Assuming Just Enough Fairness to make Session Types Complete for Lock-freedom

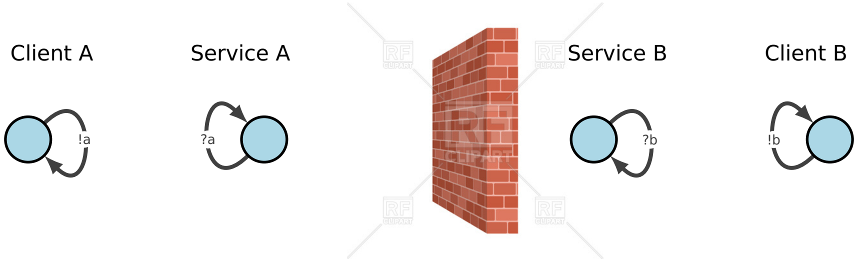
ACM/IEEE LICS 2021 36th Annual Symposium on Logic in Computer Science

Rob van Glabbeek[1], Peter Höfner[2], and Ross Horne[3]

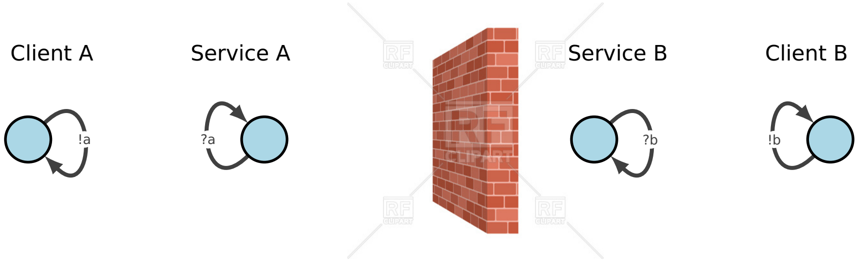
1. Data61, CSIRO and UNSW, Sydney, Australia
2. Australian National University, Canberra, Australia
3. Computer Science, University of Luxembourg, Esch-sur-Alzette, Luxembourg

29 June – 02 July, 2021

Fairness Assumptions and Liveness Properties

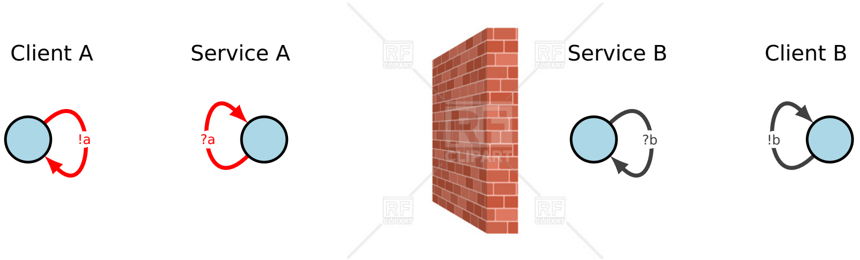


Fairness Assumptions and Liveness Properties



Liveness property:
Everyone wishing to trade eventually does so.

Fairness Assumptions and Liveness Properties

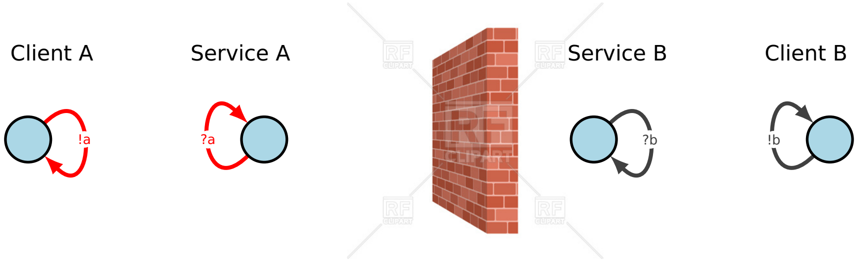


Liveness property:
Everyone wishing to trade eventually does so.

A path:

$\textit{Client A} \rightarrow \textit{Service A} : a$

Fairness Assumptions and Liveness Properties

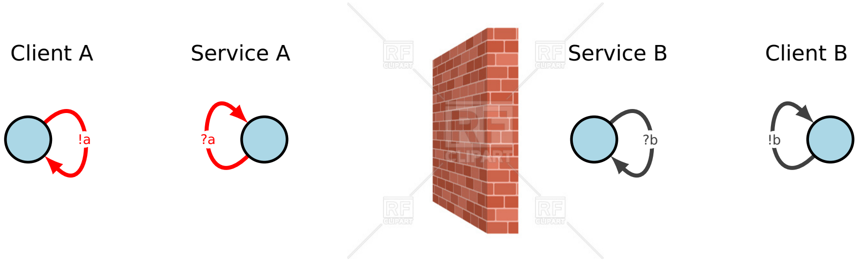


Liveness property:
Everyone wishing to trade eventually does so.

A path:

$\mathit{Client\ A} \rightarrow \mathit{Service\ A} : a \,;\, \color{red}{\mathit{Client\ A} \rightarrow \mathit{Service\ A} : a}$

Fairness Assumptions and Liveness Properties

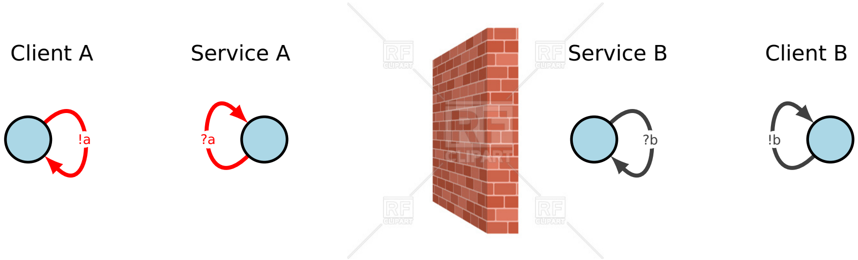


Liveness property:
Everyone wishing to trade eventually does so.

