Behavioral Subtyping

Luca Padovani
Dipartimento di Informatica, Università di Torino
http://www.di.unito.it/~padovani/
padovani@di.unito.it

Course references

- S Gay, M Hole, **Subtyping for session types in the pi calculus**, Acta Informatica 42(2/3):191-225, 2005
- L Padovani, **Fair Subtyping for Open Session Types**, ICALP 2013, LNCS 7966:373-384

Outline

1. Basic notions
    - Motivation
    - Informal review of subtyping
    - Subtyping for finite session types
2. Recursive session types
    - Subtyping for recursive session types
    - Subtyping algorithm
    - Further reading
3. Fair subtyping
    - Motivation
    - A liveness-preserving subtyping
    - Characterizing fair subtyping
    - Two issues
    - Further reading

Outline

Motivations for subtyping

$$\frac{\Gamma_1 \vdash e : s \qquad s = t \qquad \Gamma_2, u : T \vdash P}{\Gamma_1 + \Gamma_2, u : \,![t]\,.T \vdash u![e].P}$$

Motivations for subtyping

$$\frac{\Gamma_1 \vdash e : s \qquad s \leqslant t \qquad \Gamma_2, u : T \vdash P}{\Gamma_1 + \Gamma_2, u : ![t].T \vdash u![e].P}$$

- relax constraints on types without compromising safety
- $\Rightarrow$ more well-typed programs

Motivations for subtyping

Can a channel with type

“send any number of messages”

be used according to the type

“send at most 3 messages”

?

- formally justify the mismatch between actual and allowed behaviors

Outline

Defining a subtype relation

$$s \leqslant t$$

Safe substitution

"it is safe to use a value of type s
where a value of type t is expected"

Set inclusion (aka "semantic subtyping", cf Castagna et al.)

$$[\![s]\!] \subseteq [\![t]\!]$$

Property preservation (cf Liskov et al.)

$$\forall \phi . (\forall x : t . \phi(x)) \Rightarrow (\forall y : s . \phi(y))$$

Defining a subtype relation

$$s \leqslant t$$

Safe substitution

"it is safe to use a value of type s
where a value of type t is expected"

Set inclusion (aka "semantic subtyping", cf Castagna et al.)

$$[\![s]\!] \subseteq [\![t]\!]$$

Property preservation (cf Liskov et al.)

$$\forall \phi.(\forall x : t.\phi(x)) \Rightarrow (\forall y : s.\phi(y))$$

Defining a subtype relation

$$s \leqslant t$$

Safe substitution

"it is safe to use a value of type s
where a value of type t is expected"

Set inclusion (aka "semantic subtyping", cf Castagna et al.)

$$[\![s]\!] \subseteq [\![t]\!]$$

Property preservation (cf Liskov et al.)

$$\forall \phi . (\forall x : t . \phi(x)) \Rightarrow (\forall y : s . \phi(y))$$

Examples

Set inclusion

$$\texttt{Even} \leqslant \texttt{Int}$$

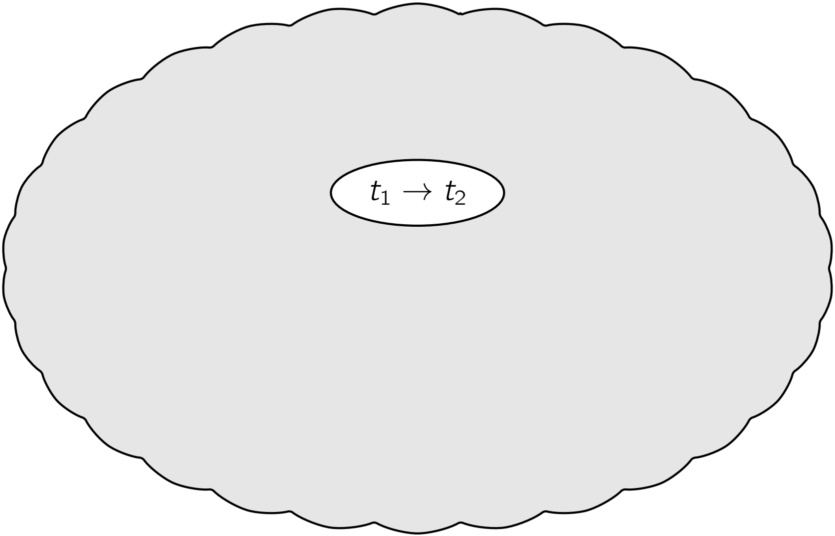
- $[\![\texttt{Even}]\!] \subseteq [\![\texttt{Int}]\!]$

Property preservation

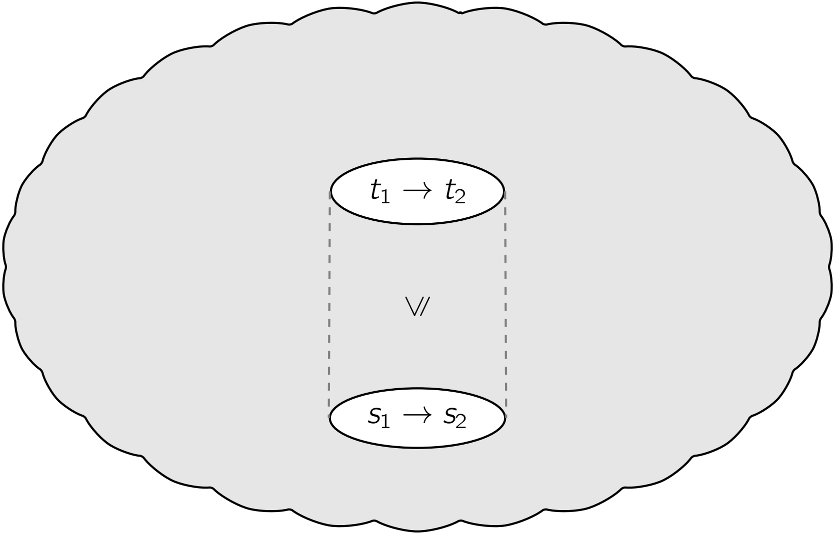
$$\{x : \texttt{Int}, y : \texttt{Int}, c : \texttt{Color}\} \leqslant \{x : \texttt{Int}, y : \texttt{Int}\}$$

- $\phi(o) =$ "o has an x field"
- $\phi(o) =$ "o has a y field"

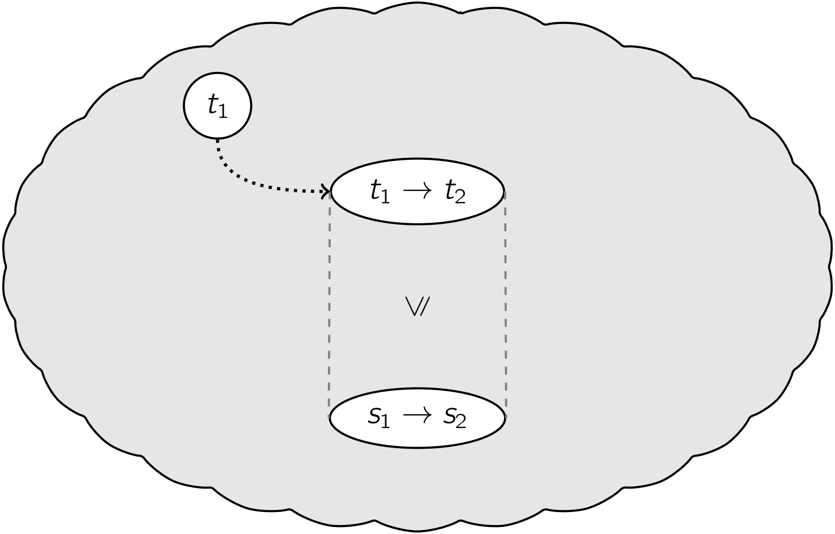
Subtyping for **function** types



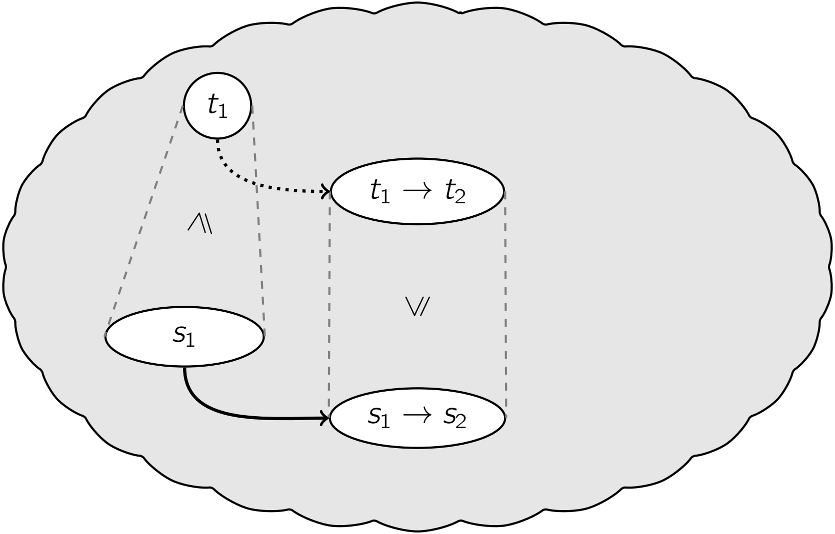
Subtyping for **function** types



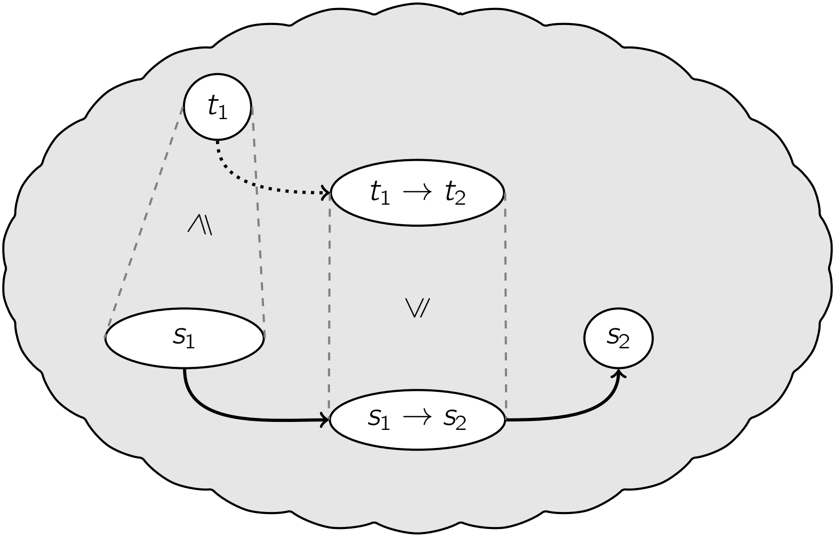
Subtyping for **function** types



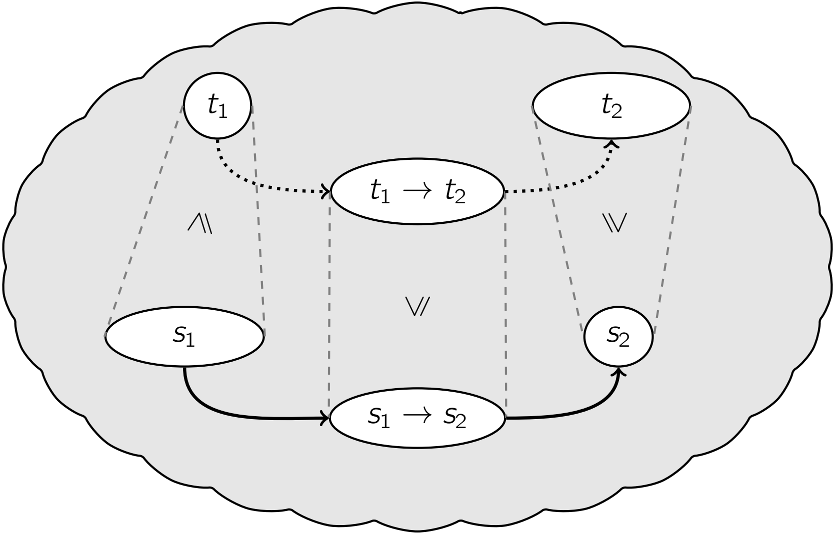
Subtyping for **function** types



Subtyping for **function** types



Subtyping for **function** types



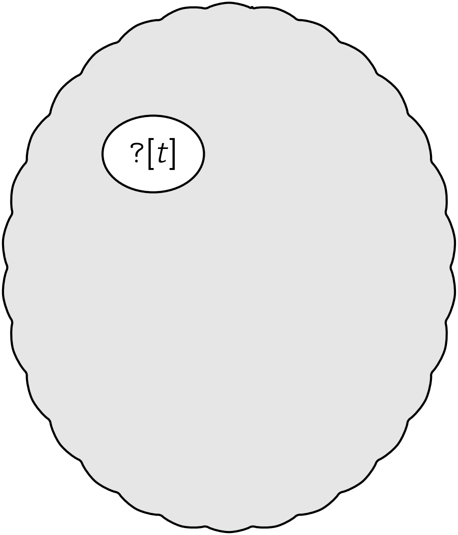
Subtyping for function types

$$\frac{t_1 \leqslant s_1 \qquad s_2 \leqslant t_2}{s_1 \to s_2 \leqslant t_1 \to t_2}$$

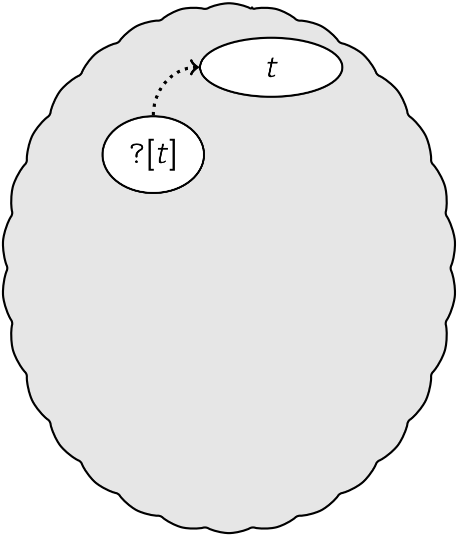
The arrow type constructor

- is **contravariant** in the domain
- is **covariant** in the codomain

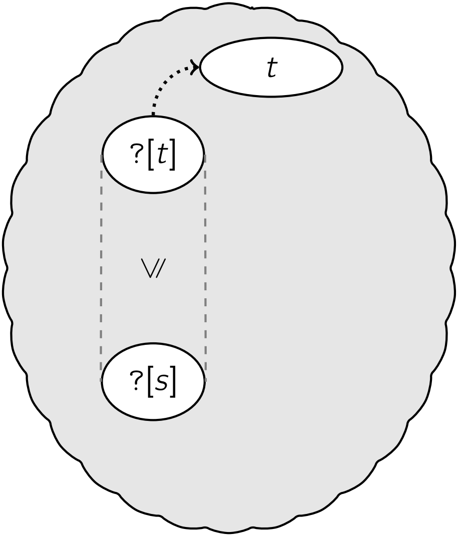
Subtyping for **input** channel types



Subtyping for **input** channel types

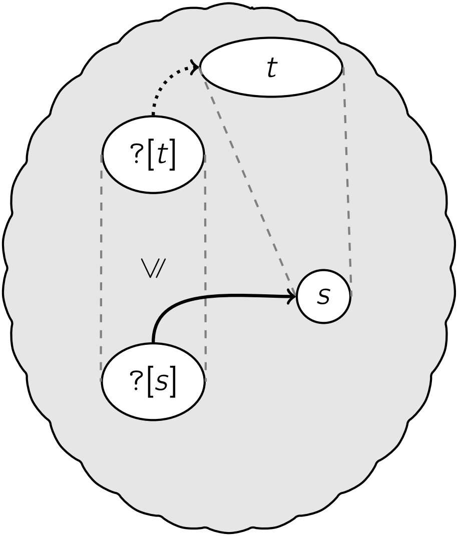


Subtyping for **input** channel types



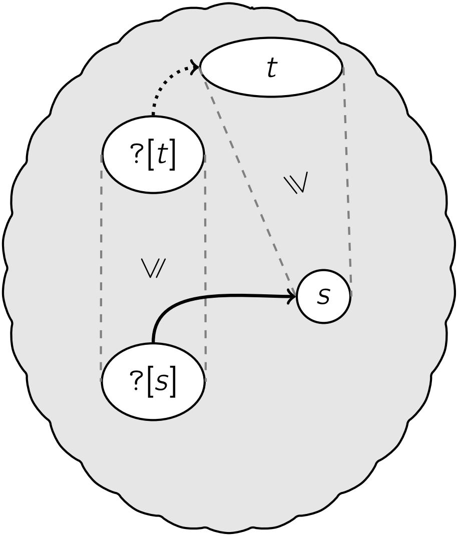
$$\frac{}{?[s] \leqslant ?[t]}$$

Subtyping for **input** channel types



$$\frac{}{?[s] \leqslant ?[t]}$$

Subtyping for **input** channel types

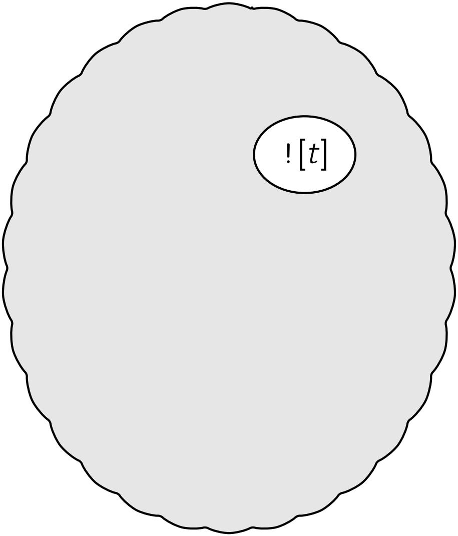


$$\frac{s \leqslant t}{?[s] \leqslant ?[t]}$$

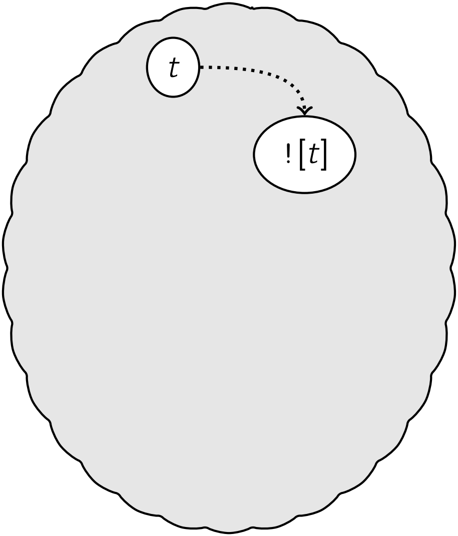
$?[\mathtt{Int}] \leqslant ?[\mathtt{Real}]$

input is covariant

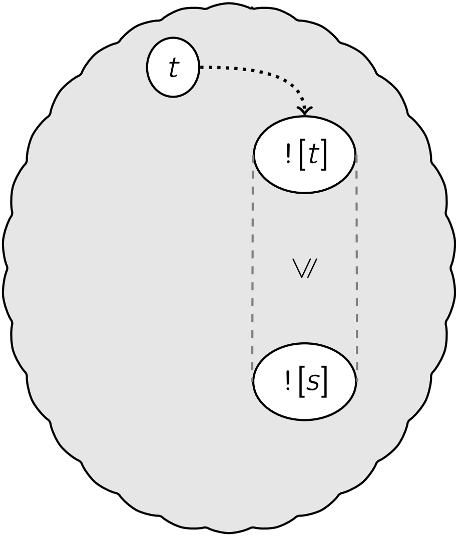
Subtyping for **output** channel types



Subtyping for **output** channel types

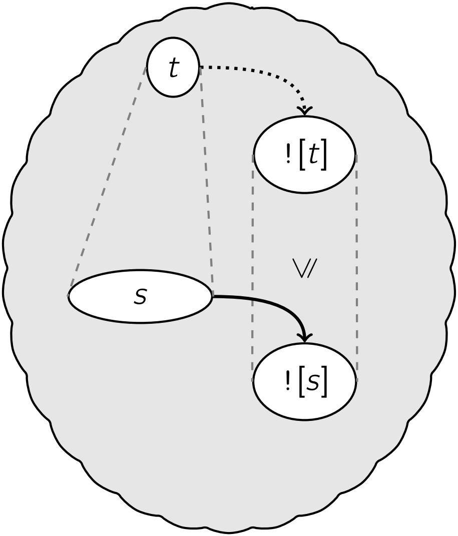


Subtyping for **output** channel types



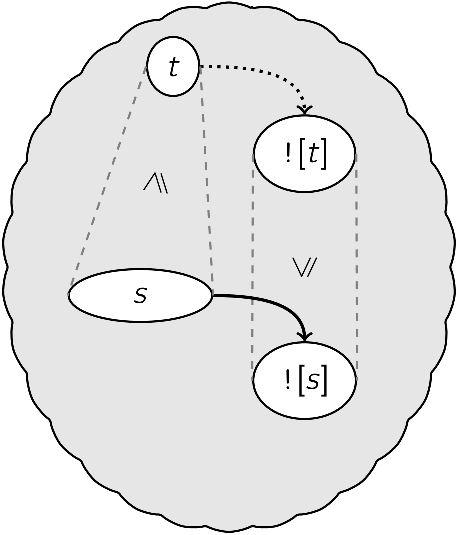
$$\overline{\;![s] \leqslant ![t]\;}$$

Subtyping for **output** channel types



$$\frac{}{![s] \leqslant ![t]}$$

Subtyping for **output** channel types



$$\frac{t \leqslant s}{![s] \leqslant ![t]}$$

$![\texttt{Real}] \leqslant ![\texttt{Int}]$

output is contravariant

Outline

Session types: syntax

T	::=		**session type**		t	::=		**type**
		`end`	(termination)				`Int`	(integer)
	|	$\texttt{?}[t]\,.\,T$	(input)			|	`Real`	(real)
	|	$\texttt{!}[t]\,.\,T$	(output)			|	T	(channel)
	|	$\&\{l_i : T_i\}_{i\in I}$	(branch)					
	|	$\oplus\{l_i : T_i\}_{i\in I}$	(choice)					

