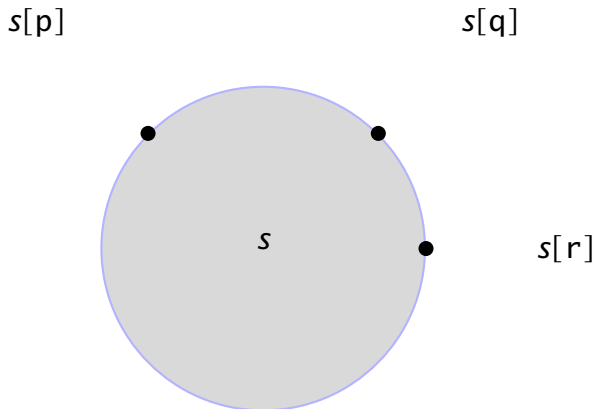


# Fair Subtyping for Multi-Party Session Types

Luca Padovani

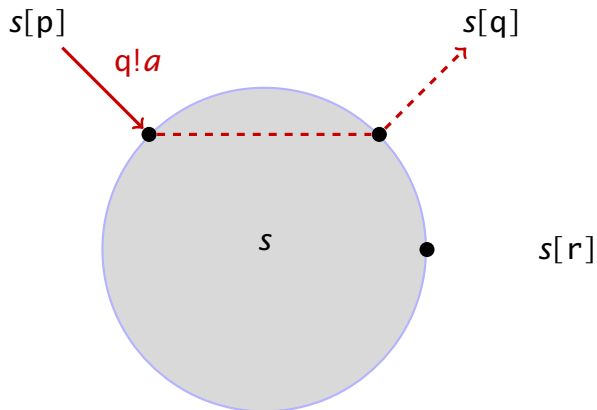
Dipartimento di Informatica, Università di Torino

# Sessions and session types



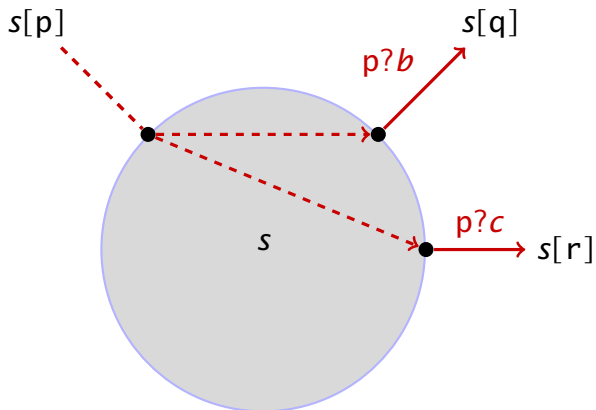
- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$

# Sessions and session types



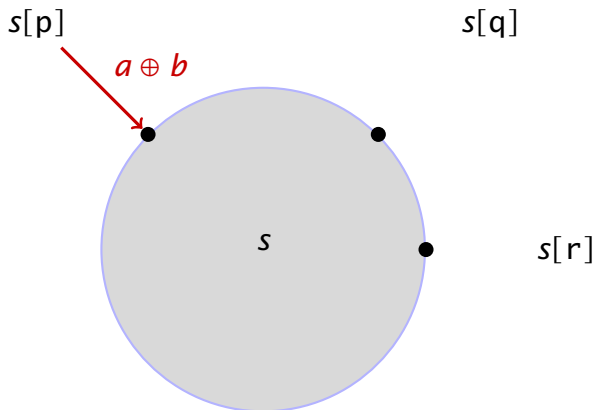
- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$

# Sessions and session types



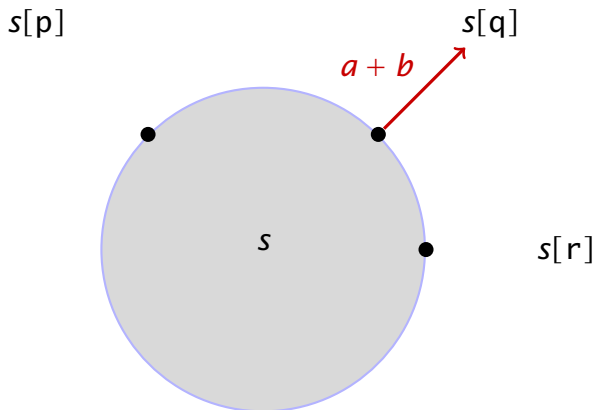
- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$