A path:

$$\mathit{Client\ A} \rightarrow \mathit{Service\ A} : a \,;\, \mathit{Client\ A} \rightarrow \mathit{Service\ A} : a \,;\, \mathit{Client\ A} \rightarrow \mathit{Service\ A} : a \ \ldots \ ✗$$

Fairness Assumptions and Liveness Properties



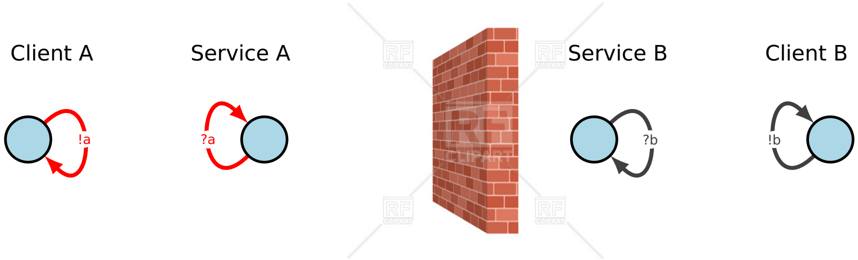
Liveness property:
Everyone wishing to trade eventually does so.

A path:

$$Client\ A \rightarrow Service\ A : a\ ;\ Client\ A \rightarrow Service\ A : a\ ;\ Client\ A \rightarrow Service\ A : a\ \ldots$$ ✗

$$\not\models \mathcal{L}(\mathrm{P})$$

Fairness Assumptions and Liveness Properties



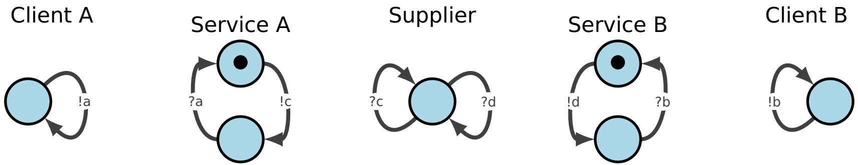
Liveness property:
Everyone wishing to trade eventually does so.

A path:

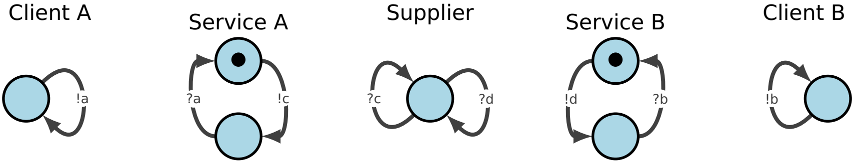
$$Client\ A \rightarrow Service\ A : a\,;\, Client\ A \rightarrow Service\ A : a\,;\, Client\ A \rightarrow Service\ A : a \ldots$$ ✗

$\not\models \mathcal{L}(\mathrm{P})$ $\models \mathcal{L}(\mathrm{J})$

Fairness Assumptions and Liveness Properties

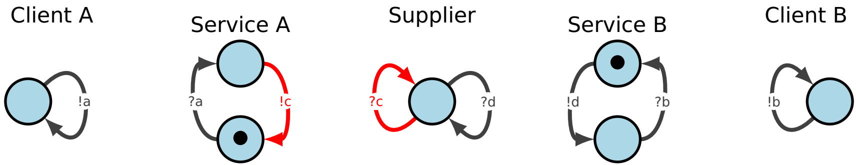


Fairness Assumptions and Liveness Properties



Liveness property:
Everyone wishing to trade eventually does so.

Fairness Assumptions and Liveness Properties

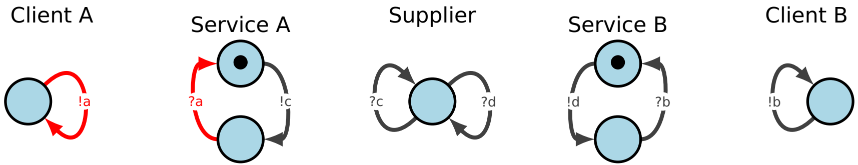


Liveness property:
Everyone wishing to trade eventually does so.

A just path:

$Service\ A \rightarrow Supplier : c$

Fairness Assumptions and Liveness Properties

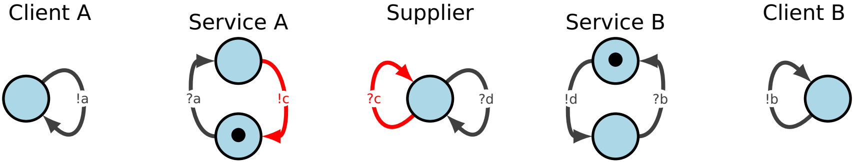


Liveness property:
Everyone wishing to trade eventually does so.

A just path:

$\mathit{Service\ A} \rightarrow \mathit{Supplier} : c\ ;\ \mathit{Client\ A} \rightarrow \mathit{Service\ A} : a$

Fairness Assumptions and Liveness Properties

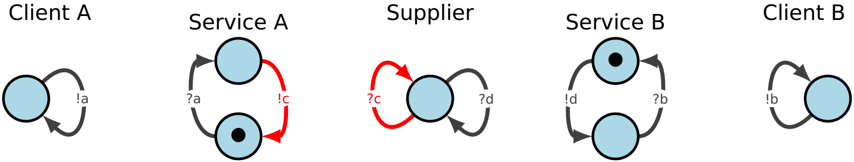


Liveness property:
Everyone wishing to trade eventually does so.

A just path:

$\mathit{Service\ A} \rightarrow \mathit{Supplier} : c\ ;\ \mathit{Client\ A} \rightarrow \mathit{Service\ A} : a\ ;\ \mathit{Service\ A} \rightarrow \mathit{Supplier} : c \ldots$ ✗

Fairness Assumptions and Liveness Properties



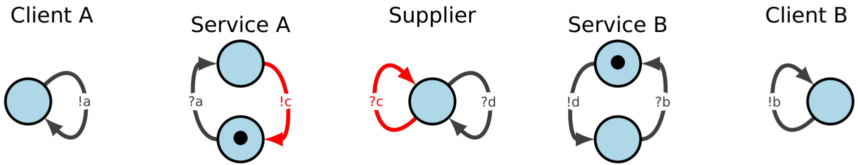
Liveness property:
Everyone wishing to trade eventually does so.

A just path:

$$\textit{Service A} \rightarrow \textit{Supplier} : c \,;\, \textit{Client A} \rightarrow \textit{Service A} : a \,;\, \textit{Service A} \rightarrow \textit{Supplier} : c \ \ldots \ ✗$$

$$\not\models \mathcal{L}(\mathbb{J})$$

Fairness Assumptions and Liveness Properties



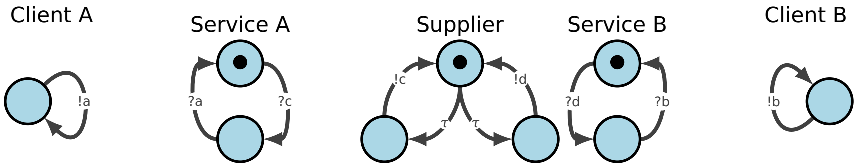
Liveness property:
Everyone wishing to trade eventually does so.

