

Fair Testing and Fair Subtyping

Luca Padovani

Dipartimento di Informatica, Università di Torino

Liveness

Session correctness = safety + liveness

Safety

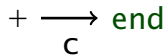
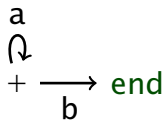
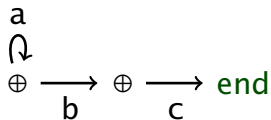
- no message of unexpected type is ever sent

Liveness

- all non-terminated participants make progress (eventually)

Example

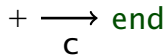
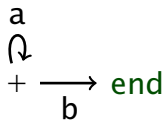
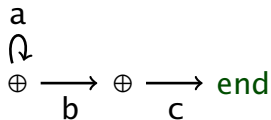
- $s[p] : \text{rec } x. (\text{a}.x \oplus \text{b}.c.\text{end})$
- $s[q] : \text{rec } x. (\text{a}.x + \text{b}.\text{end})$
- $s[r] : c.\text{end}$



Is this session correct?

Example

- $s[p] : \text{rec } x. (\text{a.}x \oplus \text{b.c.end})$
- $s[q] : \text{rec } x. (\text{a.}x + \text{b.end})$
- $s[r] : \text{c.end}$



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

end \leq_{GH} end

$$\frac{T_i \leq_{\text{GH}} S_i \quad (i \in I)}{\sum_{i \in I} a_i.T_i \leq_{\text{GH}} \sum_{i \in I \cup J} a_i.S_i}$$

$$\frac{T_i \leq_{\text{GH}} S_i \quad (i \in I)}{\bigoplus_{i \in I \cup J} a_i.T_i \leq_{\text{GH}} \bigoplus_{i \in I} a_i.S_i}$$

Subtyping for session types

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

end \leq_{GH} end

$$\boxed{\frac{T_i \leq_{\text{GH}} S_i \quad (i \in I)}{\sum_{i \in I} a_i.T_i \leq_{\text{GH}} \sum_{i \in I \cup J} a_i.S_i}}$$

covariant input

$$\frac{T_i \leq_{\text{GH}} S_i \quad (i \in I)}{\bigoplus_{i \in I \cup J} a_i.T_i \leq_{\text{GH}} \bigoplus_{i \in I} a_i.S_i}$$

Subtyping for session types

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

end \leq_{GH} end

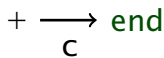
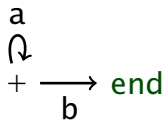
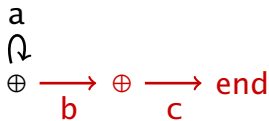
$$\frac{T_i \leq_{\text{GH}} S_i \quad (i \in I)}{\sum_{i \in I} a_i.T_i \leq_{\text{GH}} \sum_{i \in I \cup J} a_i.S_i}$$

$$\boxed{\frac{T_i \leq_{\text{GH}} S_i \quad (i \in I)}{\bigoplus_{i \in I \cup J} a_i.T_i \leq_{\text{GH}} \bigoplus_{i \in I} a_i.S_i}}$$

contravariant output

Example (with subtyping)

- $s[p] : \text{rec } x.(\text{a.x} \oplus \text{b.c.end})$
- $s[q] : \text{rec } x.(\text{a.x} + \text{b.end})$
- $s[r] : \text{c.end}$



Example (with subtyping)

- $s[p] : \text{rec } x. a.x$
- $s[q] : \text{rec } x.(a.x + b.\text{end})$
- $s[r] : c.\text{end}$

a
↻
⊕

a
↻
+ \xrightarrow{b} end

+ \xrightarrow{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

Subtyping and liveness

$$\text{end} \leq_{\text{GH}} \sum_{i \in I} a_i.T_i$$

$$\bigoplus_{i \in I} a_i.T_i \leq_{\text{GH}} \text{end}$$

Fair testing

- Arend Rensink, Walter Vogler, **Fair testing**, 2007

What's wrong with fair testing?

$$P \sqsubseteq^c Q$$

$$P \sqsubseteq^+ Q$$

- \sqsubseteq^+ is interesting, but not a pre-congruence ☹
- \sqsubseteq^c is a pre-congruence, but $\text{tr}(P) = \text{tr}(Q)$ ☹
- \sqsubseteq^+ and \sqsubseteq^c have denotational characterizations ☹

