Subtyping for Binary Sessions[1]

[1] Simon J. Gay, Malcolm Hole: Subtyping for session types in the pi calculus. Acta Inf. (2005)

Example

```
let end_echo_client ep =
    let ep = Session.select (fun x → `End x) ep
    in Session.close ep
```

```
val end_echo_client:  ⊕[End : end] → unit
```

```
val opt_echo_service : &[End : end, Msg : ?α.!α.end] → unit
```

Note that:

$$\overline{\oplus[\mathsf{End} : \mathbf{end}]} = \&[\mathsf{End} : \mathbf{end}] \neq \&[\mathsf{End} : \mathbf{end}, \mathsf{Msg} : ?\alpha.!\alpha.\mathbf{end}]$$

This is handled by a notion of subtyping (or safe substitution)

Subtipado en Lambda Calculus

- El sistema de tipos descarta programas incorrectos.
- Pero también programas “buenos”.

Subtipado en Lambda Calculus

- El sistema de tipos descarta programas incorrectos.
- Pero también programas “buenos”.
  - $(\lambda x : \texttt{float}.x > .0)\ 1$
- Queremos mayor flexibilidad y disminuir la cantidad de programas buenos que se descartan.

Principio de sustitutividad

$$\sigma \leqslant \tau$$

- Lectura: "En todo contexto donde se espera una expresión de tipo τ, puede utilizarse una de tipo σ en su lugar sin que ello genere un error"

Principio de sustitutividad

$$\sigma \leqslant \tau$$

- Lectura: "En todo contexto donde se espera una expresión de tipo τ, puede utilizarse una de tipo σ en su lugar sin que ello genere un error"
- Esto se refleja con una nueva regla de tipado llamada Subsumption:

$$\frac{\Gamma \vdash M : \sigma \qquad \sigma \leqslant \tau}{\Gamma \vdash M : \tau}\,[\text{T-Subs}]$$

Subtipado de tipos base

- Para los tipos base asumimos que nos informan de qué manera están relacionados; por ejemplo

$$\begin{array}{lcl} \texttt{nat} & \leqslant & \texttt{int} \\ \texttt{int} & \leqslant & \texttt{float} \end{array}$$

Subtipado como preorden

Subtipado como preorden

$$\frac{}{\sigma \leqslant \sigma}\,[\text{S-Refl}] \qquad \frac{\sigma \leqslant \tau \quad \tau \leqslant \rho}{\sigma \leqslant \rho}\,[\text{S-Trans}]$$

Nota

- Sin antisimetría, ni simetría

Subtipado de tipos función

Subtipado de tipos función

$$\frac{\sigma' \leqslant \sigma \quad \tau \leqslant \tau'}{\sigma \to \tau \;\leqslant\; \sigma' \to \tau'}\text{[S-Func]}$$

- Observar que el sentido de $\leqslant$ se da "vuelta" para el tipo del argumento de la función pero no para el tipo del resultado
- Se dice que el constructor de tipos función es contravariante en su primer argumento y covariante en el segundo.

Subtipado de tipos función

$$\frac{\sigma' \leqslant \sigma \quad \tau \leqslant \tau'}{\sigma \to \tau \;\leqslant\; \sigma' \to \tau'}\text{[S-Func]}$$

Subtipado de tipos función

$$\frac{\sigma' \leqslant \sigma \quad \tau \leqslant \tau'}{\sigma \to \tau \;\leqslant\; \sigma' \to \tau'}\,\text{[S-Func]}$$

Si un contexto/programa P espera una expresión f de tipo $\sigma' \to \tau'$ puede recibir otra de tipo $\sigma \to \tau$ si dan las condiciones indicadas

- Toda aplicación de f se hace sobre un argumento de tipo σ'
- El argumento se coerciona al tipo σ
- Luego se aplica la función, cuyo tipo real $\sigma \to \tau$
- Finalmente se coerciona el resultado a τ', el tipo del resultado que espera P

Agregando subsumption

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\text{[T-Var]} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\text{[T-App]}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau}\text{[T-Abs]} \qquad \frac{\Gamma \vdash M : \sigma \quad \sigma \leqslant \tau}{\Gamma \vdash M : \tau}\text{[T-Subs]}$$

- Con subsumption ya no son dirigidas por sintaxis.
- No es evidente cómo implementar un algoritmo de chequeo de tipos a partir de las reglas.

