

# 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<sup>1</sup>, Peter Höfner<sup>2</sup>, and Ross Horne<sup>3</sup>

1. Data61, CSIRO and UNSW, Sydney, Australia

2. Australian National University, Canberra, Australia

3. Computer Science, University of Luxembourg, Esch-sur-Alzette, Luxembourg

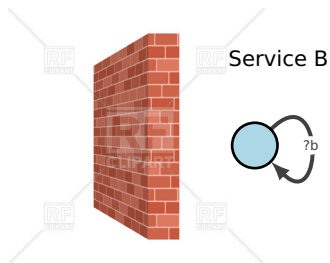
29 June – 02 July, 2021

# Fairness Assumptions and Liveness Properties

Client A



Service A



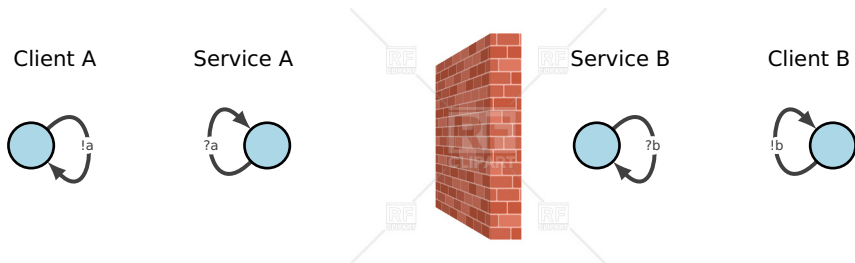
Service B



Client B

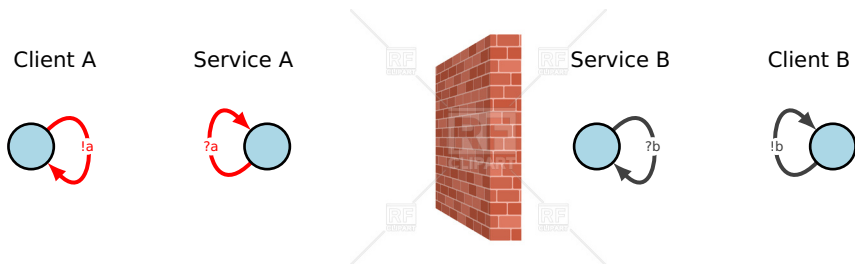


# Fairness Assumptions and Liveness Properties



*Liveness property:*  
Everyone wishing to trade eventually does so.

# Fairness Assumptions and Liveness Properties

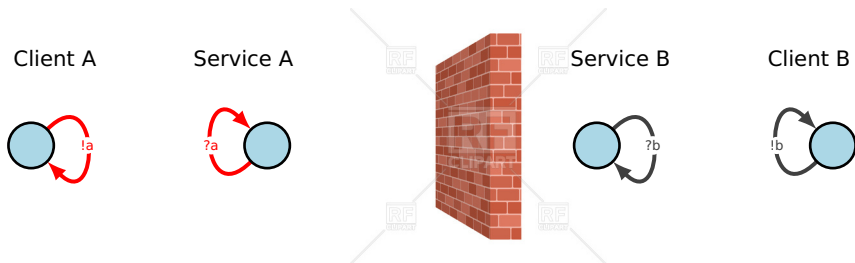


*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path:*

*Client A → Service A : a*

# Fairness Assumptions and Liveness Properties

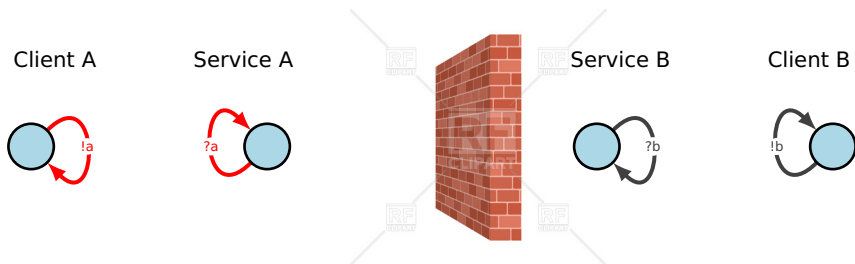


*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path:*

*Client A*  $\rightarrow$  *Service A* : *a* ; *Client A*  $\rightarrow$  *Service A* : *a*

# Fairness Assumptions and Liveness Properties

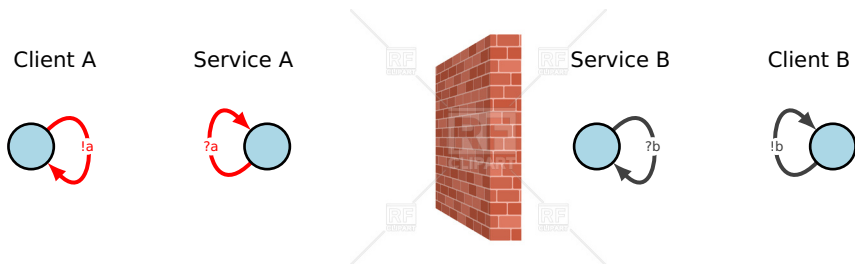


*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path:*

*Client A* → *Service A*:*a*; *Client A* → *Service A*:*a*; *Client A* → *Service A*:*a* ... **X**