What's wrong with fair testing?

$$P \sqsubseteq^c Q$$

$$P \sqsubseteq^+ Q$$

- \sqsubseteq^+ is interesting, but not a pre-congruence ☹
- \sqsubseteq^c is a pre-congruence, but $\text{tr}(P) = \text{tr}(Q)$ ☹
- \sqsubseteq^+ and \sqsubseteq^c have denotational characterizations ☹

What's wrong with fair testing?

$$P \sqsubseteq^c Q$$

$$P \sqsubseteq^+ Q$$

- \sqsubseteq^+ is interesting, but not a pre-congruence ☹
- \sqsubseteq^c is a pre-congruence, but $\text{tr}(P) = \text{tr}(Q)$ ☹
- \sqsubseteq^+ and \sqsubseteq^c have denotational characterizations ☹

What's wrong with fair testing?

$$P \sqsubseteq^c Q$$

$$P \sqsubseteq^+ Q$$

- \sqsubseteq^+ is interesting, but not a pre-congruence ☹️
- \sqsubseteq^c is a pre-congruence, but $\text{tr}(P) = \text{tr}(Q)$ ☹️
- \sqsubseteq^+ and \sqsubseteq^c have denotational characterizations ☹️

Fair Testing

\sqsubseteq^+ and recursion

$$\begin{array}{l} \tau.a.x + \tau.b \sqsubseteq^+ \tau.a.x \\ \text{rec } x.(\tau.a.x + \tau.b) \not\sqsubseteq^+ \text{rec } x.\tau.a.x \end{array}$$

How is \sqsubseteq^+ defined

- 1 $P \text{ shd } t$ iff P passes test t
- 2 $P \sqsubseteq Q$ iff $P \text{ shd } t$ implies $Q \text{ shd } t$ for every t
- 3 $P \sqsubseteq Q$ iff $P\sigma \sqsubseteq Q\sigma$ for every σ
- 4 $P \sqsubseteq^+ Q$ iff $C[P] \sqsubseteq C[Q]$ for every first-order C



$P \sqsubseteq^c Q$ implies $\text{tr}(P) = \text{tr}(Q)$

$\tau.\text{yes} + \tau.\text{no} \not\sqsubseteq^c \tau.\text{no}$

$C[-] = \text{rec } x.((\tau.x + \text{yes}.a) |_{\text{yes}} -) / \text{yes}, \text{no}$

$C[\tau.\text{yes} + \tau.\text{no}] \text{ shd } a. \checkmark \quad \neg(C[\tau.\text{no}] \text{ shd } a. \checkmark)$

Strategy

- 1 consider **finite-state processes** only
- 2 $P \leq Q$ iff $C[P] \text{ shd } t$ implies $C[Q] \text{ shd } t$ for every C and t

\sqsubseteq^c




\sqsubseteq^+

- \leq is almost as large as \sqsubseteq^+ 
- \leq is a pre-congruence 
- \leq has behavioral/axiomatic characterizations 

Strategy

- 1 consider **finite-state processes** only
- 2 $P \leq Q$ iff $C[P] \text{ shd } t$ implies $C[Q] \text{ shd } t$ for every C and t




$$\sqsubseteq^c \quad \dots \leq \quad \dots \quad \sqsubseteq^+$$

- \leq is almost as large as \sqsubseteq^+ 
- \leq is a pre-congruence 
- \leq has behavioral/axiomatic characterizations 

Strategy

- 1 consider **finite-state processes** only
- 2 $P \leq Q$ iff $C[P] \text{ shd } t$ implies $C[Q] \text{ shd } t$ for every C and t




$$\sqsubseteq^c \quad \dots \leq \quad \dots \quad \sqsubseteq^+$$

- \leq is almost as large as \sqsubseteq^+ 
- \leq is a pre-congruence 
- \leq has behavioral/axiomatic characterizations 

Strategy

- 1 consider **finite-state processes** only
- 2 $P \leq Q$ iff $C[P] \text{ shd } t$ implies $C[Q] \text{ shd } t$ for every C and t

$$\sqsubseteq^c \quad \dots \leq \quad \dots \quad \sqsubseteq^+$$

- \leq is almost as large as \sqsubseteq^+ 
- \leq is a pre-congruence 
- \leq has behavioral/axiomatic characterizations 

When is it safe to prune traces?