A just path:

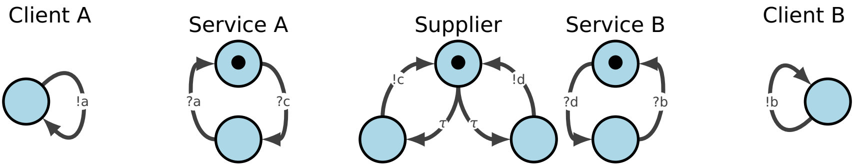
$Service\ A \rightarrow Supplier : c\, ; Client\ A \rightarrow Service\ A : a\, ; Service\ A \rightarrow Supplier : c \ldots$ ✗

$\not\models \mathcal{L}(\mathrm{J})$ $\models \mathcal{L}(\mathrm{SC})$

Fairness Assumptions and Liveness Properties

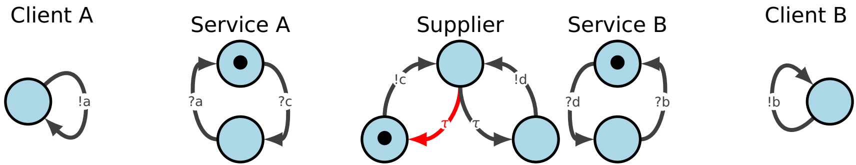


Fairness Assumptions and Liveness Properties



Liveness property:
Everyone wishing to trade eventually does so.

Fairness Assumptions and Liveness Properties

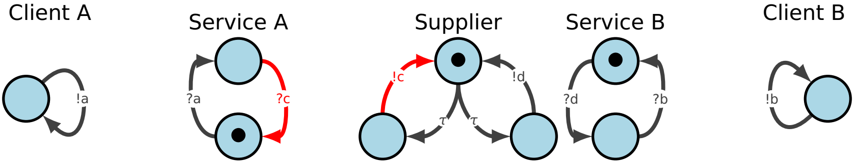


Liveness property:
Everyone wishing to trade eventually does so.

A path where components are strongly fair:

τ

Fairness Assumptions and Liveness Properties

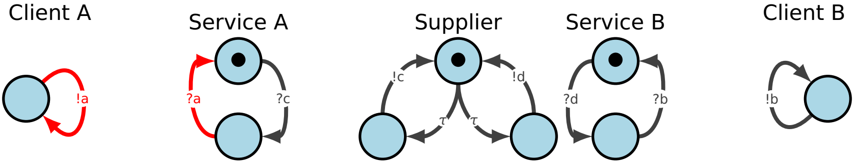


Liveness property:
Everyone wishing to trade eventually does so.

A path where components are strongly fair:

τ; *Supplier* → *Service A* : *c*

Fairness Assumptions and Liveness Properties

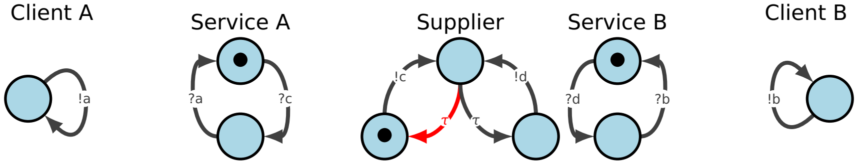


Liveness property:
Everyone wishing to trade eventually does so.

A path where components are strongly fair:

$\tau; Supplier \rightarrow Service\ A : c\ ; Client\ A \rightarrow Service\ A : a$

Fairness Assumptions and Liveness Properties

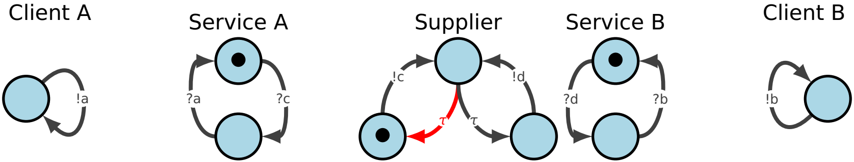


Liveness property:
Everyone wishing to trade eventually does so.

A path where components are strongly fair:

$$\tau ; Supplier \rightarrow Service\ A : c \, ; Client\ A \rightarrow Service\ A : a \, ; \tau \ldots$$ ✗

Fairness Assumptions and Liveness Properties



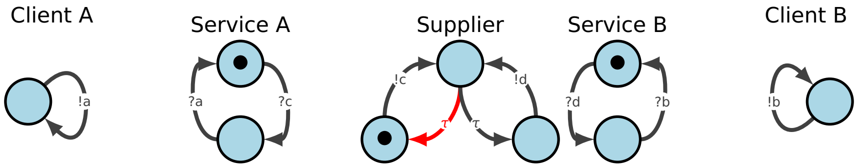
Liveness property:
Everyone wishing to trade eventually does so.

A path where components are strongly fair:

$$\tau; Supplier \rightarrow Service\ A : c\ ; Client\ A \rightarrow Service\ A : a\ ; \tau \ldots ✗$$

$$\not\models \mathcal{L}(\mathrm{SC})$$

Fairness Assumptions and Liveness Properties



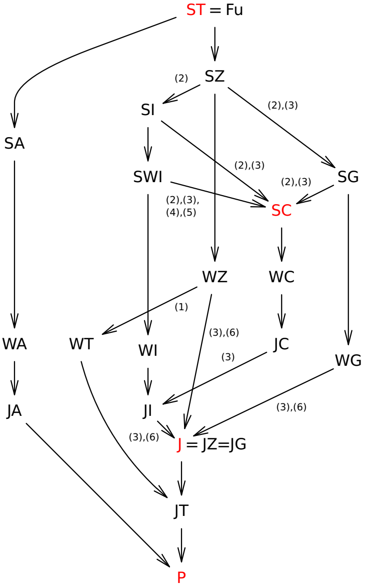
Liveness property:
Everyone wishing to trade eventually does so.

