

# Binary Session Types

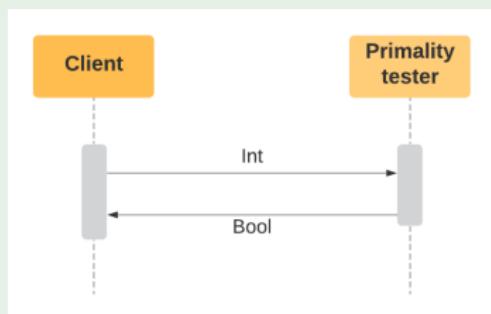
Hernán Melgratti

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

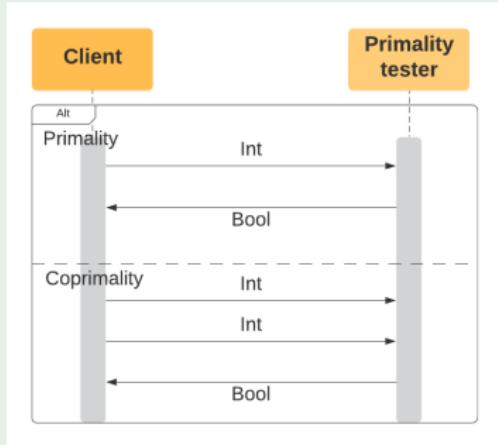
### First example



- ▶ 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**

## Informally

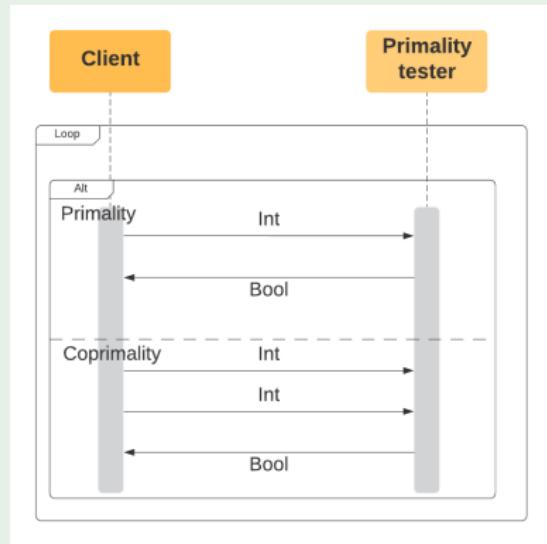
### Choices



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

## Informally

## Infinite interactions



## 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

$S, T ::=$	$\text{end}$	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
$s, t ::=$	$S$	A session type
	$\text{int, bool}$	basic types
	$\dots$	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

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

$f = ?\text{int}.!\text{bool.end}$   
 $g = ?f.!\text{bool.end}$

It resembles

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

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

## File

$\text{File} = ?\text{open}.Opened$

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

## Function that processes a file

$\text{Client}_1 = !(\text{File}).?\text{int.end}$   
 $\text{Client}_2 = !(\text{Opened}).?\text{int.end}$

## Duality

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

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

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

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

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

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

# 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

We choose<sup>1</sup>

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

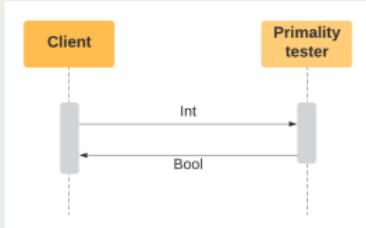
---

<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)  
 $P_{\text{server}} = x^+?(y:\text{int}).x^+!\text{true}.0 \quad (\text{faulty})$   
 $P_{\text{client}} = x^-!1.x^-?(z:\text{bool}).0$
- ▶ The system is the parallel composition of the two processes

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

# Syntax of Processes

## Polarities

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

## Values (more in general expressions)

$v, w ::= \begin{array}{ll} x^p, y^q, \dots & (\text{polarised}) \text{ variables } \mathcal{X} = \{x, y, \dots\} \\ | () & \text{unit value} \\ | \text{true, false} & \text{boolean values} \\ | \dots & \text{expressions} \end{array}$

## Processes

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

# Syntax of Types

## Session Types

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

- ▶  $\rightarrow \subseteq P \times (\mathcal{X} \cup \{\tau\}) \times (\mathcal{L} \cup \{-\}) \times P$
  
- ▶  $(P, \alpha, l, Q) \in \rightarrow$ 
  - ▶ means  $P$  evolves to  $Q$  after communicating the choice  $l$  on the session  $\alpha$
  - ▶ is abbreviated as  $P \xrightarrow[\alpha;l]{} Q$
- ▶  $\tau$  stands for a hidden session
- ▶  $-$  for no choice

# Operational semantics

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

## Substitution

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

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

## Free names

fn

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

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

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

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

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

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

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

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

# Operational semantics

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

## Substitution

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

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

## Operational semantics

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

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

## Operational semantics

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

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

## Operational semantics

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

$$\frac{\begin{array}{c} p \in \{+, -\} \\[1ex] i \in I \end{array}}{x^p \triangleleft \mathsf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathsf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathsf{l}_i} P \mid Q_i} \text{ [R-Select]}$$
$$\frac{P \xrightarrow{x, \mathsf{l}} P' \quad S \xrightarrow{\mathsf{l}} T}{(\nu x : \textcolor{violet}{S}) P \xrightarrow{\tau, -} (\nu x : \textcolor{violet}{T}) P'} \text{ [R-NewS]}$$

Semantics of Types

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

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

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

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

# Operational semantics

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

$$\frac{\begin{array}{c} p \in \{+, -\} \\[1ex] i \in I \end{array}}{x^p \triangleleft \mathsf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathsf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathsf{l}_i} P \mid Q_i} \text{ [R-Select]}$$
$$\frac{P \xrightarrow{x, \mathsf{l}} P' \quad S \xrightarrow{\mathsf{l}} T}{(\nu x : \textcolor{violet}{S}) P \xrightarrow{\tau, -} (\nu x : \textcolor{violet}{T}) P'} \text{ [R-NewS]}$$

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

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

## Structural equivalence

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

# Operational semantics

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

$$\frac{i \in I}{x^p \triangleleft \mathbf{l}_i . P \mid x^{\bar{p}} \triangleright [\mathbf{l}_j : Q_j]_{j \in I} \xrightarrow{x, \mathbf{l}_i} P \mid Q_i} \text{ [R-Select]}$$
$$\frac{P \xrightarrow{x, \mathbf{l}} P' \quad S \xrightarrow{\mathbf{l}} T}{(\nu x : \textcolor{violet}{S})P \xrightarrow{\tau, -} (\nu x : \textcolor{violet}{T})P'} \text{ [R-NewS]}$$
$$\frac{P \xrightarrow{\alpha, \mathbf{l}} P' \quad \alpha \neq x}{(\nu x : \textcolor{violet}{S})P \xrightarrow{\alpha, \mathbf{l}} (\nu x : \textcolor{violet}{S})P'} \text{ [R-New]}$$
$$\frac{P \xrightarrow{\alpha, \mathbf{l}} P'}{P | Q \xrightarrow{\alpha, \mathbf{l}} P' | Q} \text{ [R-Par]}$$
$$\frac{P \equiv Q \quad Q \xrightarrow{\alpha, \mathbf{l}} Q' \quad Q' \equiv P'}{P \xrightarrow{\alpha, \mathbf{l}} P'} \text{ [R-Cong]}$$