Never for generic processes

- $P \sqsubseteq^c Q$ implies $\text{tr}(P) = \text{tr}(Q)$ **because** Q can be **restarted**

When is it safe to prune traces?

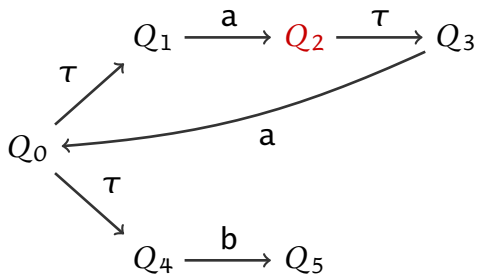
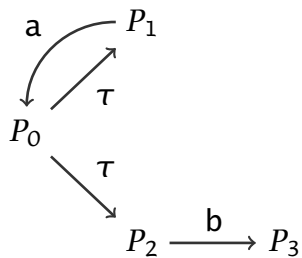
Never for generic processes

- $P \sqsubseteq^c Q$ implies $\text{tr}(P) = \text{tr}(Q)$ **because** Q can be **restarted**

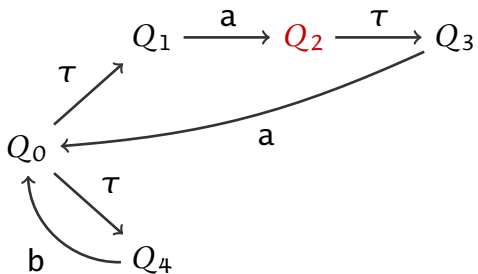
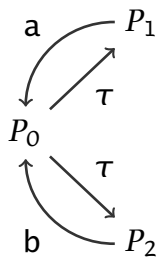
Sometimes for finite-state processes

- $P \sqsubseteq^c Q$ implies $\text{tr}(P) = \text{tr}(Q)$ **if** Q can be **restarted**

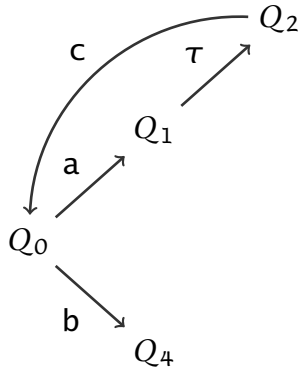
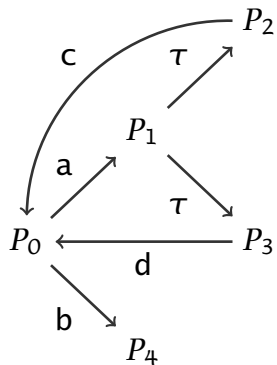
Example 1



Example 2



Example 3 (restartable \neq no exit path)



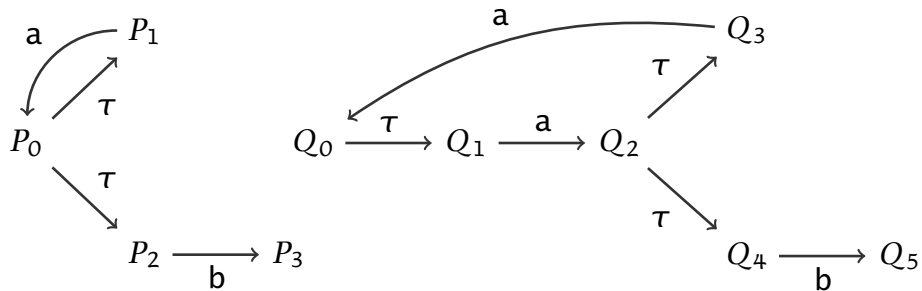
Detecting unrestartable processes

- $P < Q$ “either $\text{tr}(P) = \text{tr}(Q)$ or no test for P restarts Q ”