A path where components are strongly fair:

$$\tau; Supplier \rightarrow Service\ A : c\, ; Client\ A \rightarrow Service\ A : a\, ; \tau \ldots ✗$$

$\not\models \mathcal{L}(\mathrm{SC})$ $\models \mathcal{L}(\mathrm{ST})$

Notions of Fairness for Finite-state Automata



Under some mild assumptions:

(1) For each synchronisation $Z \subseteq \mathcal{I}$, and for each network state $\mathbb{N}$, there is at most one transition t with $instr(t) = Z$ that is enabled in $\mathbb{N}$.

(2) $\mathcal{I}$ is finite.

(3) There is a function $cp : \mathcal{I} \to \mathcal{C}$, where $\mathcal{C}$ is the set of components or locations in the network, such that $comp(t) = \{cp(I) \mid I \in instr(t)\}$ for all transitions t.

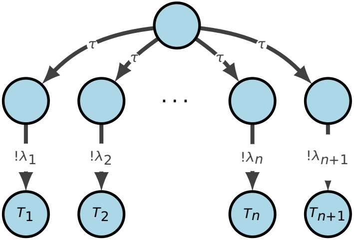
(4) If an instruction I is enabled in a state $\mathbb{N}$, it is also requested.

(5) If instruction I is requested in network state $\mathbb{N}$ and u is a transition from $\mathbb{N}$ to $\mathbb{N}'$ such that $cp(I) \notin comp(u)$, then I is still requested in $\mathbb{N}'$.

(6) If $t \smile u$ with $source(t) = source(u)$, then $\exists v \in Tr$ with $source(v) = target(u)$ and $instr(v) = instr(t)$.

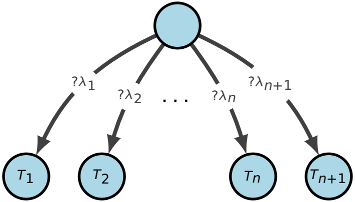
Restricting to Session Calculi

Internal choice between outputs:



$$\bigoplus_{i \in I} p_i!\lambda_i;\mathrm{T}_i$$

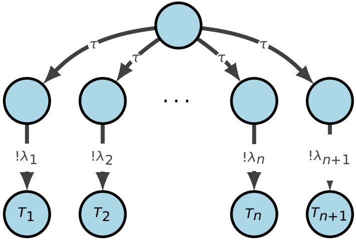
External choice between inputs:



$$\sum_{i \in I} p_i?\lambda_i;\mathrm{T}_i$$

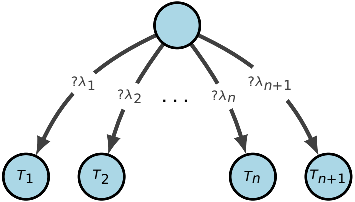
Restricting to Session Calculi

Internal choice between outputs:



$$\bigoplus_{i \in I} p_i!\lambda_i; \mathrm{T}_i$$

External choice between inputs:

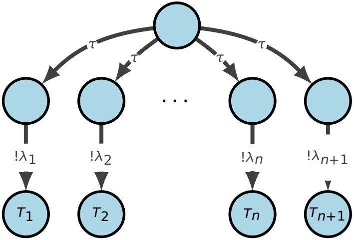


$$\sum_{i \in I} p_i?\lambda_i; \mathrm{T}_i$$

Plus guarded recursion.

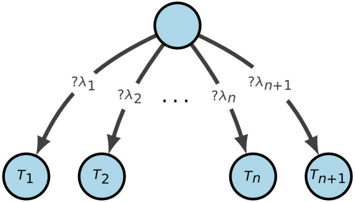
Restricting to Session Calculi

Internal choice between outputs:



$$\bigoplus_{i \in I} p_i!\lambda_i; \mathrm{T}_i$$

External choice between inputs:

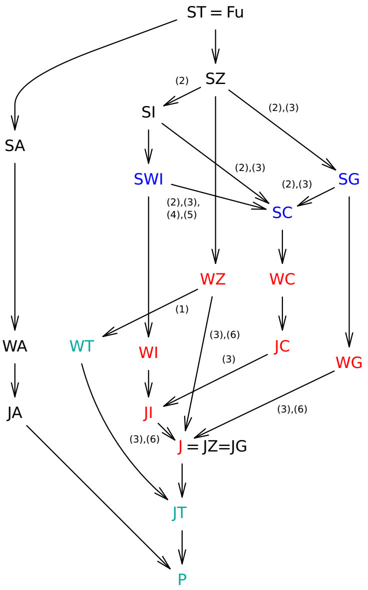


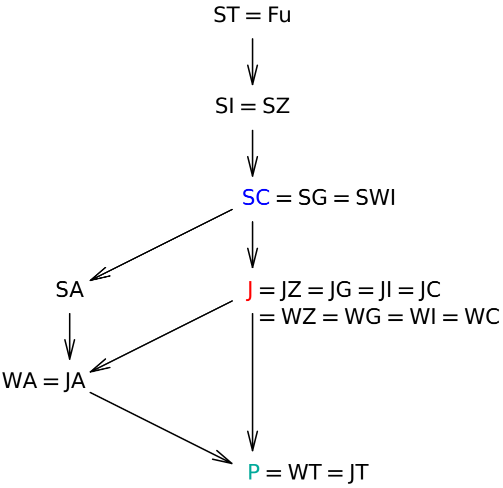
$$\sum_{i \in I} p_i?\lambda_i; \mathrm{T}_i$$

Plus guarded recursion.

$$\begin{array}{ll}
 & ClientA[\![\mu X.ServiceA!a; X]\!] \\
\parallel & ServiceA[\![\mu X.Supplier?c; ClientA?a; X]\!] \\
\parallel & Supplier[\![\mu X.(ServiceA!c; X \oplus ServiceB!d; X)]\!] \\
\parallel & ServiceB[\![\mu X.Supplier?d; ClientB?b; X]\!] \\
\parallel & ClientB[\![\mu X.ServiceB!b; X]\!]
\end{array}$$

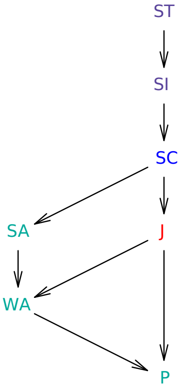
Notions of Fairness for a Synchronous Session Calculus





Lock-freedom for a Synchronous Session Calculus

Lock-freedom ($\mathcal{L}(\mathcal{F})$): Along any $\mathcal{F}$-fair path, if a component has not successfully terminated, then it must eventually act.