# Sessions and session types



- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$

# Sessions and session types



- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$

# Digression

$t \rightarrow s$

`!t.?s.end`

# Session correctness = safety + liveness

## Safety

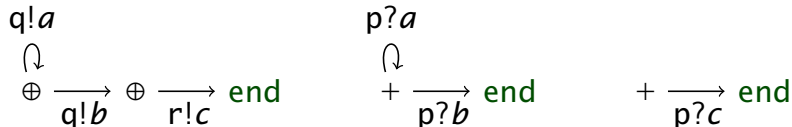
- no message of unexpected type is ever sent

## Liveness

- all non-terminated participants (eventually) make progress

## Example: multi-party session

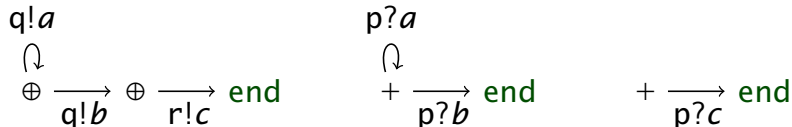
- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$



Is this session correct?

## Example: multi-party session

- $s[p] : T = q!a.T \oplus q!b.r!c.end$
- $s[q] : S = p?a.S + p?b.end$
- $s[r] : p?c.end$



Is this session correct? **Yes, under a fairness assumption**

# Subtyping for session types

- Simon Gay, Malcolm Hole, **Subtyping for session types in the pi calculus**, 2005

$\text{end} \leq_{\text{GH}} \text{end}$

$$\frac{T_1 \leq_{\text{GH}} S_1}{p?a.T_1 \leq_{\text{GH}} p?a.S_1 + p?b.S_2}$$

$$\frac{T_1 \leq_{\text{GH}} S_1}{p!a.T_1 \oplus p!b.T_2 \leq_{\text{GH}} p!a.S_1}$$

# Subtyping for session types

- Simon Gay, Malcolm Hole, **Subtyping for session types in the pi calculus**, 2005

$\text{end} \leq_{\text{GH}} \text{end}$

$$\frac{T_1 \leq_{\text{GH}} S_1}{p?a.T_1 \leq_{\text{GH}} p?a.S_1 + p?b.S_2}$$

covariant input

$$\frac{T_1 \leq_{\text{GH}} S_1}{p!a.T_1 \oplus p!b.T_2 \leq_{\text{GH}} p!a.S_1}$$

# Subtyping for session types

- Simon Gay, Malcolm Hole, **Subtyping for session types in the pi calculus**, 2005

$\text{end} \leq_{\text{GH}} \text{end}$

$$\frac{T_1 \leq_{\text{GH}} S_1}{p?a.T_1 \leq_{\text{GH}} p?a.S_1 + p?b.S_2}$$

$$\frac{T_1 \leq_{\text{GH}} S_1}{p!a.T_1 \oplus p!b.T_2 \leq_{\text{GH}} p!a.S_1}$$

contravariant output

# Digression

$\text{int} \leq \text{real}$

- it is safe to use a **value** of type `int` where a **value** of type `real` is expected

$!\text{real} \leq_{\text{GH}} !\text{int}$

- it is safe to use a **channel** of type `!int` where a **channel** of type `!real` is expected, or
- it is safe to use a **process** that sends `int`'s where a **process** that sends `real`'s is expected

# Digression

$\text{int} \leq \text{real}$

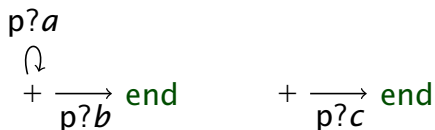
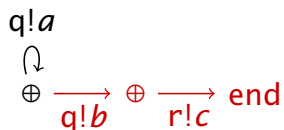
- it is safe to use a **value** of type `int` where a **value** of type `real` is expected