In branches and choices

- I non-empty and finite
- $l_i = l_j$ implies $i = j$

Session types: informal semantics

`end`	no operation allowed
$?[t].T$	receive a message of type t then behave according to T
$![t].T$	send a message of type t then behave according to T
$\&\{l_i : T_i\}_{i\in I}$	wait for one of the labels l_k from the set $\{l_i \mid i \in I\}$ then behave according to T_k
$\oplus\{l_i : T_i\}_{i\in I}$	choose and send a label l_k from the set $\{l_i \mid i \in I\}$ then behave according to T_k

Example

Player

$$\oplus \left\{ \begin{array}{l} \texttt{play} : \texttt{!}[\texttt{Real}] . \& \left\{ \begin{array}{l} \texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \end{array} \right\} \\ \texttt{quit} : \texttt{end} \end{array} \right\}$$

Gaming service

$$\& \left\{ \begin{array}{l} \texttt{play} : \texttt{?}[\texttt{Real}] . \oplus \left\{ \begin{array}{l} \texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \end{array} \right\} \\ \texttt{quit} : \texttt{end} \end{array} \right\}$$

Session types: subtype relation

[s-int-int]
$$\mathtt{Int} \leqslant \mathtt{Int}$$

[s-real-real]
$$\mathtt{Real} \leqslant \mathtt{Real}$$

[s-int-real]
$$\mathtt{Int} \leqslant \mathtt{Real}$$

[s-end]
$$\mathtt{end} \leqslant \mathtt{end}$$

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

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

[s-branch]
$$\frac{I \subseteq J \qquad S_i \leqslant T_i \ ^{i \in I}}{\&\{l_i : S_i\}_{i \in I} \leqslant \&\{l_j : T_j\}_{j \in J}}$$

[s-choice]
$$\frac{J \subseteq I \qquad S_j \leqslant T_j \ ^{j \in I}}{\oplus\{l_i : S_i\}_{i \in I} \leqslant \oplus\{l_j : T_j\}_{j \in J}}$$

Examples

$$\oplus \left\{ \begin{matrix} \texttt{play} : ![\text{Real}] . \& \left\{ \begin{matrix} \texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \end{matrix} \right\} \\ \texttt{quit} : \texttt{end} \end{matrix} \right\} \leqslant \oplus \{ \texttt{quit} : \texttt{end} \}$$

$$\oplus \left\{ \begin{matrix} \texttt{play} : ![\text{Real}] . \& \left\{ \begin{matrix} \texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \end{matrix} \right\} \\ \texttt{quit} : \texttt{end} \end{matrix} \right\} \leqslant \oplus \left\{ \texttt{play} : ![\text{Int}] . \& \left\{ \begin{matrix} \texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \\ \texttt{tie} : \texttt{end} \end{matrix} \right\} \right\}$$

Exercises

$?[![\mathtt{Int}].\mathtt{end}].\mathtt{end}$

$?[![\mathtt{Real}].\mathtt{end}].\mathtt{end}$

$![![\mathtt{Int}].\mathtt{end}].\mathtt{end}$

$![![\mathtt{Real}].\mathtt{end}].\mathtt{end}$

$$\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end}\end{Bmatrix}$$

$$\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end} \\ \mathtt{tie} : \mathtt{end}\end{Bmatrix}$$

$$!\left[\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end}\end{Bmatrix}\right].\mathtt{end}$$

$$!\left[\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end} \\ \mathtt{tie} : \mathtt{end}\end{Bmatrix}\right].\mathtt{end}$$

$$\&\begin{Bmatrix}\mathtt{win} : ?[\mathtt{Real}].\mathtt{end} \\ \mathtt{loss} : ![\mathtt{Real}].\mathtt{end}\end{Bmatrix}$$

$$\&\begin{Bmatrix}\mathtt{win} : ?[\mathtt{Int}].\mathtt{end} \\ \mathtt{loss} : ![\mathtt{Int}].\mathtt{end}\end{Bmatrix}$$

Exercises

$$?[!\,[\mathtt{Int}].\mathtt{end}].\mathtt{end} \qquad \geqslant \qquad ?[!\,[\mathtt{Real}].\mathtt{end}].\mathtt{end}$$

$$![!\,[\mathtt{Int}].\mathtt{end}].\mathtt{end} \qquad\qquad ![!\,[\mathtt{Real}].\mathtt{end}].\mathtt{end}$$

$$\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end}\end{Bmatrix} \qquad\qquad \oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end} \\ \mathtt{tie} : \mathtt{end}\end{Bmatrix}$$

$$!\left[\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end}\end{Bmatrix}\right].\mathtt{end} \qquad\qquad !\left[\oplus\begin{Bmatrix}\mathtt{win} : \mathtt{end} \\ \mathtt{loss} : \mathtt{end} \\ \mathtt{tie} : \mathtt{end}\end{Bmatrix}\right].\mathtt{end}$$

$$\&\begin{Bmatrix}\mathtt{win} : ?[\mathtt{Real}].\mathtt{end} \\ \mathtt{loss} : ![\mathtt{Real}].\mathtt{end}\end{Bmatrix} \qquad\qquad \&\begin{Bmatrix}\mathtt{win} : ?[\mathtt{Int}].\mathtt{end} \\ \mathtt{loss} : ![\mathtt{Int}].\mathtt{end}\end{Bmatrix}$$

Exercises

$$?[\,![\mathtt{Int}].\mathsf{end}].\mathsf{end} \quad \geqslant \quad ?[\,![\mathtt{Real}].\mathsf{end}].\mathsf{end}$$

$$![\,![\mathtt{Int}].\mathsf{end}].\mathsf{end} \quad \leqslant \quad ![\,![\mathtt{Real}].\mathsf{end}].\mathsf{end}$$

$$\oplus\begin{Bmatrix}\mathtt{win}: \mathsf{end}\\ \mathtt{loss}: \mathsf{end}\end{Bmatrix} \qquad \oplus\begin{Bmatrix}\mathtt{win}: \mathsf{end}\\ \mathtt{loss}: \mathsf{end}\\ \mathtt{tie}: \mathsf{end}\end{Bmatrix}$$

$$!\left[\oplus\begin{Bmatrix}\mathtt{win}: \mathsf{end}\\ \mathtt{loss}: \mathsf{end}\end{Bmatrix}\right].\mathsf{end} \qquad !\left[\oplus\begin{Bmatrix}\mathtt{win}: \mathsf{end}\\ \mathtt{loss}: \mathsf{end}\\ \mathtt{tie}: \mathsf{end}\end{Bmatrix}\right].\mathsf{end}$$

$$\&\begin{Bmatrix}\mathtt{win}: ?[\mathtt{Real}].\mathsf{end}\\ \mathtt{loss}: ![\mathtt{Real}].\mathsf{end}\end{Bmatrix} \qquad \&\begin{Bmatrix}\mathtt{win}: ?[\mathtt{Int}].\mathsf{end}\\ \mathtt{loss}: ![\mathtt{Int}].\mathsf{end}\end{Bmatrix}$$

Exercises

$$?[\,![\texttt{Int}].\mathsf{end}].\mathsf{end} \quad \geqslant \quad ?[\,![\texttt{Real}].\mathsf{end}].\mathsf{end}$$

$$![\,![\texttt{Int}].\mathsf{end}].\mathsf{end} \quad \leqslant \quad ![\,![\texttt{Real}].\mathsf{end}].\mathsf{end}$$

$$\oplus\begin{Bmatrix}\texttt{win} : \mathsf{end} \\ \texttt{loss} : \mathsf{end}\end{Bmatrix} \quad \geqslant \quad \oplus\begin{Bmatrix}\texttt{win} : \mathsf{end} \\ \texttt{loss} : \mathsf{end} \\ \texttt{tie} : \mathsf{end}\end{Bmatrix}$$

$$!\left[\oplus\begin{Bmatrix}\texttt{win} : \mathsf{end} \\ \texttt{loss} : \mathsf{end}\end{Bmatrix}\right].\mathsf{end} \qquad !\left[\oplus\begin{Bmatrix}\texttt{win} : \mathsf{end} \\ \texttt{loss} : \mathsf{end} \\ \texttt{tie} : \mathsf{end}\end{Bmatrix}\right].\mathsf{end}$$

$$\&\begin{Bmatrix}\texttt{win} : ?[\texttt{Real}].\mathsf{end} \\ \texttt{loss} : ![\texttt{Real}].\mathsf{end}\end{Bmatrix} \qquad \&\begin{Bmatrix}\texttt{win} : ?[\texttt{Int}].\mathsf{end} \\ \texttt{loss} : ![\texttt{Int}].\mathsf{end}\end{Bmatrix}$$

Exercises

$$?[\,![\texttt{Int}].\texttt{end}].\texttt{end} \quad \geqslant \quad ?[\,![\texttt{Real}].\texttt{end}].\texttt{end}$$

$$![\,![\texttt{Int}].\texttt{end}].\texttt{end} \quad \leqslant \quad ![\,![\texttt{Real}].\texttt{end}].\texttt{end}$$

$$\oplus\begin{Bmatrix}\texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end}\end{Bmatrix} \quad \geqslant \quad \oplus\begin{Bmatrix}\texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \\ \texttt{tie} : \texttt{end}\end{Bmatrix}$$

$$!\left[\oplus\begin{Bmatrix}\texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end}\end{Bmatrix}\right].\texttt{end} \quad \leqslant \quad !\left[\oplus\begin{Bmatrix}\texttt{win} : \texttt{end} \\ \texttt{loss} : \texttt{end} \\ \texttt{tie} : \texttt{end}\end{Bmatrix}\right].\texttt{end}$$

$$\&\begin{Bmatrix}\texttt{win} : ?[\texttt{Real}].\texttt{end} \\ \texttt{loss} : ![\texttt{Real}].\texttt{end}\end{Bmatrix} \qquad \qquad \&\begin{Bmatrix}\texttt{win} : ?[\texttt{Int}].\texttt{end} \\ \texttt{loss} : ![\texttt{Int}].\texttt{end}\end{Bmatrix}$$

Exercises

$$?[\,![\mathtt{Int}].\mathsf{end}].\mathsf{end} \quad\geqslant\quad ?[\,![\mathtt{Real}].\mathsf{end}].\mathsf{end}$$

$$![\,![\mathtt{Int}].\mathsf{end}].\mathsf{end} \quad\leqslant\quad ![\,![\mathtt{Real}].\mathsf{end}].\mathsf{end}$$

$$\oplus\begin{Bmatrix}\mathtt{win} : \mathsf{end} \\ \mathtt{loss} : \mathsf{end}\end{Bmatrix} \quad\geqslant\quad \oplus\begin{Bmatrix}\mathtt{win} : \mathsf{end} \\ \mathtt{loss} : \mathsf{end} \\ \mathtt{tie} : \mathsf{end}\end{Bmatrix}$$

$$!\left[\oplus\begin{Bmatrix}\mathtt{win} : \mathsf{end} \\ \mathtt{loss} : \mathsf{end}\end{Bmatrix}\right].\mathsf{end} \quad\leqslant\quad !\left[\oplus\begin{Bmatrix}\mathtt{win} : \mathsf{end} \\ \mathtt{loss} : \mathsf{end} \\ \mathtt{tie} : \mathsf{end}\end{Bmatrix}\right].\mathsf{end}$$

$$\&\begin{Bmatrix}\mathtt{win} : ?[\mathtt{Real}].\mathsf{end} \\ \mathtt{loss} : ![\mathtt{Real}].\mathsf{end}\end{Bmatrix} \quad\nleqslant\ngeqslant\quad \&\begin{Bmatrix}\mathtt{win} : ?[\mathtt{Int}].\mathsf{end} \\ \mathtt{loss} : ![\mathtt{Int}].\mathsf{end}\end{Bmatrix}$$

Basic properties

Proposition

$\leqslant$ *is a partial order*

Proposition

$s \leqslant t$ *can be decided in linear time*

Proof.

Subtyping rules are syntax directed.
Let $|t|$ be the number of subterms in t. In a derivation for $s \leqslant t$ each subterm of s (resp. t) occurs at most once. □

Basic properties

Proposition

$\leqslant$ is a partial order

Proposition

$s \leqslant t$ can be decided in linear time

Proof.

Subtyping rules are syntax directed.
Let $|t|$ be the number of subterms in t. In a derivation for $s \leqslant t$ each subterm of s (resp. t) occurs at most once. □

Type system (excerpt)

$$\frac{}{\vdash u![v].P}\ [\text{t-outS}]$$

Type system (excerpt)

$$\frac{}{u : \,!\,[T]\,.\,T' \vdash u!\,[v].P}\ \text{[t-outS]}$$

Type system (excerpt)

$$\frac{}{v : S, u : \mathtt{!}[T]\,.\,T' \vdash u\mathtt{!}[v].P}\ \text{[t-outS]}$$

Type system (excerpt)

$$\frac{S \leqslant T}{v : S, u : \mathtt{!}[T]\,.\,T' \vdash u\mathtt{!}[v].P}\ \text{[t-outS]}$$

Type system (excerpt)

$$\frac{S \leqslant T \qquad \Gamma, u : T' \vdash P}{\Gamma, v : S, u : {!}[T]\,.\,T' \vdash u{!}[v].P}\ \text{[t-outS]}$$

Type system (excerpt)

$$\frac{S \leqslant T \qquad \Gamma, u : T' \vdash P}{\Gamma, v : S, u : ![T].T' \vdash u![v].P} \text{ [t-outS]}$$

$$\frac{}{\vdash u \triangleright \{l_j : P_j\}_{j \in J}} \text{ [t-offer]}$$

Type system (excerpt)

$$\frac{S \leqslant T \qquad \Gamma, u : T' \vdash P}{\Gamma, v : S, u : ![T].T' \vdash u![v].P} \text{ [t-outS]}$$

$$\frac{}{u : \&\{l_i : T_i\}_{i \in I} \vdash u \triangleright \{l_j : P_j\}_{j \in J}} \text{ [t-offer]}$$

Type system (excerpt)

$$\frac{S \leqslant T \qquad \Gamma, u : T' \vdash P}{\Gamma, v : S, u : \,![T]\,.\,T' \vdash u![v].P} \text{ [t-outS]}$$

$$\frac{I \subseteq J}{u : \&\{l_i : T_i\}_{i \in I} \vdash u \triangleright \{l_j : P_j\}_{j \in J}} \text{ [t-offer]}$$

Type system (excerpt)

$$\frac{S \leqslant T \qquad \Gamma, u : T' \vdash P}{\Gamma, v : S, u : {!}[T].T' \vdash u{!}[v].P} \text{ [t-outS]}$$

$$\frac{I \subseteq J \qquad \Gamma, u : T_i \vdash P_i \ ^{(i \in I)}}{\Gamma, u : \&\{l_i : T_i\}_{i \in I} \vdash u \triangleright \{l_j : P_j\}_{j \in J}} \text{ [t-offer]}$$

Type system (continued)

$$\frac{S \leqslant T \qquad \Gamma, u : T', x : T \vdash P}{\Gamma, u : ?[S]\,.\,T' \vdash u?(x).P} \text{ [t-inS]}$$

$$\frac{k \in I \qquad \Gamma, u : T_k \vdash P}{\Gamma, u : \oplus\{l_i : T_i\}_{i \in I} \vdash u \lhd l_k.P} \text{ [t-choose]}$$

The substitution lemma

Lemma (substitution)

If

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

then

- $\Gamma, a : S \vdash P\{a/x\}$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$$\frac{T \leqslant S' \qquad \Gamma', u : T' \vdash Q}{\Gamma', u : \,![S']\,.T', x : T \vdash u![x].Q} \Rightarrow \overline{\qquad\qquad\qquad\qquad}$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$a, x \notin \mathsf{fn}(Q)$

$$\frac{T \leqslant S' \qquad \Gamma', u : T' \vdash Q}{\Gamma', u : {!}[S'].T', x : T \vdash u{!}[x].Q} \Rightarrow \overline{\qquad\qquad\qquad}$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$a, x \notin \mathsf{fn}(Q)$

$$\frac{T \leqslant S' \qquad \Gamma', u : T' \vdash Q}{\Gamma', u : \,![S'].T', x : T \vdash u![x].Q} \Rightarrow \frac{\Gamma', u : T' \vdash Q}{}$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

transitivity

$$\frac{T \leqslant S' \qquad \Gamma', u : T' \vdash Q}{\Gamma', u : {!}[S'].T', x : T \vdash u![x].Q} \Rightarrow \frac{S \leqslant S' \qquad \Gamma', u : T' \vdash Q}{}$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$$\frac{T \leqslant S' \qquad \Gamma', u : T' \vdash Q}{\Gamma', u : \,![S']\,.T', x : T \vdash u![x].Q} \Rightarrow \frac{S \leqslant S' \qquad \Gamma', u : T' \vdash Q}{\Gamma', u : \,![S']\,.T', a : S \vdash u![a].Q}$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$$\frac{s \leqslant t \qquad \Gamma', x : T' \vdash Q}{\Gamma', x : {!}[t]\,.T', u : s \vdash x{!}[u].Q} \Rightarrow \frac{}{}$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$$\frac{s \leqslant t \qquad \Gamma', x : T' \vdash Q}{\Gamma', x : \,![t]\,.T', u : s \vdash x![u].Q} \Rightarrow \overline{\qquad\qquad}$$

$T = \,![t]\,.T'$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$$\frac{s \leqslant t \qquad \Gamma', x : T' \vdash Q}{\Gamma', x : ![t].T', u : s \vdash x![u].Q} \Rightarrow \overline{}$$

$$![s'].S' = S \leqslant T = ![t].T'$$

$$t \leqslant s' \qquad S' \leqslant T'$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

ind. hyp. $(S' \leqslant T')$

$$\frac{s \leqslant t \qquad \Gamma', x : T' \vdash Q}{\Gamma', x : \,!\,[t]\,.\,T', u : s \vdash x\,!\,[u].Q} \Rightarrow \frac{\Gamma', a : S' \vdash Q\{a/x\}}{}$$

$$!\,[s']\,.S' = S \leqslant T = \,!\,[t]\,.\,T'$$

$$t \leqslant s' \qquad S' \leqslant T'$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

transitivity

$$\frac{s \leqslant t \qquad \Gamma', x : T' \vdash Q}{\Gamma', x : {!}[t]\,.T', u : s \vdash x!\,[u].Q} \Rightarrow \frac{s \leqslant s' \qquad \Gamma', a : S' \vdash Q\{a/x\}}{}$$

$${!}[s']\,.S' = S \leqslant T = {!}[t]\,.T'$$

$$t \leqslant s' \qquad S' \leqslant T'$$

The substitution lemma: proof

Hypotheses

- $\Gamma, x : T \vdash P$
- $a \notin \mathsf{dom}(\Gamma)$
- $S \leqslant T$

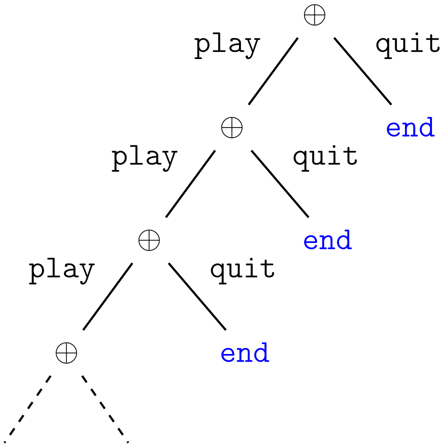
Thesis

- $\Gamma, a : S \vdash P\{a/x\}$

$$\frac{s \leqslant t \qquad \Gamma', x : T' \vdash Q}{\Gamma', x : \,![t]\,.T', u : s \vdash x![u].Q} \Rightarrow \frac{s \leqslant s' \qquad \Gamma', a : S' \vdash Q\{a/x\}}{\Gamma', a : \,![s']\,.S', u : S \vdash a![u].Q\{a/x\}}$$

Outline

Recursive session types: motivation



- **infinite** protocols
- **finite** but **arbitrarily long** protocols

Recursive session types: syntax

- infinite supply of type variables X, Y, ...

$$
\begin{array}{rclll}
T & ::= & & & \textbf{session type} \\
 & & \vdots & & \text{as before} \\
 & | & X & & \text{(session type variable)} \\
 & | & \mu X.T & & \text{(recursion)}
\end{array}
$$

- types identified modulo renaming of bound type variables
- $\mu X_1 \cdots \mu X_n.X_1$ subterms forbidden

Example

$$\mu X.\oplus\begin{Bmatrix}\texttt{play} : X \\ \texttt{quit} : \texttt{end}\end{Bmatrix}$$

Example

$$\mu X. \oplus \begin{Bmatrix} \texttt{play} : X \\ \texttt{quit} : \texttt{end} \end{Bmatrix}$$

Example

$$\mu X.\oplus\begin{Bmatrix}\texttt{play} : X \\ \texttt{quit} : \color{blue}{\texttt{end}}\end{Bmatrix}$$

$$\wr\wr$$

$$\oplus\begin{Bmatrix}\texttt{play} : \mu X.\oplus\begin{Bmatrix}\texttt{play} : X \\ \texttt{quit} : \color{blue}{\texttt{end}}\end{Bmatrix} \\ \texttt{quit} : \color{blue}{\texttt{end}}\end{Bmatrix}$$

Example

$$\mu X.\oplus\left\{\begin{array}{l}\mathtt{play} : X\\ \mathtt{quit} : \mathtt{end}\end{array}\right\}$$

$$\wr$$

$$\oplus\left\{\begin{array}{l}\mathtt{play} : \mu X.\oplus\left\{\begin{array}{l}\mathtt{play} : X\\ \mathtt{quit} : \mathtt{end}\end{array}\right\}\\ \mathtt{quit} : \mathtt{end}\end{array}\right\}$$

$$\wr$$

$$\oplus\left\{\begin{array}{l}\mathtt{play} : \oplus\left\{\begin{array}{l}\mathtt{play} : \mu X.\oplus\left\{\begin{array}{l}\mathtt{play} : X\\ \mathtt{quit} : \mathtt{end}\end{array}\right\}\\ \mathtt{quit} : \mathtt{end}\end{array}\right\}\\ \mathtt{quit} : \mathtt{end}\end{array}\right\}$$

$$\wr$$

$$\vdots$$

On contractiveness

Intuition

$$\mu X.T$$

denotes the (possibly infinite) protocol that is solution of the equation

$$X = T$$

Problem

$\mu X.X$ that is the equation $X = X$

has infinitely many solutions, which one do we mean?

