

Introduction to (Finite) Binary Session Types

Hernán Melgratti

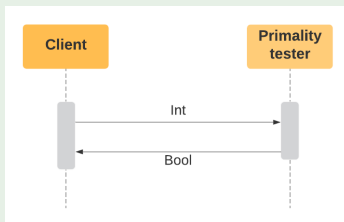
ICC University of Buenos Aires-Conicet

11 February 2020 @ Pisa

Informally

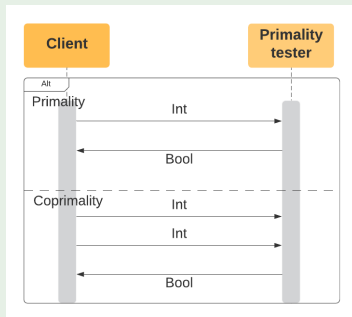
- ▶ A session type defines a communication protocol
- ▶ In the binary case, it describes the messages exchanged between two parties

First example

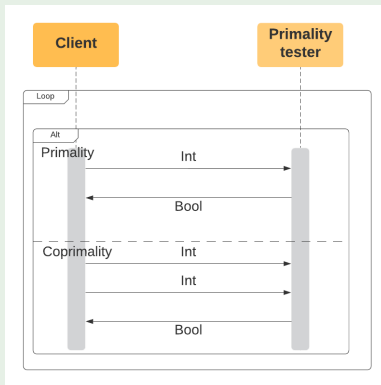


- ▶ We rely on a textual description; the flow is described from the point of view of one of the participants
- ▶ `Tester = ?int.!bool.end`
 - `?t` : a receive of a value of type `t`
 - `._` : followed by
 - `!t` : a send of a value of type `t`
 - `end` : a terminated session
- ▶ `Client = !int.?bool.end`
- ▶ `Tester` and `Client` behave **dually**

Choices



- ▶ $\text{Tester} = \&[\text{Pr} : ?\text{int}.\text{!bool}.\text{end}, \text{Co} : ?\text{int}.\text{?int}.\text{!bool}.\text{end}]$
 - ▶ $\&[\mathcal{L}_i : T_i]_{i \in I}$: **Offering** several alternatives, each of them identified by the *label* \mathcal{L}_i
- ▶ $\text{Client} = \oplus[\text{Pr} : \text{!int}.\text{?bool}.\text{end}, \text{Co} : \text{!int}.\text{!int}.\text{?bool}.\text{end}]$
 - ▶ $\oplus[\mathcal{L}_i : T_i]_{i \in I}$: **Selecting** one of the alternatives identified by the *labels* \mathcal{L}_i
- ▶ Tester and Client behave **dually**



- ```

▶ Tester = &[Pr : ?int.!bool.Tester,
 Co : ?int.?int.!bool.Tester]
▶ Client = ⊕[Pr : !int.?bool.Client,
 Co : !int.!int.?bool.Client]

```

## Modelling a function

$f : \text{int} \rightarrow \text{bool}$

```
f = ?int.!bool.end
```

Invocation

```
inv = !int.?bool.end
```

## Modelling an object (Typestate)

### File

`File = ?mode.Opened`

`Opened = &[read :  $\oplus$ [eof : Opened, val : !string.Opened], close : end]`

### Client

`Client = !mode.Reading`

`Reading =  $\oplus$ [read : &[eof : Reading, val : !string.Reading], close : end]`

# Syntax of Types

## Session Types

|                 |                                         |                        |
|-----------------|-----------------------------------------|------------------------|
| $S, T ::=$      | <code>end</code>                        | terminated session     |
|                 | <code>?t.S</code>                       | receive (input)        |
|                 | <code>!t.S</code>                       | send (output)          |
|                 | <code>&amp;[l_i : T_i]_{i \in I}</code> | branch                 |
|                 | <code>⊕[l_i : T_i]_{i \in I}</code>     | select                 |
|                 | <code>μX.S</code>                       | recursive session type |
|                 | <code>X</code>                          | session type variable  |
| $s, t ::=$      | <code>S</code>                          | A session type         |
|                 | <code>int, bool</code>                  | basic types            |
|                 | <code>...</code>                        | other types            |
| $\mathcal{L} =$ | <code>{l, l_1, ...}</code>              | Set of labels          |

### Remark

- ▶ The grammar allows terms like `?S.T`
- ▶ For instance, `?(?int.end).!bool.end` vs `?int.!bool.end`

