Session Types Revisited

Ornela Dardha

School of Computing Science
University of Glasgow

January 10, 2014

(Joint work with Elena Giachino and Davide Sangiorgi)

Gentle Intro

- In complex distributed systems communicating participants agree on a protocol to follow, specifying *type* and *direction* of data exchanged.
- Session types are a formalism to model structured communication-based programming.
- Designed for
  - process calculi
  - multithreaded functional languages
  - object-oriented languages
  - ...
- Guarantee *privacy*, *communication safety* and *session fidelity*.

Example of Session Types

Distributed Auction System:
sellers, that want to sell items,
auctioneers, that sell items on their behalf,
bidders, that bid for an item being auctioned.

$$\text{seller}: \oplus\{\textbf{selling} : !\mathit{Item}.!\mathit{Price}.\&\{\textbf{sold} : ?\mathit{Price}.\mathit{end}, \textbf{not} : \mathit{end}\}\}$$

$$\text{auctioneer}: \&\{\textbf{selling} : ?\mathit{Item}.?\mathit{Price}.\oplus\{\textbf{sold} : !\mathit{Price}.\mathit{end}, \textbf{not} : \mathit{end}\}, \textbf{register} : ?\mathit{Id}.!\mathit{Item}.?\mathit{Bid}.\mathit{end}\}$$

$$\text{bidder}: \oplus\{\textbf{register} : !\mathit{Id}.?\mathit{Item}.!\mathit{Bid}.\mathit{end}\}$$

Key words for sessions

1. Sequentiality of input/output operations explicitly indicating type of data transmitted.
   (Guarantees *session fidelity*)

Key words for sessions

1. Sequentiality of input/output operations explicitly indicating type of data transmitted.
(Guarantees *session fidelity*)
2. Duality of session types corresponding to opposite endpoints of a session channel.
(Guarantees *communication safety*)

Key words for sessions

1. Sequentiality of input/output operations explicitly indicating type of data transmitted.
   (Guarantees *session fidelity*)
2. Duality of session types corresponding to opposite endpoints of a session channel.
   (Guarantees *communication safety*)
3. Connection establishes a fresh session channel between two parties. (Guarantees *privacy*)

Background on π types

- Channel type $\sharp T$: types a channel used in input or output to transmit values of type T, many times.

Background on π types

- Channel type $\sharp T$: types a channel used in input or output to transmit values of type T, many times.
- Input/output channel type iT/oT: types a channel used *only* in input/output to transmit values of type T, many times.

Background on π types

- Channel type $\sharp T$: types a channel used in input or output to transmit values of type T, many times.
- Input/output channel type iT/oT: types a channel used *only* in input/output to transmit values of type T, many times.
- Linear input/output type $\ell_i T/\ell_o T$: types a channel used *only* in input/output and *exactly once* to transmit values of type T.

Background on π types

- Channel type $\sharp T$: types a channel used in input or output to transmit values of type T, many times.
- Input/output channel type iT/oT: types a channel used *only* in input/output to transmit values of type T, many times.
- Linear input/output type $\ell_i T/\ell_o T$: types a channel used *only* in input/output and *exactly once* to transmit values of type T.
- Linearized channel: linear channel used *multiple times* but only in a sequential manner $(?, !)$ and with the same carried type.

Background on π types

- Channel type $\sharp T$: types a channel used in input or output to transmit values of type T, many times.
- Input/output channel type iT/oT: types a channel used *only* in input/output to transmit values of type T, many times.
- Linear input/output type $\ell_i T/\ell_o T$: types a channel used *only* in input/output and *exactly once* to transmit values of type T.
- Linearized channel: linear channel used *multiple times* but only in a sequential manner $(?, !)$ and with the same carried type.
- Variant type: labelled disjoint union of types $(\&, \oplus)$.