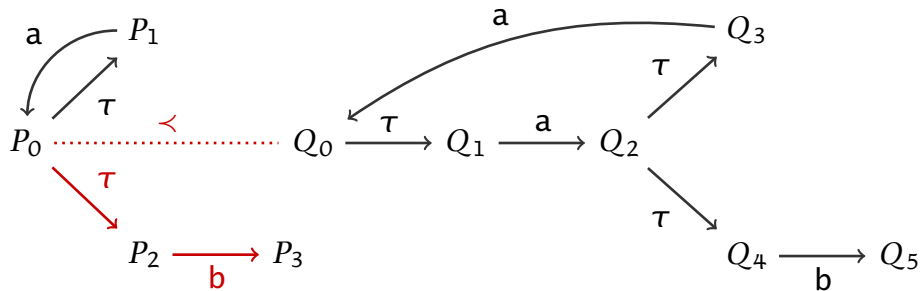
$$\frac{\forall s \in \text{tr}(P) \setminus \text{tr}(Q) : \exists t \leq s : P \xRightarrow{t} P' \wedge Q \xRightarrow{t} Q' \wedge P' < Q'}{P < Q}$$

- Every path that distinguishes P from Q ...
- ...goes through a state P' of P and a state Q' of Q ...
- ...such that P' and Q' are slightly less different.

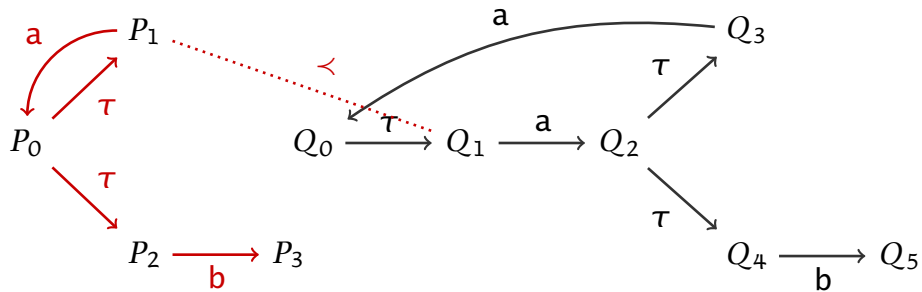
Example 4



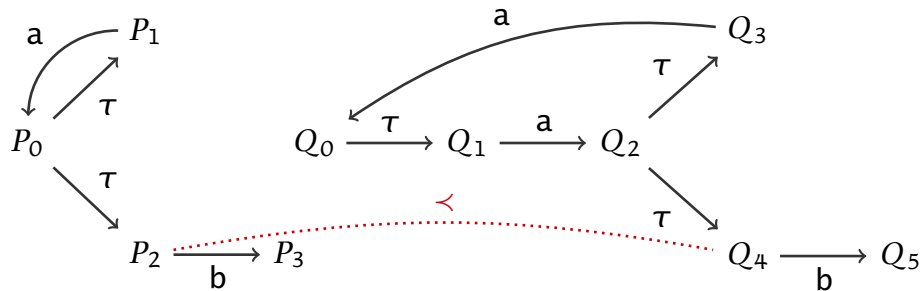
Example 4



Example 4



Example 4



Fair Subtyping

Symmetric fair testing

✓ is synchronizing

$$T \text{ shd } S \stackrel{\text{def}}{\iff} T \mid S \stackrel{s}{\implies} T' \mid S' \quad \text{implies} \quad T' \mid S' \stackrel{t\checkmark}{\implies}$$

Subtyping and “emptyness”

$\text{rec } x.a.x \preceq \text{rec } x.b.x$

Subtyping and “emptiness”