$!\text{real} \leq_{\text{GH}} !\text{int}$

- it is safe to use a **channel** of type `!int` where a **channel** of type `!real` is expected, or
- it is safe to use a **process** that sends `int`'s where a **process** that sends `real`'s is expected

# Example: multi-party session (and subtyping)

- $p : T = q!a.T \oplus q!b.r!c.end$
- $q : S = p?a.S + p?b.end$
- $r : p?c.end$



# Example: multi-party session (and subtyping)

- $p : T = q!a.T$
- $q : S = p?a.S + p?b.end$
- $r : p?c.end$

$$q!a$$
$$\Downarrow$$
$$\oplus$$
$$p?a$$
$$\Downarrow$$
$$+ \xrightarrow{p?b} end$$
$$+ \xrightarrow{p?c} end$$

Is this session correct?

# Dyadic vs multi-party sessions

In the dyadic setting. . .

- $\leq_{GH}$  preserves both safety and liveness

In the multi-party setting. . .

- $\leq_{GH}$  preserves safety
- $\leq_{GH}$  does not (necessarily) preserve liveness

# How to fix subtyping

session  $M = T_1 \mid \dots \mid T_n$

Definition (**correct session**)

- $M$  **correct** if  $M \Rightarrow N$  implies  $N \Rightarrow \text{end} \mid \dots \mid \text{end}$

Definition (fair subtyping)

- $\llbracket T \rrbracket = \{M \mid (T \mid M) \text{ is correct}\}$
- $T \leq S$  iff  $\llbracket T \rrbracket \subseteq \llbracket S \rrbracket$

# How to fix subtyping

session  $M = T_1 \mid \dots \mid T_n$

Definition (**correct session**)

- $M$  **correct** if  $M \Rightarrow N$  implies  $N \Rightarrow \text{end} \mid \dots \mid \text{end}$

Definition (fair subtyping)

- $\llbracket T \rrbracket = \{M \mid (T \mid M) \text{ is } \mathbf{correct}\}$
- $T \leq S$  iff  $\llbracket T \rrbracket \subseteq \llbracket S \rrbracket$

# Dilemma

$\leq_{GH}$  versus  $\leq$

- $\leq_{GH}$  is intuitive but unsound
- $\leq$  is sound but obscure

# $\leq_{GH}$ and $\leq$ are incomparable

$$\begin{aligned} T &= p!a.T \\ S &= q?b.S \end{aligned}$$

$$T \leq S$$

$$\llbracket T \rrbracket = \llbracket S \rrbracket = \emptyset$$

$$\begin{aligned} T &\not\leq_{GH} S \\ S &\not\leq_{GH} T \end{aligned}$$

# $\leq_{\text{GH}}$ and $\leq$ are incomparable

$$\begin{aligned} T &= p!a.T \\ S &= q?b.S \end{aligned}$$

$$T \leq S$$

$$\llbracket T \rrbracket = \llbracket S \rrbracket = \emptyset$$

$$\begin{aligned} T &\not\leq_{\text{GH}} S \\ S &\not\leq_{\text{GH}} T \end{aligned}$$

not viable

$$\llbracket \text{fail} \rrbracket = \llbracket T \rrbracket = \llbracket S \rrbracket = \dots = \emptyset$$

$$\llbracket \dots \rrbracket \neq \emptyset$$

viable

$$T \leq S \Rightarrow T \leq_{\text{GH}} S$$

When does  $\leq_{\text{GH}}$  imply  $\leq$ ?

$T \leq S$  implies  $\text{traces}(S) \subseteq \text{traces}(T)$

When does  $\leq_{\text{GH}}$  imply  $\leq$ ?