## Examples

$f : \text{int} \rightarrow \text{bool}$

```
f = ?int.!bool.end
```

```
g = ?f.!bool.end
```

It resembles

$$g : (\text{int} \rightarrow \text{bool}) \rightarrow \text{bool}$$

but it is not the same (**more to come**)

### File

```
File = ?mode.Opened
```

```
Opened = &[read : \oplus [eof : Opened, val : !string.Opened], close : end]
```

### Function that processes a file

```
Client1 = !File.?int.end
```

```
Client2 = !Opened.?int.end
```



# Duality

$\overline{S}$  is the dual of  $S$

$$\overline{\text{end}} = \text{end}$$

$$\overline{?t.S} = !t.\overline{S}$$

$$\overline{!t.S} = ?t.\overline{S}$$

$$\overline{\&[\mathcal{L}_i : \overline{T}_i]_{i \in I}} = \oplus[\mathcal{L}_i : T_i]_{i \in I}$$

$$\overline{\oplus[\mathcal{L}_i : T_i]_{i \in I}} = \&[\mathcal{L}_i : \overline{T}_i]_{i \in I}$$

## Goal

Determine whether a program implements a protocol (a session type)

1. Fix a language for writing programs
2. Define a relation between programs and session types that states that a program behaves as prescribed by the types

We choose <sup>1</sup>

1. A language with message-passing communication based on synchronous channels
2. Session types are associated with channels

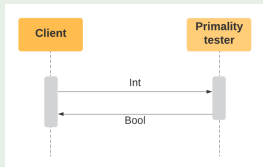
---

<sup>1</sup>Simon J. Gay, Malcolm Hole: Subtyping for session types in the pi calculus. Acta Inf. (2005)

# Programs

- ▶ Roughly, each participant is implemented by a process (i.e., a thread)
- ▶ Processes communicate through *session channels*
- ▶ A session channel  $x$  has two endpoints  $x^+$  and  $x^-$
- ▶ A process sends and receives messages on a session endpoint

## Tester



Tester = ?**int**.!**bool**.end

- ▶ We give an implementation over the session endpoints  $x^+$  (for the server) and  $x^-$  (for the client)

$P_{\text{server}} = x^+?(y:\text{int}).x^+!\text{true}.0$  (faulty)

$P_{\text{client}} = x^-!1.x^-?(z:\text{bool}).Q$

- ▶ The system is the parallel composition of the two processes

$(\nu x:\text{Tester})(P_{\text{server}} \mid P_{\text{client}})$

# Syntax of Processes

## Polarities

$p ::= + \mid - \mid \epsilon$

Optional polarities

## Values (more in general expressions)

|            |                            |                                                       |
|------------|----------------------------|-------------------------------------------------------|
| $v, w ::=$ | $x^p, y^q, \dots$          | (polarised) variables $\mathcal{X} = \{x, y, \dots\}$ |
|            | $()$                       | unit value                                            |
|            | <b>true</b> , <b>false</b> | boolean values                                        |
|            | $\dots$                    | expressions                                           |

## Processes

|            |                                                     |                      |
|------------|-----------------------------------------------------|----------------------|
| $P, Q ::=$ | $0$                                                 | terminated process   |
|            | $x^p?(y:\mathbf{t}).P$                              | input                |
|            | $x^p!v.P$                                           | output               |
|            | $x^p \triangleright [\mathbf{l}_i : P_i]_{i \in I}$ | branch               |
|            | $x^p \triangleleft \mathbf{l}.P$                    | select               |
|            | $P Q$                                               | parallel composition |
|            | $(\nu x:\mathbf{S})P$                               | channel creation     |
|            | $!P$                                                | replication          |

# Syntax of Types

## Session Types

|            |                                   |                        |
|------------|-----------------------------------|------------------------|
| $S, T ::=$ | <b>end</b>                        | terminated session     |
|            | $?t.S$                            | receive (input)        |
|            | $!t.S$                            | send (output)          |
|            | $\&[\iota_i : T_i]_{i \in I}$     | branch                 |
|            | $\oplus[\iota_i : T_i]_{i \in I}$ | select                 |
|            | $\mu X.S$                         | recursive session type |
|            | $X$                               | session type variable  |
| $s, t ::=$ | $S$                               | A session type         |
|            | <b>int, bool</b>                  | basic types            |
|            | ...                               | other types            |