Key words for π

We saw:

1. Sequentiality
2. Duality
3. Connection

Key words for π

We saw:

1. Sequentiality
2. Duality
3. Connection

1. Linearity forces a π channel to be used exactly once.

Key words for π

We saw:

1. Sequentiality
2. Duality
3. Connection

1. Linearity forces a π channel to be used exactly once.
2. Capability of input/output of the same π channel split between two partners.

Key words for π

We saw:

1. Sequentiality
2. Duality
3. Connection

1. Linearity forces a π channel to be used exactly once.
2. Capability of input/output of the same π channel split between two partners.
3. Restriction construct permits the creation of fresh private π channels.

Standard π- types

$\tau ::=$	$\emptyset[\widetilde{T}]$	channel with no capability
	$\ell_{\mathtt{i}}\ [\widetilde{T}]$	linear input
	$\ell_{\mathtt{o}}\ [\widetilde{T}]$	linear output
	$\ell_{\sharp}\ [\widetilde{T}]$	linear connection
$T ::=$	τ	linear channel type
	$\langle l_i\_T_i \rangle_{i \in I}$	variant type
	$\sharp T$	standard channel type
	$\mathtt{Bool}$	boolean type
	$\cdots$	other constructs

Standard π- processes

$P, Q ::=$	$\mathbf{0}$	inaction
	$x!\langle\tilde{v}\rangle.P$	output
	$x?(\tilde{y}).P$	input
	$P \mid Q$	composition
	$(\boldsymbol{\nu} x)P$	channel restriction
	$\textbf{case}\ v\ \textbf{of}\ \{l_i\_x_i \triangleright P_i\}_{i \in I}$	case process
$v ::=$	x	variable
	b	boolean values
	l_v	variant value

Semantics

Just to understand *case normalisation*...

(Rπ- Com)

$$x!\langle\tilde{v}\rangle.P \mid x?(\tilde{z}).Q \rightarrow P \mid Q[\tilde{v}/\tilde{z}]$$

(Rπ- Case)

$$\textbf{case}\; l_j\_v \;\textbf{of}\; \{l_i\_x_i \triangleright P_i\}_{i \in I} \rightarrow P_j[v/x_j] \quad j \in I$$

Session Types

$$
\begin{array}{rlll}
S & ::= & \mathtt{end} & \text{termination} \\
 & & !T.S & \text{send} \\
 & & ?T.S & \text{receive} \\
 & & \oplus\{l_i : S_i\}_{i \in I} & \text{select} \\
 & & \&\{l_i : S_i\}_{i \in I} & \text{branch} \\
 & & & \\
T & ::= & S & \text{session type} \\
 & & \sharp T & \text{standard channel type} \\
 & & \mathtt{Bool} & \text{boolean type} \\
 & & \cdots & \text{other constructs}
\end{array}
$$

Session Processes

$P, Q ::=$	$\mathbf{0}$	inaction
	$x!\langle v\rangle.P$	output
	$x?(y).P$	input
	$x \triangleleft l_j.P$	selection
	$x \triangleright \{l_i : P_i\}_{i\in I}$	branching
	$P \mid Q$	composition
	$(\boldsymbol{\nu} xy)P$	session restriction
	$(\boldsymbol{\nu} x)P$	channel restriction
$v ::=$	x	variable
	b	boolean values

Types Encoding

$$
\begin{aligned}
[\![\mathtt{Bool}]\!] &\stackrel{\mathtt{def}}{=} \mathtt{Bool}\\
[\![\mathtt{end}]\!] &\stackrel{\mathtt{def}}{=} \emptyset[\;]\\
[\![!T.S]\!] &\stackrel{\mathtt{def}}{=} \ell_{\mathtt{o}}\ [[\![T]\!],[\![\overline{S}]\!]]\\
[\![?T.S]\!] &\stackrel{\mathtt{def}}{=} \ell_{\mathtt{i}}\ [[\![T]\!],[\![S]\!]]\\
[\![\oplus\{l_i : T_i\}_{i\in I}]\!] &\stackrel{\mathtt{def}}{=} \ell_{\mathtt{o}}\ [\langle l_i\_[\![\overline{T_i}]\!]\rangle_{i\in I}]\\
[\![\&\{l_i : T_i\}_{i\in I}]\!] &\stackrel{\mathtt{def}}{=} \ell_{\mathtt{i}}\ [\langle l_i\_[\![T_i]\!]\rangle_{i\in I}]
\end{aligned}
$$