# Fairness Assumptions and Liveness Properties



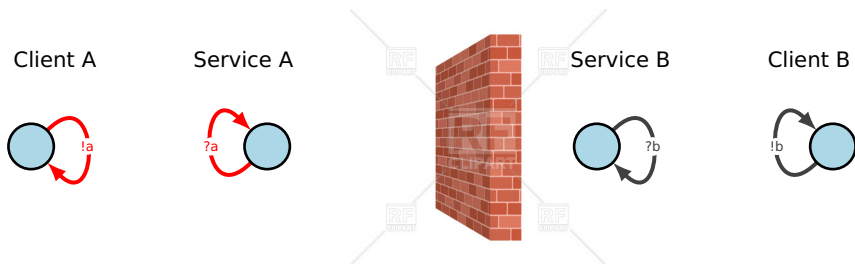
*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path:*

*Client A*  $\rightarrow$  *Service A*:a; *Client A*  $\rightarrow$  *Service A*:a; *Client A*  $\rightarrow$  *Service A*:a ...  $\times$

$\not\in \mathcal{L}(P)$

# 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 ...  $\times$

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

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

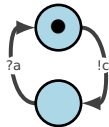


# Fairness Assumptions and Liveness Properties

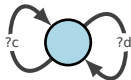
Client A



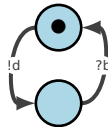
Service A



Supplier



Service B



Client B

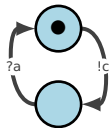


# Fairness Assumptions and Liveness Properties

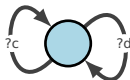
Client A



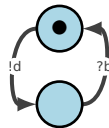
Service A



Supplier



Service B



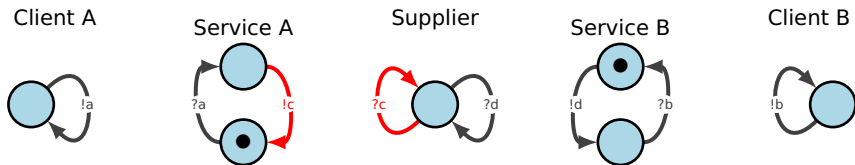
Client B



*Liveness property:*

Everyone wishing to trade eventually does so.

# Fairness Assumptions and Liveness Properties



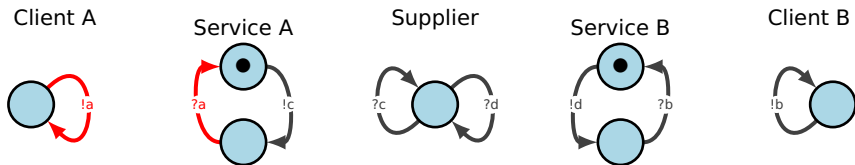
*Liveness property:*

Everyone wishing to trade eventually does so.

*A just path:*

*Service A  $\rightarrow$  Supplier: c*

# Fairness Assumptions and Liveness Properties



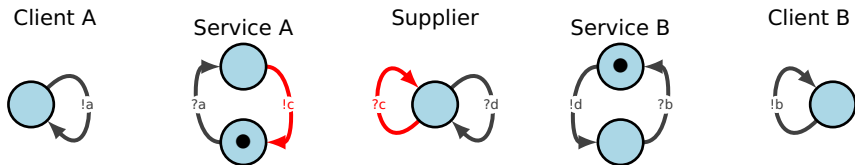
*Liveness property:*

Everyone wishing to trade eventually does so.

*A just path:*

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

# Fairness Assumptions and Liveness Properties



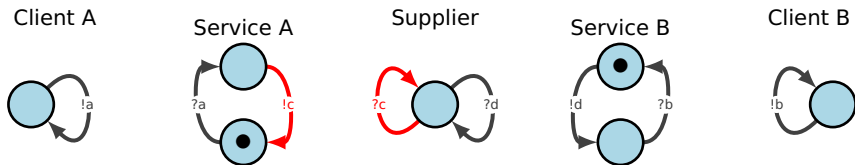
*Liveness property:*

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 ... X

# Fairness Assumptions and Liveness Properties



*Liveness property:*

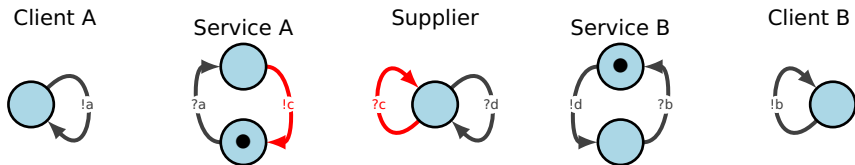
Everyone wishing to trade eventually does so.