$T \leq S$  implies  $\text{traces}(S) \subseteq \text{traces}(T)$

**almost, anyway**

# When does $\leq_{GH}$ imply $\leq$ ?

$T \leq S$  implies  $\text{traces}(S) \subseteq \text{traces}(T)$

## Idea

- 1 define  $T - S$  so that  $\text{traces}(T - S) = \text{traces}(T) \setminus \text{traces}(S)$
- 2 check whether  $T - S$  is viable
  - if  $M \mid (T - S)$  is correct, then  $M$  does not “use” any trace in  $\text{traces}(T)$  if it is also in  $\text{traces}(S)$

# When does $\leq_{\text{GH}}$ imply $\leq$ ?

$T \leq S$  implies  $\text{traces}(S) \subseteq \text{traces}(T)$

## Idea

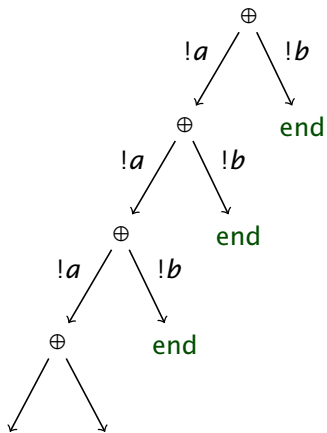
- 1 define  $T - S$  so that  $\text{traces}(T - S) = \text{traces}(T) \setminus \text{traces}(S)$
- 2 check whether  $T - S$  is viable
  - if  $M \mid (T - S)$  is correct, then  $M$  does not “use” any trace in  $\text{traces}(T)$  if it is also in  $\text{traces}(S)$

## Theorem

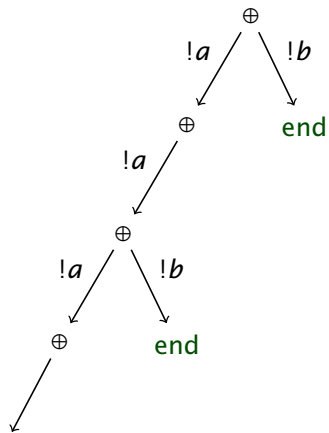
$T \leq S$  iff  $T \leq_{\text{GH}} S$  and  $T - S$  is not viable