“Cableando” subsumption dentro de las demás reglas

- Un análisis rápido determina que el único lugar donde se precisa subtipar es al aplicar una función a un argumento
- Esto sugiere la siguiente formulación donde

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\,[\text{T-Var}] \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \rho \quad \rho \leqslant \sigma}{\Gamma \vdash M\,N : \tau}\,[\text{T-App}]$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau}\,[\text{T-Abs}]$$

Syntax of Types

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

Typing (without subtyping)

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\text{[T-Par]} \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\text{[T-Res]}$$

$$\frac{\Gamma, x^p : S, y : t \vdash P}{\Gamma, x^p : ?t.S \vdash x^p?(y{:}\mathtt{t}).P}\text{[T-In]} \qquad \frac{\Gamma_1 \vdash v : t \qquad \Gamma_2, x^p : S \vdash P}{\Gamma_1 + (\Gamma_2, x^p : !t.S) \vdash x^p!v.P}\text{[T-Out]}$$

$$\frac{\Gamma, x^p : S_j \vdash P \qquad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathtt{l}_j.P}\text{[T-Choice]} \qquad \frac{\Gamma, x^p : S_i \vdash P_i \quad \forall i \in I}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_i : P_i]_{i \in I}}\text{[T-Branch]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\text{[T-Nil]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\text{[T-Par]}$$

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P}{\Gamma_1 + (\Gamma_2, x^p : \,!s\,.\,S) \vdash x^p\,!\,v\,.\,P}\text{[T-Out]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\text{[T-Nil]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\text{[T-Par]}$$

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : {!s}.S) \vdash x^p!v.P}\text{[T-Out]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\text{[T-Nil]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\text{[T-Par]}$$

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : !s.S) \vdash x^p!v.P}\text{[T-Out]}$$

$$\frac{\Gamma, x^p : S, y : t \vdash P}{\Gamma, x^p : ?s.S \vdash x^p?(y{:}t).P}\text{[T-In]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\text{[T-Nil]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\ \text{[T-Par]}$$

$$\frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\ \text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : !s.S) \vdash x^p!v.P}\ \text{[T-Out]}$$

$$\frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant t}{\Gamma, x^p : ?s.S \vdash x^p?(y{:}t).P}\ \text{[T-In]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\ \text{[T-Nil]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\text{[T-Par]} \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : \mathtt{!}s\,\mathtt{.}\,S) \vdash x^p\,\mathtt{!}\,v\,\mathtt{.}\,P}\text{[T-Out]} \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant t}{\Gamma, x^p : \mathtt{?}s\,\mathtt{.}\,S \vdash x^p\,\mathtt{?}(y{:}\mathtt{t})\,\mathtt{.}\,P}\text{[T-In]}$$

$$\frac{\Gamma, x^p : S_j \vdash P \qquad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i\in I} \vdash x^p \triangleleft \mathtt{l}_j\,\mathtt{.}\,P}\text{[T-Choice]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\text{[T-Nil]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\,[\text{T-Par}] \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\,[\text{T-Res}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : \mathtt{!}s\,\mathtt{.}\,S) \vdash x^p\mathtt{!}v\,\mathtt{.}\,P}\,[\text{T-Out}] \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant t}{\Gamma, x^p : \mathtt{?}s\,\mathtt{.}\,S \vdash x^p\mathtt{?}(y{:}t)\,\mathtt{.}\,P}\,[\text{T-In}]$$

$$\frac{\Gamma, x^p : S_j \vdash P \qquad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathtt{l}_j\,\mathtt{.}\,P}\,[\text{T-Choice}]$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\,[\text{T-Nil}]$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\;[\text{T-Par}] \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\;[\text{T-Res}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : \mathtt{!}s\,\mathtt{.}\,S) \vdash x^p\mathtt{!}v\,\mathtt{.}\,P}\;[\text{T-Out}] \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant t}{\Gamma, x^p : \mathtt{?}s\,\mathtt{.}\,S \vdash x^p\mathtt{?}(y{:}t)\,\mathtt{.}\,P}\;[\text{T-In}]$$

$$\frac{\Gamma, x^p : S_j \vdash P \qquad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i\in I} \vdash x^p \triangleleft \mathtt{l}_j\,\mathtt{.}\,P}\;[\text{T-Choice}] \qquad \frac{}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i\in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j\in J}}\;[\text{T-Branch}]$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\;[\text{T-Nil}]$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\,\text{[T-Par]} \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\,\text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : {!s}.S) \vdash x^p ! v . P}\,\text{[T-Out]} \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant t}{\Gamma, x^p : {?s}.S \vdash x^p ? (y{:}t) . P}\,\text{[T-In]}$$

$$\frac{\Gamma, x^p : S_j \vdash P \qquad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathtt{l}_j . P}\,\text{[T-Choice]} \qquad \frac{I \subseteq J \quad \Gamma, x^p : S_i \vdash P_i \quad \forall i \in I}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j \in J}}\,\text{[T-Branch]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\,\text{[T-Nil]}$$