*A just path:*

*Service A*  $\rightarrow$  *Supplier*:c ; *Client A*  $\rightarrow$  *Service A*:a ; *Service A*  $\rightarrow$  *Supplier*:c ... X

$\neq \mathcal{L}()$

# 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 ... **X**

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

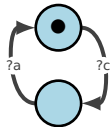
$\models \mathcal{L}(SC)$

# Fairness Assumptions and Liveness Properties

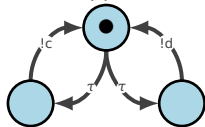
Client A



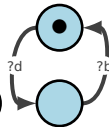
Service A



Supplier



Service B

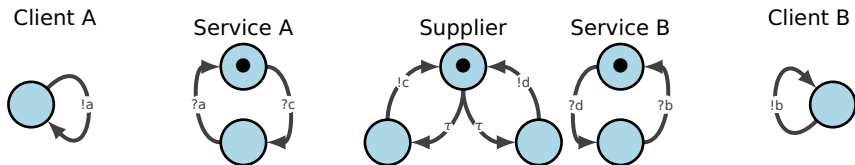


Client B





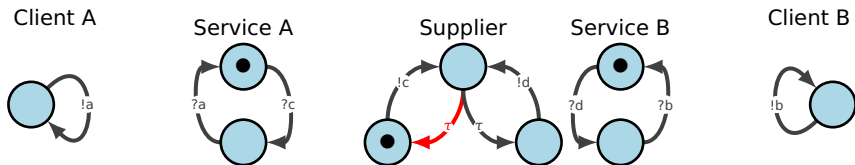
# Fairness Assumptions and Liveness Properties



*Liveness property:*

Everyone wishing to trade eventually does so.

# Fairness Assumptions and Liveness Properties

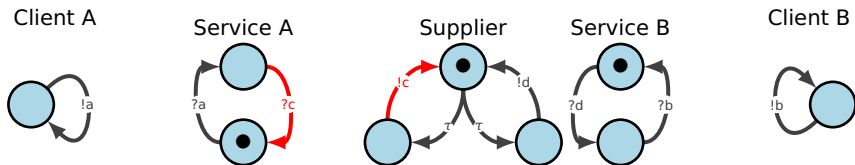


*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path where components are strongly fair:*

$\tau$

# Fairness Assumptions and Liveness Properties

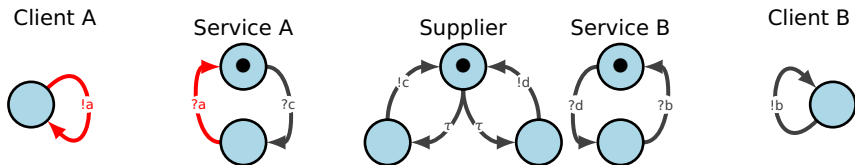


*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path where components are strongly fair:*

$\tau; \text{Service A:c}$

# Fairness Assumptions and Liveness Properties



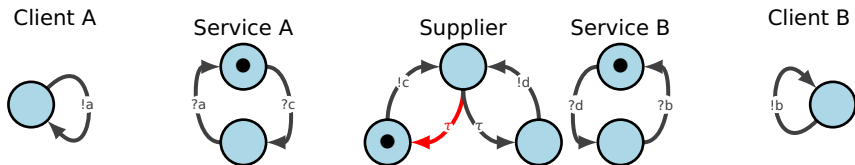
*Liveness property:*

Everyone wishing to trade eventually does so.

*A path where components are strongly fair:*

$\tau; \text{Supplier} \rightarrow \text{Service A}; c; \text{Client A} \rightarrow \text{Service A}; a$

# Fairness Assumptions and Liveness Properties



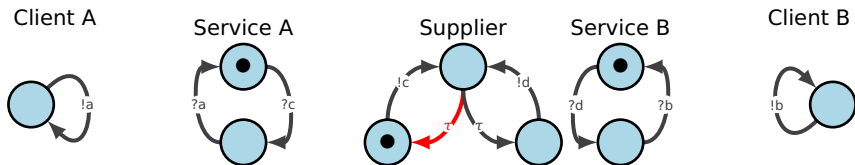
*Liveness property:*

Everyone wishing to trade eventually does so.

*A path where components are strongly fair:*

$\tau; \text{Supplier} \rightarrow \text{Service A}; c; \text{Client A} \rightarrow \text{Service A}; a; \tau \dots \times$

# Fairness Assumptions and Liveness Properties



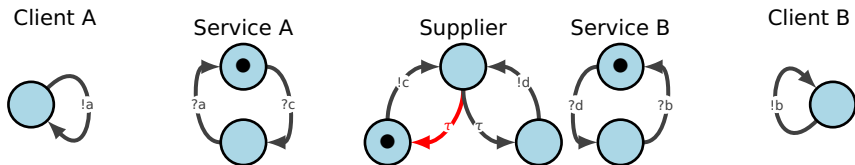
*Liveness property:*  
Everyone wishing to trade eventually does so.

*A path where components are strongly fair:*

$\tau; \text{Supplier} \rightarrow \text{Service A}; c; \text{Client A} \rightarrow \text{Service A}; a; \tau \dots \times$

$\not\in \mathcal{L}(\text{SC})$

# Fairness Assumptions and Liveness Properties



*Liveness property:*

Everyone wishing to trade eventually does so.