# Experiment 1

$$T = !a.T \oplus !b.end$$

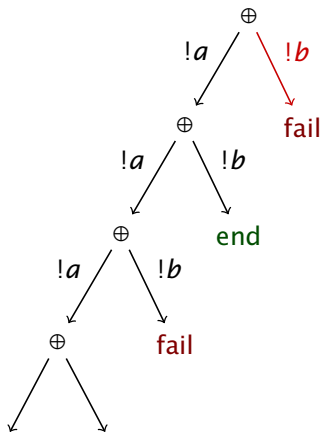


$$S = !a.!a.S \oplus !b.end$$

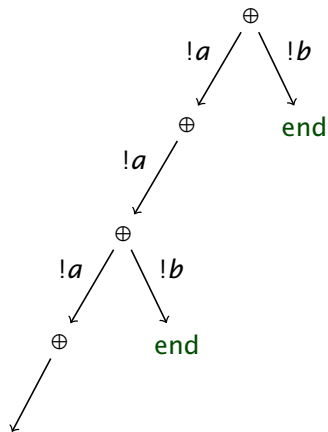


# Experiment 1

$T - S \leqslant ?$



$S = !a.!a.S \oplus !b.end$

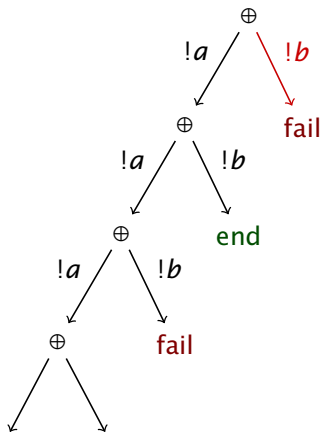


Murphy's law

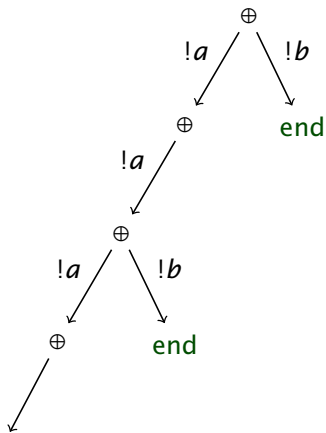
If you can fail, you will fail

# Experiment 1

$T - S \leq \text{fail}$



$S = !a.!a.S \oplus !b.\text{end}$



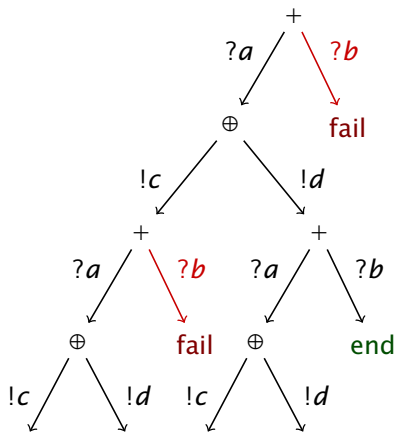
Murphy's law

If you can fail, you will fail

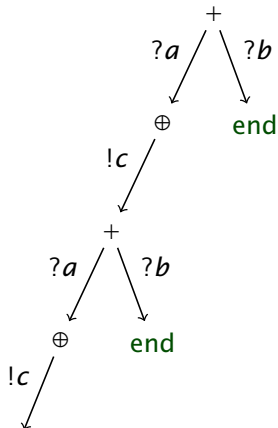


# Experiment 2

$$T - S \leqslant ?$$



$$S = ?a.!c.S + ?b.end$$



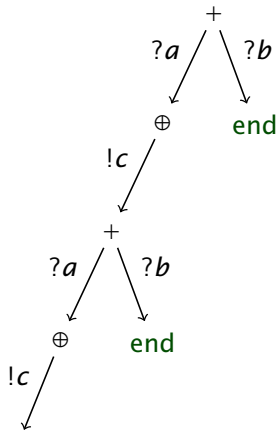
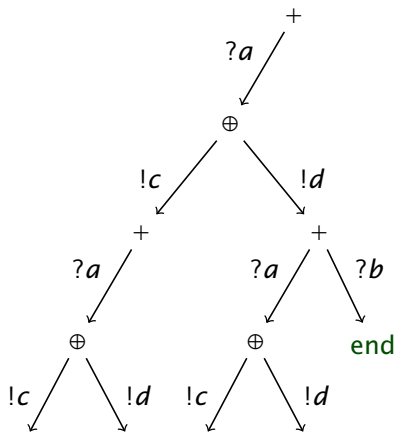
## Dual of Murphy's law

If you don't fail enough, your partner won't fail

# Experiment 2

$$T - S \leq ?a.(!c.(T - S) \oplus !d.T)$$

$$S = ?a.!c.S + ?b.end$$



## Dual of Murphy's law

If you don't fail enough, your partner won't fail

# Axiomatization of $\leq$

(S-Fail)  
 $\text{fail} \leq T$

(S-End)  
 $\text{end} \leq \text{end}$

(S-Input)

$$\frac{T_1 \leq S_1}{p?a.T_1 \leq p?a.S_1 + p?b.S_2}$$

(S-Output)

$$\frac{T_1 \leq S_1 \quad T - S \leq \text{fail}}{T = p!a.T_1 \oplus p!b.T_2 \leq p!a.S_1 = S}$$

# Axiomatization of $\leq$

(S-Fail)  
**fail**  $\leq T$

(S-End)  
**end**  $\leq \mathbf{end}$

(S-Input)

$$\frac{T_1 \leq S_1}{p?a.T_1 \leq p?a.S_1 + p?b.S_2}$$

(S-Output)

$$\frac{T_1 \leq S_1 \quad T - S \leq \mathbf{fail}}{T = p!a.T_1 \oplus p!b.T_2 \leq p!a.S_1 = S}$$

# Axiomatization of $\leq$

$$\begin{array}{l} \text{(S-Fail)} \\ \text{fail} \leq T \end{array}$$

$$\begin{array}{l} \text{(S-End)} \\ \text{end} \leq \text{end} \end{array}$$

(S-Input)

$$\frac{T_1 \leq S_1}{p?a.T_1 \leq p?a.S_1 + p?b.S_2}$$

(S-Output)

$$\frac{T_1 \leq S_1 \quad T - S \leq \text{fail}}{T = p!a.T_1 \oplus p!b.T_2 \leq p!a.S_1 = S}$$

(S-Path)

$$\frac{\exists S \in \pi : S \leq \text{fail} \quad (T \downarrow \pi)}{T \leq \text{fail}}$$

