Binary Session Types

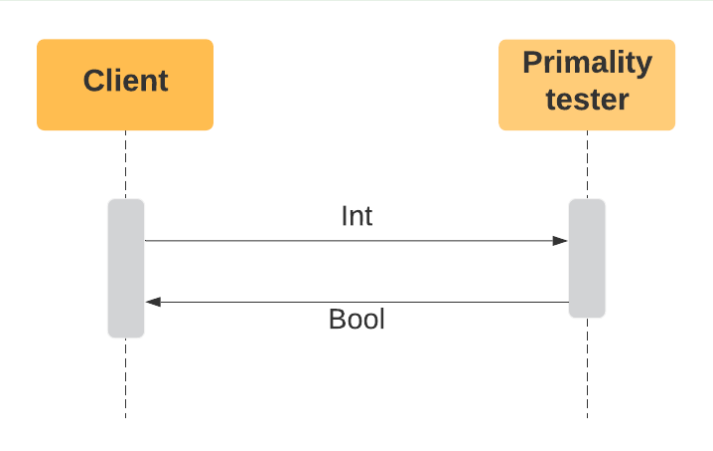
Hernán Melgratti

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



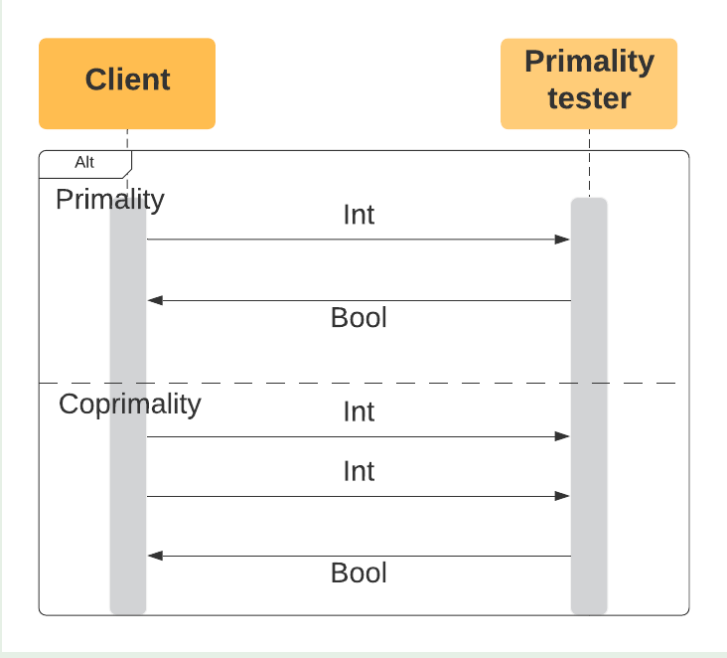
- We rely on a textual description; the flow is described from the point of view of one of the participants
- `Tester` = `?int.!bool.end`

  `?`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

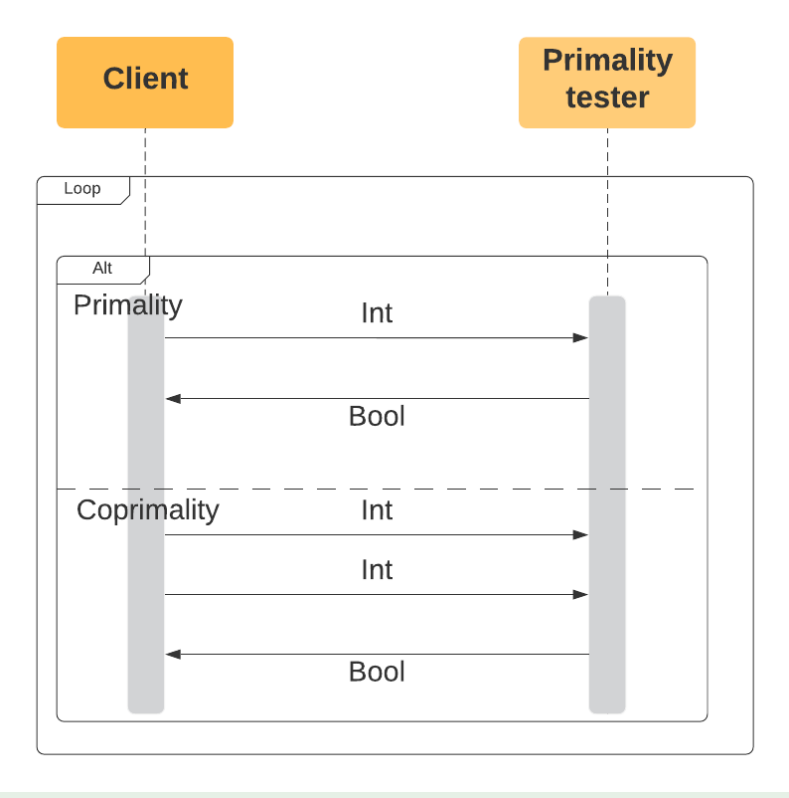
Choices



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



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

Modelling a function

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

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

Invocation

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

Modelling an object (Typestate)

File

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

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

Client

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

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

Syntax of Types

Session Types

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

Remark

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

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

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

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

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]

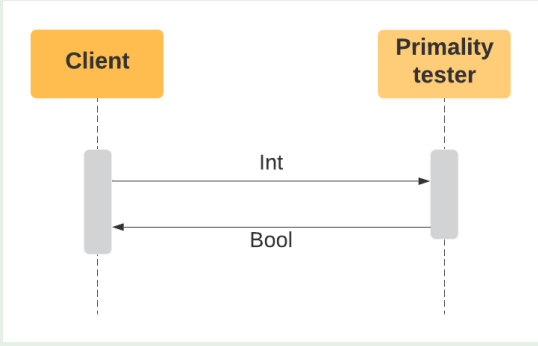
1. A language with message-passing communication based on synchronous channels
2. Session types are associated with channels

[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)
  $P_{\texttt{server}} = x^+\texttt{?}(y\texttt{:int}).x^+\texttt{!true.0}$ *(faulty)*
  $P_{\texttt{client}} = x^-\texttt{!1}.x^-\texttt{?}(z\texttt{:bool}).\texttt{Q}$
- 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 value
$\mid$	$\texttt{true}, \texttt{false}$	boolean values
$\mid$	$\ldots$	expressions

Processes

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

Syntax of Types

Session Types

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