A Gentle Introduction to Session Types

Chuta Sano

McGill University

July 9, 2024

Roadmap: an analogy with the λ-calculus

- functions
- processes

Roadmap: an analogy with the λ-calculus

- functions
- λ-calculus

- processes
- π-calculus

Roadmap: an analogy with the λ-calculus

- functions
- λ-calculus
  - type system to prevent nontermination
  - type preservation
  - progress

- processes
- π-calculus
  - type system to prevent **deadlocks**
  - session fidelity
  - deadlock freedom

Roadmap: an analogy with the λ-calculus

- functions
- λ-calculus
  - type system to prevent nontermination
  - type preservation
  - progress
- extensions to the type system

- processes
- π-calculus
  - type system to prevent **deadlocks**
  - session fidelity
  - deadlock freedom
- extensions to the type system

Roadmap: an analogy with the λ-calculus

- functions
- λ-calculus
  - type system to prevent nontermination
  - type preservation
  - progress
- extensions to the type system
  - recursion, subtyping, substructural
  - dependent types and polymorphism

- processes
- π-calculus
  - type system to prevent **deadlocks**
  - session fidelity
  - deadlock freedom
- extensions to the type system
  - (equi-)recursion, subtyping
  - polymorphism, sharing, etc.

The π-calculus[1]

A process P is given by the following grammar:

$P \mid Q$	(Concurrently execute P and Q)
$\nu x.P$	(Allocate fresh channel x)
$x \leftarrow \mathrm{recv}(c); P$	(Receive a channel from c and bind to x)
$\mathrm{send}(c)\ a; P$	(Send the channel a across c)
0	(A nullary process that does nothing)

[1] Milner 1980.

The π-calculus[1]

A process P is given by the following grammar:

$P \mid Q$	(Concurrently execute P and Q)
$\nu x.P$	(Allocate fresh channel x)
$x \leftarrow \text{recv}(c); P$	(Receive a channel from c and bind to x)
$\text{send}(c)\ a; P$	(Send the channel a across c)
0	(A nullary process that does nothing)

Fun fact: The untyped π-calculus can embed the untyped λ-calculus!

[1] Milner 1980.

Problems with the π-calculus

Problems with the π-calculus

Deadlock (kinda):

$$\nu x.(0 \mid y \leftarrow \mathsf{recv}(x); 0)$$

Problems with the π-calculus

Deadlock (kinda):

$$\nu x.(0 \mid y \leftarrow \mathsf{recv}(x); 0)$$

Nondeterminism:

$$\nu x, y_1, y_2.(\mathsf{send}(x)\ y_1; \mathsf{send}(x)\ y_2; 0 \mid z_1 \leftarrow \mathsf{recv}(x); 0 \mid z_2 \leftarrow \mathsf{recv}(x); 0)$$

Type systems to the rescue![2]

Key ideas:

[2] Honda 1993; Honda, Vasconcelos, and Kubo 1998.

Type systems to the rescue![2]

Key ideas:

- Every free channel must be assigned a session type – something that "encodes" the protocol

[2]Honda 1993; Honda, Vasconcelos, and Kubo 1998.

Type systems to the rescue![2]

Key ideas:

- Every free channel must be assigned a session type – something that "encodes" the protocol
- Channels cannot be duplicated

[2] Honda 1993; Honda, Vasconcelos, and Kubo 1998.

Type systems to the rescue![2]

Key ideas:

- Every free channel must be assigned a session type – something that "encodes" the protocol
- Channels cannot be duplicated
- Channels cannot be freely thrown away

[2] Honda 1993; Honda, Vasconcelos, and Kubo 1998.

Type systems to the rescue![2]

Key ideas:

- Every free channel must be assigned a session type – something that "encodes" the protocol
- Channels cannot be duplicated
- Channels cannot be freely thrown away

Note: I am not showing the original syntax...

[2] Honda 1993; Honda, Vasconcelos, and Kubo 1998.

Type systems to the rescue![2]

Key ideas:

- Every free channel must be assigned a session type – something that "encodes" the protocol
- Channels cannot be duplicated
- Channels cannot be freely thrown away

Note: I am not showing the original syntax...
Let's just move on...

[2] Honda 1993; Honda, Vasconcelos, and Kubo 1998.