# Notation

- ▶ for a polarity  $p$ , we write  $\bar{p}$  for the complementary endpoint

$$\bar{+} = - \qquad \bar{-} = + \qquad \bar{\epsilon} = \epsilon$$

- ▶ we identify  $x^\epsilon$  with  $x$

# Typing

## Goal

Determine whether a program implements a protocol (a session type)

1. Fix a language for writing programs
2. Define a relation between programs and session types that states that a program **behaves** as prescribed by the types

# Operational semantics

Given in terms of a *Labelled Transition System* (LTS)  $(P, \longrightarrow)$  where

- ▶  $\longrightarrow \subseteq P \times (\mathcal{X} \cup \{\tau\}) \times (\mathcal{L} \cup \{-\}) \times P$

- ▶  $(P, \alpha, \mathfrak{l}, Q) \in \longrightarrow$ 
  - ▶ means  $P$  evolves to  $Q$  after communicating the choice  $\mathfrak{l}$  on the session  $\alpha$
  - ▶ is abbreviated as  $P \xrightarrow{\alpha, \mathfrak{l}} Q$
- ▶  $\tau$  stands for a hidden session
- ▶  $-$  for no choice



# Operational semantics

$$x^p!v.P \mid x^{\bar{p}}?(y:\mathbf{t}).Q \xrightarrow{x,\bar{}} P \mid Q\{v/y\} \text{ [R-Comm]}$$

## Substitution

$$\begin{aligned} x\{v/x\} &= v \\ x^p\{v/y\} &= x^p \end{aligned} \quad \text{if } x \neq y$$

$$\begin{aligned} 0\{v/y\} &= 0 \\ (P|Q)\{v/y\} &= P\{v/y\}|Q\{v/y\} \\ (x^p?(z:\mathbf{t}).P)\{v/y\} &= x^p\{v/y\}?(z:\mathbf{t}).P\{v/y\} \quad \text{if } z \notin \text{fn}(v) \cup \{y\} \end{aligned}$$

## Free names

*fn*

$$\begin{aligned}\text{fn}(\text{true}) &= \text{fn}(\text{false}) = \text{fn}(() ) = \emptyset \\ \text{fn}(x^p) &= \{x^p\}\end{aligned}$$

$$\text{fn}(0) = \emptyset$$

$$\text{fn}(P|Q) = \text{fn}(Q) \cup \text{fn}(P)$$

$$\text{fn}(x^p?(y:\text{t}).P) = \{x^p\} \cup (\text{fn}(P) \setminus \{y\})$$

$$\text{fn}(x^p!v.P) = \{x^p\} \cup \text{fn}(v) \cup \text{fn}(P)$$

$$\text{fn}(x^p \triangleright [\mathsf{l}_i : P_i]_{i \in I}) = \{x^p\} \cup \left( \bigcup_i \text{fn}(P_i) \right)$$

$$\text{fn}(x^p \triangleleft \mathsf{l}.P) = \{x^p\} \cup \text{fn}(P)$$

$$\text{fn}((\nu x:\text{S})P) = \text{fn}(P) \setminus \{x, x^+, x^-\}$$

# Operational semantics

$$x^p ! v . P \mid x^{\bar{p}} ? (y : \mathbf{t}) . Q \xrightarrow{x, \bar{v}} P \mid Q\{v/y\} \text{ [R-Comm]}$$

## Substitution

$$\begin{aligned} x\{v/x\} &= v \\ x^p\{v/y\} &= x^p && \text{if } x \neq y \end{aligned}$$

$$\begin{aligned} 0\{v/y\} &= 0 \\ (P \mid Q)\{v/y\} &= P\{v/y\} \mid Q\{v/y\} \\ (x^p ? (z : \mathbf{t}) . P)\{v/y\} &= x^p\{v/y\} ? (z : \mathbf{t}) . P\{v/y\} && \text{if } z \notin \text{fn}(v) \cup \{y\} \\ (x^p ! w . P)\{v/y\} &= x^p\{v/y\} ! w\{v/y\} . P\{v/y\} \\ (x^p \triangleright [\mathbf{l}_i : P_i]_{i \in I})\{v/y\} &= x^p\{v/y\} \triangleright [\mathbf{l}_i : P_i\{v/y\}]_{i \in I} \\ (x^p \triangleleft \mathbf{l} . P)\{v/y\} &= x^p\{v/y\} \triangleleft \mathbf{l} . P\{v/y\} \\ ((\nu x : \mathbf{S}) P)\{v/y\} &= (\nu x : \mathbf{S}) P\{v/y\} && \text{if } x \notin \text{fn}(v) \cup \{y\} \end{aligned}$$