*A path where components are strongly fair:*

$\tau; \text{Supplier} \rightarrow \text{Service A}; c; \text{Client A} \rightarrow \text{Service A}; a; \tau \dots \times$

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

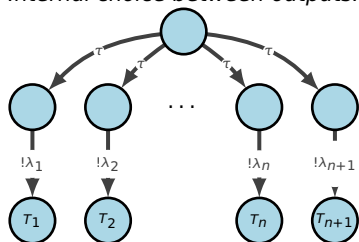
$\models \mathcal{L}(\text{ST})$





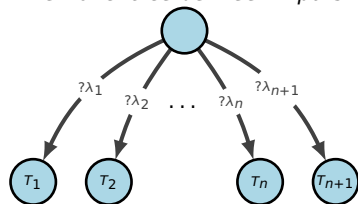
# Restricting to Session Calculi

Internal choice between outputs:



$$\bigoplus_{i \in I} \rho_i !\lambda_i ; T_i$$

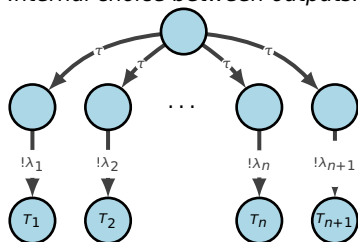
External choice between inputs:



$$\sum_{i \in I} \rho_i ?\lambda_i ; T_i$$

# Restricting to Session Calculi

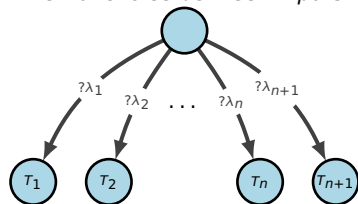
Internal choice between outputs:



$$\bigoplus_{i \in I} \rho_i !\lambda_i ; T_i$$

Plus guarded recursion.

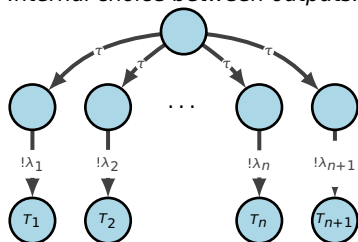
External choice between inputs:



$$\sum_{i \in I} \rho_i ?\lambda_i ; T_i$$

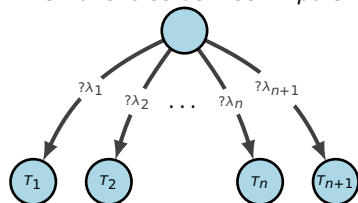
# Restricting to Session Calculi

Internal choice between outputs:



$$\bigoplus_{i \in I} p_i !\lambda_i ; T_i$$

External choice between inputs:

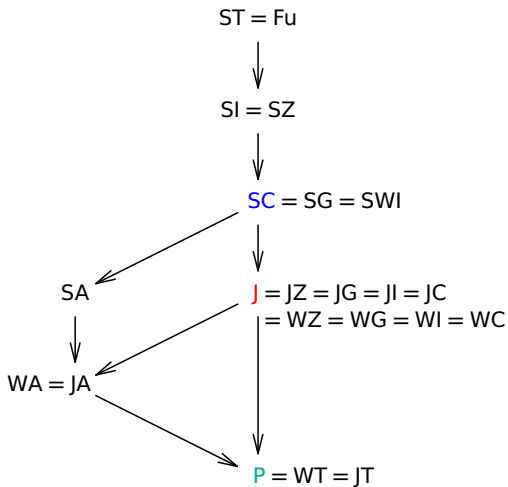
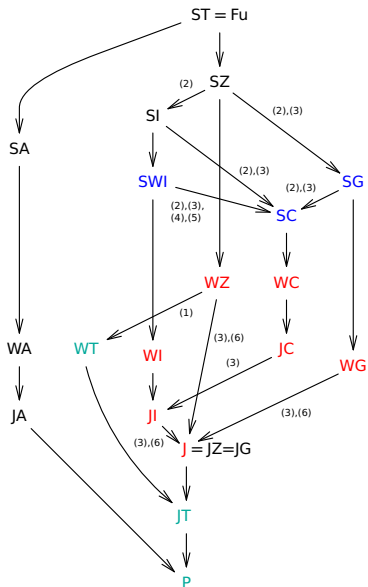


$$\sum_{i \in I} p_i ?\lambda_i ; T_i$$

Plus guarded recursion.

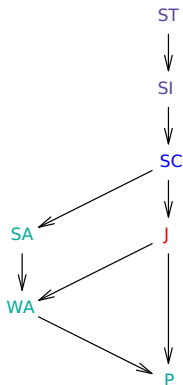
```
ClientA [μX. ServiceA!a; X]
|| ServiceA [μX. Supplier?c; ClientA?a; X]
|| Supplier [μX. (ServiceA!c; X ⊕ ServiceB!d; X)]
|| ServiceB [μX. Supplier?d; ClientB?b; X]
|| ClientB [μX. ServiceB!b; X]
```