deadlock-freedom

$$\Uparrow$$

$$\mathcal{L}(\mathsf{SI}) = \mathcal{L}(\mathsf{ST})$$

$$\Uparrow$$

$$\mathcal{L}(\mathsf{SC})$$

$$\Uparrow$$

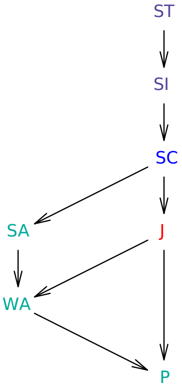
$$\mathcal{L}(\mathsf{J})$$

$$\Uparrow$$

$$\mathcal{L}(\mathsf{P}) = \mathcal{L}(\mathsf{SA}) = \mathcal{L}(\mathsf{WA})$$

Lock-freedom for a Synchronous Session Calculus

Lock-freedom ($\mathcal{L}(\mathcal{F})$): Along any $\mathcal{F}$-fair path, if a component has not successfully terminated, then it must eventually act.



deadlock-freedom

$\Uparrow$

$\mathcal{L}(\mathsf{SI}) = \mathcal{L}(\mathsf{ST})$

$\Uparrow$

$\mathcal{L}(\mathsf{SC})$

$\Uparrow$

$\mathcal{L}(\mathsf{J})$

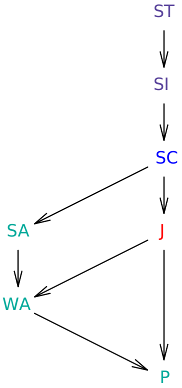
$\Uparrow$

$\mathcal{L}(\mathsf{P}) = \mathcal{L}(\mathsf{SA}) = \mathcal{L}(\mathsf{WA})$

Contravariance: more satisfaction if you consider less traces.

Lock-freedom for a Synchronous Session Calculus

Lock-freedom ($\mathcal{L}(\mathcal{F})$): Along any $\mathcal{F}$-fair path, if a component has not successfully terminated, then it must eventually act.



deadlock-freedom

$$\Uparrow$$

$$\mathcal{L}(\mathsf{SI}) = \mathcal{L}(\mathsf{ST}) = \textit{Padovani}$$

$$\Uparrow$$

$$\mathcal{L}(\mathsf{SC})$$

$$\Uparrow$$

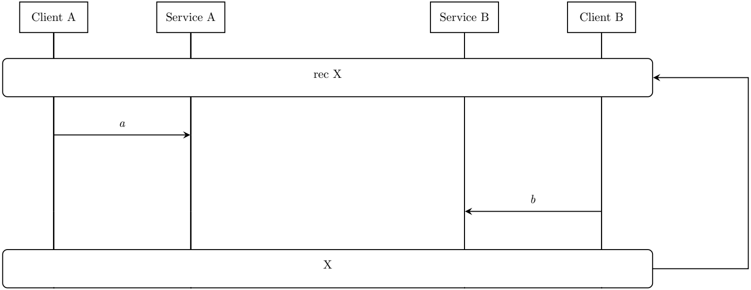
$$\mathcal{L}(\mathsf{J})$$

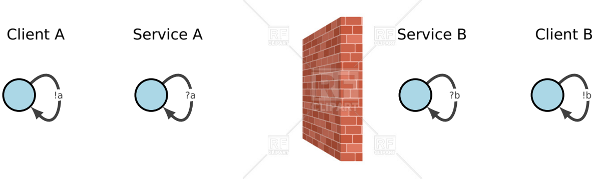
$$\Uparrow$$

$$\mathcal{L}(\mathsf{P}) = \mathcal{L}(\mathsf{SA}) = \mathcal{L}(\mathsf{WA})$$

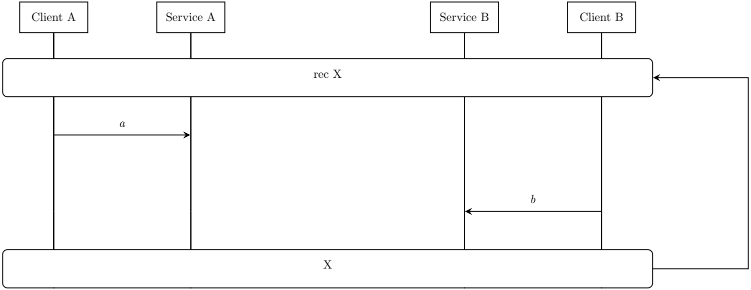
Contravariance: more satisfaction if you consider less traces.

Global session types and guarded types

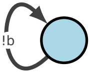




Global session types and guarded types

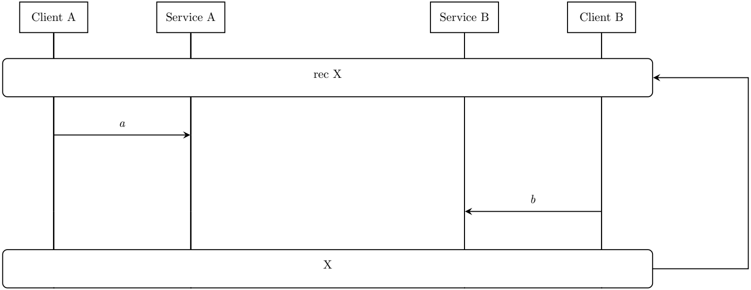


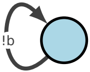
Projection of Client B:



$\vdash \quad \mu X.ServiceB!b;X$

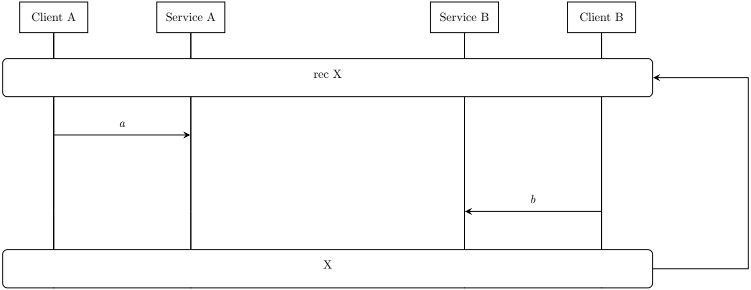
Global session types and guarded types

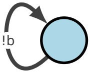


Projection of Client B: 

$\vdash \mu X.ServiceB!b;X$ Guarded!