Example of Encoding: Types 1/2

Let $x : T$ and $y : \overline{T}$ where

$$T = ?\mathtt{Int}.?\mathtt{Int}.!\mathtt{Bool}.\mathtt{end}$$

and

$$\overline{T} = !\mathtt{Int}.!\mathtt{Int}.?\mathtt{Bool}.\mathtt{end}$$

Example of Encoding: Types 2/2

The encoding of these types is as follows:

$$[\![T]\!] = \ell_{\mathtt{i}}\ [\mathtt{Int}, \ell_{\mathtt{i}}\ [\mathtt{Int}, \ell_{\mathtt{o}}\ [\mathtt{Bool}, \emptyset[]]]]$$

and

$$[\![\overline{T}]\!] = \ell_{\mathtt{o}}\ [\mathtt{Int}, \ell_{\mathtt{i}}\ [\mathtt{Int}, \ell_{\mathtt{o}}\ [\mathtt{Bool}, \emptyset[]]]]$$

Example of Encoding: Types 2/2

The encoding of these types is as follows:

$$[\![T]\!] = \ell_{\mathtt{i}}\ [\mathtt{Int}, \ell_{\mathtt{i}}\ [\mathtt{Int}, \ell_{\mathtt{o}}\ [\mathtt{Bool}, \emptyset[]]]]$$

and

$$[\![\overline{T}]\!] = \ell_{\mathtt{o}}\ [\mathtt{Int}, \ell_{\mathtt{i}}\ [\mathtt{Int}, \ell_{\mathtt{o}}\ [\mathtt{Bool}, \emptyset[]]]]$$

NB

duality on session types boils down to opposite capabilities (i/o) of channel types, only in the outermost level!

Terms Encoding

$$
\begin{array}{rcl}
[\![x]\!]_f & \stackrel{\text{def}}{=} & f_x \\
[\![b]\!]_f & \stackrel{\text{def}}{=} & b \\
[\![\mathbf{0}]\!]_f & \stackrel{\text{def}}{=} & \mathbf{0} \\
[\![x!\langle v\rangle.P]\!]_f & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} c)f_x!\langle v, c\rangle.[\![P]\!]_{f,\{x\mapsto c\}} \\
[\![x?(y).P]\!]_f & \stackrel{\text{def}}{=} & f_x?(y, c).[\![P]\!]_{f,\{x\mapsto c\}} \\
[\![x \triangleleft l_j.P]\!]_f & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} c)f_x!\langle l_j\_c\rangle.[\![P]\!]_{f,\{x\mapsto c\}} \\
[\![x \triangleright \{l_i : P_i\}_{i\in I}]\!]_f & \stackrel{\text{def}}{=} & f_x?(y).\ \textbf{case}\ y\ \textbf{of}\ \{l_i\_c \triangleright [\![P_i]\!]_{f,\{x\mapsto c\}}\}_{i\in I} \\
[\![P \mid Q]\!]_f & \stackrel{\text{def}}{=} & [\![P]\!]_f \mid [\![Q]\!]_f \\
[\![(\boldsymbol{\nu} xy)P]\!]_f & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} c)[\![P]\!]_{f,\{x,y\mapsto c\}}
\end{array}
$$

Example of Encoding: Terms 1/2

$$\texttt{server} \stackrel{\text{def}}{=} x?(nr1).x?(nr2).x!\langle nr1 == nr2\rangle.\mathbf{0}$$

$$\texttt{client} \stackrel{\text{def}}{=} y!\langle 3\rangle.y!\langle 5\rangle.y?(eq).\mathbf{0}$$