## Operational semantics

$$x^p ! v . P \mid x^{\bar{p}} ? (y : \mathbf{t}) . Q \xrightarrow{x, -} P \mid Q \{v/y\} \text{ [R-Comm]}$$

$$x^p \triangleleft \mathbf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathbf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathbf{l}_i} P \mid Q_i$$

## Operational semantics

$$x^p ! v . P \mid x^{\bar{p}} ? (y : \mathbf{t}) . Q \xrightarrow{x, \bar{v}} P \mid Q \{v/y\} \quad [\text{R-Comm}]$$

$$\frac{i \in I}{x^p \triangleleft \mathbf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathbf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathbf{l}_i} P \mid Q_i} \quad [\text{R-Select}]$$

# Operational semantics

$$x^p ! v . P \mid x^{\bar{p}} ? (y : \textcolor{violet}{t}) . Q \xrightarrow{x, \bar{p}} P \mid Q\{v/y\} \quad [\text{R-Comm}]$$

$$\frac{p \in \{+, -\} \quad i \in I}{x^p \triangleleft \textcolor{teal}{l}_i . P \mid x^{\bar{p}} \triangleright [\textcolor{teal}{l}_j : Q_j]_{j \in I} \xrightarrow{x, \textcolor{teal}{l}_i} P \mid Q_i} \quad [\text{R-Select}]$$

$$\frac{P \xrightarrow{x, \textcolor{teal}{l}} P' \quad S \xrightarrow{\textcolor{teal}{l}} T}{(\nu x : \textcolor{violet}{S}) P \xrightarrow{\tau, \bar{p}} (\nu x : \textcolor{violet}{T}) P'} \quad [\text{R-NewS}]$$

## Semantics of Types

$$? t . S \xrightarrow{\bar{p}} S$$

$$! t . S \xrightarrow{\bar{p}} S$$

$$\& [\textcolor{teal}{l}_i : T_i]_{i \in I} \xrightarrow{\textcolor{teal}{l}_i} T_i$$

$$\oplus [\textcolor{teal}{l}_i : T_i]_{i \in I} \xrightarrow{\textcolor{teal}{l}_i} T_i$$

# Operational semantics

$$x^p ! v . P \mid x^{\bar{p}}?(y:\mathbf{t}) . Q \xrightarrow{x, \bar{-}} P \mid Q\{v/y\} \quad [\text{R-Comm}]$$

$$\frac{i \in I}{x^p \triangleleft \mathbf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathbf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathbf{l}_i} P \mid Q_i} \quad [\text{R-Select}]$$

$$\frac{P \xrightarrow{x, \mathbf{l}} P' \quad S \xrightarrow{x, \mathbf{l}} T}{(\nu x:\mathbf{S})P \xrightarrow{\tau, \bar{-}} (\nu x:\mathbf{T})P'} \quad [\text{R-NewS}]$$

$$\frac{P \xrightarrow{\alpha, \mathbf{l}} P' \quad \alpha \neq x}{(\nu x:\mathbf{S})P \xrightarrow{\alpha, \mathbf{l}} (\nu x:\mathbf{S})P'} \quad [\text{R-New}]$$

$$\frac{P \xrightarrow{\alpha, \mathbf{l}} P'}{P \mid Q \xrightarrow{\alpha, \mathbf{l}} P' \mid Q} \quad [\text{R-Par}]$$

## Structural equivalence

$$\begin{aligned}P|0 &\equiv P \\P|Q &\equiv Q|P \\(P|Q)|R &\equiv Q|(P|R) \\(\nu x:S)(\nu y:T)P &\equiv (\nu y:T)(\nu x:S)P \\(\nu x:S)P|Q &\equiv (\nu x:S)(P|Q) && \text{if } x^P \notin \text{fn}(Q) \\(\nu x:S)0 &\equiv 0 && \text{if } S = \text{end}\end{aligned}$$