Global session types and guarded types

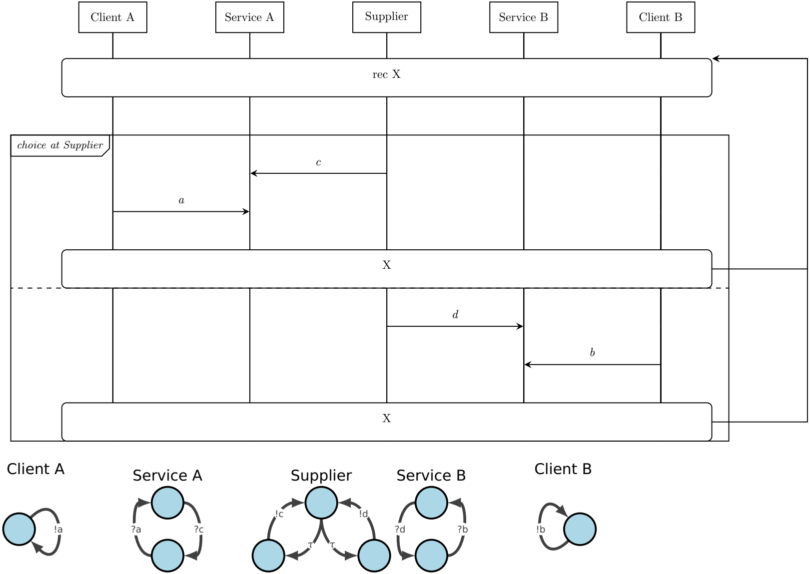


Projection of Client B: 

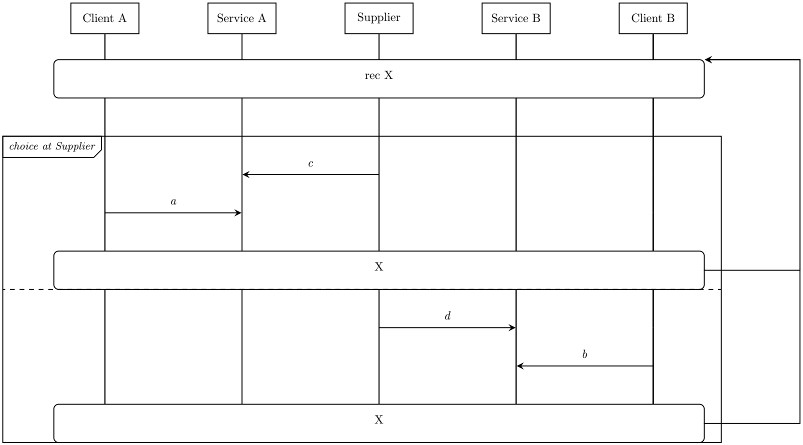
$\vdash \quad \mu X.ServiceB!b;X$ Guarded!

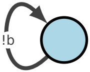
So $\mathcal{L}(P)$ is unsound with respect to typeability.

Global session types and guarded types



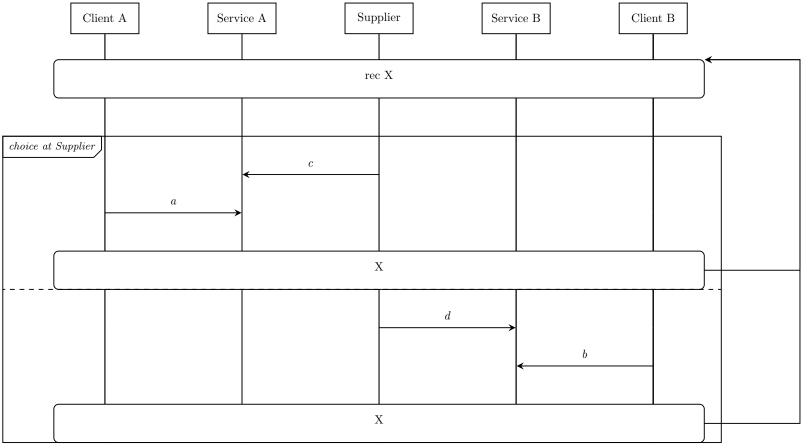
Global session types and guarded types

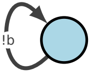


Projection of Client B: 

$\vdash \quad \mu X.(X \sqcap ServiceB!b; X)$

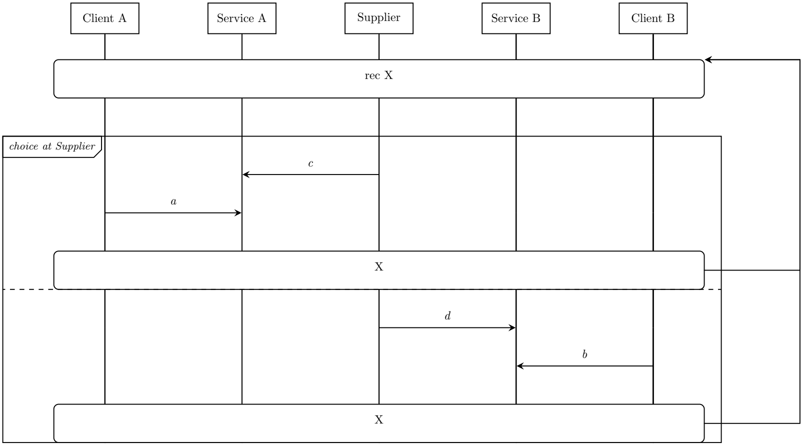
Global session types and guarded types

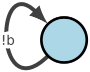


Projection of Client B: 

$\vdash \; \mu X.(X \sqcap ServiceB!b; X)$ Not Guarded!

Global session types and guarded types



Projection of Client B: 

$\vdash \quad \mu X.(X \sqcap ServiceB!b;X)$ Not Guarded!

So $\mathcal{L}$(ST) is incomplete with respect to typeability.