# 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*



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



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



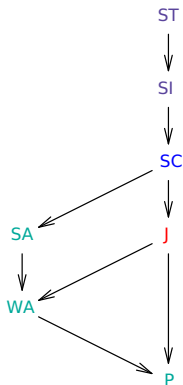
$$\mathcal{L}(\text{J})$$



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

# 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*



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



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



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

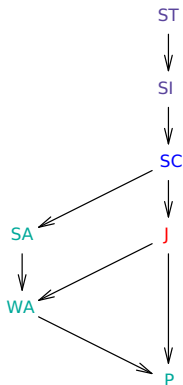


$$\mathcal{L}(\text{P}) = \mathcal{L}(\text{SA}) = \mathcal{L}(\text{WA})$$

*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*



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



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



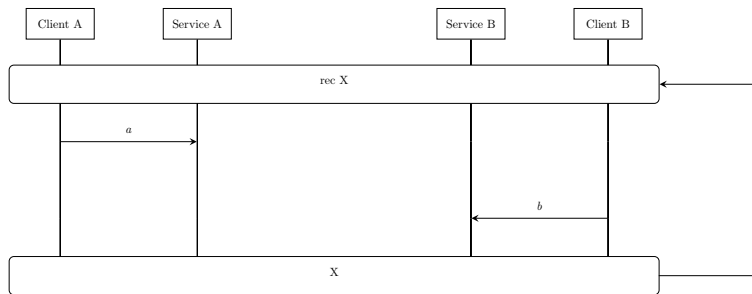
$$\mathcal{L}(\text{J})$$



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

*Contravariance*: more satisfaction if you consider less traces.

# Global session types and guarded types



Client A



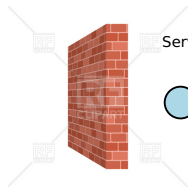
Service A



Service B

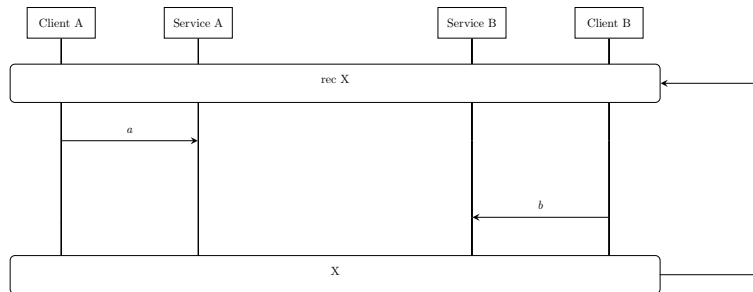



Client B



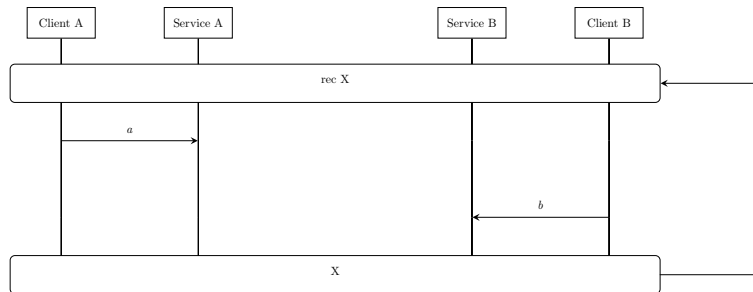


# Global session types and guarded types



Projection of Client B:   $\vdash \mu X. \text{ServiceB}!b; X$

# Global session types and guarded types



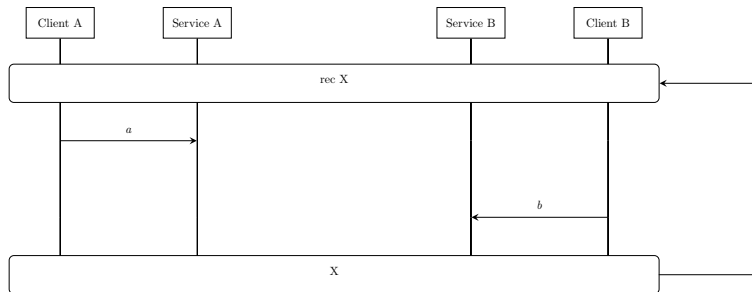
Projection of Client B:




$\vdash \mu X. \text{ServiceB}!b; X$

Guarded!

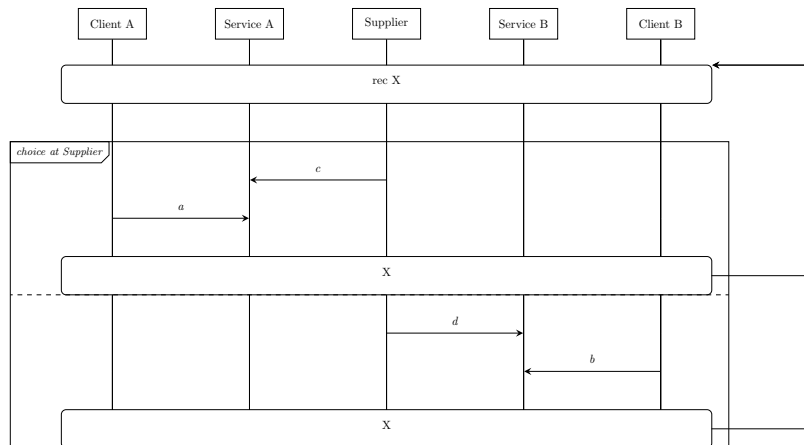
# Global session types and guarded types