Typing the π-calculus

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Parallel composition and channel abstraction:

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Parallel composition and channel abstraction:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \quad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Parallel composition and channel abstraction:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \qquad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Sending and receiving channels:

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Parallel composition and channel abstraction:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \qquad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Sending and receiving channels:

$$\frac{P \vdash \Delta, c : B, x : A}{x \leftarrow \mathsf{recv}(c); P \vdash \Delta, c : A \parr B} \qquad \frac{P \vdash \Delta, c : B}{\mathsf{send}(c)\ a; P \vdash \Delta, c : A \otimes B, a : A}$$

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Parallel composition and channel abstraction:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \quad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Sending and receiving channels:

$$\frac{P \vdash \Delta, c : B, x : A}{x \leftarrow \mathrm{recv}(c); P \vdash \Delta, c : A \parr B} \quad \frac{P \vdash \Delta, c : B}{\mathrm{send}(c)\ a; P \vdash \Delta, c : A \otimes B, a : A}$$

Null process:

Typing the π-calculus

"P is a process that communicates alongside channels $a_1, \ldots, a_n$"

$$P \vdash \underbrace{a_1 : A_1, \ldots, a_n : A_n}_{\Delta}$$

Parallel composition and channel abstraction:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \quad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Sending and receiving channels:

$$\frac{P \vdash \Delta, c : B, x : A}{x \leftarrow \mathrm{recv}(c); P \vdash \Delta, c : A \parr B} \quad \frac{P \vdash \Delta, c : B}{\mathrm{send}(c)\ a; P \vdash \Delta, c : A \otimes B, a : A}$$

Null process:

$$\frac{}{0 \vdash \cdot}$$

Problem with channel abstraction

Recall:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \quad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Problem with channel abstraction

Recall:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \quad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Problem: All channels must have two endpoints! Consider:

$$\nu x{:}A \parr B.(y \leftarrow \mathsf{recv}(x); \ldots \mid 0)$$

Problem with channel abstraction

Recall:

$$\frac{P \vdash \Delta_1 \quad Q \vdash \Delta_2}{P \mid Q \vdash \Delta_1, \Delta_2} \qquad \frac{P \vdash \Delta, x : A}{\nu x{:}A.P \vdash \Delta}$$

Problem: All channels must have two endpoints! Consider:

$$\nu x{:}A \parr B.(y \leftarrow \mathsf{recv}(x); \ldots \mid 0)$$

Any ideas?

Attempt: combine channel abstraction and composition

$$\frac{P \vdash \Delta_1, x{:}A \quad Q \vdash \Delta_2, x{:}A?}{\nu x{:}A.(P \mid Q) \vdash \Delta_1, \Delta_2}$$

Attempt: combine channel abstraction and composition

$$\frac{P \vdash \Delta_1, x{:}A \quad Q \vdash \Delta_2, x{:}A?}{\nu x{:}A.(P \mid Q) \vdash \Delta_1, \Delta_2}$$

What should the type of x be in Q?

Attempt: combine channel abstraction and composition

$$\frac{P \vdash \Delta_1, x{:}A \quad Q \vdash \Delta_2, x{:}\overline{A}}{\nu x{:}A.(P \mid Q) \vdash \Delta_1, \Delta_2}$$

What should the type of x be in Q? Duality!

$$\overline{A \otimes B} = \overline{A} \parr \overline{B}$$
$$\overline{A \parr B} = \overline{A} \otimes \overline{B}$$

Choices

Choices

"Send and receive labels"

$$\frac{P \vdash \Delta, x{:}A_i}{x.l_i; P \vdash \Delta, x{:} \oplus \{\overline{l : A}\}} \qquad \frac{(\forall i)\ P_i \vdash \Delta, x{:}A_i}{\mathsf{case}(x)\{\overline{l \Rightarrow P}\} \vdash \Delta, x{:}\&\{\overline{l : A}\}}$$

Summary:[3]

Type	Interpretation (provider)	Process Term	Cont.
1	Close channel (terminate)	$\text{close}(x)$	-
$\bot$	Wait for channel to close	$\text{wait}(x); P$	-
$A \otimes B$	Send channel of type A	$\text{send}(x)\ y; P$	B
$A \parr B$	Receive channel of type A	$y \leftarrow \text{recv}(x); P$	B
$\oplus\{\overline{l : A}\}$	Send a label $i \in \bar{l}$	$x.l; P$	A_i
$\&\{\overline{l : A}\}$	Receive and branch on $i \in \bar{l}$	$\text{case}(x)\{\overline{l \Rightarrow P}\}$	A_i

[3] Caires and Pfenning 2010; Wadler 2012.

Equi-recursion and examples

Equi-recursion and examples

Natural numbers:

$$\text{nat} = \oplus\{\text{succ} : \text{nat} \\ \text{zero} : 1\}$$

Equi-recursion and examples

Natural numbers:

$$\mathsf{nat} = \oplus\{\mathsf{succ} : \mathsf{nat} \\ \mathsf{zero} : 1\}$$

Queue:

$$\mathsf{queue} = \&\{\mathsf{enq} : A \multimap \mathsf{queue} \\ \mathsf{deq} : A \otimes \mathsf{queue}\}$$

Drawing time

Drawing time

Audience: shout out any reasonable natural number.

Anyway...

- (Message-passing) process calculi model concurrent computation via processes that communicate across channels
- Session types apply to channels and are "protocols" that processes must follow when interacting with channels
- Channels are linear; they cannot be duplicated nor thrown away
- See also: sharing[4], multiparty session types[5], GV[6]

[4] Balzer and Pfenning 2017.
[5] Carbone, Honda, and Yoshida 2008.
[6] Gay and Vasconcelos 2010.