Theorem (Courcelle 1983)

Contractive (systems of) equations have ***exactly*** *one solution*

On contractiveness

Intuition

$$\mu X.T$$

denotes the (possibly infinite) protocol that is solution of the equation

$$X = T$$

Problem

$\mu X.X$ that is the equation $X = X$

has infinitely many solutions, which one do we mean?

Theorem (Courcelle 1983)

Contractive (systems of) equations have ***exactly*** *one solution*

On contractiveness

Intuition

$$\mu X.T$$

denotes the (possibly infinite) protocol that is solution of the equation

$$X = T$$

Problem

$\mu X.X$ that is the equation $X = X$

has infinitely many solutions, which one do we mean?

Theorem (Courcelle 1983)

Contractive (systems of) equations have **exactly** *one solution*

Subtyping recursive types: the Amber rule

$$\frac{}{\mu X.S \leqslant \mu Y.T}$$

Subtyping recursive types: the Amber rule

$$\frac{S \leqslant T}{\mu X.S \leqslant \mu Y.T}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Example

$$\frac{}{\vdash \mu X.!\,[\mathtt{Real}]\,.X \leqslant \mu Y.!\,[\mathtt{Int}]\,.Y}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Example

$$\frac{\overline{X \leqslant Y \vdash \texttt{!}[\texttt{Real}].X \leqslant \texttt{!}[\texttt{Int}].Y}}{\vdash \mu X.\texttt{!}[\texttt{Real}].X \leqslant \mu Y.\texttt{!}[\texttt{Int}].Y}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Example

$$\dfrac{\dfrac{\overline{X \leqslant Y \vdash \texttt{Int} \leqslant \texttt{Real}}}{X \leqslant Y \vdash \,!\,[\texttt{Real}]\,.X \leqslant \,!\,[\texttt{Int}]\,.Y}}{\vdash \mu X.!\,[\texttt{Real}]\,.X \leqslant \mu Y.!\,[\texttt{Int}]\,.Y}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Example

$$\frac{\dfrac{}{X \leqslant Y \vdash \texttt{Int} \leqslant \texttt{Real}} \qquad \dfrac{}{X \leqslant Y \vdash X \leqslant Y}}{\dfrac{X \leqslant Y \vdash \,!\,[\texttt{Real}]\,.X \leqslant \,!\,[\texttt{Int}]\,.Y}{\vdash \mu X.!\,[\texttt{Real}]\,.X \leqslant \mu Y.!\,[\texttt{Int}]\,.Y}}$$

Subtyping recursive types: the Amber rule

$$\frac{\Sigma, X \leqslant Y \vdash S \leqslant T}{\Sigma \vdash \mu X.S \leqslant \mu Y.T} \qquad \frac{(X \leqslant Y) \in \Sigma}{\Sigma \vdash X \leqslant Y}$$

Example

$$\dfrac{\dfrac{\overline{X \leqslant Y \vdash \texttt{Int} \leqslant \texttt{Real}} \qquad \overline{X \leqslant Y \vdash X \leqslant Y}}{X \leqslant Y \vdash \texttt{!}[\texttt{Real}].X \leqslant \texttt{!}[\texttt{Int}].Y}}{\vdash \mu X.\texttt{!}[\texttt{Real}].X \leqslant \mu Y.\texttt{!}[\texttt{Int}].Y}$$

Problem

$$\mu X.\texttt{!}[\texttt{Int}].X \not\leqslant \mu Y.\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].Y \qquad \texttt{end} \not\leqslant \mu X.\texttt{end}$$

- some types are not related even though the protocols they denote are related (or equal)

Unfolding

A recursive session type… $\mu X.T$

…and its unfolding $T\{\mu X.T/X\}$

Proposition

As we unfold a session type, the number of topmost μ's decreases

Idea

We can unfold a session type up to its topmost non-μ constructor

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

Unfolding

A recursive session type...
$\mu X.T$

...and its unfolding
$T\{\mu X.T/X\}$

Proposition

As we unfold a session type, the number of topmost μ's decreases

Idea

We can unfold a session type up to its topmost non-μ constructor

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

Unfolding

A recursive session type. . .
$\mu X.T$

. . . and its unfolding
$T\{\mu X.T/X\}$

Proposition

As we unfold a session type, the number of topmost μ's decreases

Idea

We can unfold a session type up to its topmost non-μ constructor

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

Unfoldings vs the Amber rule

$$T \stackrel{\text{def}}{=} \mu X.\,!\,[\texttt{Int}]\,.\,!\,[\texttt{Int}]\,.X$$

$$\frac{}{T \leqslant \,!\,[\texttt{Int}]\,.\,T}$$

Unfoldings vs the Amber rule

$$T \stackrel{\text{def}}{=} \mu X.\,\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].X$$

$$\frac{\overline{\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].T \leqslant \texttt{!}[\texttt{Int}].T}}{T \leqslant \texttt{!}[\texttt{Int}].T}$$

Unfoldings vs the Amber rule

$$T \stackrel{\text{def}}{=} \mu X.\,\texttt{!}[\texttt{Int}]\,.\,\texttt{!}[\texttt{Int}]\,.X$$

$$\dfrac{\dfrac{\overline{\texttt{!}[\texttt{Int}]\,.\,T \leqslant T}}{\texttt{!}[\texttt{Int}]\,.\,\texttt{!}[\texttt{Int}]\,.\,T \leqslant \texttt{!}[\texttt{Int}]\,.\,T}}{T \leqslant \texttt{!}[\texttt{Int}]\,.\,T}$$

Unfoldings vs the Amber rule

$$T \stackrel{\text{def}}{=} \mu X.\,!\,[\texttt{Int}]\,.\,!\,[\texttt{Int}]\,.X$$

$$\dfrac{\dfrac{\dfrac{\overline{!\,[\texttt{Int}]\,.\,T \leqslant\, !\,[\texttt{Int}]\,.\,!\,[\texttt{Int}]\,.\,T}}{!\,[\texttt{Int}]\,.\,T \leqslant T}}{!\,[\texttt{Int}]\,.\,!\,[\texttt{Int}]\,.\,T \leqslant\, !\,[\texttt{Int}]\,.\,T}}{T \leqslant\, !\,[\texttt{Int}]\,.\,T}$$

Unfoldings vs the Amber rule

$$T \stackrel{\text{def}}{=} \mu X.\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].X$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{T \leqslant \texttt{!}[\texttt{Int}].T}}{\texttt{!}[\texttt{Int}].T \leqslant \texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].T}}{\texttt{!}[\texttt{Int}].T \leqslant T}}{\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].T \leqslant \texttt{!}[\texttt{Int}].T}}{T \leqslant \texttt{!}[\texttt{Int}].T}$$

Unfoldings vs the Amber rule

$$T \stackrel{\text{def}}{=} \mu X.\,\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].X$$

$$\begin{array}{c}
\vdots \\
\hline
T \leqslant \texttt{!}[\texttt{Int}].T \\
\hline
\texttt{!}[\texttt{Int}].T \leqslant \texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].T \\
\hline
\texttt{!}[\texttt{Int}].T \leqslant T \\
\hline
\texttt{!}[\texttt{Int}].\texttt{!}[\texttt{Int}].T \leqslant \texttt{!}[\texttt{Int}].T \\
\hline
T \leqslant \texttt{!}[\texttt{Int}].T
\end{array}$$

Observations

- no unfolding yields terms with matching μ's
- we must give up the idea of using a finite derivation for relating recursive types

Type simulation

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $s = \texttt{Int}$ and $t = \texttt{Real}$
- $s = t = \texttt{Real}$
- $\mathsf{unfold}(s) = ?[s'].S$ and $\mathsf{unfold}(t) = ?[t'].T$ and $s' \mathrel{\mathscr{R}} t'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = ![s'].S$ and $\mathsf{unfold}(t) = ![t'].T$ and $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I \subseteq J$ and $S_i \mathrel{\mathscr{R}} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathrel{\mathscr{R}} T_j$ for every $j \in J$

Type simulation

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathscr{R} t$ implies either

- $s = t = \texttt{Int}$
- $s = \texttt{Int}$ and $t = \texttt{Real}$
- $s = t = \texttt{Real}$
- $\mathsf{unfold}(s) = ?[s'].S$ and $\mathsf{unfold}(t) = ?[t'].T$ and $s' \mathscr{R} t'$ and $S \mathscr{R} T$
- $\mathsf{unfold}(s) = ![s'].S$ and $\mathsf{unfold}(t) = ![t'].T$ and $t' \mathscr{R} s'$ and $S \mathscr{R} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I \subseteq J$ and $S_i \mathscr{R} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathscr{R} T_j$ for every $j \in J$

Type simulation

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $s = \texttt{Int}$ and $t = \texttt{Real}$
- $s = t = \texttt{Real}$
- $\mathsf{unfold}(s) = {?}[s'].S$ and $\mathsf{unfold}(t) = {?}[t'].T$ and $s' \mathrel{\mathscr{R}} t'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = {!}[s'].S$ and $\mathsf{unfold}(t) = {!}[t'].T$ and $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I \subseteq J$ and $S_i \mathrel{\mathscr{R}} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathrel{\mathscr{R}} T_j$ for every $j \in J$

Type simulation

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $s = \texttt{Int}$ and $t = \texttt{Real}$
- $s = t = \texttt{Real}$
- $\mathsf{unfold}(s) = ?[s'].S$ and $\mathsf{unfold}(t) = ?[t'].T$ and $s' \mathrel{\mathscr{R}} t'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = ![s'].S$ and $\mathsf{unfold}(t) = ![t'].T$ and $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I \subseteq J$ and $S_i \mathrel{\mathscr{R}} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathrel{\mathscr{R}} T_j$ for every $j \in J$

Type simulation

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $s = \texttt{Int}$ and $t = \texttt{Real}$
- $s = t = \texttt{Real}$
- $\mathsf{unfold}(s) = \texttt{?}[s'].S$ and $\mathsf{unfold}(t) = \texttt{?}[t'].T$ and $s' \mathrel{\mathscr{R}} t'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \texttt{!}[s'].S$ and $\mathsf{unfold}(t) = \texttt{!}[t'].T$ and $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I \subseteq J$ and $S_i \mathrel{\mathscr{R}} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathrel{\mathscr{R}} T_j$ for every $j \in J$

Subtyping for recursive session types

Definition (subtyping)

Let $\leqslant$ be the largest type simulation, that is

$$\leqslant \stackrel{\text{def}}{=} \bigcup_{\mathscr{R} \text{ is a type simulation}} \mathscr{R}$$

Example

Given $T \stackrel{\text{def}}{=} \mu X.\mathtt{!}[\mathtt{Int}].\mathtt{!}[\mathtt{Int}].X$ show that $T \leqslant \mathtt{!}[\mathtt{Int}].T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \mathtt{Int}$
- $\mathsf{unfold}(s) = \mathtt{!}[s'].S' \qquad \mathsf{unfold}(t) = \mathtt{!}[t'].T' \qquad t' \mathrel{\mathscr{R}} s' \qquad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.\,\texttt{!}[\texttt{Int}]\,.\,\texttt{!}[\texttt{Int}]\,.X$ show that $T \leqslant \texttt{!}[\texttt{Int}]\,.\,T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T, \texttt{!}[\texttt{Int}]\,.\,T), (\texttt{!}[\texttt{Int}]\,.\,T, T), (\texttt{Int}, \texttt{Int})\}$$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $\mathsf{unfold}(s) = \texttt{!}[s']\,.S' \quad \mathsf{unfold}(t) = \texttt{!}[t']\,.\,T' \quad t' \mathrel{\mathscr{R}} s' \quad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.\,![\mathtt{Int}].\,![\mathtt{Int}].X$ show that $T \leqslant\ ![\mathtt{Int}].T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T, ![\mathtt{Int}].T), (![\mathtt{Int}].T, T), (\mathtt{Int}, \mathtt{Int})\}$$

$s = T$
$t = ![\mathtt{Int}].T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \mathtt{Int}$
- $\mathrm{unfold}(s) = ![s'].S' \quad \mathrm{unfold}(t) = ![t'].T' \quad t' \mathrel{\mathscr{R}} s' \quad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.\,![\mathtt{Int}].\,![\mathtt{Int}].X$ show that $T \leqslant\ ![\mathtt{Int}].T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T,\ ![\mathtt{Int}].T), (![\mathtt{Int}].T,\ T), (\mathtt{Int}, \mathtt{Int})\}$$

$\mathsf{unfold}(s) =\ ![\mathtt{Int}].![\mathtt{Int}].T$
$\mathsf{unfold}(t) =\ ![\mathtt{Int}].T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \mathtt{Int}$
- $\mathsf{unfold}(s) =\ ![s'].S' \qquad \mathsf{unfold}(t) =\ ![t'].T' \qquad t' \mathrel{\mathscr{R}} s' \qquad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.\,!\,[\texttt{Int}]\,.\,!\,[\texttt{Int}]\,.X$ show that $T \leqslant\, !\,[\texttt{Int}]\,.\,T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T,\, !\,[\texttt{Int}]\,.\,T),\, (!\,[\texttt{Int}]\,.\,T,\, T),\, (\texttt{Int}, \texttt{Int})\}$$

$s' = t' = \texttt{Int}$
$S' = \,!\,[\texttt{Int}]\,.\,T$
$T' = T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \,\mathscr{R}\, t$ implies either

- $s = t = \texttt{Int}$
- $\text{unfold}(s) = \,!\,[s']\,.S'$ $\quad$ $\text{unfold}(t) = \,!\,[t']\,.\,T'$ $\quad$ $t' \,\mathscr{R}\, s'$ $\quad$ $S' \,\mathscr{R}\, T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.\,!\,[\mathtt{Int}]\,.\,!\,[\mathtt{Int}]\,.X$ show that $T \leqslant\, !\,[\mathtt{Int}]\,.\,T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T,\, !\,[\mathtt{Int}]\,.\,T), (!\,[\mathtt{Int}]\,.\,T,\, T), (\mathtt{Int}, \mathtt{Int})\}$$

$s = \,!\,[\mathtt{Int}]\,.\,T$
$t = T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \mathtt{Int}$
- $\mathsf{unfold}(s) = \,!\,[s']\,.S' \quad \mathsf{unfold}(t) = \,!\,[t']\,.\,T' \quad t' \mathrel{\mathscr{R}} s' \quad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.![\mathtt{Int}].![\mathtt{Int}].X$ show that $T \leqslant ![\mathtt{Int}].T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T, ![\mathtt{Int}].T), (![\mathtt{Int}].T, T), (\mathtt{Int}, \mathtt{Int})\}$$

$\mathsf{unfold}(s) = ![\mathtt{Int}].T$
$\mathsf{unfold}(t) = ![\mathtt{Int}].![\mathtt{Int}].T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \mathtt{Int}$
- $\mathsf{unfold}(s) = ![s'].S' \quad \mathsf{unfold}(t) = ![t'].T' \quad t' \mathrel{\mathscr{R}} s' \quad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.![\texttt{Int}].![\texttt{Int}].X$ show that $T \leqslant ![\texttt{Int}].T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T, ![\texttt{Int}].T), (![\texttt{Int}].T, T), (\texttt{Int}, \texttt{Int})\}$$

$s' = t' = \texttt{Int}$
$S' = T$
$T' = ![\texttt{Int}].T$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $\mathsf{unfold}(s) = ![s'].S' \quad \mathsf{unfold}(t) = ![t'].T' \quad t' \mathrel{\mathscr{R}} s' \quad S' \mathrel{\mathscr{R}} T'$

Example

Given $T \stackrel{\text{def}}{=} \mu X.![\texttt{Int}].![\texttt{Int}].X$ show that $T \leqslant ![\texttt{Int}].T$

$$\mathscr{R} \stackrel{\text{def}}{=} \{(T, ![\texttt{Int}].T), (![\texttt{Int}].T, T), (\texttt{Int}, \texttt{Int})\}$$

$s = t = \texttt{Int}$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $\mathsf{unfold}(s) = ![s'].S' \quad \mathsf{unfold}(t) = ![t'].T' \quad t' \mathrel{\mathscr{R}} s' \quad S' \mathrel{\mathscr{R}} T'$

More examples

$$\mu X.\oplus\begin{Bmatrix}\texttt{play} : X \\ \texttt{quit} : \texttt{end}\end{Bmatrix} \leqslant \mu Y.\oplus\left\{\texttt{play} : \oplus\begin{Bmatrix}\texttt{play} : Y \\ \texttt{quit} : \texttt{end}\end{Bmatrix}\right\}$$

$$\mu X.\oplus\begin{Bmatrix}\texttt{play} : X \\ \texttt{quit} : \texttt{end}\end{Bmatrix} \leqslant \mu Y.\oplus\{\texttt{play} : Y\}$$

$$\mu X.\oplus\begin{Bmatrix}\texttt{play} : X \\ \texttt{quit} : \texttt{end}\end{Bmatrix} \leqslant \oplus\{\texttt{quit} : \texttt{end}\}$$

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ *is transitive*

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ *is transitive*

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s,t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathscr{R} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\text{unfold}(s) = ![s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\text{unfold}(u) = ![u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\text{unfold}(t) = ![t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathscr{R} s'$ and $S \mathscr{R} T$ by definition of $\mathscr{R}$.

... □

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ *is transitive*

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s, t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathrel{\mathscr{R}} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\mathsf{unfold}(s) = ![s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\mathsf{unfold}(u) = ![u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\mathsf{unfold}(t) = ![t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$ by definition of $\mathscr{R}$.

... □

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ *is transitive*

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s,t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathrel{\mathscr{R}} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\mathsf{unfold}(s) = \,![s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\mathsf{unfold}(u) = \,![u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\mathsf{unfold}(t) = \,![t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$ by definition of $\mathscr{R}$.

... □

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ *is transitive*

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s, t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathrel{\mathscr{R}} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\mathsf{unfold}(s) = {!}[s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\mathsf{unfold}(u) = {!}[u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\mathsf{unfold}(t) = {!}[t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$ by definition of $\mathscr{R}$.