Projection of Client B:   $\vdash \mu X. \text{ServiceB}!b; X$  **Guarded!**

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

# Global session types and guarded types



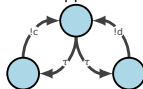
Client A



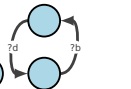
Service A



Supplier



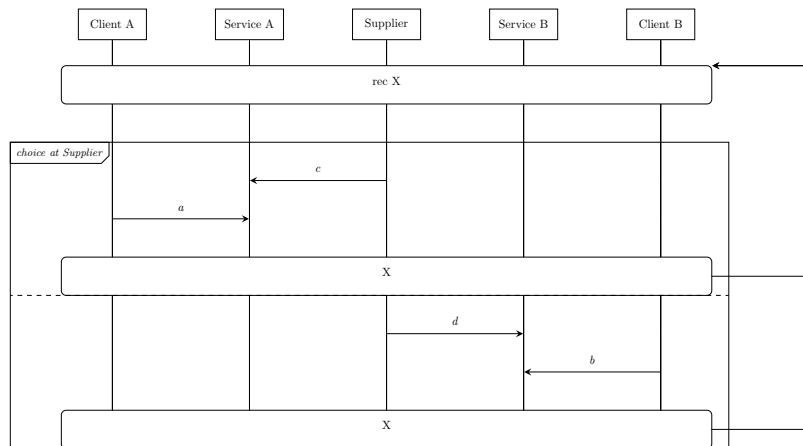
Service B



Client B



# Global session types and guarded types

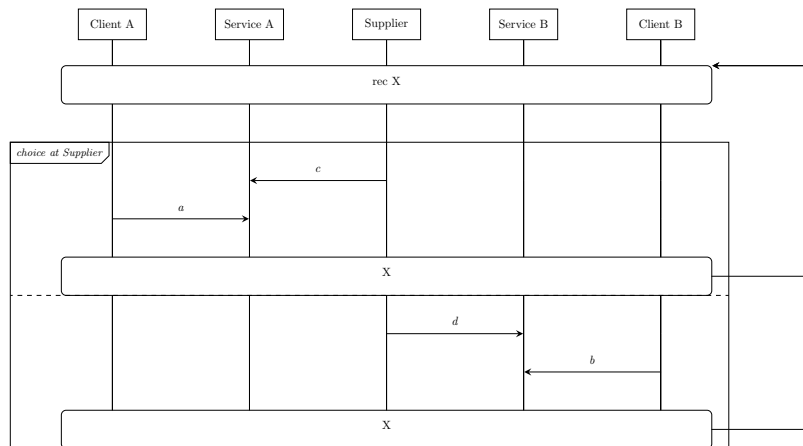


Projection of Client B:



$\vdash \mu X.(X \sqcap \text{ServiceB}!b; X)$

# Global session types and guarded types

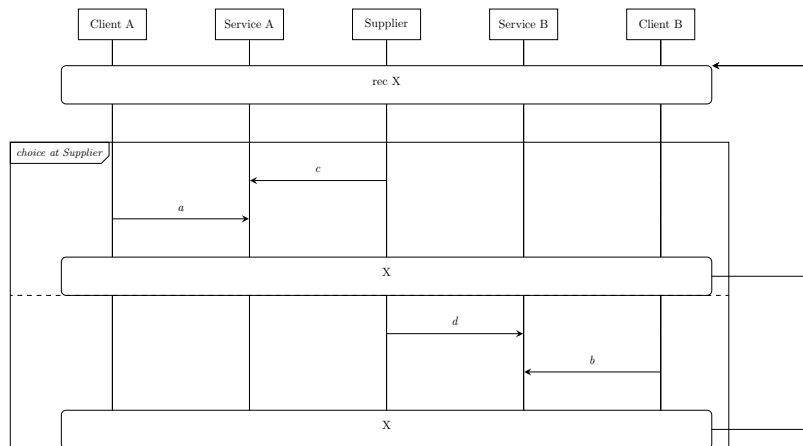


Projection of Client B:



$\vdash \mu X.(X \sqcap \text{ServiceB}!b; X)$  **Not Guarded!**

# Global session types and guarded types



Projection of Client B:



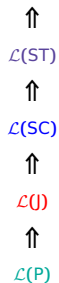
$\vdash \mu X.(X \sqcap \text{ServiceB}!b; X)$  **Not Guarded!**

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

# Soundness and Completeness for Race-free Networks

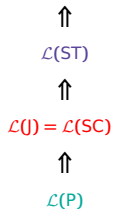
*session calculus:*

*deadlock-freedom*



*race-free session calculus:*

*deadlock-freedom*

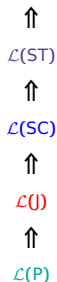




# Soundness and Completeness for Race-free Networks

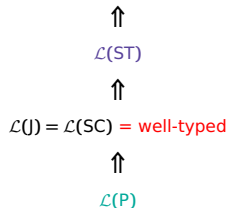
*session calculus:*

*deadlock-freedom*



*race-free session calculus:*

*deadlock-freedom*



## Theorem (soundness)

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

## Theorem (completeness)

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

# Completeness does not depend on race-freedom

## Theorem (completeness)

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

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

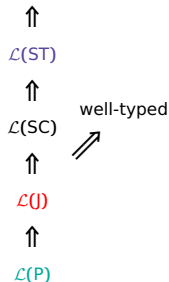
# Completeness does not depend on race-freedom