The system is given by

$$(\boldsymbol{\nu} xy)\ \big(\texttt{server} \mid \texttt{client}\big)$$

The encoding of the above system is

$$[\![(\boldsymbol{\nu} xy)\ \big(\texttt{server} \mid \texttt{client}\big)]\!]_f = (\boldsymbol{\nu}\mathrm{z})\ [\![\big(\texttt{server} \mid \texttt{client}\big)]\!]_{f,\{\mathrm{x,y}\mapsto\mathrm{z}\}}$$

Example of Encoding: Terms 2/2

Where the encodings of `server` and `client` processes are as follows:

$$[\![\texttt{server}]\!]_{f,\{\mathrm{x,y}\mapsto \mathrm{z}\}} \stackrel{\texttt{def}}{=} z?(nr1, c).c?(nr2, c').(\boldsymbol{\nu} c'')c!\langle nr1 == nr2, c''\rangle.\mathbf{0}$$

$$[\![\texttt{client}]\!]_{f,\{\mathrm{x,y}\mapsto \mathrm{z}\}} \stackrel{\texttt{def}}{=} (\boldsymbol{\nu} c)z!\langle 3, c\rangle.(\boldsymbol{\nu} c')c!\langle 5, c'\rangle.c'?(eq, c'').\mathbf{0}$$

Output actions create new channels c, c', c'' which are sent to the communicating party along with the value.

Guaranteeing Communication Properties

- Privacy is guaranteed because a channel is used *at most* once.
- Communication safety is guaranteed because a channel is used *at least* once.
- Session fidelity is guaranteed because of continuation-passing.

Theorem (Correctness of the Encoding)

$\Gamma \vdash P$ *if and only if* $[\![\Gamma]\!]_f \vdash [\![P]\!]_f$.

Theorem (Correctness of the Encoding)

$\Gamma \vdash P$ *if and only if* $[\![\Gamma]\!]_f \vdash [\![P]\!]_f$.

Theorem (Operational Correspondence)

Let P be a process in the π-calculus with sessions. The following hold.

1. *If $P \to P'$ then $\exists\ Q$ such that $[\![P]\!]_f \to Q$ and $Q \hookrightarrow [\![P']\!]_f$, where $\hookrightarrow$ denotes a structural congruence* possibly *extended with a* case normalisation.
2. *If $[\![P]\!]_f \to\equiv Q$ then, $\exists\ P'$ such that $P \to P'$ and $Q \to^{*}\equiv [\![P']\!]_{f'}$.*

Theorem (Correctness of the Encoding)

$\Gamma \vdash P$ *if and only if* $[\![\Gamma]\!]_f \vdash [\![P]\!]_f$.

Theorem (Operational Correspondence)

Let P be a process in the π-calculus with sessions. The following hold.

1. *If $P \to P'$ then $\exists\ Q$ such that $[\![P]\!]_f \to Q$ and $Q \hookrightarrow [\![P']\!]_f$, where $\hookrightarrow$ denotes a structural congruence* possibly *extended with a* case normalisation.
2. *If $[\![P]\!]_f \to\equiv Q$ then, $\exists\ P'$ such that $P \to P'$ and $Q \to^{*}\equiv [\![P']\!]_{f'}$.*

Corollary

Subject Reduction and Type Soundness on session types.

Extensions of the Encoding of Sessions

Subtyping

Theorem

$T <: T'$ if and only if $[\![T]\!] \leq [\![T']\!]$.

Subtyping

Theorem

$T <: T'$ if and only if $[\![T]\!] \leq [\![T']\!]$.

Derived from the encoding:

- Reflexivity and Transitivity of Subtyping.
- Lemmas (ex. Substitution, Narrowing...) follow from the corresponding ones in π.
- Nothing to prove for other type constructs added.