Unsound variant of [T-out]

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad s \leqslant t}{\Gamma_1 + (\Gamma_2, x^p : {!s}.S) \vdash x^p ! v . P}\,\text{[T-Out-Bad]}$$

Unsound variant of [T-out]

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad s \leqslant t}{\Gamma_1 + (\Gamma_2, x^p : {!s}.S) \vdash x^p ! v . P}\text{[T-Out-Bad]}$$

Example

Assume $\mathtt{nat} \leqslant \mathtt{int} \leqslant \mathtt{float}$. Then the following derivation would be possible

$$\frac{\vdash 1.2 : \mathtt{float} \qquad \dfrac{x^+ : \mathsf{end}\ \text{completed}}{x^+ : \mathsf{end} \vdash 0}\text{[T-Nil]} \qquad \mathtt{int} \leqslant \mathtt{float}}{x^+ : {!\mathtt{int}}.\mathsf{end} \vdash x^+ ! 1.2 . 0}\text{[T-Out-Bad]}$$

And clearly, $x^+ ! 1.2 . P$ does not use x^+ as described by ${!\mathtt{int}}.\mathsf{end}$

Unsound variant of [T-in]

$$\frac{\Gamma, x^p : S, y : t \vdash P \quad t \leqslant s}{\Gamma, x^p : ?s\,.\,S \vdash x^p?(y{:}\mathtt{t})\,.\,P}\text{[T-In-Bad]}$$

Unsound variant of [T-in]

$$\frac{\Gamma, x^p : S, y : t \vdash P \qquad t \leqslant s}{\Gamma, x^p : ?s.S \vdash x^p?(y{:}t).P}\,[\text{T-In-Bad}]$$

Example

Assume $\mathtt{nat} \leqslant \mathtt{int} \leqslant \mathtt{float}$. Then the following derivation would be possible

$$\dfrac{\dfrac{\vdots}{x^+ : \mathsf{end}, y^+ : !\mathtt{int}.\mathsf{end}, z : \mathtt{int} \vdash y^+!z.0} \qquad \mathtt{int} \leqslant \mathtt{float}}{x^+ : ?\mathtt{float}.\mathsf{end}, y^+ : !\mathtt{int}.\mathsf{end} \vdash x^+?(z{:}\mathtt{int}).y^+!z.0}\,[\text{T-In-Bad}]$$

And clearly, the process violates the communication on y^+ when a $\mathtt{float}$ is received on x^+

Unsound variant of [T-Branch]

$$\frac{J \subseteq I \qquad \Gamma, x^p : S_i \vdash P_i \qquad \forall j \in J}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j \in J}}\ \text{[T-Branch-Bad]}$$

Unsound variant of [T-Branch]

$$\frac{J \subseteq I \quad \Gamma, x^p : S_i \vdash P_i \quad \forall j \in J}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j \in J}}\ \text{[T-Branch-Bad]}$$

Example

Assume $\mathtt{nat} \leqslant \mathtt{int} \leqslant \mathtt{float}$. Then the following derivation would be possible

$$\frac{\{1\} \subseteq \{1,2\} \qquad \dfrac{\vdots}{x^+ : \mathsf{end} \vdash 0}}{x^+ : \&[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}] \vdash x^+ \triangleright [\mathtt{l}_1 : 0]}\ \text{[T-Branch-Bad]}$$

The process cannot proceed if the peer choses $\mathtt{l}_2$, e.g.,

$$(\nu x{:}\&[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}])(x^+ \triangleright [\mathtt{l}_1 : 0] \mid x^- \triangleleft \mathtt{l}_2.0)$$

Expectation about subtyping relation

$$\frac{\Gamma, x^p : s \vdash P \qquad t \leqslant s}{\Gamma, x^p : t \vdash P}$$

Subtyping for non-recursive types

$$\mathsf{end} \leqslant \mathsf{end} \text{ [S-End]}$$

Subtyping for non-recursive types

$$\mathsf{end} \leqslant \mathsf{end} \; \text{[S-End]}$$

$$\frac{s \leqslant t \qquad S \leqslant T}{?s\,.\,S \leqslant ?t\,.\,T}\,\text{[S-InS]}$$

