Introduction to (Finite) Binary Session Types

Hernán Melgratti

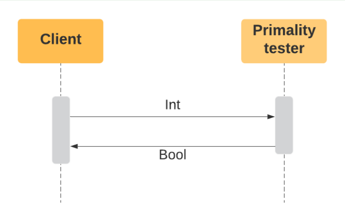
ICC University of Buenos Aires-Conicet

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

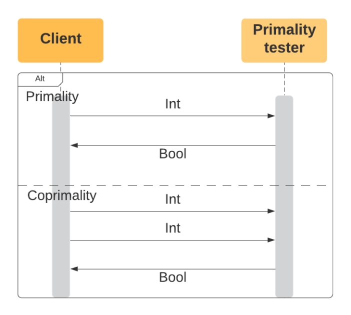


- We rely on a textual description; the flow is described from the point of view of one of the participants
- `Tester` $=$ `?int.!bool.end`

  `?`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

Informally

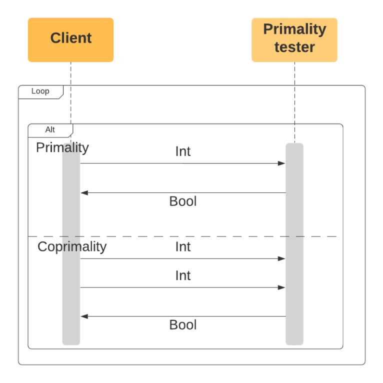
Choices



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

Informally

Infinite interactions



- $\mathtt{Tester} = \&[\mathtt{Pr} : ?\mathtt{int}.!\mathtt{bool}.\mathtt{Tester},\ \mathtt{Co} : ?\mathtt{int}.?\mathtt{int}.!\mathtt{bool}.\mathtt{Tester}]$
- $\mathtt{Client} = \oplus[\mathtt{Pr} : !\mathtt{int}.?\mathtt{bool}.\mathtt{Client},\ \mathtt{Co} : !\mathtt{int}.!\mathtt{int}.?\mathtt{bool}.\mathtt{Client}]$

Modelling a function

f : int $\rightarrow$ bool

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

Invocation

$\texttt{inv} = \texttt{!int.?bool.end}$

Modelling an object (Typestate)

File

$$\mathtt{File} = ?\mathtt{mode}.\mathtt{Opened}$$

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

Client

$$\mathtt{Client} = !\mathtt{mode}.\mathtt{Reading}$$

$$\mathtt{Reading} = \oplus[\mathit{read} : \&[\mathit{eof} : \mathtt{Reading}, \mathit{val} : !\mathtt{string}.\mathtt{Reading}], \mathit{close} : \mathtt{end}]$$

Syntax of Types

Session Types

$S, T ::=$		$\mathtt{end}$	terminated session
	$\mid$	$?t.S$	receive (input)
	$\mid$	$!t.S$	send (output)
	$\mid$	$\&[\mathtt{l}_i : T_i]_{i \in I}$	branch
	$\mid$	$\oplus[\mathtt{l}_i : T_i]_{i \in I}$	select
	$\mid$	$\mu X.S$	recursive session type
	$\mid$	X	session type variable
$s, t ::=$		S	A session type
	$\mid$	$\mathtt{int}, \mathtt{bool}$	basic types
	$\mid$	...	other types
$\mathcal{L} \;=$		$\{\mathtt{l}, \mathtt{l}_1, \ldots\}$	Set of labels

Remark

- The grammar allows terms like $?S.T$
- For instance, $?(?\mathtt{int}.\mathtt{end}).!\mathtt{bool}.\mathtt{end}$ vs $?\mathtt{int}.!\mathtt{bool}.\mathtt{end}$

Examples

$f : \texttt{int} \to \texttt{bool}$

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

It resembles

$$g : (\texttt{int} \to \texttt{bool}) \to \texttt{bool}$$

but it is not the same (more to come)

File

$$\texttt{File} = \texttt{?mode.Opened}$$

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

Function that processes a file

$$\texttt{Client}_1 = \texttt{!File.?int.end}$$
$$\texttt{Client}_2 = \texttt{!Opened.?int.end}$$