# Operational semantics

$$x^p ! v . P \mid x^{\bar{p}}?(y:\mathbf{t}).Q \xrightarrow{x, \bar{-}} P \mid Q\{v/y\} \text{ [R-Comm]}$$

$$\frac{i \in I}{x^p \triangleleft \mathbf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathbf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathbf{l}_i} P \mid Q_i} \text{ [R-Select]}$$

$$\frac{P \xrightarrow{x, \mathbf{l}} P' \quad S \xrightarrow{x, \mathbf{l}} T}{(\nu x:\mathbf{S})P \xrightarrow{\tau, \bar{-}} (\nu x:\mathbf{T})P'} \text{ [R-NewS]}$$

$$\frac{P \xrightarrow{\alpha, \mathbf{l}} P' \quad \alpha \neq x}{(\nu x:\mathbf{S})P \xrightarrow{\alpha, \mathbf{l}} (\nu x:\mathbf{S})P'} \text{ [R-New]}$$

$$\frac{P \xrightarrow{\alpha, \mathbf{l}} P'}{P \mid Q \xrightarrow{\alpha, \mathbf{l}} P' \mid Q} \text{ [R-Par]}$$

$$\frac{P \equiv Q \quad Q \xrightarrow{\alpha, \mathbf{l}} Q' \quad Q' \equiv P'}{P \xrightarrow{\alpha, \mathbf{l}} P'} \text{ [R-Cong]}$$

## Type Judgement

$$\Gamma \vdash P$$

$P$  uses channels as specified by  $\Gamma$

## Environments $\Gamma$

- ▶ Partial function from polarized names to types
- ▶ Written  $x_1^{p_1} : t_1, x_2^{p_2} : t_2, \dots, x_n^{p_n} : t_n$
- ▶ Its satisfies one of the following conditions
  - ▶  $x^+, x^-, x \notin \text{dom}(\Gamma)$
  - ▶  $x \in \text{dom}(\Gamma)$  and  $x^+, x^- \notin \text{dom}(\Gamma)$
  - ▶  $x^p \in \text{dom}(\Gamma)$  and  $p \in \{+, -\}$  and  $x^{\bar{p}}, x \notin \text{dom}(\Gamma)$
  - ▶  $x^+, x^- \in \text{dom}(\Gamma)$  and  $x \notin \text{dom}(\Gamma)$

# Typing

$$x^+ : ?\text{int}.\text{!bool}.\text{end} \vdash x^+?(y:\text{int}).x^+!\text{true}.0$$
$$x^+ : ?\text{int}.\text{!bool}.\text{end} \not\vdash x^+?(y:\text{int}).x^+!y.0$$
$$x^+ : ?\text{int}.\text{end}, y^- : \text{!int}.\text{end} \vdash x^+?(z:\text{int}).y^-!z.0$$
$$x^+ : ?\text{int}.\text{end}, y^- : \text{!bool}.\text{end} \not\vdash 0$$
$$\vdash (\nu x:?\text{int}.\text{end})(x^+?(z:\text{int}).0 \mid x^-!1.0)$$
$$\not\vdash (\nu x:?\text{int}.\text{end})(x^+?(z:\text{int}).0)$$
$$\not\vdash (\nu x:?\text{int}.\text{end})(x^+?(z:\text{int}).0 \mid x^-!1.0 \mid x^-!2.0)$$

# Typing

$\nVdash (\nu x: ?\text{int}. ?\text{int}. \text{end})(x^+?(z:\text{int}). x^+?(z:\text{int}). 0 \mid x^-!1. 0 \mid x^-!2. 0)$

Think about

$(\nu x: ?\text{int}. !\text{int}. \text{end})($   
     $x^+?(z:\text{int}). x^+!(z+1). 0 \mid$   
     $x^+?(z:\text{int}). x^+!(z+1). 0 \mid$   
     $x^-!1. x^-?(z:\text{int}). Q_1 \mid$   
     $x^-!2. x^-?(z:\text{int}). Q_2 \quad )$

## Typing

$$\frac{\Gamma_1 \vdash P_1 \quad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2} [\text{T-Par}]$$

$$x^+ : \text{Tester}, x^- : \overline{\text{Tester}} \vdash P_{\text{server}} \mid P_{\text{client}}$$

where

**Tester** = ?int.!bool.end

$P_{\text{server}} = x^+?(y:\text{int}).x^+!\text{true}.0$  (*faulty*)

