logical Predicate

- Terminating by construction.
- Inductive on typing derivations: $\mathcal{L}[\Gamma; \Delta \vdash T]$
  - $P \in \mathcal{L}[\Gamma; y{:}A, \Delta \vdash T]$ if $\forall R \in \mathcal{L}[y : A].(\nu y)(R \mid P) \in \mathcal{L}[\Gamma; \Delta \vdash T]$.
- Base case is inductive on types:
  - $P \in \mathcal{L}[z{:}A \multimap B] \triangleq$ if $P \xrightarrow{z(y)} P'$ then $\forall Q \in \mathcal{L}[y{:}A].(\nu y)(P' \mid Q) \in \mathcal{L}[z{:}B]$
  - …

Typing implies Termination

If $\Gamma; \Delta \vdash P :: T$ then $P \in \mathcal{L}[\Gamma; \Delta \vdash T]$.

Proof Conversions

Termination and Equivalence

Typed Equivalence

- Relational generalization of the predicate.
- Inductive on typing derivations: $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$
  - If $\Gamma; \Delta, y{:}A \vdash P \mathcal{R} Q :: T$ then $\forall R. \vdash R :: y{:}A, \Gamma; \Delta \vdash (\nu y)(P \mid R) \mathcal{R} (\nu y)(Q \mid R) :: T$.

Typed Equivalence

- Relational generalization of the predicate.
- Inductive on typing derivations: $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$
  - If $\Gamma; \Delta, y{:}A \vdash P \mathcal{R} Q :: T$ then $\forall R. \vdash R :: y{:}A, \Gamma; \Delta \vdash (\nu y)(P \mid R) \mathcal{R} (\nu y)(Q \mid R) :: T$.
- Base case is inductive on types:
  - $\vdash P \mathcal{R} Q :: x{:}A \multimap B$ iff $P \xrightarrow{x(y)} P'$ implies $Q \stackrel{x(y)}{\Longrightarrow} Q'$ and $\forall R. \cdot \vdash R :: y{:}A$ we have $\vdash (\nu y)(P \mid R) \mathcal{R} (\nu y)(Q \mid R) :: x{:}B$
  - ...

Proof Conversions

Termination and Equivalence

Typed Equivalence

- Relational generalization of the predicate.
- Inductive on typing derivations: $\Gamma; \Delta \vdash P \mathcal{R} Q :: T$
  - If $\Gamma; \Delta, y{:}A \vdash P \mathcal{R} Q :: T$ then $\forall R. \vdash R :: y{:}A, \Gamma; \Delta \vdash (\nu y)(P \mid R) \mathcal{R} (\nu y)(Q \mid R) :: T$.
- Base case is inductive on types:
  - $\vdash P \mathcal{R} Q :: x{:}A \multimap B$ iff $P \xrightarrow{x(y)} P'$ implies $Q \stackrel{x(y)}{\Longrightarrow} Q'$ and $\forall R. \cdot \vdash R :: y{:}A$ we have $\vdash (\nu y)(P \mid R) \mathcal{R} (\nu y)(Q \mid R) :: x{:}B$
  - ...

Soundness of Commuting Conversions

If $\Gamma; \Delta \vdash P \simeq_c Q :: T$ then $\Gamma; \Delta \vdash P \approx Q :: T$

Type Isomorphisms

Definition and Validation

Type Isomorphism ($A \simeq B$)

Types A and B are iso. if there are proofs π_A of $B \vdash A$ and π_B of $A \vdash B$, composing in both direction to identity.

Type Isomorphisms

Definition and Validation

Type Isomorphism ($A \simeq B$)

Types A and B are iso. if there are proofs π_A of $B \vdash A$ and π_B of $A \vdash B$, composing in both direction to identity.

Session Type Isomorphisms ($A \simeq_S B$)

Session types A and B are iso. if there are processes P and Q:

- $x{:}A \vdash P :: y{:}B$ and $y{:}B \vdash Q :: x{:}A$.

Type Isomorphisms

Definition and Validation

Type Isomorphism ($A \simeq B$)

Types A and B are iso. if there are proofs π_A of $B \vdash A$ and π_B of $A \vdash B$, composing in both direction to identity.

Session Type Isomorphisms ($A \simeq_S B$)

Session types A and B are iso. if there are processes P and Q:

- $x{:}A \vdash P :: y{:}B$ and $y{:}B \vdash Q :: x{:}A$.
- $x{:}A \vdash (\nu y)(P \mid Q) \approx [x \leftrightarrow z] :: z{:}A$.
- $y{:}B \vdash (\nu x)(Q \mid P) \approx [y \leftrightarrow z] :: z{:}B$.

Type Isomorphisms

Definition and Validation

Type Isomorphism ($A \simeq B$)

Types A and B are iso. if there are proofs π_A of $B \vdash A$ and π_B of $A \vdash B$, composing in both direction to identity.

Session Type Isomorphisms ($A \simeq_S B$)

Session types A and B are iso. if there are processes P and Q:

- $x{:}A \vdash P :: y{:}B$ and $y{:}B \vdash Q :: x{:}A$.
- $x{:}A \vdash (\nu y)(P \mid Q) \approx [x \leftrightarrow z] :: z{:}A$.
- $y{:}B \vdash (\nu x)(Q \mid P) \approx [y \leftrightarrow z] :: z{:}B$.

Validating Isomorphisms

If $A \simeq B$ then $A \simeq_S B$.

Summary

So far:

- Explored a logical interpretation of session-based concurrency
- Explain concurrency theoretic concepts using logic
- Map logical phenomena to concurrency theory
- Clean and elegant reasoning through logic.

Not in this talk:

- Asynchrony [DeYoung12]
- Parametricity [Caires et al12]
- Inductive and Coinductive Session Types [Toninho et al14,LindleyMorris16]
- Monitoring and blame [Jia et al16]
- Shared resources and non-determinism [Atkey et al16,BalzerPfenning17]
- Multiparty sessions [Carbone et al15,16]

Thank you!
Questions?

Session Types and Linear Logic

Bernardo Toninho and Nobuko Yoshida

University of Bath,
November 28, 2017