Duality

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

$$
\begin{aligned}
\overline{\mathsf{end}} &= \mathsf{end} \\
\overline{?t.S} &= !t.\overline{S} \\
\overline{!t.S} &= ?t.\overline{S} \\
\overline{\&[\mathtt{l}_i : T_i]_{i\in I}} &= \oplus[\mathtt{l}_i : \overline{T_i}]_{i\in I} \\
\overline{\oplus[\mathtt{l}_i : T_i]_{i\in I}} &= \&[\mathtt{l}_i : \overline{T_i}]_{i\in I}
\end{aligned}
$$

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 [1]

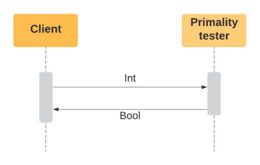
1. A language with message-passing communication based on synchronous channels
2. Session types are associated with channels

[1] 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



$$\texttt{Tester} = \texttt{?int.!bool.end}$$

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

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

  $P_{\texttt{client}} = x^-!1.x^-?(z{:}\texttt{bool}).Q$
- The system is the parallel composition of the two processes

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

Syntax of Processes

Polarities

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

Values (more in general expressions)

$v, w ::=$	$x^p, y^q, \ldots$	(polarised) variables $\mathcal{X} = \{x, y, \ldots\}$
$\mid$	$()$	unit vaue
$\mid$	$\mathtt{true}, \mathtt{false}$	boolean values
$\mid$	$\ldots$	expressions

Processes

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

Syntax of Types

Session Types

$S, T ::=$		$\mathtt{end}$	terminated session
	$\mid$	$?t.S$	receive (input)
	$\mid$	$!t.S$	send (output)
	$\mid$	$\&[\mathtt{l}_i : T_i]_{i \in I}$	branch
	$\mid$	$\oplus[\mathtt{l}_i : T_i]_{i \in I}$	select
	$\mid$	$\mu X.S$	recursive session type
	$\mid$	X	session type variable
$s, t ::=$		S	A session type
	$\mid$	$\mathtt{int}, \mathtt{bool}$	basic types
	$\mid$	$\ldots$	other types

Notation

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

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

- we identify x^{ϵ} 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

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

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

Operational semantics

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

Substitution

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

Free names

fn

$$
\begin{aligned}
\mathsf{fn}(\mathtt{true}) = \mathsf{fn}(\mathtt{false}) \;&=\; \mathsf{fn}(()) \;=\; \emptyset \\
\mathsf{fn}(x^p) &= \{x^p\} \\
\\
\mathsf{fn}(0) &= \emptyset \\
\mathsf{fn}(P|Q) &= \mathsf{fn}(Q) \cup \mathsf{fn}(P) \\
\mathsf{fn}(x^p\,\mathtt{?}(y{:}\mathtt{t})\,\mathtt{.}\,P) &= \{x^p\} \cup (\mathsf{fn}(P) \setminus \{y\}) \\
\mathsf{fn}(x^p\,\mathtt{!}\,v\,\mathtt{.}\,P) &= \{x^p\} \cup \mathsf{fn}(v) \cup \mathsf{fn}(P) \\
\mathsf{fn}(x^p \triangleright [\mathtt{l}_i : P_i]_{i \in I}) &= \{x^p\} \cup (\bigcup_i \mathsf{fn}(P_i)) \\
\mathsf{fn}(x^p \triangleleft \mathtt{l}\,\mathtt{.}\,P) &= \{x^p\} \cup \mathsf{fn}(P) \\
\mathsf{fn}((\nu x{:}S)P) &= \mathsf{fn}(P) \setminus \{x, x^+, x^-\}
\end{aligned}
$$

Operational semantics

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

Substitution

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

Operational semantics

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

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

Operational semantics

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

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

Operational semantics

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

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

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

Semantics of Types

$$?t \,.\, S \xrightarrow{-} S \qquad !t \,.\, S \xrightarrow{-} S$$

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

Operational semantics

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

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

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

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

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

Structural equivalence

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

Operational semantics

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

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

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

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

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

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