Encoding Parametric Polymorphism

$$\begin{aligned} [\![X]\!] &\stackrel{\text{def}}{=} X \\ [\![\langle X; T\rangle]\!] &\stackrel{\text{def}}{=} \langle X; [\![T]\!]\rangle \end{aligned}$$

$$\begin{aligned} [\![\langle T; v\rangle]\!]_f &\stackrel{\text{def}}{=} \langle [\![T]\!]; f_v\rangle \\ [\![\textbf{open } v \textbf{ as } (X; x) \textbf{ in } P]\!]_f &\stackrel{\text{def}}{=} \textbf{open } f_v \textbf{ as } (X; f_x) \textbf{ in } [\![P]\!]_f \end{aligned}$$

Encoding Bounded Polymorphism

$$
\begin{aligned}
[\![B]\!] &\stackrel{\texttt{def}}{=} B \\
[\![\oplus\{l_i(X_i <: B_i) : T_i\}_{i\in I}]\!] &\stackrel{\texttt{def}}{=} \ell_{\texttt{o}}\ [\langle l_i(X_i \leq B_i)\_[\![\overline{T_i}]\!]\rangle_{i\in I}] \\
[\![\&\{l_i(X_i <: B_i) : T_i\}_{i\in I}]\!] &\stackrel{\texttt{def}}{=} \ell_{\texttt{i}}\ [\langle l_i(X_i \leq B_i)\_[\![T_i]\!]\rangle_{i\in I}]
\end{aligned}
$$

$$
\begin{aligned}
[\![x \lhd l_j(B).P]\!]_f &\stackrel{\texttt{def}}{=} (\boldsymbol{\nu} c) f_x!\langle l_j(B)\_c\rangle.[\![P]\!]_{f,\{x\mapsto c\}} \\
[\![x \rhd \{l_i(X_i <: B_i) : P_i\}_{i\in I}]\!]_f &\stackrel{\texttt{def}}{=} f_x?(y). \\
&\quad \textbf{case } y \textbf{ of } \{l_i(X_i \leq B_i)\_c \rhd [\![P_i]\!]_{f,\{x\mapsto c\}}\}
\end{aligned}
$$

Parametric and Bounded Polymorphism

Theorem (Correctness of Typing Unpacking)

$\Gamma; \Delta \vdash \textbf{open}\ v\ \textbf{as}\ (X; x)\ \textbf{in}\ P$ *if and only if*
$[\![\Gamma; \Delta]\!]_f \vdash [\![\textbf{open}\ v\ \textbf{as}\ (X; x)\ \textbf{in}\ P]\!]_f$.

Parametric and Bounded Polymorphism

Theorem (Correctness of Typing Unpacking)

$\Gamma; \Delta \vdash \textbf{open } v \textbf{ as } (X; x) \textbf{ in } P$ *if and only if*
$[\![\Gamma; \Delta]\!]_f \vdash [\![\textbf{open } v \textbf{ as } (X; x) \textbf{ in } P]\!]_f$.

Theorem (Correctness of Typing Bounded Polymorphic Processes)

$\Gamma; \Delta \vdash Q$ *if and only if* $[\![\Gamma; \Delta]\!]_f \vdash [\![Q]\!]_f$, *where either*
$Q = x \triangleleft l_j(B).P$, *or* $Q = x \triangleright \{l_i(X_i \leq B_i) : P_i\}_{i \in I}$

Observations

Derived from the encoding:

- Modification in the calculus as expected
- Encoding of polymorphism constructs: an homomorphism.
- Again, Subject Reduction and Type Soundness derived for free (considering just the constructs added).

Higher-Order

$$
\begin{array}{rlll}
\sigma & ::= & T & \text{general type} \\
 & & \Diamond & \text{process type} \\
 & & & \\
T & ::= & T \rightarrow \sigma & \text{functional type} \\
 & & T \xrightarrow{1} \sigma & \text{linear functional type} \\
 & & & \\
P & ::= & PQ & \text{application} \\
 & & v & \text{values} \\
 & & & \\
v & ::= & \lambda x : T.P & \text{abstraction}
\end{array}
$$

And encoding is an homomorphism...