$P_{\text{client}} = x^-!1.x^-(z:\text{bool}).Q$

## Typing

$$\frac{\Gamma_1 \vdash P_1 \quad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2} [\text{T-Par}]$$

$$x^+ : \text{Tester}, x^- : \overline{\text{Tester}} \not\models P_{\text{server}} \mid P_{\text{client}} \mid P_{\text{client}}$$

where

`Tester` = ?`int`!`bool`.`end`

$P_{\text{server}} = x^+?(y:\text{int}).x^+!\text{true}.0$  (*faulty*)

$P_{\text{client}} = x^-!1.x^-(z:\text{bool}).Q$

# Typing

$$\frac{\Gamma_1 \vdash P_1 \quad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2} [\text{T-Par}]$$

## Context split

$$\begin{aligned}\Gamma + x^+ : t &= \Gamma, x^+ : t && \text{if } x, x^+ \notin \text{dom}(\Gamma) \\ \Gamma + x^- : t &= \Gamma, x^- : t && \text{if } x, x^- \notin \text{dom}(\Gamma) \\ (\Gamma, x : t) + x : t &= \Gamma, x : t && \text{if } t \text{ is not a session type}\end{aligned}$$

Extended on context as

$$\begin{aligned}\Gamma + \quad \quad \quad \emptyset &= \Gamma \\ \Gamma + (x^p : t, \Delta) &= (\Gamma + x^p : t) + \Delta\end{aligned}$$

Linear usage of session endpoints

# Typing

$$\frac{\Gamma_1 \vdash P_1 \quad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2} [\text{T-Par}]$$

$$\frac{\Gamma, x^+ : S, x^- : \bar{S} \vdash P}{\Gamma \vdash (\nu x : S) P} [\text{T-Res}]$$

$$(\nu x : \text{Tester})(P_{\text{server}} \mid P_{\text{client}})$$

where

`Tester = ?int.!bool.end`

$P_{\text{server}} = x^+?(y:\text{int}).x^+!\text{true}.0$  (*faulty*)

$P_{\text{client}} = x^-!1.x^-(z:\text{bool}).Q$



# Typing

$$\frac{\Gamma_1 \vdash P_1 \quad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2} \text{[T-Par]}$$

$$\frac{\Gamma, x^p : S, y : t \vdash P}{\Gamma, x^p : ?t.S \vdash x^p?(y:t).P} \text{[T-In]}$$

$$\frac{\Gamma, x^+ : S, x^- : \bar{S} \vdash P}{\Gamma \vdash (\nu x:S)P} \text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P}{\Gamma_1 + (\Gamma_2, x^p : !t.S) \vdash x^p!v.P} \text{[T-Out]}$$

Auxiliary Typing on expressions  $\Gamma \vdash v : t$

$$\begin{array}{ll} \emptyset \vdash \text{true} : \text{bool} & \emptyset \vdash \text{false} : \text{bool} \\ \emptyset \vdash () : \text{unit} & x^p : t \vdash x^p : t \end{array}$$

# Typing

$$\frac{\Gamma_1 \vdash P_1 \quad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2} [\text{T-Par}]$$

$$\frac{\Gamma, x^+ : S, x^- : \bar{S} \vdash P}{\Gamma \vdash (\nu x : S) P} [\text{T-Res}]$$

$$\frac{\Gamma, x^p : S, y : t \vdash P}{\Gamma, x^p : ?t.S \vdash x^p ?(y : t).P} [\text{T-In}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P}{\Gamma_1 + (\Gamma_2, x^p : !t.S) \vdash x^p !v.P} [\text{T-Out}]$$

$$\frac{\Gamma, x^p : S_j \vdash P \quad j \in I}{\Gamma, x^p : \oplus [\mathfrak{L}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathfrak{L}_I.P} [\text{T-Choice}]$$

$$\frac{\Gamma, x^p : S_i \vdash P_i \quad \forall i \in I}{\Gamma, x^p : \& [\mathfrak{L}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathfrak{L}_i : P_i]_{i \in I}} [\text{T-Branch}]$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0} [\text{T-Nil}]$$

$\Gamma$  completed if  $\Gamma(x^p) = S$  implies  $S = \text{end}$

## Terminology

We say  $P$  is *well-typed* if there exists  $\Gamma$  s.t.  $\Gamma \vdash P$

## Example

Is  $P$  well-typed?

