

Binary Session Types

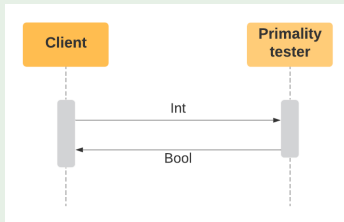
Hernán Melgratti

ICC University of Buenos Aires-Conicet

Informally

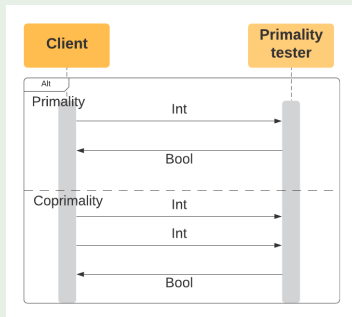
- ▶ A session type defines a communication protocol
- ▶ In the binary case, it describes the messages exchanged between two parties

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**

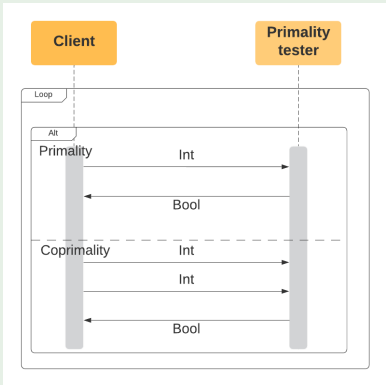
Choices



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

Informally

Infinite interactions



- ```

▶ 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 = ?open.Opened`

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

### Client

`Client = !open.Reading`

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

# Syntax of Types

## Session Types

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

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

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

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

$$\overline{\&[\iota_i : T_i]_{i \in I}} = \oplus[\iota_i : \overline{T_i}]_{i \in I}$$

$$\overline{\oplus[\iota_i : T_i]_{i \in I}} = \&[\iota_i : \overline{T_i}]_{i \in I}$$

## 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

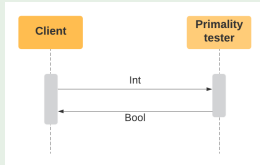
---

<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)

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

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

- ▶ 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                                            |
|            | <code>true, false</code> | boolean values                                        |
|            | $\dots$                  | expressions                                           |

## Processes

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

# Syntax of Types

## Session Types

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

## Operational semantics

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

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

$$\blacktriangleright (P, \alpha, \mathfrak{l}, Q) \in \longrightarrow$$

- ▶ means  $P$  evolves to  $Q$  after communicating the choice  $\mathfrak{l}$  on the session  $\alpha$
- ▶ is abbreviated as  $P \xrightarrow{\alpha, \mathfrak{l}} Q$

$$\blacktriangleright \tau \text{ stands for a hidden session}$$

$$\blacktriangleright - \text{ for no choice}$$

## 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

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

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

## Free names

*fn*

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

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

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

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

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

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

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

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

# 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

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

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

## 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

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

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

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

## Semantics of Types

$$? t . S \xrightarrow{-} S$$

$$! t . S \xrightarrow{-} S$$

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

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

# Operational semantics

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

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

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

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

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

## 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

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

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

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

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

$$(\nu x : \mathbf{S})P \xrightarrow{\alpha, \mathbf{l}} (\nu x : \mathbf{S})P'$$

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

$$P|Q \xrightarrow{\alpha, \mathbf{l}} P'|Q$$

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