Higher-Order Results

Theorem (Correctness: Typing HOπ Processes)

$\Phi; \Gamma; \mathcal{S} \vdash P : \sigma$, *if and only if* $[\![\Phi; \Gamma; \mathcal{S}]\!]_f \vdash [\![P]\!]_f : [\![\sigma]\!]$.

Higher-Order Results

Theorem (Correctness: Typing HOπ Processes)

$\Phi; \Gamma; \mathcal{S} \vdash P : \sigma$, *if and only if* $[\![\Phi; \Gamma; \mathcal{S}]\!]_f \vdash [\![P]\!]_f : [\![\sigma]\!]$.

Derived from the encoding:

- Session π augmented with cbv λ.
- Encoding of the new constructs: an homomorphism.
- Again, Subject Reduction and Type Soundness derived for free.

Conclusions 1/2

Conclusions 1/2

- Presented an encoding of session types into ordinary π types.

Conclusions 1/2

- Presented an encoding of session types into ordinary π types.
- Encoding proved faithful, in that it allows us to derive all the basic properties of session types from π types.

Conclusions 1/2

- Presented an encoding of session types into ordinary π types.
- Encoding proved faithful, in that it allows us to derive all the basic properties of session types from π types.
- Encoding proved robust (Subtyping, Polymorphism, HO).

Conclusions 2/2

Conclusions 2/2

- Elimination of redundancy: syntax of types and terms in sessions.

Conclusions 2/2

- Elimination of redundancy: syntax of types and terms in sessions.
- Derivation of properties: subject reduction and type soundness in sessions come as straightforward corollaries from the theory of π.

Conclusions 2/2

- Elimination of redundancy: syntax of types and terms in sessions.
- Derivation of properties: subject reduction and type soundness in sessions come as straightforward corollaries from the theory of π.
- Duality on session types boils down to opposite capabilities of standard channel types.

Conclusions 2/2

- Elimination of redundancy: syntax of types and terms in sessions.
- Derivation of properties: subject reduction and type soundness in sessions come as straightforward corollaries from the theory of π.
- Duality on session types boils down to opposite capabilities of standard channel types.
- Robustness of the encoding allows us to easily obtain extensions of the session calculus.

Questions?

About the encoding...

Theorem (Operational Correspondence)

Let P be a process in the π-calculus with sessions. The following hold.

1. *If $P \to P'$ then, $[\![P]\!]_f \to^* \equiv [\![P']\!]_f$;*
2. *If $[\![P]\!]_f \to \equiv Q$ then, $\exists\ P', \mathcal{E}[\cdot]$ such that $\mathcal{E}[P] \to \mathcal{E}[P']$ and $Q \to^* \equiv [\![P']\!]_{f'}$, where either $f' = f$ or $\mathtt{dom}(f') = \mathtt{dom}(f) \cup BV(\mathcal{E}[\cdot])$.*

Subtyping in standard π- calculus

$$\frac{}{T \leq T}\ (\mathrm{S}\pi\text{- }\textsc{Refl}) \qquad \frac{T \leq T' \qquad T' \leq T''}{T \leq T''}\ (\mathrm{S}\pi\text{- }\textsc{Trans})$$

$$\frac{\widetilde{T} \leq \widetilde{T'}}{\ell_{\mathtt{i}}\ [\widetilde{T}] \leq \ell_{\mathtt{i}}\ [\widetilde{T'}]}\ (\mathrm{S}\pi\text{- }\mathtt{ii}) \qquad \frac{\widetilde{T'} \leq \widetilde{T}}{\ell_{\mathtt{o}}\ [\widetilde{T}] \leq \ell_{\mathtt{o}}\ [\widetilde{T'}]}\ (\mathrm{S}\pi\text{- }\mathtt{oo})$$

$$\frac{I \subseteq J \qquad T_i \leq T'_j \quad \forall i \in I}{\langle l_i\_T_i\rangle_{i \in I} \leq \langle l_j\_T'_j\rangle_{j \in J}}\ (\mathrm{S}\pi\text{- }\textsc{Variant})$$