(S-Murphy)

$$\frac{k \in \{1, 2\} \quad T_k \leq \text{fail}}{p!a_1.T_1 \oplus p!a_2.T_2 \leq \text{fail}}$$

# Axiomatization of $\leq$

(S-Fail)  
 $\text{fail} \leq T$

(S-End)  
 $\text{end} \leq \text{end}$

(S-Input)

$T_1 \leq S_1$

(S-Output)

$T_1 \leq S_1$

$T - S \leq \text{fail}$

$p?a.T_1 \leq$  **every path  $\pi$  to success...**  $T_1 \oplus p!b.T_2 \leq p!a.S_1 = S$

(S-Path)

$\exists S \in \pi : S \leq \text{fail} \quad (T \downarrow \pi)$

$T \leq \text{fail}$

(S-Murphy)

$k \in \{1, 2\} \quad T_k \leq \text{fail}$

$p!a_1.T_1 \oplus p!a_2.T_2 \leq \text{fail}$

# Axiomatization of $\leq$

(S-Fail)  
 $\text{fail} \leq T$

(S-End)  
 $\text{end} \leq \text{end}$

(S-Input)

$T_1 \leq S_1$

(S-Output)

$T_1 \leq S_1$

$T - S \leq \text{fail}$

$p?a.T_1 \leq$

**every path  $\pi$  to success...**

$T_1 \oplus p!b.T_2 \leq p!a.S_1 = S$

**... goes through some node  $S$  that...**

(S-Path)

$\exists S \in \pi : S \leq \text{fail} \quad (T \downarrow \pi)$

$T \leq \text{fail}$

(S-Murphy)

$k \in \{1, 2\} \quad T_k \leq \text{fail}$

$p!a_1.T_1 \oplus p!a_2.T_2 \leq \text{fail}$

# Axiomatization of $\leq$

(S-Fail)  
 $\text{fail} \leq T$

(S-End)  
 $\text{end} \leq \text{end}$

(S-Input)

$$T_1 \leq S_1$$

(S-Output)

$$T_1 \leq S_1$$

$$T - S \leq \text{fail}$$

$$\frac{p?a.T_1 \leq T_1 \leq S_1 \quad T_1 \oplus p!b.T_2 \leq p!a.S_1 = S}{p?a.T_1 \leq T_1 \oplus p!b.T_2 \leq p!a.S_1 = S}$$

every path  $\pi$  to success...

... goes through some node  $S$  that...

(S-Path)

$$\exists S \in \pi : S \leq \text{fail} \quad (T \downarrow \pi)$$

$$T \leq \text{fail}$$

(S-Murphy)

$$k \in \{1, 2\} \quad T_k \leq \text{fail}$$

$$p!a_1.T_1 \oplus p!a_2.T_2 \leq \text{fail}$$

... leads to failure

## (Fair) subtyping = (fair) testing preorder

- $P$  passes test  $T$
- $P \sqsubseteq Q$  iff  $P$  passes test  $T$  implies  $Q$  passes test  $T$

### “Unfair” testing

- De Nicola, Hennessy, **Testing equivalences for processes**, 1983
- ... similar properties of  $\leq_{GH}$

### Fair testing

- Cleaveland, Natarajan, **Divergence and fair testing**, 1995
- Rensink, Vogler, **Fair testing**, 2007

# Fair testing vs fair subtyping

## Fair testing

- + generic processes
- denotational (= obscure) characterization
- no complete axiomatization
- exponential

## Fair subtyping

- simple processes (session types)
- + operational (= hopefully less obscure) characterization
- + complete axiomatization
- + polynomial

# More on fair subtyping

- Padovani, **Fair Subtyping for Multi-Party Session Types**, COORDINATION 2011
  - + formal definitions and proofs
  - + algorithms (viability, normal form, subtyping)
  
- long version coming up on my home page in **3 days**
  - + higher-order session types
  - + details of axiomatization

# Challenges

“fair” type checking?

# Fair Subtyping for Multi-Party Session Types

Luca Padovani

Dipartimento di Informatica, Università di Torino