... □

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ is transitive

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s,t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathrel{\mathscr{R}} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\mathsf{unfold}(s) = {!}[s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\mathsf{unfold}(u) = {!}[u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\mathsf{unfold}(t) = {!}[t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$ by definition of $\mathscr{R}$.

... □

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ *is transitive*

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s, t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathrel{\mathscr{R}} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\mathsf{unfold}(s) = ![s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\mathsf{unfold}(u) = ![u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\mathsf{unfold}(t) = ![t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$ by definition of $\mathscr{R}$.

...

□

$\leqslant$ is a pre-order (but not a partial order)

Lemma

$\leqslant$ is transitive

Proof.

It is enough to show that

$$\mathscr{R} \stackrel{\text{def}}{=} \{(s,t) \mid \exists u : s \leqslant u \wedge u \leqslant t\}$$

is a type simulation.

Suppose $s \mathrel{\mathscr{R}} t$. Then $s \leqslant u$ and $u \leqslant t$ for some type u.

...

Suppose $\mathsf{unfold}(s) = ![s'].S$. From the hypothesis $s \leqslant u$ we deduce that $\mathsf{unfold}(u) = ![u'].U$ and $u' \leqslant s'$ and $S \leqslant U$. From the hypothesis $u \leqslant t$ we deduce that $\mathsf{unfold}(t) = ![t'].T$ and $t' \leqslant u'$ and $U \leqslant T$. Then $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$ by definition of $\mathscr{R}$.

... □

Outline

Subtyping algorithm

```
subtype(s_0, t_0)
  ℛ := ∅
  𝒮 := {(s_0, t_0)}
  while 𝒮 ≠ ℛ do
    let (s, t) ∈ 𝒮 \ ℛ
    let s' = unfold(s)
    let t' = unfold(t)
    if s' = ![s''].S and t' = ![t''].T then
      𝒮 := 𝒮 ∪ {(t'', s''), (S, T)}
    else if s' = ⊕{l_i : S_i}_{i∈I} and t' = ⊕{l_j : T_j}_{j∈J} and J ⊆ I then
      𝒮 := 𝒮 ∪ {(S_j, T_j)}_{j∈J}
    else ···
    else return false
    ℛ := ℛ ∪ {(s, t)}
  return true
```

Subtyping algorithm

subtype(s_0, t_0)

$\mathscr{R} := \emptyset$ pairs that have been checked

$\mathscr{S} := \{(s_0, t_0)\}$

```
while 𝒮 ≠ ℛ do
  let (s, t) ∈ 𝒮 \ ℛ
  let s′ = unfold(s)
  let t′ = unfold(t)
  if s′ = ![s″].S and t′ = ![t″].T then
    𝒮 := 𝒮 ∪ {(t″, s″), (S, T)}
  else if s′ = ⊕{l_i : S_i}_{i∈I} and t′ = ⊕{l_j : T_j}_{j∈J} and J ⊆ I then
    𝒮 := 𝒮 ∪ {(S_j, T_j)}_{j∈J}
  else ···
  else return false
  ℛ := ℛ ∪ {(s, t)}
return true
```

Subtyping algorithm

```
subtype(s_0, t_0)
  R := ∅                      ← pairs that have been checked
  S := {(s_0, t_0)}           ← pairs that must be in R if s_0 ⩽ t_0
  while S ≠ R do
    let (s, t) ∈ S \ R
    let s' = unfold(s)
    let t' = unfold(t)
    if s' = ![s''].S and t' = ![t''].T then
      S := S ∪ {(t'', s''), (S, T)}
    else if s' = ⊕{l_i : S_i}_{i∈I} and t' = ⊕{l_j : T_j}_{j∈J} and J ⊆ I then
      S := S ∪ {(S_j, T_j)}_{j∈J}
    else ···
    else return false
    R := R ∪ {(s, t)}
  return true
```

Subtyping algorithm: example

$$s_0 \stackrel{\text{def}}{=} \mu X.\oplus \begin{Bmatrix} \texttt{play} : X \\ \texttt{quit} : \texttt{end} \end{Bmatrix} \qquad t_0 \stackrel{\text{def}}{=} \mu Y.\oplus \left\{ \texttt{play} : \oplus \begin{Bmatrix} \texttt{play} : Y \\ \texttt{quit} : \texttt{end} \end{Bmatrix} \right\}$$

$$\mathscr{R}^0 = \emptyset$$

$$\mathscr{S}^0 = \{(s_0, t_0)\}$$

$$\mathscr{R}^1 = \{(s_0, t_0)\}$$

$$\mathscr{S}^1 = \left\{(s_0, t_0), \left(s_0, \oplus \begin{Bmatrix} \texttt{play} : t_0 \\ \texttt{quit} : \texttt{end} \end{Bmatrix}\right)\right\}$$

$$\mathscr{R}^2 = \left\{(s_0, t_0), \left(s_0, \oplus \begin{Bmatrix} \texttt{play} : t_0 \\ \texttt{quit} : \texttt{end} \end{Bmatrix}\right)\right\}$$

$$\mathscr{S}^2 = \left\{(s_0, t_0), \left(s_0, \oplus \begin{Bmatrix} \texttt{play} : t_0 \\ \texttt{quit} : \texttt{end} \end{Bmatrix}\right), (\texttt{end}, \texttt{end})\right\}$$

Subtyping algorithm: example

$$s_0 \stackrel{\text{def}}{=} \mu X.\oplus \begin{Bmatrix} \texttt{play} : X \\ \texttt{quit} : \texttt{end} \end{Bmatrix} \qquad t_0 \stackrel{\text{def}}{=} \mu Y.\oplus \left\{ \texttt{play} : \oplus \begin{Bmatrix} \texttt{play} : Y \\ \texttt{quit} : \texttt{end} \end{Bmatrix} \right\}$$

$$\mathscr{R}^0 = \emptyset$$

$$\mathscr{S}^0 = \{(s_0, t_0)\}$$

$$\mathscr{R}^1 = \{(s_0, t_0)\}$$

$$\mathscr{S}^1 = \left\{(s_0, t_0), \left(s_0, \oplus \begin{Bmatrix} \texttt{play} : t_0 \\ \texttt{quit} : \texttt{end} \end{Bmatrix}\right)\right\}$$

$$\mathscr{R}^2 = \left\{(s_0, t_0), \left(s_0, \oplus \begin{Bmatrix} \texttt{play} : t_0 \\ \texttt{quit} : \texttt{end} \end{Bmatrix}\right)\right\}$$

$$\mathscr{S}^2 = \left\{(s_0, t_0), \left(s_0, \oplus \begin{Bmatrix} \texttt{play} : t_0 \\ \texttt{quit} : \texttt{end} \end{Bmatrix}\right), (\texttt{end}, \texttt{end})\right\}$$

Subtyping algorithm: correctness

subtype(s_0, t_0)

  $\mathscr{R} := \emptyset$

  $\mathscr{S} := \{(s_0, t_0)\}$

  while $\mathscr{S} \neq \mathscr{R}$ **do**

    let $(s, t) \in \mathscr{S} \setminus \mathscr{R}$

    let $s' =$ unfold(s)

    let $t' =$ unfold(t)

    if $s' = ![s''].S$ **and** $t' = ![t''].T$ **then**

      $\mathscr{S} := \mathscr{S} \cup \{(t'', s''), (S, T)\}$

    else if $s' = \oplus\{l_i : S_i\}_{i \in I}$ **and** $t' = \oplus\{l_j : T_j\}_{j \in J}$ **and** $J \subseteq I$ **then**

      $\mathscr{S} := \mathscr{S} \cup \{(S_j, T_j)\}_{j \in J}$

    else $\cdots$

    else return false

    $\mathscr{R} := \mathscr{R} \cup \{(s,$

  return true

Invariant each pair of types in $\mathscr{R}$ has been checked and the continuations are in $\mathscr{S}$

there is no type simulation that contains (s, t), let alone (s_0, t_0)

$\mathscr{R}$ is a type simulation that contains (s_0, t_0) hence $s_0 \leqslant t_0$

Subtyping algorithm: completeness

```
subtype(s₀, t₀)
  ℛ := ∅
  𝒮 := {(s₀, t₀)}
  while 𝒮 ≠ ℛ do
    let (s, t) ∈ 𝒮 \ ℛ
    let s′ = unfold(s)
    let t′ = unfold(t)
    if s′ = ![s″].S and t′ = ![t″].T then
      𝒮 := 𝒮 ∪ {(t″, s″), (S, T)}
    else if s′ = ⊕{l_i : S_i}_{i∈I} and t′ = ⊕{l_j : T_j}_{j∈J} and J ⊆ I then
      𝒮 := 𝒮 ∪ {(S_j, T_j)}_{j∈J}
    else ···
    else return false
    ℛ := ℛ ∪ {(s, t)}
  return true
```

Subtyping algorithm: completeness

```
subtype(s₀, t₀)
  ℛ := ∅
  𝒮 := {(s₀, t₀)}
  while 𝒮 ≠ ℛ do
    let (s, t) ∈ 𝒮 \ ℛ
    let s′ = unfold(s)
    let t′ = unfold(t)
    if s′ = ![s″].S and t′ = ![t″].T then
      𝒮 := 𝒮 ∪ {(t″, s″), (S, T)}
    else if s′ = ⊕{l_i : S_i}_{i∈I} and t′ = ⊕{l_j : T_j}_{j∈J} and J ⊆ I then
      𝒮 := 𝒮 ∪ {(S_j, T_j)}_{j∈J}
    else ···
    else return false
    ℛ := ℛ ∪ {(s, t)}
  return true
```

will it ever terminate?

Computing the subterms of a type

$$\mathrm{Sub}(t) \stackrel{\text{def}}{=} \begin{cases} \{t\} \cup \mathrm{Sub}(s) \cup \mathrm{Sub}(T) & \text{if } t = ?[s].T \\ & \quad \text{or } t = ![s].T \\ \{t\} \cup \bigcup_{i \in I} \mathrm{Sub}(T_i) & \text{if } t = \&\{l_i : T_i\}_{i \in I} \\ & \quad \text{or } t = \oplus\{l_i : T_i\}_{i \in I} \\ \{t\} \cup \mathrm{Sub}(T)\{t/X\} & \text{if } t = \mu X.T \\ \{t\} & \text{otherwise} \end{cases}$$

$$\mathrm{Sub}(\{t_1, \ldots, t_n\}) \stackrel{\text{def}}{=} \bigcup_{1 \le i \le n} \mathrm{Sub}(t_i)$$

Properties of Sub

Proposition

Sub(t) *is* **finite** *for every* t

Proof.

Easy induction on t. □

Lemma (closure)

Let H, K *be sets of types. Then:*

- $H \subseteq \mathrm{Sub}(H)$
- $H \subseteq K$ *implies* $\mathrm{Sub}(H) \subseteq \mathrm{Sub}(K)$
- $\mathrm{Sub}(\mathrm{Sub}(H)) = \mathrm{Sub}(H)$

Idempotency of Sub

Proof.

We prove $\mathsf{Sub}(\mathsf{Sub}(t)) = \mathsf{Sub}(t)$. Using the first two properties of Sub we have $\{t\} \subseteq \mathsf{Sub}(t)$ hence $\mathsf{Sub}(t) \subseteq \mathsf{Sub}(\mathsf{Sub}(t))$.

We prove $\mathsf{Sub}(\mathsf{Sub}(t)) \subseteq \mathsf{Sub}(t)$ by induction on t.

Suppose $t = \mu X.T$. Then

$$\mathsf{Sub}(t) = \{t\} \cup \mathsf{Sub}(T)\{t/X\} \qquad \text{def. of Sub}$$

$$\begin{aligned}
\mathsf{Sub}(\mathsf{Sub}(t)) &= \mathsf{Sub}(t) \cup \mathsf{Sub}(\mathsf{Sub}(T)\{t/X\}) && \text{def.} \\
&\subseteq \mathsf{Sub}(t) \cup \mathsf{Sub}(\mathsf{Sub}(T))\{t/X\} && \text{conjecture} \\
&\subseteq \mathsf{Sub}(t) \cup \mathsf{Sub}(T)\{t/X\} && \text{ind. hyp.} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}$$

□

Idempotency of Sub

Proof.

We prove $\mathsf{Sub}(\mathsf{Sub}(t)) = \mathsf{Sub}(t)$. Using the first two properties of Sub we have $\{t\} \subseteq \mathsf{Sub}(t)$ hence $\mathsf{Sub}(t) \subseteq \mathsf{Sub}(\mathsf{Sub}(t))$.

We prove $\mathsf{Sub}(\mathsf{Sub}(t)) \subseteq \mathsf{Sub}(t)$ by induction on t.

Suppose $t = \mu X.T$. Then

$$\mathsf{Sub}(t) = \{t\} \cup \mathsf{Sub}(T)\{t/X\} \qquad \text{def. of Sub}$$

$$\begin{aligned}
\mathsf{Sub}(\mathsf{Sub}(t)) &= \mathsf{Sub}(t) \cup \mathsf{Sub}(\mathsf{Sub}(T)\{t/X\}) && \text{def.} \\
&\subseteq \mathsf{Sub}(t) \cup \mathsf{Sub}(\mathsf{Sub}(T))\{t/X\} && \text{conjecture} \\
&\subseteq \mathsf{Sub}(t) \cup \mathsf{Sub}(T)\{t/X\} && \text{ind. hyp.} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}$$

□

Idempotency of Sub

Proof.

We prove $\mathsf{Sub}(\mathsf{Sub}(t)) = \mathsf{Sub}(t)$. Using the first two properties of Sub we have $\{t\} \subseteq \mathsf{Sub}(t)$ hence $\mathsf{Sub}(t) \subseteq \mathsf{Sub}(\mathsf{Sub}(t))$.

We prove $\mathsf{Sub}(\mathsf{Sub}(t)) \subseteq \mathsf{Sub}(t)$ by induction on t.

Suppose $t = \mu X.T$. Then

$$\mathsf{Sub}(t) = \{t\} \cup \mathsf{Sub}(T)\{t/X\} \qquad \text{def. of Sub}$$

$$\begin{aligned}
\mathsf{Sub}(\mathsf{Sub}(t)) &= \mathsf{Sub}(t) \cup \mathsf{Sub}(\mathsf{Sub}(T)\{t/X\}) && \text{def.} \\
&\subseteq \mathsf{Sub}(t) \cup \mathsf{Sub}(\mathsf{Sub}(T))\{t/X\} && \text{conjecture} \\
&\subseteq \mathsf{Sub}(t) \cup \mathsf{Sub}(T)\{t/X\} && \text{ind. hyp.} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}$$

□

Idempotency of Sub

Proof.

We prove $\mathrm{Sub}(\mathrm{Sub}(t)) = \mathrm{Sub}(t)$. Using the first two properties of Sub we have $\{t\} \subseteq \mathrm{Sub}(t)$ hence $\mathrm{Sub}(t) \subseteq \mathrm{Sub}(\mathrm{Sub}(t))$.

We prove $\mathrm{Sub}(\mathrm{Sub}(t)) \subseteq \mathrm{Sub}(t)$ by induction on t.

Suppose $t = \mu X.T$. Then

$$\mathrm{Sub}(t) = \{t\} \cup \mathrm{Sub}(T)\{t/X\} \qquad \text{def. of Sub}$$

$$\begin{aligned}
\mathrm{Sub}(\mathrm{Sub}(t)) &= \mathrm{Sub}(t) \cup \mathrm{Sub}(\mathrm{Sub}(T)\{t/X\}) && \text{def.} \\
&\subseteq \mathrm{Sub}(t) \cup \mathrm{Sub}(\mathrm{Sub}(T))\{t/X\} && \text{conjecture} \\
&\subseteq \mathrm{Sub}(t) \cup \mathrm{Sub}(T)\{t/X\} && \text{ind. hyp.} \\
&= \mathrm{Sub}(t) && \text{def. of Sub}
\end{aligned}$$

□

Key closure lemma

Lemma

Let $\mathsf{btv}(t) \cap \mathsf{ftv}(S) = \emptyset$*. Then* $\mathsf{Sub}(t\{S/Y\}) \subseteq \mathsf{Sub}(t)\{S/Y\} \cup \mathsf{Sub}(S)$*.*

Proof.

By induction on t. Suppose $t = \mu X.T$ and $X \neq Y$. Then

$$\begin{aligned}
\mathsf{Sub}(t\{S/Y\}) &= \mathsf{Sub}(\mu X.T\{S/Y\}) \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T\{S/Y\})\{t\{S/Y\}/X\} \\
&\subseteq \{t\{S/Y\}\} \cup (\mathsf{Sub}(T)\{S/Y\} \cup \mathsf{Sub}(S))\{t\{S/Y\}/X\} \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T)\{S/Y\}\{t\{S/Y\}/X\} \cup \mathsf{Sub}(S) \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T)\{t/X\}\{S/Y\} \cup \mathsf{Sub}(S) \\
&= (\{t\} \cup \mathsf{Sub}(T)\{t/X\})\{S/Y\} \cup \mathsf{Sub}(S) \\
&= \mathsf{Sub}(t)\{S/Y\} \cup \mathsf{Sub}(S)
\end{aligned}$$

□

Key closure lemma

Lemma

Let $\mathsf{btv}(t) \cap \mathsf{ftv}(S) = \emptyset$*. Then* $\mathsf{Sub}(t\{S/Y\}) \subseteq \mathsf{Sub}(t)\{S/Y\} \cup \mathsf{Sub}(S)$*.*

Proof.

By induction on t. Suppose $t = \mu X.T$ and $X \neq Y$. Then

$$\begin{aligned}
\mathsf{Sub}(t\{S/Y\}) &= \mathsf{Sub}(\mu X.T\{S/Y\}) \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T\{S/Y\})\{t\{S/Y\}/X\} \\
&\subseteq \{t\{S/Y\}\} \cup (\mathsf{Sub}(T)\{S/Y\} \cup \mathsf{Sub}(S))\{t\{S/Y\}/X\} \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T)\{S/Y\}\{t\{S/Y\}/X\} \cup \mathsf{Sub}(S) \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T)\{t/X\}\{S/Y\} \cup \mathsf{Sub}(S) \\
&= (\{t\} \cup \mathsf{Sub}(T)\{t/X\})\{S/Y\} \cup \mathsf{Sub}(S) \\
&= \mathsf{Sub}(t)\{S/Y\} \cup \mathsf{Sub}(S)
\end{aligned}$$

□

Key closure lemma

Lemma

Let $\mathsf{btv}(t) \cap \mathsf{ftv}(S) = \emptyset$. *Then* $\mathsf{Sub}(t\{S/Y\}) \subseteq \mathsf{Sub}(t)\{S/Y\} \cup \mathsf{Sub}(S)$.

Proof.

By induction on t. Suppose $t = \mu X.T$ and $X \neq Y$. Then

$$
\begin{aligned}
\mathsf{Sub}(t\{S/Y\}) &= \mathsf{Sub}(\mu X.T\{S/Y\}) \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T\{S/Y\})\{t\{S/Y\}/X\} \\
&\subseteq \{t\{S/Y\}\} \cup (\mathsf{Sub}(T)\{S/Y\} \cup \mathsf{Sub}(S))\{t\{S/Y\}/X\} \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T)\{S/Y\}\{t\{S/Y\}/X\} \cup \mathsf{Sub}(S) \\
&= \{t\{S/Y\}\} \cup \mathsf{Sub}(T)\{t/X\}\{S/Y\} \cup \mathsf{Sub}(S) \\
&= (\{t\} \cup \mathsf{Sub}(T)\{t/X\})\{S/Y\} \cup \mathsf{Sub}(S) \\
&= \mathsf{Sub}(t)\{S/Y\} \cup \mathsf{Sub}(S)
\end{aligned}
$$

□

Key closure lemma

Lemma

Let $\mathrm{btv}(t) \cap \mathrm{ftv}(S) = \emptyset$*. Then* $\mathrm{Sub}(t\{S/Y\}) \subseteq \mathrm{Sub}(t)\{S/Y\} \cup \mathrm{Sub}(S)$*.*

Proof.

By induction on t. Suppose $t = \mu X.T$ and $X \neq Y$. Then

$$
\begin{aligned}
\mathrm{Sub}(t\{S/Y\}) &= \mathrm{Sub}(\mu X.T\{S/Y\}) \\
&= \{t\{S/Y\}\} \cup \mathrm{Sub}(T\{S/Y\})\{t\{S/Y\}/X\} \\
&\subseteq \{t\{S/Y\}\} \cup (\mathrm{Sub}(T)\{S/Y\} \cup \mathrm{Sub}(S))\{t\{S/Y\}/X\} \\
&= \{t\{S/Y\}\} \cup \mathrm{Sub}(T)\{S/Y\}\{t\{S/Y\}/X\} \cup \mathrm{Sub}(S) \\
&= \{t\{S/Y\}\} \cup \mathrm{Sub}(T)\{t/X\}\{S/Y\} \cup \mathrm{Sub}(S) \\
&= (\{t\} \cup \mathrm{Sub}(T)\{t/X\})\{S/Y\} \cup \mathrm{Sub}(S) \\
&= \mathrm{Sub}(t)\{S/Y\} \cup \mathrm{Sub}(S)
\end{aligned}
$$

□

Relation between unfold and Sub

Lemma

$\mathsf{unfold}(t) \in \mathsf{Sub}(t)$

Proof.