Semantics of Bounded Polymorphism

$$(\boldsymbol{\nu} xy)(x \triangleleft l_j(B).P \mid y \triangleright \{l_i(X_i <: B_i) : P_i\}_{i \in I} \mid R) \rightarrow$$
$$(\boldsymbol{\nu} xy)(P \mid P_j[B/X_j] \mid R) \quad j \in I$$

$$\textbf{case}\, l_j(B)\_v\ \textbf{of}\ \{l_i(X_i \leq B_i)\_x_i \triangleright P\}_{i \in I} \rightarrow P_j[B/X_j][v/x_j] \quad j \in I$$

Parametric Polymorphism

Example of polymorphism in pi with/without sessions:

$$
\begin{aligned}
&x : !\langle X; D\rangle.end\ ,\ y : ?\langle X; D\rangle.end \\
&\qquad \vdash x!\langle Int; 5\rangle \mid y?(z).\ \texttt{open}\ z\ \texttt{as}\ (X; w)\ \texttt{in}\ nj!\langle w\rangle \\
&\qquad \longrightarrow\ \texttt{open}\ \langle Int; 5\rangle\ \texttt{as}\ (X; w)\ \texttt{in}\ nj!\langle w\rangle \\
&\qquad \longrightarrow nj!\langle 5\rangle
\end{aligned}
$$

Encoding Higher-Order

$$
\begin{aligned}
[\![T \xrightarrow{1} \sigma]\!] &\stackrel{\texttt{def}}{=} [\![T]\!] \xrightarrow{1} \sigma \\
[\![T \rightarrow \sigma]\!] &\stackrel{\texttt{def}}{=} [\![T]\!] \rightarrow \sigma
\end{aligned}
$$

$$
\begin{aligned}
[\![\lambda x : T.P]\!]_f &\stackrel{\texttt{def}}{=} \lambda x : [\![T]\!].[\![P]\!]_f \\
[\![PQ]\!]_f &\stackrel{\texttt{def}}{=} [\![P]\!]_f [\![Q]\!]_f
\end{aligned}
$$

Where $\sigma ::= T \mid \Diamond$

Up-side-down point of view

- Linear channel transmitting a value and *a new* linear channel $(\nu b)\overline{a}\langle v, b\rangle \ldots$

Up-side-down point of view

- Linear channel transmitting a value and *a new* linear channel $(\nu b)\overline{a}\langle v, b\rangle \ldots$
- Linear channel transmitting a value and itself $\overline{a}\langle v, a\rangle \ldots$

Up-side-down point of view

- Linear channel transmitting a value and *a new* linear channel $(\nu b)\overline{a}\langle v, b\rangle \ldots$
- Linear channel transmitting a value and itself $\overline{a}\langle v, a\rangle \ldots$
- Linear channel transmitting a value $\overline{a}\langle v\rangle \ldots$

Up-side-down point of view

- Linear channel transmitting a value and *a new* linear channel $(\nu b)\overline{a}\langle v, b\rangle \ldots$
- Linear channel transmitting a value and itself $\overline{a}\langle v, a\rangle \ldots$
- Linear channel transmitting a value $\overline{a}\langle v\rangle \ldots$

NB

Session types are an optimisation of linear π types.

New typing rule for output

$$\frac{\Gamma_1 \vdash x : \ell_{\mathsf{o}}\ [\widetilde{T}] \qquad \widetilde{\Gamma}_2, x : \ell_\alpha\ [\widetilde{S}] \vdash \tilde{v} : \widetilde{T} \qquad \Gamma_3, x : \ell_{\overline{\alpha}}\ [\widetilde{S}] \vdash P}{\Gamma_1 \uplus \widetilde{\Gamma}_2 \uplus \Gamma_3 \vdash x!\langle \tilde{v} \rangle.P}$$