Subtyping for non-recursive types

$$\mathsf{end} \leqslant \mathsf{end} \;\text{[S-End]}$$

$$\frac{s \leqslant t \qquad S \leqslant T}{?s.S \leqslant ?t.T}\;\text{[S-InS]} \qquad\qquad \frac{t \leqslant s \qquad S \leqslant T}{!s.S \leqslant !t.T}\;\text{[S-OutS]}$$

Subtyping for non-recursive types

$$\mathsf{end} \leqslant \mathsf{end} \;\text{[S-End]}$$

$$\frac{s \leqslant t \qquad S \leqslant T}{?s.S \leqslant ?t.T}\;\text{[S-InS]}$$

$$\frac{t \leqslant s \qquad S \leqslant T}{!s.S \leqslant !t.T}\;\text{[S-OutS]}$$

$$\frac{I \subseteq J \qquad \forall i \in I. S_i \leqslant T_i}{\&[\mathtt{l}_i : S_i]_{i \in I} \leqslant \&[\mathtt{l}_j : T_i]_{j \in J}}\;\text{[S-Branch]}$$

Subtyping for non-recursive types

$$\mathbf{end} \leqslant \mathbf{end} \; [\text{S-End}]$$

$$\frac{s \leqslant t \qquad S \leqslant T}{?s.S \leqslant ?t.T}\,[\text{S-InS}] \qquad \frac{t \leqslant s \qquad S \leqslant T}{!s.S \leqslant !t.T}\,[\text{S-OutS}]$$

$$\frac{I \subseteq J \qquad \forall i \in I.S_i \leqslant T_i}{\&[\mathtt{l}_i : S_i]_{i \in I} \leqslant \&[\mathtt{l}_j : T_i]_{j \in J}}\,[\text{S-Branch}] \qquad \frac{J \subseteq I \qquad \forall j \in J.S_j \leqslant T_j}{\oplus[\mathtt{l}_i : S_i]_{i \in I} \leqslant \oplus[\mathtt{l}_j : T_i]_{j \in J}}\,[\text{S-Choice}]$$

Unsound variant of [S-InS]

$$\frac{t \leqslant s \qquad S \leqslant T}{?s.S \leqslant ?t.T}\ \text{[S-InS-Bad]}$$

Unsound variant of [S-InS]

$$\frac{t \leqslant s \qquad S \leqslant T}{?s.S \leqslant ?t.T}\ \text{[S-InS-Bad]}$$

Example

Assume $\texttt{nat} \leqslant \texttt{int} \leqslant \texttt{float}$. Then, $?\texttt{float}.\mathsf{end} \leqslant ?\texttt{int}.\mathsf{end}$.

Moreover, the following holds

$$\frac{}{x^+ : ?\texttt{int}.\mathsf{end}, y^+ : !\texttt{int}.\mathsf{end} \vdash x^+?(z{:}\texttt{int}).y^+!z.0}\ \text{[T-In]}$$

Unsound variant of [S-InS]

$$\frac{t \leqslant s \qquad S \leqslant T}{?s.S \leqslant ?t.T}\,\text{[S-InS-Bad]}$$

Example

Assume $\mathtt{nat} \leqslant \mathtt{int} \leqslant \mathtt{float}$. Then, $?\mathtt{float}.\mathsf{end} \leqslant ?\mathtt{int}.\mathsf{end}$.

Moreover, the following holds

$$\frac{}{x^+ : ?\mathtt{int}.\mathsf{end}, y^+ : !\mathtt{int}.\mathsf{end} \vdash x^+?(z{:}\mathtt{int}).y^+!z.0}\,\text{[T-In]}$$

Contrastingly, the following should not hold

$$\frac{}{x^+ : ?\mathtt{float}.\mathsf{end}, y^+ : !\mathtt{int}.\mathsf{end} \vdash x^+?(z{:}\mathtt{int}).y^+!z.0}\,\text{[T-In]}$$

Unsound variant of [S-OutS]

$$\frac{\color{darkred}{s \leqslant t} \qquad S \leqslant T}{!s.S \leqslant !t.T}\,\text{[S-OutS-Bad]}$$

Unsound variant of [S-OutS]

$$\dfrac{{\color{darkred} s \leqslant t} \qquad S \leqslant T}{!s.S \leqslant\, !t.T}\,\text{[S-OutS-Bad]}$$

Example

Assume $\mathtt{nat} \leqslant \mathtt{int} \leqslant \mathtt{float}$. Then, $!\mathtt{int}.\mathtt{end} \leqslant\, !\mathtt{float}.\mathtt{end}$.