Soundness and Completeness for Race-free Networks

session calculus:

deadlock-freedom

$\Uparrow$

$\mathcal{L}(\mathsf{ST})$

$\Uparrow$

$\mathcal{L}(\mathsf{SC})$

$\Uparrow$

$\mathcal{L}(\mathsf{J})$

$\Uparrow$

$\mathcal{L}(\mathsf{P})$

race-free session calculus:

deadlock-freedom

$\Uparrow$

$\mathcal{L}(\mathsf{ST})$

$\Uparrow$

$\mathcal{L}(\mathsf{J}) = \mathcal{L}(\mathsf{SC})$

$\Uparrow$

$\mathcal{L}(\mathsf{P})$

Soundness and Completeness for Race-free Networks

session calculus:

deadlock-freedom

$\Uparrow$

$\mathcal{L}(\mathrm{ST})$

$\Uparrow$

$\mathcal{L}(\mathrm{SC})$

$\Uparrow$

$\mathcal{L}(\mathrm{J})$

$\Uparrow$

$\mathcal{L}(\mathrm{P})$

race-free session calculus:

deadlock-freedom

$\Uparrow$

$\mathcal{L}(\mathrm{ST})$

$\Uparrow$

$\mathcal{L}(\mathrm{J}) = \mathcal{L}(\mathrm{SC})$ = well-typed

$\Uparrow$

$\mathcal{L}(\mathrm{P})$

Theorem (soundness)

$\mathbb{N}$ *well-typed and race-free* $\quad\Rightarrow\quad \mathbb{N} \models \mathcal{L}(J)$.

Theorem (completeness)

$\mathbb{N} \models \mathcal{L}(J) \quad\Rightarrow\quad \mathbb{N}$ *well-typed.*

Completeness does not depend on race-freedom

Theorem (completeness)

$\mathbb{N} \models \mathcal{L}(J) \quad \Rightarrow \quad \mathbb{N}$ *well-typed.*

Can synthesise a global session type whenever $\mathcal{L}(\mathsf{J})$ satisfied.

Completeness does not depend on race-freedom

Theorem (completeness)

$\mathbb{N} \models \mathcal{L}(J) \quad \Rightarrow \quad \mathbb{N}$ *well-typed.*

Can synthesise a global session type whenever $\mathcal{L}(\mathrm{J})$ satisfied.

deadlock-freedom

$\Uparrow$

$\mathcal{L}(\mathrm{ST})$

$\Uparrow$

$\mathcal{L}(\mathrm{SC})$ $\quad$ well-typed

$\Uparrow \quad \Nearrow$

$\mathcal{L}(\mathrm{J})$

$\Uparrow$

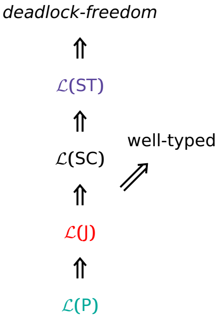
$\mathcal{L}(\mathrm{P})$

Completeness does not depend on race-freedom

Theorem (completeness)

$\mathbb{N} \models \mathcal{L}(J) \quad \Rightarrow \quad \mathbb{N}$ *well-typed.*

Can synthesise a global session type whenever $\mathcal{L}$(J) satisfied.



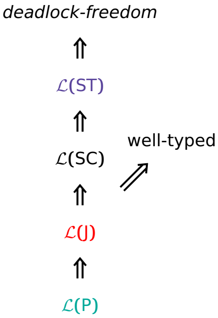
Can we strengthen such that "$\mathbb{N} \models \mathcal{L}(\mathrm{SC}) \quad \Rightarrow \quad \mathbb{N}$ well-typed" holds?

Completeness does not depend on race-freedom

Theorem (completeness)

$\mathbb{N} \models \mathcal{L}(J) \quad \Rightarrow \quad \mathbb{N}$ *well-typed.*

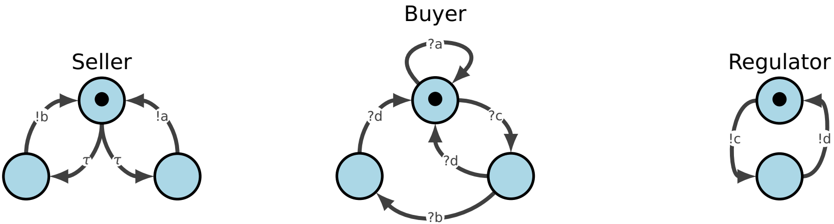
Can synthesise a global session type whenever $\mathcal{L}$(J) satisfied.



Can we strengthen such that "$\mathbb{N} \models \mathcal{L}(\mathrm{SC}) \quad \Rightarrow \quad \mathbb{N}$ well-typed" holds? ✗

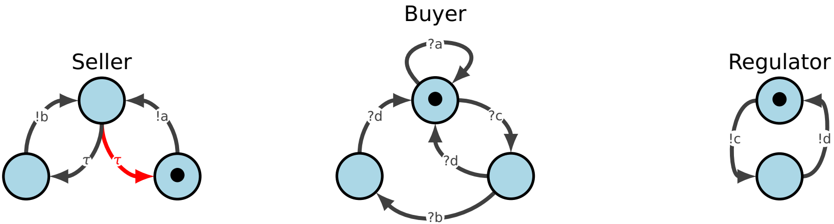
$\mathcal{L}$(SC) Incomparable to Well-Typed

$\not\models \mathcal{L}(\mathrm{J})$ $\models \mathcal{L}(\mathrm{SC})$



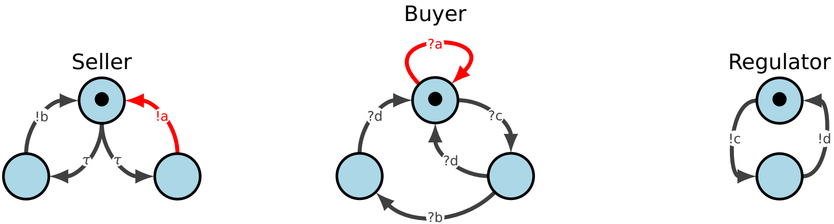
$\mathcal{L}$(SC) Incomparable to Well-Typed

$\not\models \mathcal{L}(\mathsf{J})$ $\qquad\qquad$ $\models \mathcal{L}(\mathsf{SC})$



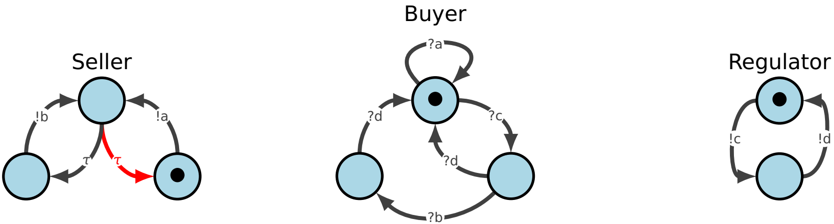
$\mathcal{L}$(SC) Incomparable to Well-Typed