$\text{rec } x.a.x \preceq \text{rec } x.b.x$

not viable

$\text{fail} \preceq \text{rec } x.a.x \preceq \text{rec } x.b.x \preceq \dots$

viable

$\text{end}, \text{rec } x.(\tau.a.x + \tau.b.\text{end}), \dots$

Reducing $<$ to non-viability

Theorem

$T < S$ iff $T - S$ is not viable

Reducing $<$ to non-viability

Theorem

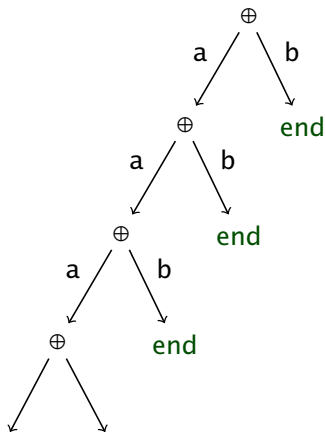
$T < S$ iff $T - S$ is not viable

end – end = fail

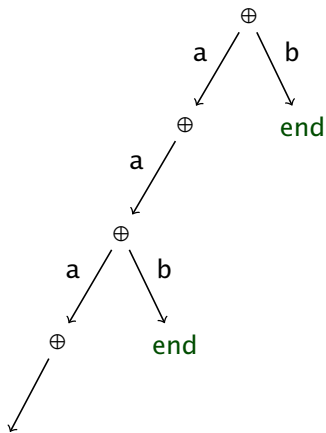
$$\begin{aligned}\sum_{i \in I} a_i \cdot T_i - \sum_{i \in I \cup J} a_i \cdot S_i &= \sum_{i \in I} a_i \cdot (T_i - S_i) \\ \bigoplus_{i \in I \cup J} a_i \cdot T_i - \bigoplus_{i \in I} a_i \cdot S_i &= \bigoplus_{i \in I} a_i \cdot (T_i - S_i) \oplus \bigoplus_{i \in J} a_i \cdot T_i\end{aligned}$$

Experiment 1

$$T = \text{rec } x.(a.x \oplus b.\text{end})$$

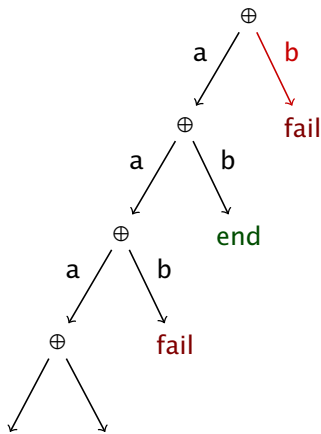


$$S = \text{rec } x.(a.a.x \oplus b.\text{end})$$

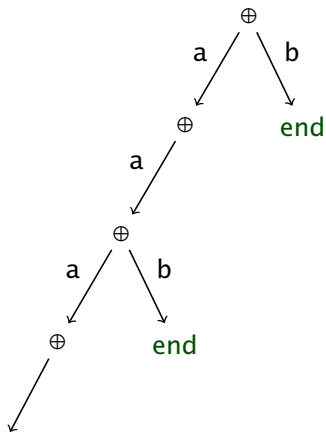


Experiment 1

$$T - S \leq ?$$



$$S = \text{rec } x.(a.a.x \oplus b.\text{end})$$

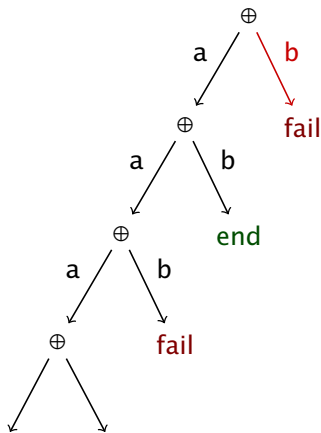


Murphy's law

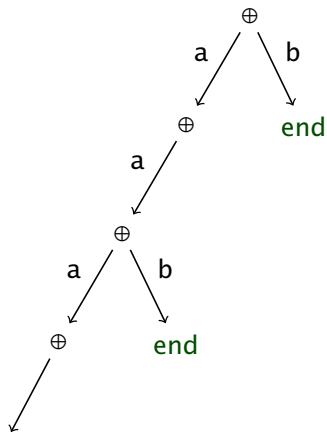
If you can fail, you will fail

Experiment 1

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



$S = \text{rec } x.(a.a.x \oplus b.\text{end})$



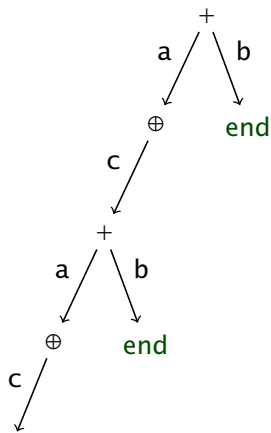
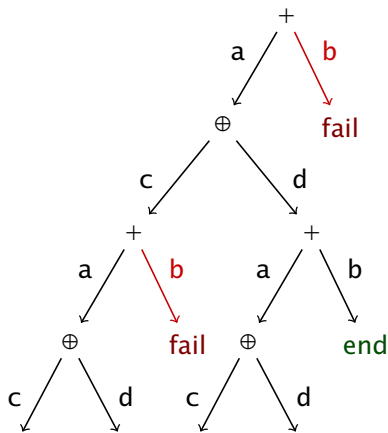
Murphy's law