Moreover, the following holds

$$\dfrac{}{x^+ : !\mathtt{float}.\mathtt{end} \vdash x^+!1.2.0}\,\text{[T-Out]}$$

Unsound variant of [S-OutS]

$$\dfrac{s \leqslant t \qquad S \leqslant T}{!s.S \leqslant\ !t.T}\,\text{[S-OutS-Bad]}$$

Example

Assume $\mathtt{nat} \leqslant \mathtt{int} \leqslant \mathtt{float}$. Then, $!\mathtt{int}.\mathtt{end} \leqslant\ !\mathtt{float}.\mathtt{end}$.

Moreover, the following holds

$$\dfrac{}{x^+ : !\mathtt{float}.\mathtt{end} \vdash x^+!1.2.0}\,\text{[T-Out]}$$

Contrastingly, the following should not hold

$$\dfrac{}{x^+ : !\mathtt{int}.\mathtt{end} \vdash x^+!1.2.0}\,\text{[T-Out]}$$

Unsound variant of [S-Branch]

$$\frac{J \subseteq I \qquad \forall j \in J. S_i \leqslant T_i}{\&[\mathtt{l}_i : S_i]_{i \in I} \leqslant \&[\mathtt{l}_j : T_i]_{j \in J}} \text{[S-Branch-Bad]}$$

Unsound variant of [S-Branch]

$$\frac{J \subseteq I \qquad \forall j \in J. S_i \leqslant T_i}{\&[\mathtt{l}_i : S_i]_{i\in I} \leqslant \&[\mathtt{l}_j : T_i]_{j \in J}}\text{[S-Branch-Bad]}$$

Example

Hence, $\&[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}] \leqslant \&[\mathtt{l}_1 : \mathsf{end}]$

Moreover, the following holds

$$\frac{\vdots}{x^+ : \&[\mathtt{l}_1 : \mathsf{end}] \vdash x^+ \triangleright [\mathtt{l}_1 : 0]}\text{[T-Branch]}$$

Unsound variant of [S-Branch]

$$\frac{J \subseteq I \qquad \forall j \in J. S_i \leqslant T_i}{\&[\mathtt{l}_i : S_i]_{i \in I} \leqslant \&[\mathtt{l}_j : T_i]_{j \in J}} \text{[S-Branch-Bad]}$$

Example

Hence, $\&[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}] \leqslant \&[\mathtt{l}_1 : \mathsf{end}]$

Moreover, the following holds

$$\frac{\vdots}{x^+ : \&[\mathtt{l}_1 : \mathsf{end}] \vdash x^+ \triangleright [\mathtt{l}_1 : 0]} \text{[T-Branch]}$$

but the following shouldn't

$$\frac{\vdots}{x^+ : \&[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}] \vdash x^+ \triangleright [\mathtt{l}_1 : 0]} \text{[T-Branch]}$$

because the process below would be well typed

$$(\nu x{:}\&[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}])(x^+ \triangleright [\mathtt{l}_1 : 0] \mid x^- \triangleleft \mathtt{l}_2.0)$$

Unsound variant of [S-Choice]

$$\frac{I \subseteq J \qquad \forall i \in I. S_j \leqslant T_j}{\oplus[\mathtt{l}_i : S_i]_{i \in I} \leqslant \oplus[\mathtt{l}_j : T_i]_{j \in J}}\text{[S-Choice-Bad]}$$

Unsound variant of [S-Choice]

$$\frac{I \subseteq J \qquad \forall i \in I. S_j \leqslant T_j}{\oplus[\mathtt{l}_i : S_i]_{i \in I} \leqslant \oplus[\mathtt{l}_j : T_j]_{j \in J}} \text{[S-Choice-Bad]}$$

Example

Hence, $\oplus[\mathtt{l}_1 : \mathsf{end}] \leqslant \oplus[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}]$

Moreover, the following holds

$$\frac{\vdots}{x^+ : \oplus[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}] \vdash x^+ \triangleleft \mathtt{l}_2.0} \text{[T-Choice]}$$