$\not\models \mathcal{L}(\mathrm{J})$ $\qquad\qquad$ $\models \mathcal{L}(\mathrm{SC})$



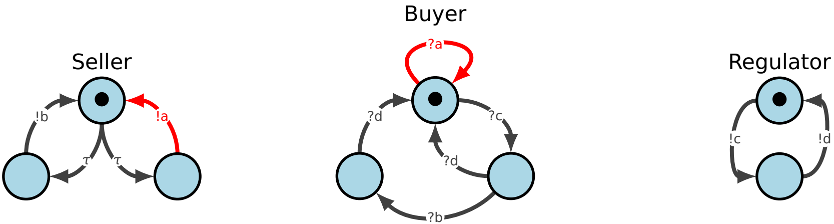
$\mathcal{L}$(SC) Incomparable to Well-Typed

$\not\models \mathcal{L}(\mathrm{J})$ $\models \mathcal{L}(\mathrm{SC})$



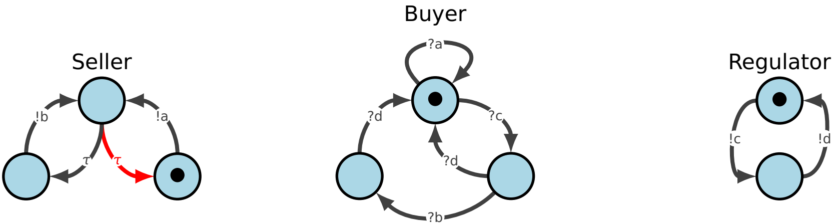
$\mathcal{L}$(SC) Incomparable to Well-Typed

$\not\models \mathcal{L}(\mathrm{J})$ $\qquad\qquad$ $\models \mathcal{L}(\mathrm{SC})$



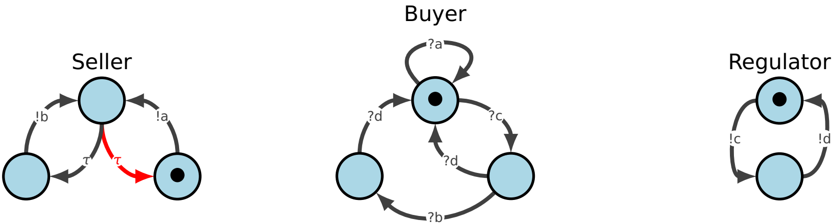
$\mathcal{L}$(SC) Incomparable to Well-Typed

$\not\models \mathcal{L}(\mathrm{J})$ $\models \mathcal{L}(\mathrm{SC})$



$\mathcal{L}$(SC) Incomparable to Well-Typed

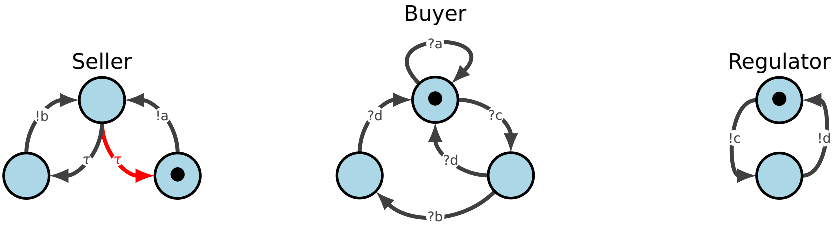
$\not\models \mathcal{L}(\mathrm{J})$ $\qquad$ $\models \mathcal{L}(\mathrm{SC})$



The network is not well typed, in line with $\mathcal{L}$(J).

$\mathcal{L}$(SC) Incomparable to Well-Typed

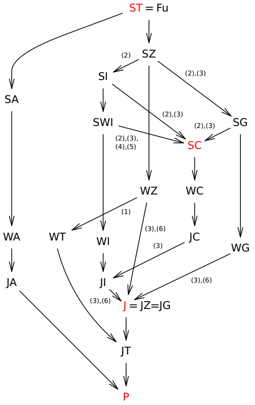
$\not\models \mathcal{L}(\mathsf{J})$ $\quad\quad$ $\models \mathcal{L}(\mathsf{SC})$



The network is not well typed, in line with $\mathcal{L}$(J).

Explanation for experts: Each global type must have a subexpression $\mathit{Seller} \to \mathit{Buyer} : a \,;\, \mathcal{G}_1 \boxplus \mathit{Seller} \to \mathit{Buyer} : b \,;\, \mathcal{G}_2$, and hence must have a reachable state $\mathbb{M}$ in which both transitions $\mathbb{M} \xrightarrow{\mathit{Seller} \to \mathit{Buyer} : a} $ and $\mathbb{M} \xrightarrow{\mathit{Seller} \to \mathit{Buyer} : b}$ are enabled. Yet there is no such reachable state.

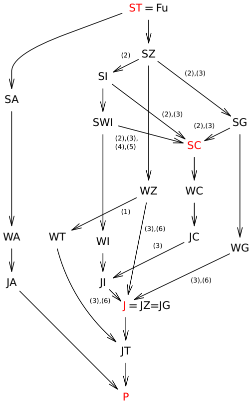
Conclusion



We considered a parametrised notion of lock-freedom and instantiated it for all established notions of fairness.

And the notion satisfying the most robust soundness and completeness properties with respect to global session types is:

Conclusion

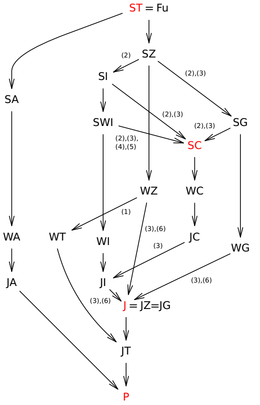


We considered a parametrised notion of lock-freedom and instantiated it for all established notions of fairness.

And the notion satisfying the most robust soundness and completeness properties with respect to global session types is:

$\mathcal{L}(\mathrm{J})$ Just Lock-Freedom

Conclusion



We considered a parametrised notion of lock-freedom and instantiated it for all established notions of fairness.

And the notion satisfying the most robust soundness and completeness properties with respect to global session types is:

$\mathcal{L}(\mathrm{J})$ Just Lock-Freedom

This is the first completeness result of it's kind for session calculi.

Session calculi look simple but proofs are non-trival and full of surprises...