If you can fail, you will fail

Experiment 2

$$T - S \leqslant ?$$

$$S = \text{rec } x.(a.c.x + b.\text{end})$$



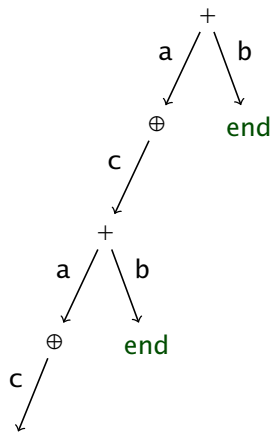
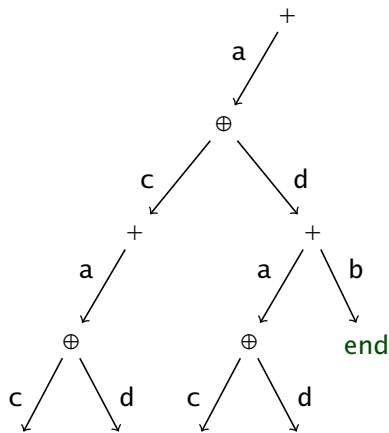
Dual of Murphy's law

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

Experiment 2

$$T - S \leq \text{rec } x.(a.(c.x \oplus d.T))$$

$$S = \text{rec } x.(a.c.x + b.\text{end})$$



Dual of Murphy's law

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

Deduction system for \leq

(S-Fail)
fail $\leq T$

(S-End)
end \leq end

(S-Input)

$$\frac{T_i \leq S_i \quad (i \in I)}{\sum_{i \in I} a_i \cdot T_i \leq \sum_{i \in I \cup J} a_i \cdot S_i}$$

(S-Output)

$$\frac{T_i \leq S_i \quad (i \in I) \quad T - S \leq \text{fail}}{T = \bigoplus_{i \in I \cup J} a_i \cdot T_i \leq \bigoplus_{i \in I} a_i \cdot S_i = S}$$

Deduction system for \leq

(S-Fail)
fail $\leq T$

(S-End)
end \leq **end**

(S-Input)

$$\frac{T_i \leq S_i \quad (i \in I)}{\sum_{i \in I} a_i.T_i \leq \sum_{i \in I \cup J} a_i.S_i}$$

(S-Output)

$$\frac{T_i \leq S_i \quad (i \in I) \quad T - S \leq \mathbf{fail}}{T = \bigoplus_{i \in I \cup J} a_i.T_i \leq \bigoplus_{i \in I} a_i.S_i = S}$$

Deduction system for \leq

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

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

(S-Input)

$$\frac{T_i \leq S_i \quad (i \in I)}{\sum_{i \in I} a_i.T_i \leq \sum_{i \in I \cup J} a_i.S_i}$$

(S-Output)

$$\frac{T_i \leq S_i \quad (i \in I) \quad T - S \leq \text{fail}}{T = \bigoplus_{i \in I \cup J} a_i.T_i \leq \bigoplus_{i \in I} a_i.S_i = S}$$

(S-Path)

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

(S-Murphy)

$$\frac{k \in I \quad T_k \leq \text{fail}}{\bigoplus_{i \in I} a_i.T_i \leq \text{fail}}$$

Deduction system for \leq

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

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

(S-Input)

$$T_i \leq S_i \quad (i \in I)$$

$$\frac{\sum_{i \in I} a_i \cdot T_i}{\sum_{i \in I} a_i \cdot S_i}$$

(S-Output)

$$T_i \leq S_i \quad (i \in I)$$

$$\frac{T - S \leq \text{fail}}{\bigoplus_{i \in I} a_i \cdot S_i = S}$$

every path π to success...