- ▶  $P = x^+?(y:\text{int}).x^+!\text{true}.0$
- ▶ **Yes!** take  $\Gamma = x^+ : ?\text{int}.\text{!bool}.\text{end}$

$$\frac{\frac{\frac{\emptyset \vdash \text{true} : \text{bool} \quad \frac{x^+ : \text{end}, y : \text{int} \text{ completed}}{\quad} [\text{T-Nil}]}{\quad} [\text{T-Out}]}{\frac{x^+ : \text{!bool}.\text{end}, y : \text{int} \vdash x^+!\text{true}.0}{x^+ : ?\text{int}.\text{!bool}.\text{end} \vdash x^+?(y:\text{int}).x^+!\text{true}.0} [\text{T-In}]}$$

## Example

Is  $P$  well-typed?

- ▶  $P = x^+?(y:!bool.end).y!true.0$
- ▶ **Yes!** take  $\Gamma = x^+ : ?(!bool.end).end$

$$\frac{\frac{\frac{\emptyset \vdash true : bool}{x^+ : end, y : !bool.end \vdash y!true.0} [T-In]}{\frac{\frac{\frac{x^+ : end, y : end \text{ completed}}{\emptyset \vdash true : bool} [T-Nil]}{x^+ : end, y : end \vdash 0} [T-Out]} [T-Out]} x^+ : ?(!bool.end).end \vdash x^+?(y:!bool.end).y!true.0$$

## Example

Is  $P$  well-typed

►  $P = x^+?(y:!\text{int}.\text{end}).y!\text{true}.0$

No!

Try with  $\Gamma = x^+ : ?(!\text{int}.\text{end}).\text{end}$

$$\frac{\frac{\frac{\emptyset \not\vdash \text{true} : \text{int} \quad x^+ : \text{end}, y : \text{end} \text{ completed}}{x^+ : \text{end}, y : \text{end} \vdash 0} [\text{T-Nil}]}{x^+ : \text{end}, y : !\text{int}.\text{end} \vdash y!1.0} [\text{T-Out}]}{x^+ : ?(!\text{int}.\text{end}).\text{end} \vdash x^+?(y:!\text{int}.\text{end}).y!1.0} [\text{T-In}]$$

## Example

Is  $P$  well-typed?

►  $P = x^+?(y:\text{int.end}).y!1.0$

No! Try with  $\Gamma = x^+ : ?(\text{int.end}).\text{end}$

There is a mismatch:  $y : \text{int.end}$  and  $y!1.0$

$$\frac{\frac{x^+ : \text{end}, y : \text{int.end} \vdash y!1.0}{x^+ : ?(\text{int.end}).\text{end} \vdash x^+?(y:\text{int.end}).y!1.0} [\text{T-In}]}{\text{[T-Out]}}$$

## Rethinking $(\text{int} \rightarrow \text{bool}) \rightarrow \text{bool}$

$?(?\text{int}.\text{!bool.end}).\text{!bool.end}$  is not  $(\text{int} \rightarrow \text{bool}) \rightarrow \text{bool}$

$g : (\text{int} \rightarrow \text{bool}) \rightarrow \text{bool}$   
 $g\ f = f\ 1$

We may implement  $g$  as

$$P_g = x^+?(y:\textcolor{violet}{t}_f).y!1.y?(z:\textcolor{blue}{bool}).x^+!z.0$$

where  $t_f = ?\text{int}.\text{!bool.end}$

But

$$x^+ : ?t_f.\text{!bool.end} \not\vdash P_g$$

However

$$x^+ : ?\overline{t_f}.\text{!bool.end} \vdash P_g$$



What about  $?(!\text{int}.\text{?bool}.\text{end}).!\text{bool}.\text{end}?$

$g : (\text{int} \rightarrow \text{bool}) \rightarrow \text{bool}$   
 $g\ y = (y\ 1) \ \&\ (y\ 2)$

We may implement  $g$  as

$$P_g = x^+?(y:\overline{t_f}).y!1.y?(z_1:\text{bool}).y!2.y?(z_2:\text{bool}).x^+!z_1\&z_2.0$$

where  $t_f = ?\text{int}.\text{!bool}.\text{end}$

However

$$x^+ : ?\overline{t_f}.\text{!bool}.\text{end} \not\models P_g$$

The parameter  $y$  **must** be used just for **one** application

## On linearity

- ▶ Consider  $P = x^+ ! y^+ . y^+ ! 1 . 0$ .
- ▶ Does the following hold?