By induction on the number n of topmost μs in t.

- ($n = 0$) Then $\mathsf{unfold}(t) = t \in \mathsf{Sub}(t)$
- ($n > 0$) Then $t = \mu X.T$, we have

$$\begin{aligned}
\mathsf{unfold}(t) &= \mathsf{unfold}(T\{t/X\}) && \text{def. of unfold} \\
&\in \mathsf{Sub}(T\{t/X\}) && \text{contr. + ind. hyp.} \\
&\subseteq \mathsf{Sub}(T)\{t/X\} \cup \mathsf{Sub}(t) && \text{key closure lemma} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}$$

□

Relation between unfold and Sub

Lemma

$\mathsf{unfold}(t) \in \mathsf{Sub}(t)$

Proof.

By induction on the number n of topmost μs in t.

- $(n = 0)$ Then $\mathsf{unfold}(t) = t \in \mathsf{Sub}(t)$
- $(n > 0)$ Then $t = \mu X.T$, we have

$$
\begin{aligned}
\mathsf{unfold}(t) &= \mathsf{unfold}(T\{t/X\}) && \text{def. of unfold} \\
&\in \mathsf{Sub}(T\{t/X\}) && \text{contr. + ind. hyp.} \\
&\subseteq \mathsf{Sub}(T)\{t/X\} \cup \mathsf{Sub}(t) && \text{key closure lemma} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}
$$

$\square$

Relation between unfold and Sub

Lemma

$\mathsf{unfold}(t) \in \mathsf{Sub}(t)$

Proof.

By induction on the number n of topmost μs in t.

- $(n = 0)$ Then $\mathsf{unfold}(t) = t \in \mathsf{Sub}(t)$
- $(n > 0)$ Then $t = \mu X.T$, we have

$$\begin{aligned}
\mathsf{unfold}(t) &= \mathsf{unfold}(T\{t/X\}) && \text{def. of unfold} \\
&\in \mathsf{Sub}(T\{t/X\}) && \text{contr. + ind. hyp.} \\
&\subseteq \mathsf{Sub}(T)\{t/X\} \cup \mathsf{Sub}(t) && \text{key closure lemma} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}$$

□

Relation between unfold and Sub

Lemma

$\mathsf{unfold}(t) \in \mathsf{Sub}(t)$

Proof.

By induction on the number n of topmost μs in t.

- $(n = 0)$ Then $\mathsf{unfold}(t) = t \in \mathsf{Sub}(t)$
- $(n > 0)$ Then $t = \mu X.T$, we have

$$
\begin{aligned}
\mathsf{unfold}(t) &= \mathsf{unfold}(T\{t/X\}) && \text{def. of unfold} \\
&\in \mathsf{Sub}(T\{t/X\}) && \text{contr. + ind. hyp.} \\
&\subseteq \mathsf{Sub}(T)\{t/X\} \cup \mathsf{Sub}(t) && \text{key closure lemma} \\
&= \mathsf{Sub}(t) && \text{def. of Sub}
\end{aligned}
$$

□

Subtyping algorithm: termination

```
subtype(s₀, t₀)
  R := ∅
  S := {(s₀, t₀)}
  while S ≠ R do
    let (s, t) ∈ S \ R
    let s' = unfold(s)
    let t' = unfold(t)
    if s' = ![s''].S and t' = ![t''].T then
      S := S ∪ {(t'', s''), (S, T)}
    else if s' = ⊕{l_i : S_i}_{i∈I} and t' = ⊕{l_j : T_j}_{j∈J} and J ⊆ I then
      S := S ∪ {(S_j, T_j)}_{j∈J}
    else ···
    else return false
    R := R ∪ {(s, t)}
  return true
```

Invariant $\mathscr{S} \subseteq (\mathsf{Sub}(s_0) \cup \mathsf{Sub}(t_0))^2$

Subtyping algorithm: complexity

- Given two types s and t, let

$$n = \max\{\#\mathsf{Sub}(s), \#\mathsf{Sub}(t)\}$$

- The algorithm performs at most n^2 iterations

- Other operations have negligible costs
  (with suitable representation of sets/session types)

Homework

- Implement the subtyping algorithm for session types

"What I cannot create, I do not understand"

Richard Feynman

Outline

General references

- B Liskov, J Wing, **A behavioral notion of subtyping**
  TOPLAS 1994
    - subtyping as property preservation

- B Pierce, **Types and Programming Languages**
  MIT Press, 2002
    - Chapters 15–19 on subtyping
    - Chapters 20–21 on recursive types

- D Sangiorgi, **Introduction to bisimulation and coinduction**
  Cambridge 2012
    - coinductively defined behavioral equivalences

Subtyping for XML

- H Hosoya, J Vouillon, B Pierce, **Regular expression types for XML**, TOPLAS 2005
- A Frisch, G Castagna, V Benzaken, **Semantic subtyping: Dealing set-theoretically with function, union, intersection, and negation types**, JACM 2008
  - XML + arrow types + boolean combinators
- S Carpineti, C Laneve, L Padovani, **PiDuce - A Project for Experimenting Web Services Technologies**, Science of Computer Programming 2009
  - XML + Web service references

Subtyping for channel types

- B Pierce, D Sangiorgi, **Typing and Subtyping for Mobile Processes**, MSCS 1996
  - covariance/contravariance for input/output
- G Castagna, R De Nicola, D Varacca, **Semantic subtyping for the pi-calculus**, TCS 2008
  - semantic subtyping for channel types
- S Gay, **Bounded polymorphism in session types**, MSCS 2008

Outline

Well-typed programs. . .

From one of Philip Wadler's talks

"Well-typed programs can't go wrong"

Milner 1978

"Well-typed programs don't get stuck"

Harper; Felleisen and Wright 1994

"Well-typed programs can't be blamed"

Wadler and Findler 2008

Well-typed programs. . .

From one of Philip Wadler's talks

"Well-typed programs can't go wrong"

Milner 1978

"Well-typed programs don't get stuck"

Harper; Felleisen and Wright 1994

"Well-typed programs can't be blamed"

Wadler and Findler 2008

But do well-typed programs do anything good at all?

Safety and liveness properties

- Lamport, **Proving the Correctness of Multiprocess Programs**, IEEE Trans. on Software Engineering, 1977

Correctness	=	safety + liveness
Safety	=	something [bad] must not happen
Liveness	=	something [good] must happen

Safety and liveness properties

- Lamport, **Proving the Correctness of Multiprocess Programs**, IEEE Trans. on Software Engineering, 1977

Correctness	=	safety + liveness
Safety	=	something [bad] must not happen
Liveness	=	something [good] must happen

Theorem (Alpern and Schneider 1984)

Each property is the intersection of a safety property and a liveness property

Running an e-commerce site

$$\mu X.\&\left\{\begin{array}{l}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{array}\right\}$$

Client 1 **Client 2** **Client 3** **Client 4**

Running an e-commerce site

$$\mu X.\&\left\{\begin{array}{l}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{array}\right\}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`			

Running an e-commerce site

$$\mu X.\& \begin{Bmatrix} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
AddToCart			
AddToCart			

Running an e-commerce site

$$\mu X.\&\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`			
`AddToCart`			
`CheckOut`			

Running an e-commerce site

$$\mu X.\&\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`		
`AddToCart`			
`CheckOut`			

Running an e-commerce site

$$\mu X.\& \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	
`AddToCart`			
`CheckOut`			

Running an e-commerce site

$$\mu X.\& \begin{Bmatrix} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	
`AddToCart`		`AddToCart`	
`CheckOut`			

Running an e-commerce site

$$\mu X.\& \begin{Bmatrix} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	
`AddToCart`		`AddToCart`	
`CheckOut`		`AddToCart`	

Running an e-commerce site

$$\mu X.\&\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	
`AddToCart`		`AddToCart`	
`CheckOut`		`AddToCart`	
		`CheckOut`	

Running an e-commerce site

$$\mu X.\&\left\{\begin{array}{l}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{array}\right\}$$

Client 1

AddToCart
AddToCart
CheckOut

Client 2

CheckOut

Client 3

AddToCart
AddToCart
AddToCart
CheckOut

Client 4

AddToCart

Running an e-commerce site

$$\mu X.\& \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	`AddToCart`
`AddToCart`		`AddToCart`	`AddToCart`
`CheckOut`		`AddToCart`	
		`CheckOut`	

Running an e-commerce site

$$\mu X.\& \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	`AddToCart`
`AddToCart`		`AddToCart`	`AddToCart`
`CheckOut`		`AddToCart`	`AddToCart`
		`CheckOut`	

Running an e-commerce site

$$\mu X.\& \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	`AddToCart`
`AddToCart`		`AddToCart`	`AddToCart`
`CheckOut`		`AddToCart`	`AddToCart`
		`CheckOut`	`AddToCart`

Running an e-commerce site

$$\mu X . \& \begin{Bmatrix} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	`AddToCart`
`AddToCart`		`AddToCart`	`AddToCart`
`CheckOut`		`AddToCart`	`AddToCart`
		`CheckOut`	`AddToCart`
			`AddToCart`

Running an e-commerce site

$$\mu X.\& \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
`AddToCart`	`CheckOut`	`AddToCart`	`AddToCart`
`AddToCart`		`AddToCart`	`AddToCart`
`CheckOut`		`AddToCart`	`AddToCart`
		`CheckOut`	`AddToCart`
			`AddToCart`
			`AddToCart`

Running an e-commerce site

$$\mu X.\&\left\{\begin{array}{l}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{array}\right\}$$

Client 1	Client 2	Client 3	Client 4
AddToCart	CheckOut	AddToCart	AddToCart
AddToCart		AddToCart	AddToCart
CheckOut		AddToCart	AddToCart
		CheckOut	AddToCart
			AddToCart
			AddToCart
			AddToCart

Running an e-commerce site

$$\mu X.\&\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

Client 1	**Client 2**	**Client 3**	**Client 4**
AddToCart	CheckOut	AddToCart	AddToCart
AddToCart		AddToCart	AddToCart
CheckOut		AddToCart	AddToCart
		CheckOut	AddToCart
			AddToCart
			AddToCart
			AddToCart
			AddToCart
			...

What happened?

You

Client 4

$$\mu X.\& \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

$$\mu X.\oplus \left\{ \begin{array}{l} \texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end} \end{array} \right\}$$

What happened?

You **Client 4**

$$\mu X.\oplus\left\{\texttt{AddToCart} : X\right\}$$

$$\not\leqslant$$

$$\mu X.\&\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

$$\mu X.\oplus\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

What happened?

You | **Client 4**

$$\mu X.\oplus\left\{\texttt{AddToCart} : X\right\}$$

$$\leqslant$$

$$\mu X.\&\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

$$\mu X.\oplus\begin{Bmatrix}\texttt{AddToCart} : X \\ \texttt{CheckOut} : \texttt{end}\end{Bmatrix}$$

- $\leqslant$ preserves safety but not (necessarily) liveness
- can we define a **liveness-preserving** subtyping relation?

Outline

Syntax

T	$::=$		**session type**
		$\texttt{end}$	(termination)
	$\mid$	$?[t].T$	(input)
	$\mid$	$![t].T$	(output)
	$\mid$	$\&\{l_i : T_i\}_{i \in I}$	(branch)
	$\mid$	$\oplus\{l_i : T_i\}_{i \in I}$	(choice)
	$\mid$	X	(session type variable)
	$\mid$	$\mu X.T$	(recursion)

- in this part of the course: first-order session types only
- extension to higher-order session types is possible

Syntax

T	$::=$		**session type**
		`end`	(termination)
	$\mid$	$?[t].T$	(input)
	$\mid$	$![t].T$	(output)
	$\mid$	$\&\{l_i : T_i\}_{i \in I}$	(branch)
	$\mid$	$\oplus\{l_i : T_i\}_{i \in I}$	(choice)
	$\mid$	X	(session type variable)
	$\mid$	$\mu X.T$	(recursion)

- in this part of the course: first-order session types only
- extension to higher-order session types is possible

LTS for session types

Branch / external choice

$$\frac{k \in I}{\&\{l_i : T_i\}_{i \in I} \xrightarrow{?l_k} T_k}$$

Choice / internal choice

$$\frac{k \in I \qquad \#I > 1}{\oplus\{l_i : T_i\}_{i \in I} \xrightarrow{\tau} \oplus\{l_k : T_k\}} \qquad \oplus\{l : T\} \xrightarrow{!l} T$$

Recursion

$$\frac{T\{\mu X.T/X\} \xrightarrow{\ell} S}{\mu X.T \xrightarrow{\ell} S}$$

LTS for session types

Branch / external choice

$$\frac{k \in I}{\&\{l_i : T_i\}_{i \in I} \xrightarrow{?l_k} T_k}$$

Choice / internal choice

$$\frac{k \in I \qquad \#I > 1}{\oplus\{l_i : T_i\}_{i \in I} \xrightarrow{\tau} \oplus\{l_k : T_k\}} \qquad \oplus\{l : T\} \xrightarrow{!l} T$$

Recursion

$$\frac{T\{\mu X.T/X\} \xrightarrow{\ell} S}{\mu X.T \xrightarrow{\ell} S}$$

LTS for session types

Branch / external choice

$$\frac{k \in I}{\&\{l_i : T_i\}_{i \in I} \xrightarrow{?l_k} T_k}$$

Choice / internal choice

$$\frac{k \in I \qquad \#I > 1}{\oplus\{l_i : T_i\}_{i \in I} \xrightarrow{\tau} \oplus\{l_k : T_k\}} \qquad \oplus\{l : T\} \xrightarrow{!l} T$$

Recursion

$$\frac{T\{\mu X.T/X\} \xrightarrow{\ell} S}{\mu X.T \xrightarrow{\ell} S}$$

Notation for the LTS

$$\begin{array}{rcll} \alpha & ::= & & \textbf{Action} \\ & & \texttt{?a} & \text{(input / receive tag)} \\ & | & \texttt{!a} & \text{(output / send tag)} \end{array}$$

$$\begin{array}{rcll} \ell & ::= & & \textbf{Label} \\ & & \alpha & \text{(visible action)} \\ & | & \tau & \text{(invisible action)} \end{array}$$

Complement of an action

$$\overline{\texttt{?a}} \stackrel{\text{def}}{=} \texttt{!a} \qquad\qquad \overline{\texttt{!a}} \stackrel{\text{def}}{=} \texttt{?a}$$

Sessions: syntax and LTS

$$
\begin{array}{rcll}
M & ::= & & \textbf{Session} \\
 & & T & \text{(participant)} \\
 & | & M \mid M & \text{(parallel composition)}
\end{array}
$$

$$\frac{M \xrightarrow{\alpha} M' \qquad N \xrightarrow{\overline{\alpha}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

$$\frac{M \xrightarrow{\ell} M'}{M \mid N \xrightarrow{\ell} M' \mid N} \qquad \frac{N \xrightarrow{\ell} N'}{M \mid N \xrightarrow{\ell} M \mid N'}$$

Sessions: syntax and LTS

$$
\begin{array}{llll}
M & ::= & & \textbf{Session} \\
 & & T & \text{(participant)} \\
 & | & M \mid M & \text{(parallel composition)}
\end{array}
$$

$$\frac{M \xrightarrow{\alpha} M' \qquad N \xrightarrow{\overline{\alpha}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

$$\frac{M \xrightarrow{\ell} M'}{M \mid N \xrightarrow{\ell} M' \mid N} \qquad \frac{N \xrightarrow{\ell} N'}{M \mid N \xrightarrow{\ell} M \mid N'}$$

Sessions: syntax and LTS

$$
\begin{array}{rcll}
M & ::= & & \textbf{Session} \\
 & & T & \text{(participant)} \\
 & | & M \mid M & \text{(parallel composition)}
\end{array}
$$

$$\frac{M \xrightarrow{\alpha} M' \qquad N \xrightarrow{\overline{\alpha}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

$$\frac{M \xrightarrow{\ell} M'}{M \mid N \xrightarrow{\ell} M' \mid N} \qquad \frac{N \xrightarrow{\ell} N'}{M \mid N \xrightarrow{\ell} M \mid N'}$$

More notation for the LTS

$\stackrel{\tau}{\Longrightarrow}$ reflexive and transitive closure of $\stackrel{\tau}{\longrightarrow}$

$\stackrel{\alpha}{\Longrightarrow}$ $\stackrel{\tau}{\Longrightarrow}\stackrel{\alpha}{\longrightarrow}\stackrel{\tau}{\Longrightarrow}$

$\stackrel{\alpha_1\cdots\alpha_n}{\Longrightarrow}$ $\stackrel{\alpha_1}{\Longrightarrow}\ \ldots\ \stackrel{\alpha_n}{\Longrightarrow}$

φ $\alpha_1 \cdots \alpha_n$ (string of visible actions)

$\overline{\varphi}$ component-wise complement of φ

Successful session

We reserve a special tag OK (not in branches) to denote success

Definition

We say that M is successful if, for every N such that

$$M \stackrel{\tau}{\Longrightarrow} N$$

there exists N' such that

$$N \stackrel{!\mathtt{OK}}{\Longrightarrow} N'$$

Example

$$\mu Y.\&\begin{Bmatrix}\mathtt{a} : Y \\ \mathtt{b} : \oplus\{\mathtt{OK} : \mathtt{end}\}\end{Bmatrix} \quad | \quad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$

More examples

$$| \quad \oplus\{\mathtt{a} : \oplus\{\mathtt{a} : \oplus\{\mathtt{b} : \mathtt{end}\}\}\}$$

$$| \quad \mathtt{end}$$

$$\mu Y.\&\begin{Bmatrix}\mathtt{a} : Y \\ \mathtt{b} : \oplus\{\mathtt{OK} : \mathtt{end}\}\end{Bmatrix} \quad | \quad \mu X.\oplus\{\mathtt{a} : X\}$$

$$| \quad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$

$$| \quad \oplus\begin{Bmatrix}\mathtt{a} : \mu X.\oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$

More examples

$$| \quad \oplus\{\mathtt{a} : \oplus\{\mathtt{a} : \oplus\{\mathtt{b} : \mathtt{end}\}\}\}$$ ☺

$$| \quad \mathtt{end}$$

$$\mu Y.\& \begin{Bmatrix} \mathtt{a} : Y \\ \mathtt{b} : \oplus\{\mathtt{OK} : \mathtt{end}\} \end{Bmatrix} \quad | \quad \mu X.\oplus\{\mathtt{a} : X\}$$

$$| \quad \mu X.\oplus \begin{Bmatrix} \mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$$

$$| \quad \oplus \begin{Bmatrix} \mathtt{a} : \mu X.\oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$$

More examples

$$\mu Y.\&\begin{cases}\mathtt{a} : Y \\ \mathtt{b} : \oplus\{\mathtt{OK} : \mathtt{end}\}\end{cases}$$

$\mid$	$\oplus\{\mathtt{a} : \oplus\{\mathtt{a} : \oplus\{\mathtt{b} : \mathtt{end}\}\}\}$	☺
$\mid$	$\mathtt{end}$	☹
$\mid$	$\mu X.\oplus\{\mathtt{a} : X\}$	
$\mid$	$\mu X.\oplus\begin{cases}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{cases}$	
$\mid$	$\oplus\begin{cases}\mathtt{a} : \mu X.\oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{cases}$	

More examples

$$| \quad \oplus\{\mathtt{a} : \oplus\{\mathtt{a} : \oplus\{\mathtt{b} : \mathtt{end}\}\}\}$$ ☺

$$| \quad \mathtt{end}$$ ☹

$$\mu Y.\&\begin{Bmatrix} \mathtt{a} : Y \\ \mathtt{b} : \oplus\{\mathtt{OK} : \mathtt{end}\} \end{Bmatrix} \quad | \quad \mu X.\oplus\{\mathtt{a} : X\}$$ ☹

$$| \quad \mu X.\oplus\begin{Bmatrix} \mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$$

$$| \quad \oplus\begin{Bmatrix} \mathtt{a} : \mu X.\oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$$