## Theorem (completeness)

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

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

*deadlock-freedom*



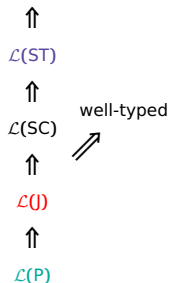
# Completeness does not depend on race-freedom

## Theorem (completeness)

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

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

*deadlock-freedom*



Can we strengthen such that “ $\mathbb{N} \models \mathcal{L}(SC) \Rightarrow \mathbb{N} \text{ well-typed}$ ” holds?

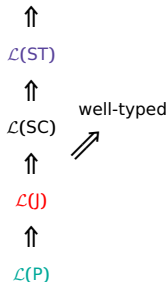
# Completeness does not depend on race-freedom

## Theorem (completeness)

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

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

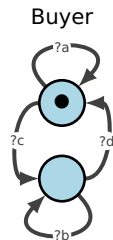
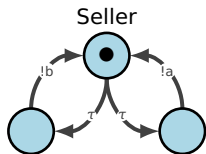
*deadlock-freedom*



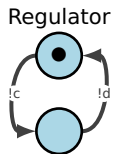
Can we strengthen such that “ $\mathbb{N} \models \mathcal{L}(SC) \Rightarrow \mathbb{N} \text{ well-typed}$ ” holds? **X**

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

$\neq \mathcal{L}(\text{U})$

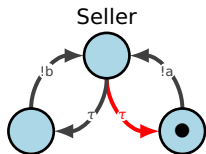


$\models \mathcal{L}(\text{SC})$

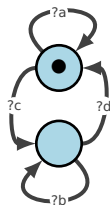


# $\mathcal{L}(\text{SC})$ Incomparable to Well-Typed

$\neq \mathcal{L}(\text{U})$

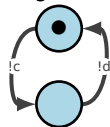


Buyer



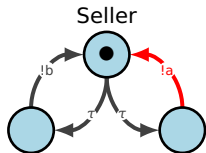
$\models \mathcal{L}(\text{SC})$

Regulator

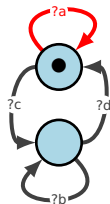


# $\mathcal{L}(\text{SC})$ Incomparable to Well-Typed

$\neq \mathcal{L}(\text{U})$

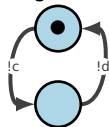


Buyer



$\models \mathcal{L}(\text{SC})$

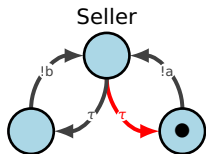
Regulator



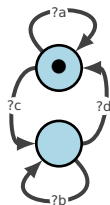


# $\mathcal{L}(\text{SC})$ Incomparable to Well-Typed

$\neq \mathcal{L}(\text{U})$

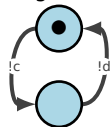


Buyer



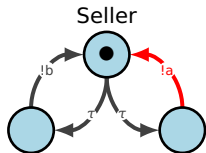
$\models \mathcal{L}(\text{SC})$

Regulator

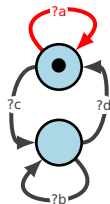


# $\mathcal{L}(\text{SC})$ Incomparable to Well-Typed

$\neq \mathcal{L}(\text{U})$

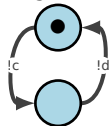


Buyer



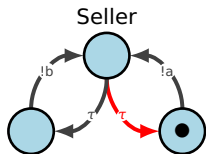
$\models \mathcal{L}(\text{SC})$

Regulator

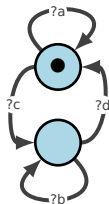


# $\mathcal{L}(\text{SC})$ Incomparable to Well-Typed

$\neq \mathcal{L}(\text{U})$

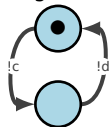


Buyer



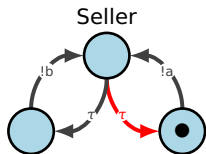
$\models \mathcal{L}(\text{SC})$

Regulator

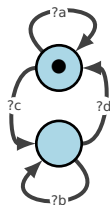


# $\mathcal{L}(\text{SC})$ Incomparable to Well-Typed

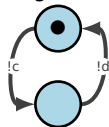
$\neq \mathcal{L}(\text{J})$



Buyer



Regulator

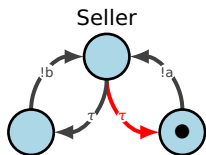


$\models \mathcal{L}(\text{SC})$

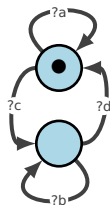
The network is not well typed, in line with  $\mathcal{L}(\text{J})$ .