Unsound variant of [S-Choice]

$$\frac{I \subseteq J \qquad \forall i \in I . S_i \leqslant T_i}{\oplus[\mathtt{l}_i : S_i]_{i \in I} \leqslant \oplus[\mathtt{l}_j : T_j]_{j \in J}}\ \text{[S-Choice-Bad]}$$

Example

Hence, $\oplus[\mathtt{l}_1 : \mathsf{end}] \leqslant \oplus[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}]$

Moreover, the following holds

$$\frac{\vdots}{x^+ : \oplus[\mathtt{l}_1 : \mathsf{end}, \mathtt{l}_2 : \mathsf{end}] \vdash x^+ \triangleleft \mathtt{l}_2 . 0}\ \text{[T-Choice]}$$

but the following shouldn't

$$\frac{\vdots}{x^+ : \oplus[\mathtt{l}_1 : \mathsf{end}] \vdash x^+ \triangleleft \mathtt{l}_2 . 0}\ \text{[T-Choice]}$$

because the process below would be well typed

$$(\nu x{:}\&[\mathtt{l}_1 : \mathsf{end}])(x^- \triangleright [\mathtt{l}_1 : 0] \mid x^+ \triangleleft \mathtt{l}_2 . 0)$$

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
	$\mid$	$\mathtt{int}, \mathtt{bool}$	
	$\mid$	[t]	service types
	$\mid$	...	other types
$\mathcal{L} \;=$		$\{\mathtt{l}, \mathtt{l}_1, \ldots\}$	Set of labels

Infinite types

Several ways of writing a type

$$\mu X.\,!\texttt{int}.X$$

Infinite types

Several ways of writing a type

$$\mu X.\mathtt{!int}.X$$
$$\mathtt{!int}.\mu Y.\mathtt{!int}.Y$$

Infinite types

Several ways of writing a type

$$\mu X.!\mathtt{int}.X$$
$$!\mathtt{int}.\mu Y.!\mathtt{int}.Y$$
$$!\mathtt{int}.!\mathtt{int}.\mu Y.!\mathtt{int}.Y$$

Infinite types

Several ways of writing a type

$$\mu X.!\mathtt{int}.X$$
$$!\mathtt{int}.\mu Y.!\mathtt{int}.Y$$
$$!\mathtt{int}.!\mathtt{int}.\mu Y.!\mathtt{int}.Y$$
$$\mu X.!\mathtt{int}.!\mathtt{int}.X$$

Infinite types

Several ways of writing a type

$$\mu X.\,!\mathtt{int}.X$$
$$!\mathtt{int}.\mu Y.\,!\mathtt{int}.Y$$
$$!\mathtt{int}.\,!\mathtt{int}.\mu Y.\,!\mathtt{int}.Y$$
$$\mu X.\,!\mathtt{int}.\,!\mathtt{int}.X$$
$$\mu X.\mu Y.\,!\mathtt{int}.X$$

Infinite types

Several ways of writing a type

$$\begin{array}{l}\mu X.!\texttt{int}.X\\ !\texttt{int}.\mu Y.!\texttt{int}.Y\\ !\texttt{int}.!\texttt{int}.\mu Y.!\texttt{int}.Y\\ \mu X.!\texttt{int}.!\texttt{int}.X\\ \mu X.\mu Y.!\texttt{int}.X\end{array}$$

unfold(_)

$$unfold(T) = \begin{cases} unfold(S\{\mu X.S/X\}) & \text{if } T = \mu X.S\\ T & \text{otherwise}\end{cases}.$$

Infinite types

Several ways of writing a type

$$\mu X.!\texttt{int}.X$$
$$!\texttt{int}.\mu Y.!\texttt{int}.Y$$
$$!\texttt{int}.!\texttt{int}.\mu Y.!\texttt{int}.Y$$
$$\mu X.!\texttt{int}.!\texttt{int}.X$$
$$\mu X.\mu Y.!\texttt{int}.X$$

unfold(_)

$$unfold(T) = \begin{cases} unfold(S\{\mu X.S/X\}) & \text{if } T = \mu X.S \\ T & \text{otherwise} \end{cases}.$$

$unfold(T)$ terminates for all t (because types are contractive)

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(s) = \mathsf{end}$ then $\mathit{unfold}(t) = \mathsf{end}$.

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $unfold(s) = \mathsf{end}$ then $unfold(t) = \mathsf{end}$.
2. If $unfold(s) = ?t_1 . S_1$ then $unfold(t) = ?t_2 . S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_1, t_2) \in \mathcal{R}$.