More examples

$$| \quad \oplus\{\mathtt{a} : \oplus\{\mathtt{a} : \oplus\{\mathtt{b} : \mathtt{end}\}\}\}$$ ☺

$$| \quad \mathtt{end}$$ ☹

$$\mu Y.\& \begin{Bmatrix} \mathtt{a} : Y \\ \mathtt{b} : \oplus\{\mathtt{OK} : \mathtt{end}\} \end{Bmatrix} \quad | \quad \mu X.\oplus\{\mathtt{a} : X\}$$ ☹

$$| \quad \mu X.\oplus \begin{Bmatrix} \mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$$ ☺

$$| \quad \oplus \begin{Bmatrix} \mathtt{a} : \mu X.\oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$$

More examples

$$\mid \ \oplus\{\texttt{a} : \oplus\{\texttt{a} : \oplus\{\texttt{b} : \texttt{end}\}\}\}$$ ☺

$$\mid \ \texttt{end}$$ ☹

$$\mu Y.\&\left\{\begin{array}{l}\texttt{a} : Y \\ \texttt{b} : \oplus\{\texttt{OK} : \texttt{end}\}\end{array}\right\} \ \mid \ \mu X.\oplus\{\texttt{a} : X\}$$ ☹

$$\mid \ \mu X.\oplus\left\{\begin{array}{l}\texttt{a} : \oplus\{\texttt{a} : X\} \\ \texttt{b} : \texttt{end}\end{array}\right\}$$ ☺

$$\mid \ \oplus\left\{\begin{array}{l}\texttt{a} : \mu X.\oplus\{\texttt{a} : X\} \\ \texttt{b} : \texttt{end}\end{array}\right\}$$ ☹

Fair subtyping

Definition

$$S \leqslant_{\mathsf{F}} T \stackrel{\text{def}}{\iff} \forall M : (M \mid S \text{ successful}) \Rightarrow (M \mid T \text{ successful})$$

Fair subtyping

Definition

$$S \leqslant_{\mathsf{F}} T \overset{\text{def}}{\iff} \forall M : (M \mid S \text{ successful}) \Rightarrow (M \mid T \text{ successful})$$

By definition

- $\leqslant_{\mathsf{F}}$ preserves success (which is a liveness property)
- $\leqslant_{\mathsf{F}}$ is a pre-order

Examples

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \leqslant \qquad \leqslant_{\mathsf{F}} \qquad \oplus\{\mathtt{a} : \oplus\{\mathtt{a} : \oplus\{\mathtt{b} : \mathtt{end}\}\}\}$$

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \leqslant \qquad \not\leqslant_{\mathsf{F}} \qquad \mu X.\oplus\{\mathtt{a} : X\}$$

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \leqslant \qquad \leqslant_{\mathsf{F}} \qquad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \leqslant \qquad \not\leqslant_{\mathsf{F}} \qquad \oplus\begin{Bmatrix}\mathtt{a} : \mu X.\oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$

One more example: the gaming service

$$\mu X.\& \left\{ \begin{array}{l} \mathtt{play} : \oplus \left\{ \begin{array}{l} \mathtt{win} : X \\ \mathtt{loss} : X \end{array} \right\} \\ \mathtt{quit} : \mathtt{end} \end{array} \right\}$$

One more example: the gaming service

$$\mu X.\&\left\{\begin{array}{l}\mathtt{play}:\oplus\left\{\begin{array}{l}\mathtt{win}:X\\ \mathtt{loss}:X\end{array}\right\}\\ \mathtt{quit}:\mathtt{end}\end{array}\right\}$$

$$\mu X.\&\left\{\begin{array}{l}\mathtt{play}:\oplus\left\{\mathtt{loss}:\&\left\{\begin{array}{l}\mathtt{play}:\oplus\left\{\begin{array}{l}\mathtt{win}:X\\ \mathtt{loss}:X\end{array}\right\}\\ \mathtt{quit}:\mathtt{end}\end{array}\right\}\right\}\\ \mathtt{quit}:\mathtt{end}\end{array}\right\}$$

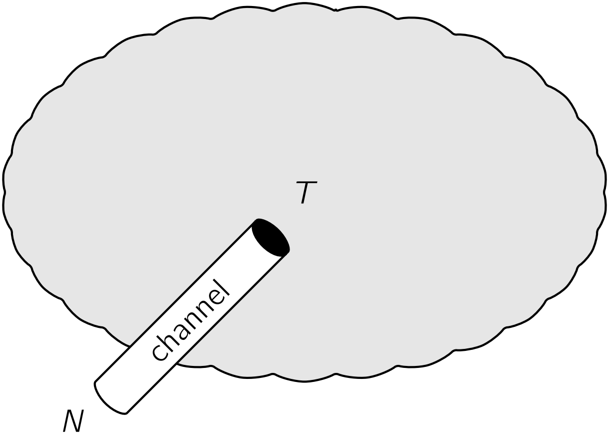
One more example: the gaming service

$$\mu X.\&\begin{Bmatrix}\mathtt{play} : \oplus\begin{Bmatrix}\mathtt{win} : X \\ \mathtt{loss} : X\end{Bmatrix} \\ \mathtt{quit} : \mathtt{end}\end{Bmatrix}$$

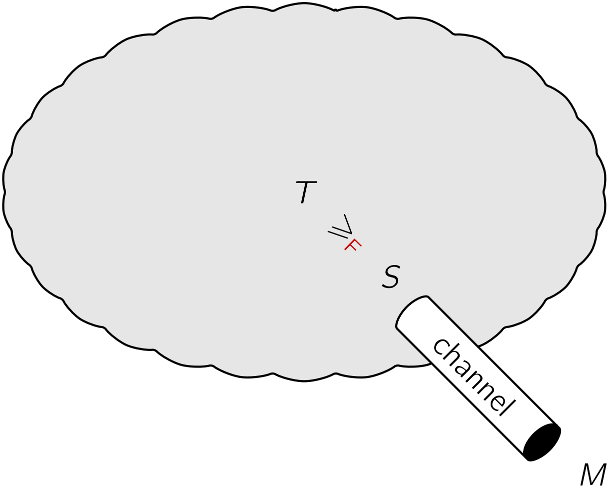
$$\overset{???}{\leqslant_{\mathsf{F}}}$$

$$\mu X.\&\begin{Bmatrix}\mathtt{play} : \oplus\left\{\mathtt{loss} : \&\begin{Bmatrix}\mathtt{play} : \oplus\begin{Bmatrix}\mathtt{win} : X \\ \mathtt{loss} : X\end{Bmatrix} \\ \mathtt{quit} : \mathtt{end}\end{Bmatrix}\right\} \\ \mathtt{quit} : \mathtt{end}\end{Bmatrix}$$

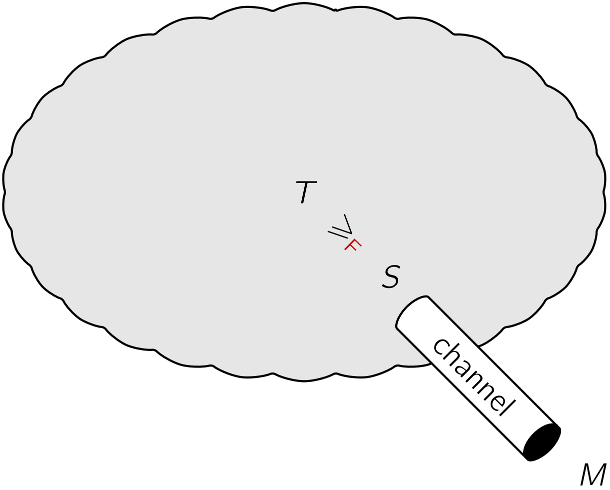
Is $\leqslant_F$ a subtyping relation?



Is $\leqslant_{\mathsf{F}}$ a subtyping relation?



Is $\leqslant_{\mathsf{F}}$ a subtyping relation? **YES**



Digression: on the definition of "success"

Definition

We say that M is successful if, for every N such that

$$M \stackrel{\tau}{\Longrightarrow} N$$

there exists N' such that

$$N \stackrel{\mathtt{!OK}}{\Longrightarrow} N'$$

Digression: on the definition of "success"

Definition

We say that M is successful if, for every N such that

$$M \stackrel{\tau}{\Longrightarrow} N$$

there exists N' such that

$$N \stackrel{\texttt{!OK}}{\Longrightarrow} N'$$

Definition (variation 1)

Same as original definition, but M has always the form $T_1 \mid T_2$

Digression: on the definition of “success”

Digression: on the definition of "success"

Definition (variation 2)

We say that M is successful if, for every N such that

$$M \stackrel{\tau}{\Longrightarrow} N \stackrel{\tau}{\not\longrightarrow}$$

we have

$$N = \texttt{end} \mid \cdots \mid \texttt{end}$$

Digression: on the definition of “success”

Definition (variation 2)

We say that M is successful if, for every N such that

$$M \stackrel{\tau}{\Longrightarrow} N \stackrel{\tau}{\not\longrightarrow}$$

we have

$$N = \texttt{end} \mid \cdots \mid \texttt{end}$$

Definition (variation 3)

We say that M is successful if, for every N such that

$$M \stackrel{\tau}{\Longrightarrow} N$$

we have

$$N \stackrel{\tau}{\Longrightarrow} \texttt{end} \mid \cdots \mid \texttt{end}$$

Homework

1. Check whether the subtyping relation induced by variation 2 coincides with $\leqslant$
2. Show that the subtyping relation induced by variation 3 has a least element

Outline

$\leqslant_{\mathsf{F}} \subseteq \leqslant$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \texttt{Int}$
- $s = \texttt{Int}$ and $t = \texttt{Real}$
- $s = t = \texttt{Real}$
- $\mathsf{unfold}(s) = ?[s'].S$ and $\mathsf{unfold}(t) = ?[t'].T$ and $s' \mathrel{\mathscr{R}} t'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = ![s'].S$ and $\mathsf{unfold}(t) = ![t'].T$ and $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I \subseteq J$ and $S_i \mathrel{\mathscr{R}} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathrel{\mathscr{R}} T_j$ for every $j \in J$

$\leqslant_{\mathsf{F}} \subseteq \leqslant$

Definition (type simulation)

We say that a relation $\mathscr{R}$ is a type simulation if $s \mathrel{\mathscr{R}} t$ implies either

- $s = t = \mathtt{Int}$
- $s = \mathtt{Int}$ and $t = \mathtt{Real}$
- $s = t = \mathtt{Real}$
- $\mathsf{unfold}(s) = ?[s'] . S$ and $\mathsf{unfold}(t) = ?[t'] . T$ and $s' \mathrel{\mathscr{R}} t'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = ![s'] . S$ and $\mathsf{unfold}(t) = ![t'] . T$ and $t' \mathrel{\mathscr{R}} s'$ and $S \mathrel{\mathscr{R}} T$
- $\mathsf{unfold}(s) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \&\{l_j : T_j\}_{j \in J}$ and $I = J$ and $S_i \mathrel{\mathscr{R}} T_i$ for every $i \in I$
- $\mathsf{unfold}(s) = \oplus\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(t) = \oplus\{l_j : T_j\}_{j \in J}$ and $J \subseteq I$ and $S_j \mathrel{\mathscr{R}} T_j$ for every $j \in J$

Proof of $\leqslant_F \subseteq \leqslant$

We must show that $\leqslant_F$ is a type simulation. Suppose $S \leqslant_F T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

$$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid S$$

is successful. From the hypothesis $S \leqslant_F T$ we deduce

$$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid T$$

is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

Exercise: deduce $I = J$.

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$

We must show that $\leqslant_{\mathsf{F}}$ is a type simulation. Suppose $S \leqslant_{\mathsf{F}} T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

$$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid S$$

is successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce

$$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid T$$

is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

Exercise: deduce $I = J$.

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$

We must show that $\leqslant_{\mathsf{F}}$ is a type simulation. Suppose $S \leqslant_{\mathsf{F}} T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

$$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid S$$

is successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce

$$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid T$$

is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

Exercise: deduce $I = J$.

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$

We must show that $\leqslant_{\mathsf{F}}$ is a type simulation. Suppose $S \leqslant_{\mathsf{F}} T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

$$\oplus\{l_k : \oplus\{\texttt{OK} : \texttt{end}\}\} \mid S$$

is successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce

$$\oplus\{l_k : \oplus\{\texttt{OK} : \texttt{end}\}\} \mid T$$

is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

Exercise: deduce $I = J$.

Proof of $\leqslant_F \subseteq \leqslant$

We must show that $\leqslant_F$ is a type simulation. Suppose $S \leqslant_F T$.

- Suppose unfold$(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

$$\oplus\{l_k : \oplus\{\texttt{OK} : \texttt{end}\}\} \mid S$$

is successful. From the hypothesis $S \leqslant_F T$ we deduce

$$\oplus\{l_k : \oplus\{\texttt{OK} : \texttt{end}\}\} \mid T$$

is also successful. This is true for any $k \in I$, hence we deduce unfold$(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

Exercise: deduce $I = J$.

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$

We must show that $\leqslant_{\mathsf{F}}$ is a type simulation. Suppose $S \leqslant_{\mathsf{F}} T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

  $$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid S$$

  is successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce

  $$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid T$$

  is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

  Exercise: deduce $I = J$.

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$

We must show that $\leqslant_{\mathsf{F}}$ is a type simulation. Suppose $S \leqslant_{\mathsf{F}} T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

  $$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid S$$

  is successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce

  $$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid T$$

  is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

  Exercise: deduce $I = J$.

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$

We must show that $\leqslant_{\mathsf{F}}$ is a type simulation. Suppose $S \leqslant_{\mathsf{F}} T$.

- Suppose $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$. Let $k \in I$. Then

  $$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid S$$

  is successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce

  $$\oplus\{l_k : \oplus\{\mathtt{OK} : \mathtt{end}\}\} \mid T$$

  is also successful. This is true for any $k \in I$, hence we deduce $\mathsf{unfold}(T) = \&\{l_i : T_i\}_{i \in J}$ with $I \subseteq J$.

  Exercise: deduce $I = J$.

Proof of $\leqslant_F \subseteq \leqslant$ (continued)

- We have $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(T) = \&\{l_j : T_j\}_{j \in J}$ with $I \subseteq J$.

  Let R be a session type such that $R \mid S_k$ is successful. Then

  $$\oplus\{l_k : R\} \mid S$$

  is also successful. From the hypothesis $S \leqslant_F T$ we deduce that

  $$\oplus\{l_k : R\} \mid T$$

  is successful too. Then

  $$\oplus\{l_k : R\} \mid T \xrightarrow{\tau} R \mid T_k$$

  must be successful. We conclude $S_k \leqslant_F T_k$.

Exercise: I cheated a little in this slide, where?

Proof of $\leqslant_F \subseteq \leqslant$ (continued)

- We have $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(T) = \&\{l_j : T_j\}_{j \in J}$ with $I \subseteq J$.

  Let R be a session type such that $R \mid S_k$ is successful. Then

  $$\oplus\{l_k : R\} \mid S$$

  is also successful. From the hypothesis $S \leqslant_F T$ we deduce that

  $$\oplus\{l_k : R\} \mid T$$

  is successful too. Then

  $$\oplus\{l_k : R\} \mid T \xrightarrow{\tau} R \mid T_k$$

  must be successful. We conclude $S_k \leqslant_F T_k$.

Exercise: I cheated a little in this slide, where?

Proof of $\leqslant_F \subseteq \leqslant$ (continued)

- We have $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$ and $\mathsf{unfold}(T) = \&\{l_j : T_j\}_{j \in J}$ with $I \subseteq J$.

  Let R be a session type such that $R \mid S_k$ is successful. Then

  $$\oplus\{l_k : R\} \mid S$$

  is also successful. From the hypothesis $S \leqslant_F T$ we deduce that

  $$\oplus\{l_k : R\} \mid T$$

  is successful too. Then

  $$\oplus\{l_k : R\} \mid T \xrightarrow{\tau} R \mid T_k$$

  must be successful. We conclude $S_k \leqslant_F T_k$.

Exercise: I cheated a little in this slide, where?

Proof of $\leqslant_F \subseteq \leqslant$ (continued)

- We have unfold$(S) = \&\{l_i : S_i\}_{i \in I}$ and unfold$(T) = \&\{l_j : T_j\}_{j \in J}$ with $I \subseteq J$.

  Let R be a session type such that $R \mid S_k$ is successful. Then

  $$\oplus\{l_k : R\} \mid S$$

  is also successful. From the hypothesis $S \leqslant_F T$ we deduce that

  $$\oplus\{l_k : R\} \mid T$$

  is successful too. Then

  $$\oplus\{l_k : R\} \mid T \xrightarrow{\tau} R \mid T_k$$

  must be successful. We conclude $S_k \leqslant_F T_k$.

Exercise: I cheated a little in this slide, where?

Proof of $\leqslant_F \subseteq \leqslant$ (continued)

- We have $\text{unfold}(S) = \&\{l_i : S_i\}_{i \in I}$ and $\text{unfold}(T) = \&\{l_j : T_j\}_{j \in J}$ with $I \subseteq J$.

  Let R be a session type such that $R \mid S_k$ is successful. Then

  $$\oplus\{l_k : R\} \mid S$$

  is also successful. From the hypothesis $S \leqslant_F T$ we deduce that

  $$\oplus\{l_k : R\} \mid T$$

  is successful too. Then

  $$\oplus\{l_k : R\} \mid T \xrightarrow{\tau} R \mid T_k$$

  must be successful. We conclude $S_k \leqslant_F T_k$.

Exercise: I cheated a little in this slide, where?

Proof of $\leqslant_{\mathsf{F}} \subseteq \leqslant$ (continued)

- We have $\mathsf{unfold}(S) = \&\{l_i : S_i\}_{i\in I}$ and $\mathsf{unfold}(T) = \&\{l_j : T_j\}_{j\in J}$ with $I \subseteq J$.

  Let R be a session type such that $R \mid S_k$ is successful. Then

  $$\oplus\{l_k : R\} \mid S$$

  is also successful. From the hypothesis $S \leqslant_{\mathsf{F}} T$ we deduce that

  $$\oplus\{l_k : R\} \mid T$$

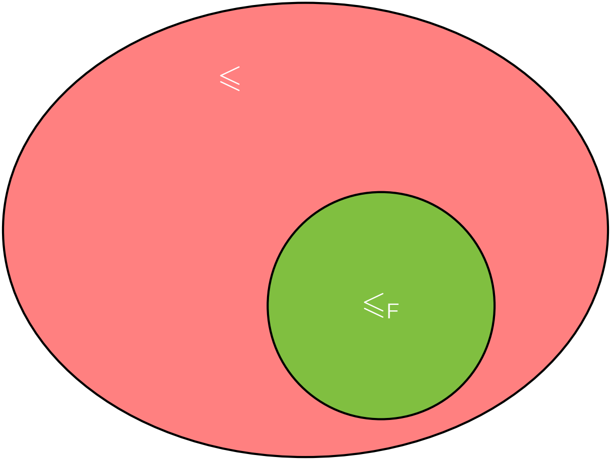
  is successful too. Then

  $$\oplus\{l_k : R\} \mid T \xrightarrow{\tau} R \mid T_k$$

  must be successful. We conclude $S_k \leqslant_{\mathsf{F}} T_k$.

Exercise: I cheated a little in this slide, where?