(S-Path)

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

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

(S-Murphy)

$$k \in I \quad T_k \leq \text{fail}$$

$$\frac{}{\bigoplus_{i \in I} a_i \cdot T_i \leq \text{fail}}$$

Deduction system for \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)

$$T_i \leq S_i \quad (i \in I)$$

$$\frac{\sum_{i \in I} a_i \cdot T_i}{\sum_{i \in I} a_i \cdot T_i}$$

(S-Output)

$$T_i \leq S_i \quad (i \in I)$$

$$\frac{\bigoplus_{i \in I} a_i \cdot T_i}{\bigoplus_{i \in I} a_i \cdot T_i}$$

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

$$\leq \bigoplus_{i \in I} a_i \cdot S_i = S$$

every path π to success...

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

(S-Path)

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

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

(S-Murphy)

$$\frac{k \in I \quad T_k \leq \text{fail}}{k \in I \quad T_k \leq \text{fail}}$$

$$\bigoplus_{i \in I} a_i \cdot T_i \leq \text{fail}$$

Deduction system for \leq

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

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

(S-Input)

$$\frac{T_i \leq S_i \quad (i \in I)}{\sum_{i \in I} a_i.T_i \leq S}$$

(S-Output)

$$\frac{T_i \leq S_i \quad (i \in I) \quad T - S \leq \text{fail}}{\sum_{i \in I} a_i.T_i \leq \bigoplus_{i \in I} a_i.S_i = S}$$

every path π to success...

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

(S-Path)

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

(S-Murphy)

$$\frac{k \in I \quad T_k \leq \text{fail}}{\bigoplus_{i \in I} a_i.T_i \leq \text{fail}}$$

... leads to failure

Fair testing vs fair subtyping

Fair testing

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

Fair subtyping

- simple processes (session types)
- + behavioral (= hopefully less obscure) characterization
- + complete deduction system
- + polynomial

Symmetric fair testing / fair subtyping

- Mario Bravetti, Gianluigi Zavattaro, **Towards a Unifying Theory for Choreography Conformance and Contract Compliance**, 2007
- Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Nirmal Desai, Viviana Patti, Munindar P. Singh, **Choice, interoperability, and conformance in interaction protocols and service choreographies**, 2009
- Arjan J. Mooij, Christian Stahl, Marc Voorhoeve, **Relating fair testing and accordance for service replaceability**, 2010
- Robi Malik, David Streader, Steve Reeves, **Fair Testing Revisited: A Process-Algebraic Characterisation of Conflicts**, 2004

More on fair subtyping

- Luca Padovani, **Fair Subtyping for Multi-Party Session Types**, 2011
 - + formal definitions and proofs
 - + algorithms (viability, normal form, subtyping)
- long version on my home page
 - + details of deduction system
 - + higher-order session types

$$\frac{T' \leq T}{?(T).S \xrightarrow{?T'} S}$$

Challenges

- complete deduction system for Ξ^c
- “fair” type checking
- mixed “fair” and “unfair” choices

Liveness and type checking

$$P = \text{rec } X.u!a\langle e \rangle.X$$

$$T = \text{rec } x.(!a\langle t \rangle.x \oplus !b\langle \rangle.\text{end})$$

$$\frac{\frac{\frac{\vdash e : t \quad \overline{\{X \mapsto u : T\}; u : T \vdash X}}{\{X \mapsto u : T\}; u : !a\langle t \rangle.T \vdash u!a\langle e \rangle.X} \quad T \leq !a\langle t \rangle.T}{\{X \mapsto u : T\}; u : T \vdash x!a\langle e \rangle.X}}{u : T \vdash \text{rec } X.u!a\langle e \rangle.X}$$

Liveness and “unfair” choices

`rec x.?password.(τ.!ok + τ.!error.x)`

Liveness and “unfair” choices

`rec x.?password.(τ.!ok + τ.!error.x)`

`rec x.(?"fidelio".!ok + ?¬"fidelio".!error.x)`

Fair Testing and Fair Subtyping

Luca Padovani

Dipartimento di Informatica, Università di Torino