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(s) = \mathsf{end}$ then $\mathit{unfold}(t) = \mathsf{end}$.
2. If $\mathit{unfold}(s) = ?t_1 . S_1$ then $\mathit{unfold}(t) = ?t_2 . S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_1, t_2) \in \mathcal{R}$.
3. If $\mathit{unfold}(s) = !\,t_1 . S_1$ then $\mathit{unfold}(t) = !\,t_2 . S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_2, t_1) \in \mathcal{R}$.

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(s) = \mathsf{end}$ then $\mathit{unfold}(t) = \mathsf{end}$.
2. If $\mathit{unfold}(s) = \mathtt{?}t_1\,\mathtt{.}\,S_1$ then $\mathit{unfold}(t) = \mathtt{?}t_2\,\mathtt{.}\,S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_1, t_2) \in \mathcal{R}$.
3. If $\mathit{unfold}(s) = \mathtt{!}\,t_1\,\mathtt{.}\,S_1$ then $\mathit{unfold}(t) = \mathtt{!}\,t_2\,\mathtt{.}\,S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_2, t_1) \in \mathcal{R}$.
4. If $\mathit{unfold}(s) = \mathtt{\&[}\mathtt{l}_i : S_i\mathtt{]}_{i \in I}$ then $\mathit{unfold}(t) = \mathtt{\&[}\mathtt{l}_j : T_j\mathtt{]}_{j \in J}$ and $I \subseteq J$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(s) = \mathsf{end}$ then $\mathit{unfold}(t) = \mathsf{end}$.
2. If $\mathit{unfold}(s) = ?t_1.S_1$ then $\mathit{unfold}(t) = ?t_2.S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_1, t_2) \in \mathcal{R}$.
3. If $\mathit{unfold}(s) = !t_1.S_1$ then $\mathit{unfold}(t) = !t_2.S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_2, t_1) \in \mathcal{R}$.
4. If $\mathit{unfold}(s) = \&[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(t) = \&[\mathtt{l}_j : T_j]_{j \in J}$ and $I \subseteq J$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.
5. If $\mathit{unfold}(s) = \oplus[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(t) = \oplus[\mathtt{l}_j : T_j]_{j \in J}$ and $J \subseteq I$ and $(S_j, T_j) \in \mathcal{R}$ for all $j \in J$.

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(s) = \mathsf{end}$ then $\mathit{unfold}(t) = \mathsf{end}$.
2. If $\mathit{unfold}(s) = ?t_1.S_1$ then $\mathit{unfold}(t) = ?t_2.S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_1, t_2) \in \mathcal{R}$.
3. If $\mathit{unfold}(s) = !t_1.S_1$ then $\mathit{unfold}(t) = !t_2.S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_2, t_1) \in \mathcal{R}$.
4. If $\mathit{unfold}(s) = \&[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(t) = \&[\mathtt{l}_j : T_j]_{j \in J}$ and $I \subseteq J$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.
5. If $\mathit{unfold}(s) = \oplus[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(t) = \oplus[\mathtt{l}_j : T_j]_{j \in J}$ and $J \subseteq I$ and $(S_j, T_j) \in \mathcal{R}$ for all $j \in J$.
6. If $\mathit{unfold}(s) = [s']$ then $\mathit{unfold}(t) = [t']$ and $(s', t') \in \mathcal{R}$.
7. if s and t are basic types, then $s \prec t$.

(Coinductive) Subtyping

The *coinductive subtyping relation* $\leqslant_c$ is defined by $S \leqslant_c T$ if and only if there exists a type simulation $\mathcal{R}$ such that $(S, T) \in \mathcal{R}$.

Coinductive subtyping

Example

$$S = \mu X . ! \texttt{int} . X$$

Coinductive subtyping

Example

$$S = \mu X.!\texttt{int}.X$$
$$T = !\texttt{float}.\mu Y.!\texttt{float}.Y$$

We show that $\mathcal{R} = \{(\texttt{int}, \texttt{float}), (T, S), (\mu Y.!\texttt{float}.Y, S)\}$ is type simulation.
Hence $T \leqslant S$

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $unfold(S) = \mathsf{end}$ then $unfold(T) = \mathsf{end}$.