The situation so far



Notation: traces

$$\mathsf{tr}(T) \stackrel{\text{def}}{=} \{\varphi \mid T \stackrel{\varphi}{\Longrightarrow}\}$$

Examples

- $\mathsf{tr}(\mathtt{end}) = \{\varepsilon\}$
- $\mathsf{tr}(\oplus\{\mathtt{a} : \mathtt{end}, \mathtt{b} : \mathtt{end}\}) = \{\varepsilon, !\mathtt{a}, !\mathtt{b}\}$
- $\mathsf{tr}(\mu X.\oplus\{\mathtt{a} : X\}) = \{\varepsilon, !\mathtt{a}, !\mathtt{a}!\mathtt{a}, \ldots\} = (!\mathtt{a})^*$
- $\mathsf{tr}(\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}) = \{\varepsilon, !\mathtt{b}, !\mathtt{a}, !\mathtt{a}!\mathtt{b}, \ldots\} = (!\mathtt{a})^*(\varepsilon + !\mathtt{b})$

Notation: traces

$$\mathsf{tr}(T) \stackrel{\text{def}}{=} \{\varphi \mid T \stackrel{\varphi}{\Longrightarrow}\}$$

Examples

- $\mathsf{tr}(\texttt{end}) = \{\varepsilon\}$
- $\mathsf{tr}(\oplus\{\mathrm{a} : \texttt{end}, \mathrm{b} : \texttt{end}\}) = \{\varepsilon, !\mathrm{a}, !\mathrm{b}\}$
- $\mathsf{tr}(\mu X.\oplus\{\mathrm{a} : X\}) = \{\varepsilon, !\mathrm{a}, !\mathrm{a}!\mathrm{a}, \ldots\} = (!\mathrm{a})^*$
- $\mathsf{tr}\left(\mu X.\oplus\left\{\begin{matrix}\mathrm{a} : X \\ \mathrm{b} : \texttt{end}\end{matrix}\right\}\right) = \{\varepsilon, !\mathrm{b}, !\mathrm{a}, !\mathrm{a}!\mathrm{b}, \ldots\} = (!\mathrm{a})^*(\varepsilon + !\mathrm{b})$

Notation: traces

$$\mathsf{tr}(T) \stackrel{\text{def}}{=} \{\varphi \mid T \stackrel{\varphi}{\Longrightarrow}\}$$

Examples

- $\mathsf{tr}(\mathtt{end}) = \{\varepsilon\}$
- $\mathsf{tr}(\oplus\{\mathtt{a} : \mathtt{end}, \mathtt{b} : \mathtt{end}\}) = \{\varepsilon, !\mathtt{a}, !\mathtt{b}\}$
- $\mathsf{tr}(\mu X.\oplus\{\mathtt{a} : X\}) = \{\varepsilon, !\mathtt{a}, !\mathtt{a}!\mathtt{a}, \ldots\} = (!\mathtt{a})^*$
- $\mathsf{tr}(\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}) = \{\varepsilon, !\mathtt{b}, !\mathtt{a}, !\mathtt{a}!\mathtt{b}, \ldots\} = (!\mathtt{a})^*(\varepsilon + !\mathtt{b})$

Notation: traces

$$\mathsf{tr}(T) \stackrel{\text{def}}{=} \{\varphi \mid T \stackrel{\varphi}{\Longrightarrow}\}$$

Examples

- $\mathsf{tr}(\mathtt{end}) = \{\varepsilon\}$
- $\mathsf{tr}(\oplus\{\mathtt{a} : \mathtt{end}, \mathtt{b} : \mathtt{end}\}) = \{\varepsilon, \mathtt{!a}, \mathtt{!b}\}$
- $\mathsf{tr}(\mu X.\oplus\{\mathtt{a} : X\}) = \{\varepsilon, \mathtt{!a}, \mathtt{!a!a}, \dots\} = (\mathtt{!a})^*$
- $\mathsf{tr}(\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}) = \{\varepsilon, \mathtt{!b}, \mathtt{!a}, \mathtt{!a!b}, \dots\} = (\mathtt{!a})^*(\varepsilon + \mathtt{!b})$

Notation: traces

$$\mathsf{tr}(T) \stackrel{\text{def}}{=} \{\varphi \mid T \stackrel{\varphi}{\Longrightarrow}\}$$

Examples

- $\mathsf{tr}(\mathtt{end}) = \{\varepsilon\}$
- $\mathsf{tr}(\oplus\{\mathtt{a} : \mathtt{end}, \mathtt{b} : \mathtt{end}\}) = \{\varepsilon, \mathtt{!a}, \mathtt{!b}\}$
- $\mathsf{tr}(\mu X.\oplus\{\mathtt{a} : X\}) = \{\varepsilon, \mathtt{!a}, \mathtt{!a!a}, \dots\} = (\mathtt{!a})^*$
- $\mathsf{tr}(\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}) = \{\varepsilon, \mathtt{!b}, \mathtt{!a}, \mathtt{!a!b}, \dots\} = (\mathtt{!a})^*(\varepsilon + \mathtt{!b})$

Notation: type continuation

Definition (continuation)

Let $T \stackrel{\alpha}{\Longrightarrow}$. The continuation of T after α is *the* session type S such that $T \stackrel{\tau}{\Longrightarrow}\stackrel{\alpha}{\longrightarrow} S$. We generalize continuations to *strings* of actions, thus

$$\begin{aligned} T(\varepsilon) &\stackrel{\text{def}}{=} T \\ T(\alpha\varphi) &\stackrel{\text{def}}{=} T(\alpha)(\varphi) \end{aligned}$$

Example

Let $T \stackrel{\text{def}}{=} \mu X.\oplus \begin{Bmatrix} \mathtt{a} : X \\ \mathtt{b} : \mathtt{end} \end{Bmatrix}$. Then

$$T(!\mathtt{b}) = T(!\mathtt{a}!\mathtt{b}) = \mathtt{end} \qquad \text{and} \qquad T(!\mathtt{a}) = T$$

Trace convergence

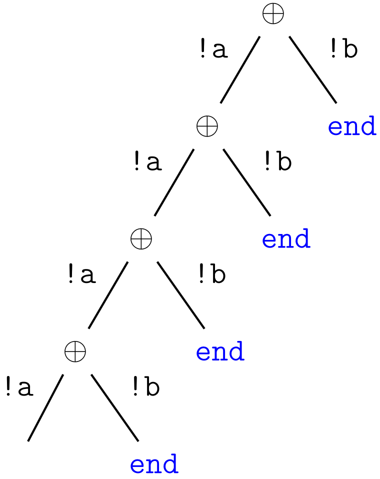
$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \mathtt{a} : S(\psi\,!\,\mathtt{a}) \sqsubseteq T(\psi\,!\,\mathtt{a})}{S \sqsubseteq T}$$

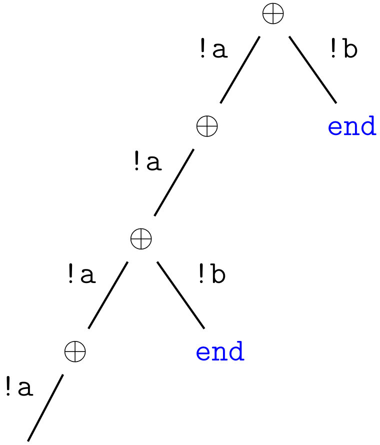
Notes

- $\sqsubseteq$ is inductively defined (least relation such that...)
- the base case is when $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$
- there is always a finite number of premises

Trace convergence: example

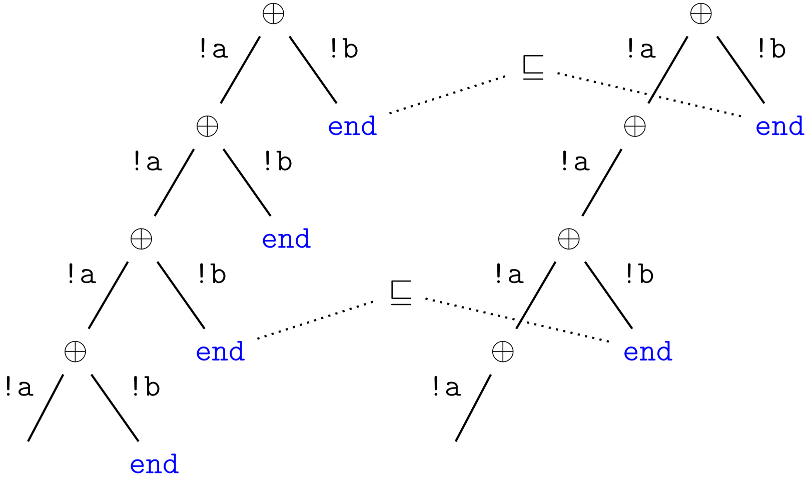
$$\mu X.\oplus\left\{\begin{array}{l}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{array}\right\} \qquad \mu X.\oplus\left\{\begin{array}{l}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{array}\right\}$$





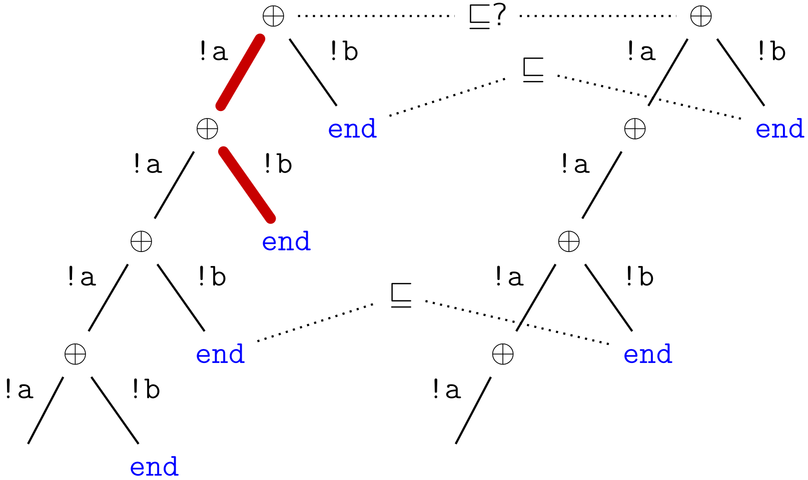
Trace convergence: example

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$



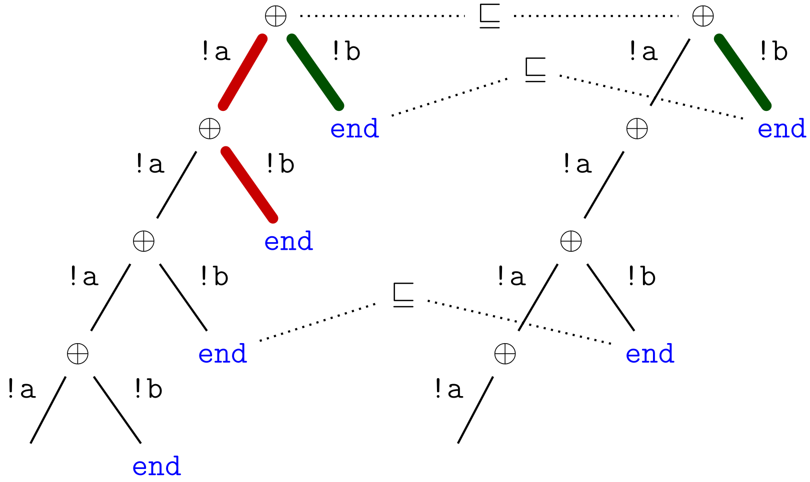
Trace convergence: example

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$



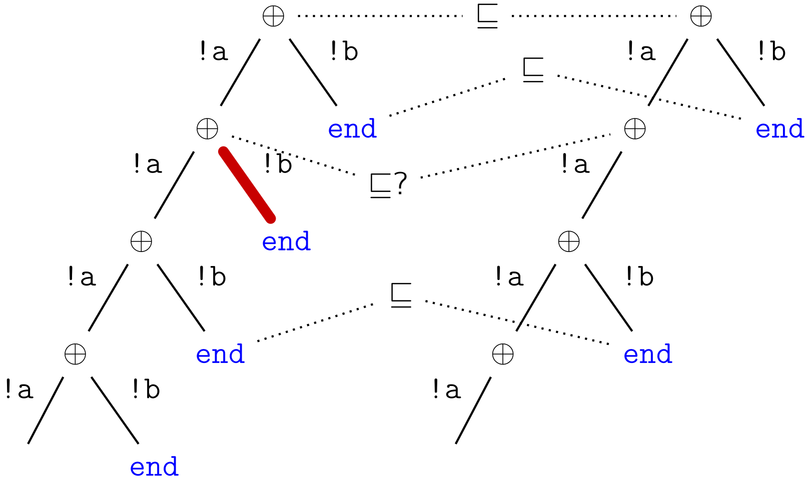
Trace convergence: example

$$\mu X.\oplus\begin{Bmatrix}\texttt{a} : X \\ \texttt{b} : \texttt{end}\end{Bmatrix} \qquad \mu X.\oplus\begin{Bmatrix}\texttt{a} : \oplus\{\texttt{a} : X\} \\ \texttt{b} : \texttt{end}\end{Bmatrix}$$



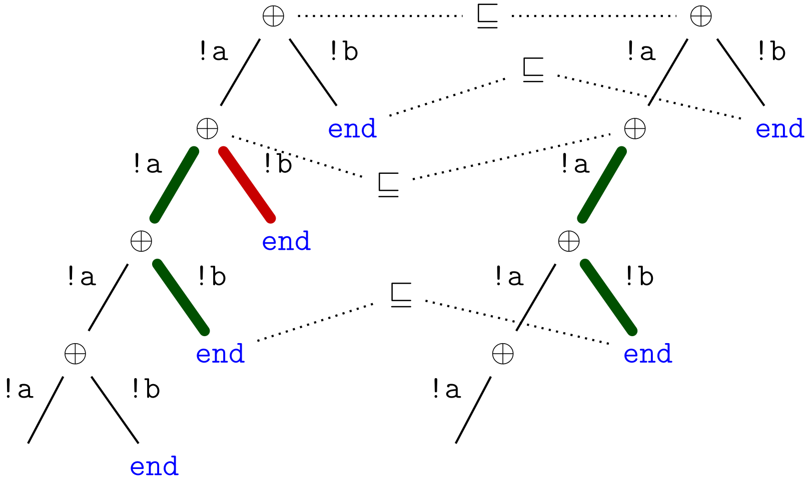
Trace convergence: example

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$



Trace convergence: example

$$\mu X.\oplus\begin{Bmatrix}\mathtt{a} : X \\ \mathtt{b} : \mathtt{end}\end{Bmatrix} \qquad \mu X.\oplus\begin{Bmatrix}\mathtt{a} : \oplus\{\mathtt{a} : X\} \\ \mathtt{b} : \mathtt{end}\end{Bmatrix}$$



Characterization of fair subtyping

Definition (fair type simulation)

We say that a type simulation $\mathscr{R}$ is fair if $\mathscr{R} \subseteq \sqsubseteq$.

Theorem

$\leqslant_{\mathsf{F}}$ is the largest fair type simulation.

Proof plan.

Let $\preccurlyeq$ denote the largest fair type simulation. We prove the inclusions $\preccurlyeq \subseteq \leqslant_{\mathsf{F}}$ and $\leqslant_{\mathsf{F}} \subseteq \preccurlyeq$. □

$\preccurlyeq \subseteq \leqslant_{F}$

Lemma

$S \preccurlyeq T$ implies $S \leqslant_{F} T$

Proof plan.

We must prove that

$$M \mid S \qquad \text{successful}$$

implies

$$M \mid T \qquad \text{successful}$$

In particular, we must show that

$$M \mid T \xrightarrow{!\mathtt{OK}}$$

□

$\preccurlyeq \subseteq \leqslant_{\mathsf{F}}$

Lemma

$S \preccurlyeq T$ implies $S \leqslant_{\mathsf{F}} T$

Proof plan.

We must prove that

$$M \mid S \qquad \text{successful}$$

implies

$$M \mid T \qquad \text{successful}$$

In particular, we must show that

$$M \mid T \stackrel{!\mathtt{OK}}{\Longrightarrow}$$

□

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \texttt{a} : S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and $\texttt{a}$ such that $S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\texttt{!a})$ for some N such that $N \mid S(\psi\texttt{!a})$ is successful. We conclude by induction hypothesis.

□

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \texttt{a} : S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and a such that $S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\texttt{!a})$ for some N such that $N \mid S(\psi\texttt{!a})$ is successful. We conclude by induction hypothesis.

□

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \mathtt{a} : S(\psi\,\mathtt{!a}) \sqsubseteq T(\psi\,\mathtt{!a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and $\mathtt{a}$ such that $S(\psi\,\mathtt{!a}) \sqsubseteq T(\psi\,\mathtt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\,\mathtt{!a})$ for some N such that $N \mid S(\psi\,\mathtt{!a})$ is successful. We conclude by induction hypothesis.

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \mathtt{a} : S(\psi\,\texttt{!}\,\mathtt{a}) \sqsubseteq T(\psi\,\texttt{!}\,\mathtt{a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and $\mathtt{a}$ such that $S(\psi\,\texttt{!}\,\mathtt{a}) \sqsubseteq T(\psi\,\texttt{!}\,\mathtt{a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\,\texttt{!}\,\mathtt{a})$ for some N such that $N \mid S(\psi\,\texttt{!}\,\mathtt{a})$ is successful. We conclude by induction hypothesis. □

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \texttt{a} : S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and a such that $S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\texttt{!a})$ for some N such that $N \mid S(\psi\texttt{!a})$ is successful. We conclude by induction hypothesis.

$\square$

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \texttt{a} : S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and $\texttt{a}$ such that $S(\psi\texttt{!a}) \sqsubseteq T(\psi\texttt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\texttt{!a})$ for some N such that $N \mid S(\psi\texttt{!a})$ is successful. We conclude by induction hypothesis. □

Lemma (key convergence lemma)

If $S \sqsubseteq T$ *and* $M \mid S$ *is successful, then* $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \mathtt{a} : S(\psi\,\mathtt{!a}) \sqsubseteq T(\psi\,\mathtt{!a})}{S \sqsubseteq T}$$

Proof.

By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and $\mathtt{a}$ such that $S(\psi\,\mathtt{!a}) \sqsubseteq T(\psi\,\mathtt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\,\mathtt{!a})$ for some N such that $N \mid S(\psi\,\mathtt{!a})$ is successful. We conclude by induction hypothesis. □

Lemma (key convergence lemma)

If $S \sqsubseteq T$ and $M \mid S$ is successful, then $M \mid T \stackrel{\texttt{!OK}}{\Longrightarrow}$.

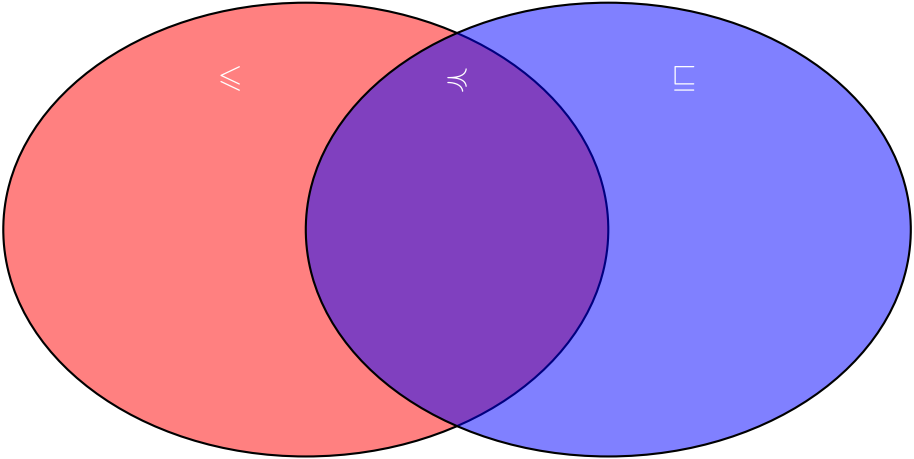
$$\frac{\forall \varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T) : \exists \psi < \varphi, \mathtt{a} : S(\psi\,\mathtt{!a}) \sqsubseteq T(\psi\,\mathtt{!a})}{S \sqsubseteq T}$$

Proof.

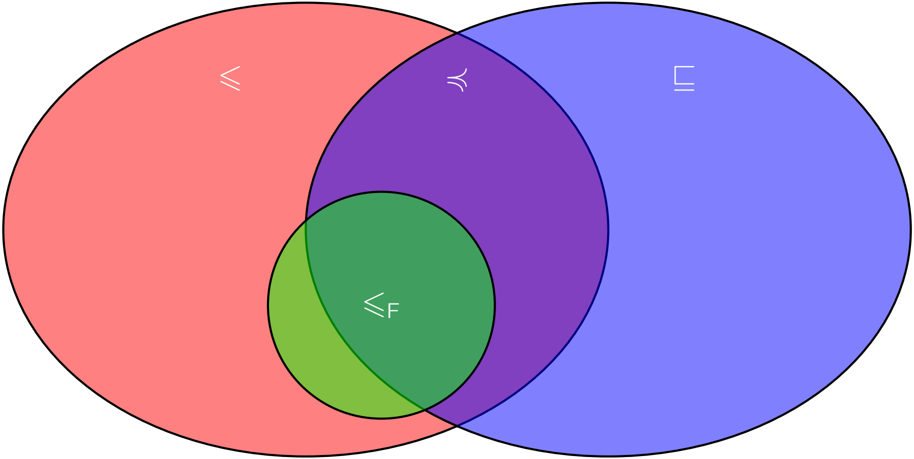
By induction on the size of the derivation of $S \sqsubseteq T$. From the hypothesis $M \mid S$ successful we have $M \stackrel{\varphi\texttt{!OK}}{\Longrightarrow}$ and $S \stackrel{\overline{\varphi}}{\Longrightarrow}$ for some φ of *minimum* length.