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

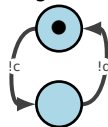
$\neq \mathcal{L}(\text{J})$



Buyer



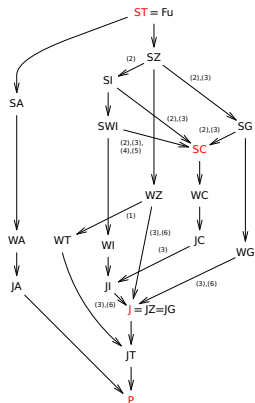
Regulator



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

*Why:* The internal choices of the Seller do not fully determine the flow of the protocol.

# Conclusion

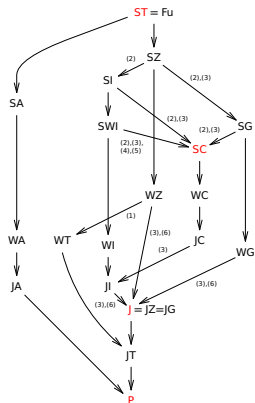


We considered a parametrised notion of lock-freedom and instantiated it for all established notions of fairness.

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



# Conclusion



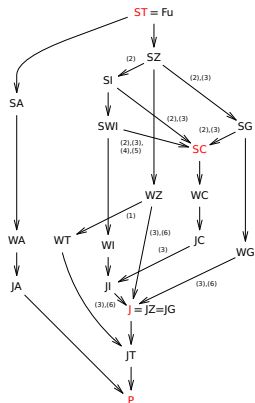
$\mathcal{L}()$

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:

# 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:

$\mathcal{L}()$

# Just Lock-Freedom

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

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

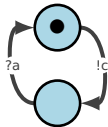


# Fairness Assumptions and Liveness Properties

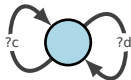
Client A



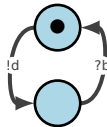
Service A



Supplier



Service B



Client B

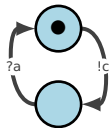


# Fairness Assumptions and Liveness Properties

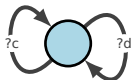
Client A



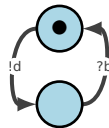
Service A



Supplier



Service B



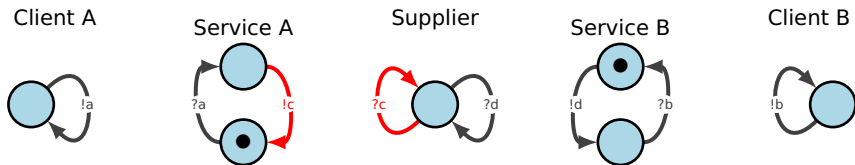
Client B



*Liveness property:*

Everyone wishing to trade eventually does so.

# Fairness Assumptions and Liveness Properties



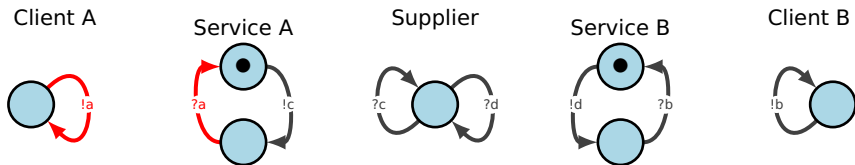
*Liveness property:*

Everyone wishing to trade eventually does so.

*A just path:*

*Service A  $\rightarrow$  Supplier: c*

# Fairness Assumptions and Liveness Properties

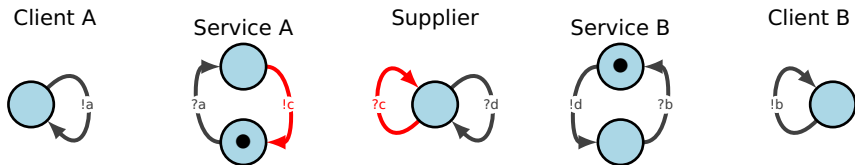


*Liveness property:*  
Everyone wishing to trade eventually does so.

*A just path:*

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

# Fairness Assumptions and Liveness Properties



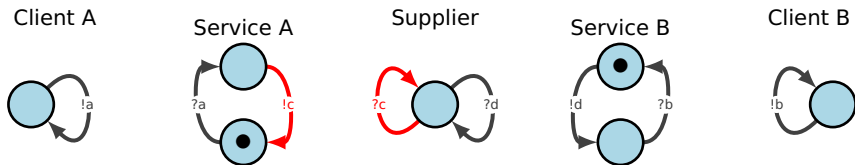
*Liveness property:*

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 ... X

# Fairness Assumptions and Liveness Properties



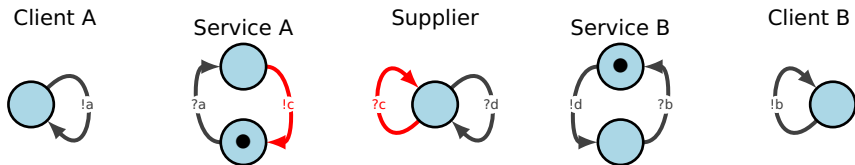
*Liveness property:*  
Everyone wishing to trade eventually does so.

*A just path:*

*Service A*  $\rightarrow$  *Supplier*:c ; *Client A*  $\rightarrow$  *Service A*:a ; *Service A*  $\rightarrow$  *Supplier*:c ... X

$\neq \mathcal{L}()$

# 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 ... X

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

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