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(S) = \mathsf{end}$ then $\mathit{unfold}(T) = \mathsf{end}$.
2. If $\mathit{unfold}(S) = {?}t_1 \,.\, S_1$ then $\mathit{unfold}(T) = {!}t_2 \,.\, S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(S) = \mathbf{end}$ then $\mathit{unfold}(T) = \mathbf{end}$.
2. If $\mathit{unfold}(S) = ?t_1 . S_1$ then $\mathit{unfold}(T) = !t_2 . S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
3. If $\mathit{unfold}(S) = !t_1 . S_1$ then $\mathit{unfold}(T) = ?t_2 . S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(S) = \mathsf{end}$ then $\mathit{unfold}(T) = \mathsf{end}$.
2. If $\mathit{unfold}(S) = \mathtt{?}t_1\mathtt{.}S_1$ then $\mathit{unfold}(T) = \mathtt{!}t_2\mathtt{.}S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
3. If $\mathit{unfold}(S) = \mathtt{!}t_1\mathtt{.}S_1$ then $\mathit{unfold}(T) = \mathtt{?}t_2\mathtt{.}S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
4. If $\mathit{unfold}(S) = \&[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(T) = \oplus[\mathtt{l}_i : T_i]_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(S) = \mathsf{end}$ then $\mathit{unfold}(T) = \mathsf{end}$.
2. If $\mathit{unfold}(S) = \,?t_1\,.\,S_1$ then $\mathit{unfold}(T) = \,!t_2\,.\,S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
3. If $\mathit{unfold}(S) = \,!t_1\,.\,S_1$ then $\mathit{unfold}(T) = \,?t_2\,.\,S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
4. If $\mathit{unfold}(S) = \&[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(T) = \oplus[\mathtt{l}_i : T_i]_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.
5. If $\mathit{unfold}(S) = \oplus[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(T) = \&[\mathtt{l}_i : T_i]_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(S) = \mathsf{end}$ then $\mathit{unfold}(T) = \mathsf{end}$.
2. If $\mathit{unfold}(S) = \mathtt{?}t_1\mathtt{.}S_1$ then $\mathit{unfold}(T) = \mathtt{!}t_2\mathtt{.}S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
3. If $\mathit{unfold}(S) = \mathtt{!}t_1\mathtt{.}S_1$ then $\mathit{unfold}(T) = \mathtt{?}t_2\mathtt{.}S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
4. If $\mathit{unfold}(S) = \mathtt{\&[}\mathtt{l}_i : S_i\mathtt{]}_{i \in I}$ then $\mathit{unfold}(T) = \mathtt{\oplus[}\mathtt{l}_i : T_i\mathtt{]}_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.
5. If $\mathit{unfold}(S) = \mathtt{\oplus[}\mathtt{l}_i : S_i\mathtt{]}_{i \in I}$ then $\mathit{unfold}(T) = \mathtt{\&[}\mathtt{l}_i : T_i\mathtt{]}_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.

(Coinductive) Duality

The *coinductive duality relation* $\perp_c$ is defined by $S \perp_c T$ if and only if there exists a duality relation $\mathcal{R}$ such that $(S, T) \in \mathcal{R}$.

Typing

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\,[\text{T-Par}] \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\,[\text{T-Res}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant_c s}{\Gamma_1 + (\Gamma_2, x^p : !s.S) \vdash x^p!v.P}\,[\text{T-Out}] \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant_c t}{\Gamma, x^p : ?s.S \vdash x^p?(y{:}t).P}\,[\text{T-In}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x : [s] \vdash P \quad t \leqslant_c s}{\Gamma_1 + (\Gamma_2, x : [s]) \vdash x^p!v.P}\,[\text{T-Out-Un}] \qquad \frac{\Gamma, x : [s], y : t \vdash P \quad s \leqslant_c t}{\Gamma, x : [s] \vdash x?(y{:}t).P}\,[\text{T-In-Un}]$$

$$\frac{\Gamma, x^p : S_j \vdash P \quad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathtt{l}_j.P}\,[\text{T-Choice}] \qquad \frac{I \subseteq J \quad \Gamma, x^p : S_i \vdash P_i \quad \forall i \in I}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j \in J}}\,[\text{T-Branch}]$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\,[\text{T-Nil}] \qquad \frac{\Gamma \vdash P \quad \Gamma \text{ Unlimited}}{\Gamma \vdash !P}\,[\text{T-Rep}]$$

Properties

Substitution

If $\Gamma, x^p : s \vdash P$ and $t \leqslant_c s$ and $\Gamma + y : t$ is defined then $\Gamma + y^q : t \vdash P\{y/x\}$

Ocaml Implementation

Subtyping

- Relying on Ocaml Subtyping

```
type (+ρ,-σ) st (* OCaml syntax for ⟨ρ, σ⟩ *)
```