$$\Gamma, x^+ : !(!\text{int.end}).\text{end}, y^+ : !\text{int.end} \vdash P$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P}{\Gamma_1 + (\Gamma_2, x^p : !t.S) \vdash x^p ! v . P} [\text{T-Out}]$$

- ▶ **No.** Why does [T-Out] ban  $P$ ?
- ▶ Take

$$Q = (\nu y : !\text{int.end})(\nu x : !(!\text{int.end}).\text{end})(P \mid x^- ? (z : !\text{int.end}). z ! 2 . 0 \mid y^- ? (z : \text{int}). 0)$$

- ▶  $Q \xrightarrow{\tau, -} Q'$  where

$$Q' = (\nu y : !\text{int.end})(\nu x : \text{end})(y^+ ! 1 . 0 \mid y^+ ! 2 . 0 \mid y^- ? (z : \text{int}). 0)$$

where two processes concurrently send on  $y^+$

- ▶ A process does not use a session endpoint after *delegating* it (i.e., sending it over a different session endpoint)

## Theorem (Type Preservation)

- ▶ If  $\Gamma \vdash P$  and  $P \xrightarrow{\tau, \bar{\tau}} Q$  then  $\Gamma \vdash Q$ .
- ▶ If  $\Gamma, x^p : S, x^{\bar{p}} : \bar{S} \vdash P$  and  $P \xrightarrow{x, \bar{x}} Q$  then  $S \xrightarrow{x, \bar{x}} T$  and  $\Gamma, x^p : T, x^{\bar{p}} : \bar{T} \vdash Q$ .

## Theorem (Type Safety)

Let  $\Gamma \vdash P$  where  $\Gamma$  balanced<sup>2</sup>

- ▶ If  $P \equiv (\nu \tilde{z} : \tilde{S})(x^p ! v . P_1 \mid x^{\bar{p}} ? (y : \tilde{t}) . P_2 \mid Q)$  with  $p \in \{+, -\}$  then  $x, x^+, x^- \notin \text{fn}(Q)$  and  $\Gamma, \tilde{z} : \tilde{S} \vdash v : t$
- ▶ If  $P \equiv (\nu \tilde{z} : \tilde{S})(x^p \triangleleft \mathfrak{l}_j . R \mid x^{\bar{p}} \triangleright [\mathfrak{l}_i : P_i]_{i \in I} \mid Q)$  with  $p \in \{+, -\}$  then  $x, x^+, x^- \notin \text{fn}(Q)$  and  $j \in I$ .

---

<sup>2</sup> $\Gamma$  is balanced if  $x^p : S$  and  $x^{\bar{p}} : T$  implies  $S = \bar{T}$

# Properties

Does the following hold?

$$\vdash (\nu x:?\text{int}.\text{end})(x^+?(z:\text{int}).x^-!z.0)$$

Yes!

- ▶ The process is well-typed and deadlocked

The type system ensures

- ▶ *Type Safety* in communication (e.g., received values are of the expected type)
- ▶ *Session Fidelity* (e.g., communication follows the flow described by the session type)
- ▶ The type system does not ensure deadlock-freedom

# Deadlock

## Deadlocked Process

$$P = x^+?(z:\text{int}).y^-!1.0 \quad | \quad y^+?(z:\text{int}).x^-!1.0$$

Is  $P$  well-typed?

$$\vdash (\nu x:?\text{int}.\text{end})(x^+?(z:\text{int}).x^-!z.0)$$

Yes!

- ▶ The process is well-typed and deadlocked
- ▶ The type system does not check the dependencies between different sessions

## Deadlock-freedom by design (linear logic approaches)

- ▶ Connection drawn between linear logic and session-typed pi-calculus gave rise to type systems that guarantee deadlock-freedom
  - ▶ Luís Caires, Frank Pfenning: Session Types as Intuitionistic Linear Propositions. CONCUR 2010.
  - ▶ Philip Wadler: Propositions as sessions. ICFP 2012.
- ▶ The type system imposes some structural constraint on programs
  - ▶ two processes share at most one channel
  - ▶ Hence, there are no circular dependencies
- ▶ Key typing rule (presentation recast)

$$\frac{\Gamma_1, x^p : S \vdash P \quad \Gamma_1, x^{\bar{p}} : \bar{S} \vdash Q}{\Gamma_1, \Gamma_2 \vdash (\nu x:S)(P \mid Q)} \text{[T-Cut]}$$