- (base case) $\mathsf{tr}(S) \subseteq \mathsf{tr}(T)$. We conclude $T \stackrel{\overline{\varphi}}{\Longrightarrow}$.
- (ind. case) Suppose $\varphi \in \mathsf{tr}(S) \setminus \mathsf{tr}(T)$ (otherwise it's easy). By def. of $\sqsubseteq$ there exist $\psi < \varphi$ and $\mathtt{a}$ such that $S(\psi\,\mathtt{!a}) \sqsubseteq T(\psi\,\mathtt{!a})$. Since $M \mid S$ is successful and φ has minimum length, we have $M \mid S \stackrel{\tau}{\Longrightarrow} N \mid S(\psi\,\mathtt{!a})$ for some N such that $N \mid S(\psi\,\mathtt{!a})$ is successful. We conclude by induction hypothesis. □

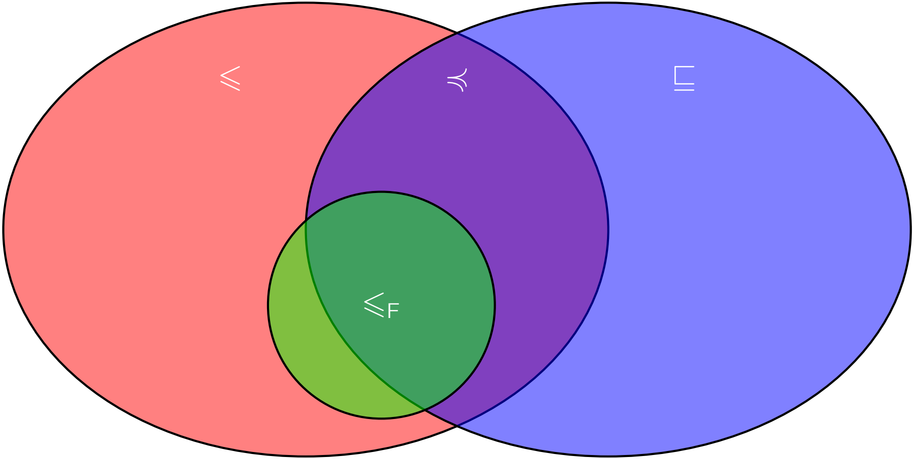
$\leqslant_{\mathsf{F}} \subseteq \preccurlyeq$



$\leqslant_{\mathsf{F}} \subseteq \preccurlyeq$



$\leqslant_{\mathsf{F}} \subseteq \preccurlyeq$



$$S \leqslant T \quad \text{and} \quad S \not\sqsubseteq T \quad \text{implies} \quad S \not\leqslant_{\mathsf{F}} T$$

Proof plan

Showing $S \not\leqslant_{\mathsf{F}} T$ amounts at finding some R such that

$$R \mid S \quad \text{is successful}$$

and

$$R \mid T \quad \text{is not}$$

under the conditions

$$S \leqslant T \quad \text{and} \quad S \not\sqsubseteq T$$

The discriminator

$$\mathscr{M}(T, S) \simeq \begin{cases} \oplus\{\mathtt{a}_i : \mathscr{M}(S_i, T_i)\}_{i \in I, S_i \not\sqsubseteq T_i} & \text{if } S \simeq \&\{\mathtt{a}_i : S_i\}_{i \in I} \\ & \quad T \simeq \&\{\mathtt{a}_j : T_j\}_{j \in J} \\ & \quad I \subseteq J \\ \&\{\mathtt{a}_j : \mathscr{M}(S_j, T_j)\}_{j \in J} & \text{if } S \simeq \oplus\{\mathtt{a}_i : S_i\}_{i \in I} \\ \quad \{\mathtt{a}_i : \oplus\{\mathtt{OK} : \mathtt{end}\}\}_{i \in I \setminus J} & \quad T \simeq \oplus\{\mathtt{a}_j : T_j\}_{j \in J} \\ & \quad J \subseteq I \end{cases}$$

- $\mathscr{M}(S, T)$ is well defined (properties of $\leqslant$ and $\not\sqsubseteq$)
- $\mathscr{M}(S, T) \mid S$ is successful (tedius)
- $\mathscr{M}(S, T) \mid T$ is unsuccessful (easy)

The discriminator

$S \not\sqsubseteq T$ implies $S_i \not\sqsubseteq T_i$ for some $i \in I$

$S \not\sqsubseteq T$ implies $S_j \not\sqsubseteq T_j$ for every $j \in J$

$$\mathscr{M}(T, S) = \begin{cases} \oplus\{\mathtt{a}_i : \mathscr{M}(S_i, T_i)\}_{i \in I, S_i \not\sqsubseteq T_i} & \text{if } S \simeq \&\{\mathtt{a}_i : S_i\}_{i \in I} \\ & \quad T \simeq \&\{\mathtt{a}_j : T_j\}_{j \in J} \\ & \quad I \subseteq J \\ \&\{\mathtt{a}_j : \mathscr{M}(S_j, T_j)\}_{j \in J} & \text{if } S \simeq \oplus\{\mathtt{a}_i : S_i\}_{i \in I} \\ \quad \{\mathtt{a}_i : \oplus\{\mathtt{OK} : \mathtt{end}\}\}_{i \in I \setminus J} & \quad T \simeq \oplus\{\mathtt{a}_j : T_j\}_{j \in J} \\ & \quad J \subseteq I \end{cases}$$

- $\mathscr{M}(S, T)$ is well defined (properties of $\leqslant$ and $\not\sqsubseteq$)
- $\mathscr{M}(S, T) \mid S$ is successful (tedius)
- $\mathscr{M}(S, T) \mid T$ is unsuccessful (easy)

The discriminator

$$\mathscr{M}(T, S) \simeq \begin{cases} \oplus\{\mathtt{a}_i : \mathscr{M}(S_i, T_i)\}_{i \in I, S_i \not\sqsubseteq T_i} & \text{if } S \simeq \&\{\mathtt{a}_i : S_i\}_{i \in I} \\ & \quad T \simeq \&\{\mathtt{a}_j : T_j\}_{j \in J} \\ & \quad I \subseteq J \\ \\ \&\{\mathtt{a}_j : \mathscr{M}(S_j, T_j)\}_{j \in J} & \text{if } S \simeq \oplus\{\mathtt{a}_i : S_i\}_{i \in I} \\ \quad \{\mathtt{a}_i : \oplus\{\mathtt{OK} : \mathtt{end}\}\}_{i \in I \setminus J} & \quad T \simeq \oplus\{\mathtt{a}_j : T_j\}_{j \in J} \\ & \quad J \subseteq I \end{cases}$$

- $\mathscr{M}(S, T)$ is well defined (properties of $\leqslant$ and $\not\sqsubseteq$)
- $\mathscr{M}(S, T) \mid S$ is successful (tedius)
- $\mathscr{M}(S, T) \mid T$ is unsuccessful (easy)

The discriminator

$$
\mathscr{M}(T, S) \simeq \begin{cases}
\oplus\{\mathtt{a}_i : \mathscr{M}(S_i, T_i)\}_{i \in I, S_i \not\sqsubseteq T_i} & \text{if } S \simeq \&\{\mathtt{a}_i : S_i\}_{i \in I} \\
& \quad T \simeq \&\{\mathtt{a}_j : T_j\}_{j \in J} \\
& \quad I \subseteq J \\
\&\{\mathtt{a}_j : \mathscr{M}(S_j, T_j)\}_{j \in J} & \text{if } S \simeq \oplus\{\mathtt{a}_i : S_i\}_{i \in I} \\
\quad \{\mathtt{a}_i : \oplus\{\mathtt{OK} : \mathtt{end}\}\}_{i \in I \setminus J} & \quad T \simeq \oplus\{\mathtt{a}_j : T_j\}_{j \in J} \\
& \quad I \subseteq J
\end{cases}
$$

receive tag $\mathtt{a}_i$ from S that cannot be sent by T

- $\mathscr{M}(S, T)$ is well defined (properties of $\leqslant$ and $\not\sqsubseteq$)
- $\mathscr{M}(S, T) \mid S$ is successful (tedius)
- $\mathscr{M}(S, T) \mid T$ is unsuccessful (easy)

Exercises

1. Show that $S \leqslant T$ implies $S \leqslant_{\mathsf{F}} T$ for all finite S, T

2. Find S, T such that $S \sqsubseteq T$ but $S \not\leqslant T$

3. Find S, T such that $S \sqsubseteq T$ but there exists $\varphi \in \mathsf{tr}(S) \cap \mathsf{tr}(T)$ such that $S(\varphi) \not\sqsubseteq T(\varphi)$

Conjecture (read: homework)

Definition

We say that M is successful if, for every N such that

$$M \stackrel{!\mathtt{REQ}}{\Longrightarrow} N$$

there exists N' such that

$$N \stackrel{!\mathtt{RESP}}{\Longrightarrow} N'$$

- See whether the subtyping relation induced by `REQ-RESP` success coincides with $\leqslant_{\mathsf{F}}$

Outline

1. Basic notions
   - Motivation
   - Informal review of subtyping
   - Subtyping for finite session types
2. Recursive session types
   - Subtyping for recursive session types
   - Subtyping algorithm
   - Further reading
3. Fair subtyping
   - Motivation
   - A liveness-preserving subtyping
   - Characterizing fair subtyping
   - Two issues
   - Further reading

Issue 1: higher-order session types

T	$::=$		**session type**
		$\texttt{end}$	(termination)
	$\mid$	$\texttt{?}[t]\texttt{.}T$	(input)
	$\mid$	$\texttt{!}[t]\texttt{.}T$	(output)
	$\mid$	$\&\{l_i : T_i\}_{i \in I}$	(branch)
	$\mid$	$\oplus\{l_i : T_i\}_{i \in I}$	(choice)
	$\mid$	X	(session type variable)
	$\mid$	$\mu X.T$	(recursion)

LTS for higher-order session types

From tags. . .

$$\frac{M \xrightarrow{\texttt{!a}} M' \qquad N \xrightarrow{\texttt{?a}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

LTS for higher-order session types

From tags. . .

$$\frac{M \xrightarrow{\texttt{!a}} M' \qquad N \xrightarrow{\texttt{?a}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

. . . to types

$$\frac{M \xrightarrow{!s} M' \qquad N \xrightarrow{?t} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

LTS for higher-order session types

From tags. . .

$$\frac{M \xrightarrow{\texttt{!a}} M' \qquad N \xrightarrow{\texttt{?a}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

. . . to types

$$\frac{M \xrightarrow{!s} M' \qquad N \xrightarrow{?t} N}{M \mid N \xrightarrow{\tau} M' \mid N'} \quad s \leqslant_{\mathsf{F}} t$$

LTS for higher-order session types

From tags...

$$\frac{M \xrightarrow{\texttt{!a}} M' \qquad N \xrightarrow{\texttt{?a}} N}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

...to types

$$\frac{M \xrightarrow{!s} M' \qquad N \xrightarrow{?t} N}{M \mid N \xrightarrow{\tau} M' \mid N'} \quad s \leqslant_{\mathsf{F}} t$$

Problem

- the LTS is used for defining $\leqslant_{\mathsf{F}}$
- $\leqslant_{\mathsf{F}}$ is used for defining the LTS

Parametric LTS and derived notions

$$\frac{M \xrightarrow{!s}_{\mathscr{S}} M' \qquad N \xrightarrow{?t}_{\mathscr{S}} N}{M \mid N \xrightarrow{\tau}_{\mathscr{S}} M' \mid N'} \quad s \,\mathscr{S}\, t$$

- $\mathscr{S}$-successful session
- $\mathscr{S}$-success induces an $\mathscr{S}$-subtyping $\mathbf{F}(\mathscr{S})$
- show that $\mathbf{F}$ has a largest fixpoint
  - **Alert!** When $\mathscr{S} \subseteq \mathscr{R}$, the number of $\mathscr{R}$-successful sessions is larger than the number of $\mathscr{S}$-successful sessions, so in principle $\mathbf{F}(\mathscr{R})$ could be smaller than or unrelated to $\mathbf{F}(\mathscr{S})$
- define $\leqslant_{\mathsf{F}}$ as the largest fixpoint of $\mathbf{F}$

Parametric LTS and derived notions

$$\frac{M \xrightarrow{!s}_{\mathscr{S}} M' \qquad N \xrightarrow{?t}_{\mathscr{S}} N}{M \mid N \xrightarrow{\tau}_{\mathscr{S}} M' \mid N'} \quad s \mathrel{\mathscr{S}} t$$

- $\mathscr{S}$-successful session
- $\mathscr{S}$-success induces an $\mathscr{S}$-subtyping $\mathbf{F}(\mathscr{S})$
- show that $\mathbf{F}$ has a largest fixpoint
  - **Alert!** When $\mathscr{S} \subseteq \mathscr{R}$, the number of $\mathscr{R}$-successful sessions is larger than the number of $\mathscr{S}$-successful sessions, so in principle $\mathbf{F}(\mathscr{R})$ could be smaller than or unrelated to $\mathbf{F}(\mathscr{S})$
- define $\leqslant_{\mathsf{F}}$ as the largest fixpoint of $\mathbf{F}$

Parametric LTS and derived notions

$$\frac{M \xrightarrow{!s}_{\mathscr{S}} M' \qquad N \xrightarrow{?t}_{\mathscr{S}} N}{M \mid N \xrightarrow{\tau}_{\mathscr{S}} M' \mid N'} \quad s \mathrel{\mathscr{S}} t$$

- $\mathscr{S}$-successful session
- $\mathscr{S}$-success induces an $\mathscr{S}$-subtyping $\mathbf{F}(\mathscr{S})$
- show that $\mathbf{F}$ has a largest fixpoint
  - **Alert!** When $\mathscr{S} \subseteq \mathscr{R}$, the number of $\mathscr{R}$-successful sessions is larger than the number of $\mathscr{S}$-successful sessions, so in principle $\mathbf{F}(\mathscr{R})$ could be smaller than or unrelated to $\mathbf{F}(\mathscr{S})$
- define $\leqslant_{\mathsf{F}}$ as the largest fixpoint of $\mathbf{F}$

Parametric LTS and derived notions

$$\frac{M \xrightarrow{!s}_{\mathscr{S}} M' \qquad N \xrightarrow{?t}_{\mathscr{S}} N}{M \mid N \xrightarrow{\tau}_{\mathscr{S}} M' \mid N'} \quad s \mathrel{\mathscr{S}} t$$

- $\mathscr{S}$-successful session
- $\mathscr{S}$-success induces an $\mathscr{S}$-subtyping $\mathbf{F}(\mathscr{S})$
- show that $\mathbf{F}$ has a largest fixpoint
  - **Alert!** When $\mathscr{S} \subseteq \mathscr{R}$, the number of $\mathscr{R}$-successful sessions is larger than the number of $\mathscr{S}$-successful sessions, so in principle $\mathbf{F}(\mathscr{R})$ could be smaller than or unrelated to $\mathbf{F}(\mathscr{S})$
- define $\leqslant_{\mathbf{F}}$ as the largest fixpoint of $\mathbf{F}$

Parametric LTS and derived notions

$$\frac{M \xrightarrow{!s}_{\mathscr{S}} M' \qquad N \xrightarrow{?t}_{\mathscr{S}} N}{M \mid N \xrightarrow{\tau}_{\mathscr{S}} M' \mid N'} \quad s \mathrel{\mathscr{S}} t$$

- $\mathscr{S}$-successful session
- $\mathscr{S}$-success induces an $\mathscr{S}$-subtyping $\mathbf{F}(\mathscr{S})$
- show that $\mathbf{F}$ has a largest fixpoint
  - **Alert!** When $\mathscr{S} \subseteq \mathscr{R}$, the number of $\mathscr{R}$-successful sessions is larger than the number of $\mathscr{S}$-successful sessions, so in principle $\mathbf{F}(\mathscr{R})$ could be smaller than or unrelated to $\mathbf{F}(\mathscr{S})$
- define $\leqslant_{\mathsf{F}}$ as the largest fixpoint of $\mathbf{F}$

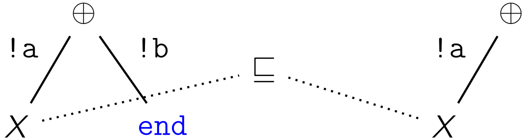
Issue 2: pre-congruence properties

$$S \leqslant_{\mathsf{F}} T \qquad \stackrel{?}{\Rightarrow} \qquad \mathscr{C}[S] \leqslant_{\mathsf{F}} \mathscr{C}[T]$$

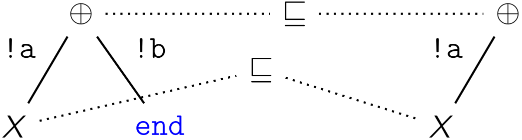
- makes sense only if the hole in $\mathscr{C}$ is in covariant position (which is *always* the case for first-order session types)
- trivially satisfied if S and T are closed
- what about *open* session types?

The naive extension is (obviously) unsound

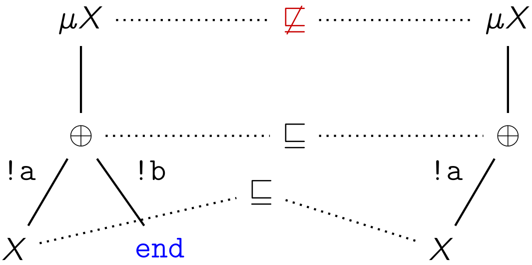
The naive extension is (obviously) unsound



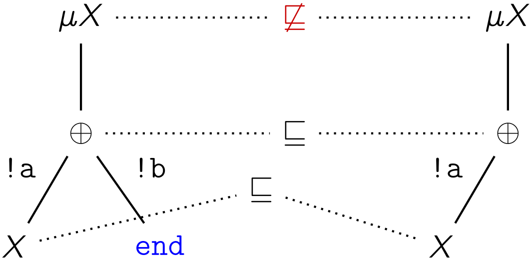
The naive extension is (obviously) unsound



The naive extension is (obviously) unsound



The naive extension is (obviously) unsound



Convergence is context-dependent

$$\frac{X \in U}{X \sqsubseteq_U X}$$

Axioms for (iso-recursive) fair subtyping

complete axiomatization

[f-end]

$$\texttt{end} \leqslant_{\mathsf{F}} \texttt{end}$$

[f-var]

$$X \leqslant_{\mathsf{F}} X$$

[f-rec]

$$\frac{T \leqslant_{\mathsf{F}} S \qquad T \sqsubseteq_{\{X\}} S}{\mu X.T \leqslant_{\mathsf{F}} \mu X.S}$$

[f-branch]

$$\frac{\forall i \in I : S_i \leqslant_{\mathsf{F}} T_i}{\&\{\mathtt{a}_i : S_i\}_{i \in I} \leqslant_{\mathsf{F}} \&\{\mathtt{a}_i : T_i\}_{i \in I}}$$

[f-choice]

$$\frac{\forall i \in I : T_i \leqslant_{\mathsf{F}} S_i}{\oplus\{\mathtt{a}_i : S_i\}_{i \in I} \leqslant_{\mathsf{F}} \oplus\{\mathtt{a}_j : T_j\}_{j \in I \cup J}}$$

Axioms for (iso-recursive) fair subtyping

complete axiomatization

[f-end]

$$\texttt{end} \leqslant_{\mathsf{F}} \texttt{end}$$

[f-var]

$$X \leqslant_{\mathsf{F}} X$$

[f-rec]

$$\frac{T \leqslant_{\mathsf{F}} S \qquad T \sqsubseteq_{\{X\}} S}{\mu X.T \leqslant_{\mathsf{F}} \mu X.S}$$

[f-branch]

$$\frac{\forall i \in I : S_i \leqslant_{\mathsf{F}} T_i}{\&\{\mathtt{a}_i : S_i\}_{i \in I} \leqslant_{\mathsf{F}} \&\{\mathtt{a}_i : T_i\}_{i \in I}}$$

[f-choice]

$$\frac{\forall i \in I : T_i \leqslant_{\mathsf{F}} S_i}{\oplus\{\mathtt{a}_i : S_i\}_{i \in I} \leqslant_{\mathsf{F}} \oplus\{\mathtt{a}_j : T_j\}_{j \in I \cup J}}$$

Proposition

There is a complete algorithm for fair subtyping that runs in $O(n^4)$

Outline

Testing equivalences

- R De Nicola and M Hennessy, **Testing equivalences for processes**, TCS 1984
- M Hennessy, **Algebraic Theory of Processes**
  MIT Press 1988
  - testing equivalences for CCS processes
- D Sangiorgi, **Introduction to bisimulation and coinduction**
  Cambridge 2012
  - survey on testing equivalences

Fair testing and liveness-preserving refinements

- V Natarajan and R Cleaveland, **Divergence and fair testing**
  ICALP 1995
  - pre-congruence issues not investigated

- A Rensink and W Vogler, **Fair testing**
  Information and Computation 2007
  - more general process algebra
  - trace equivalence
  - no trace/axiomatic characterization
  - exponential decision algorithm

Models of higher-order session types

- L Padovani, **Fair Subtyping for Multi-Party Session Types**, MSCS (to appear, but on my homepage)

- G Bernardi and M Hennessy, **Using higher-order contracts to model session types**, CONCUR 2014 (to appear, but on arXiv)

Type systems for liveness properties

- N Kobayashi, **A type system for lock-free processes**, Information and Computation 2002
- S Debois, T Hildebrandt, T Slaats, N Yoshida, **Type Checking Liveness for Collaborative Processes with Bounded and Unbounded Recursion**, FORTE 2014
- L Padovani, **Deadlock and lock freedom in the linear π-calculus**, CSL-LICS 2014

  (later this week)
- . . .