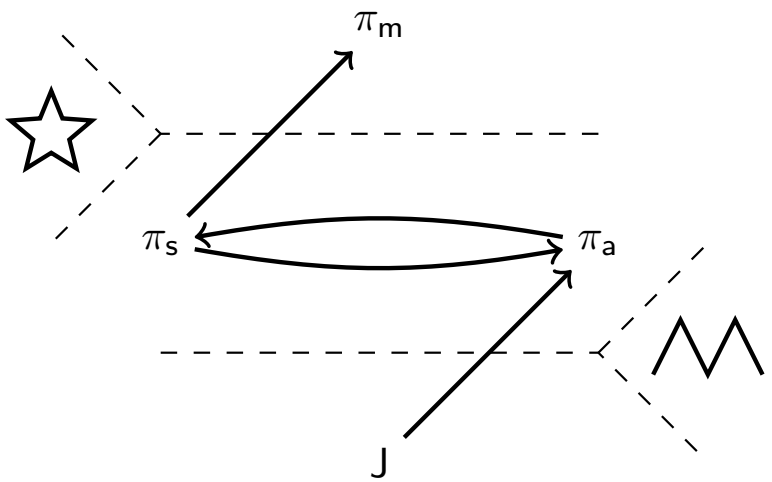
Kirstin Peters
Uwe Nestmann

"False Distribution"
Translational Expressiveness
Comparing Process Calculi Using Encodings



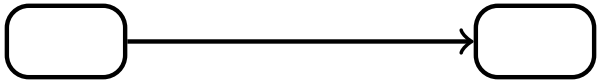
$$[\![P \mid Q]\!] \stackrel{?}{=} [\![P]\!] \mid [\![Q]\!]$$

Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)

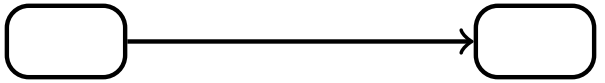
Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)

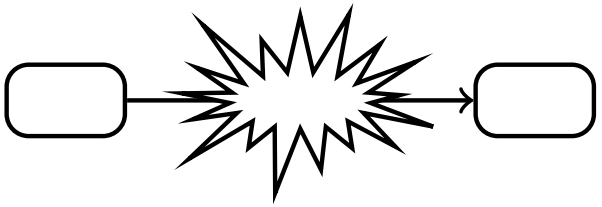


Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)



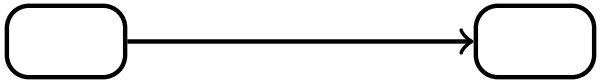
Synchronous Communication:



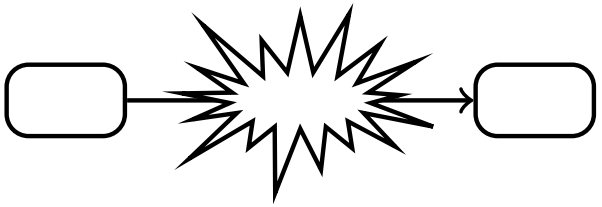
- instantaneous

Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)

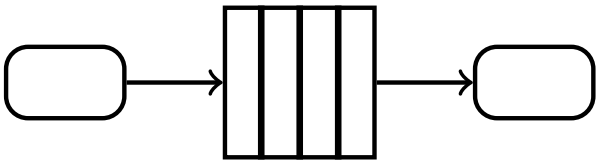


Synchronous Communication:



- instantaneous

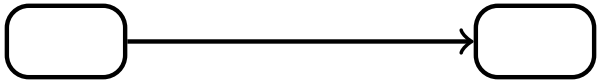
Asynchronous Communication:



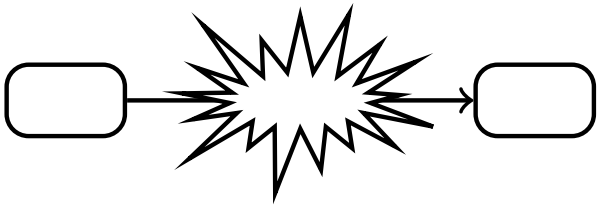
- takes time

Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)

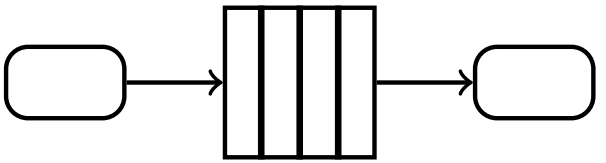


Synchronous Communication:



- instantaneous
- *abstract* specification

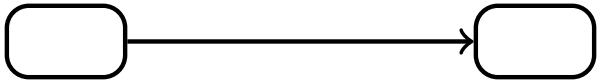
Asynchronous Communication:



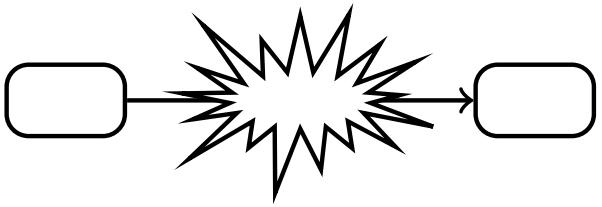
- takes time

Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)

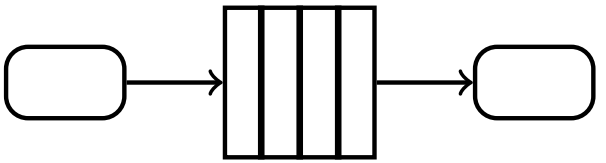


Synchronous Communication:



- instantaneous
- *abstract* specification

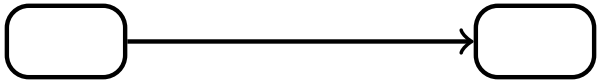
Asynchronous Communication:



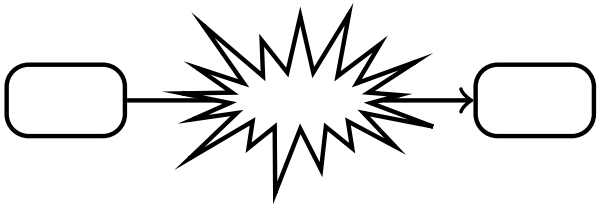
- takes time
- *concrete* implementation

Synchronous and Asynchronous Interaction in Dist. Sys.

(* joint project with TU Braunschweig [Goltz, Schicke, Glabbeek] *)

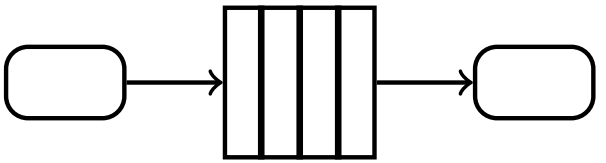


Synchronous Communication:



- instantaneous
- *abstract* specification

Asynchronous Communication:



- takes time
- *concrete* implementation

Implement asynchrony in synchrony while preserving distributability?

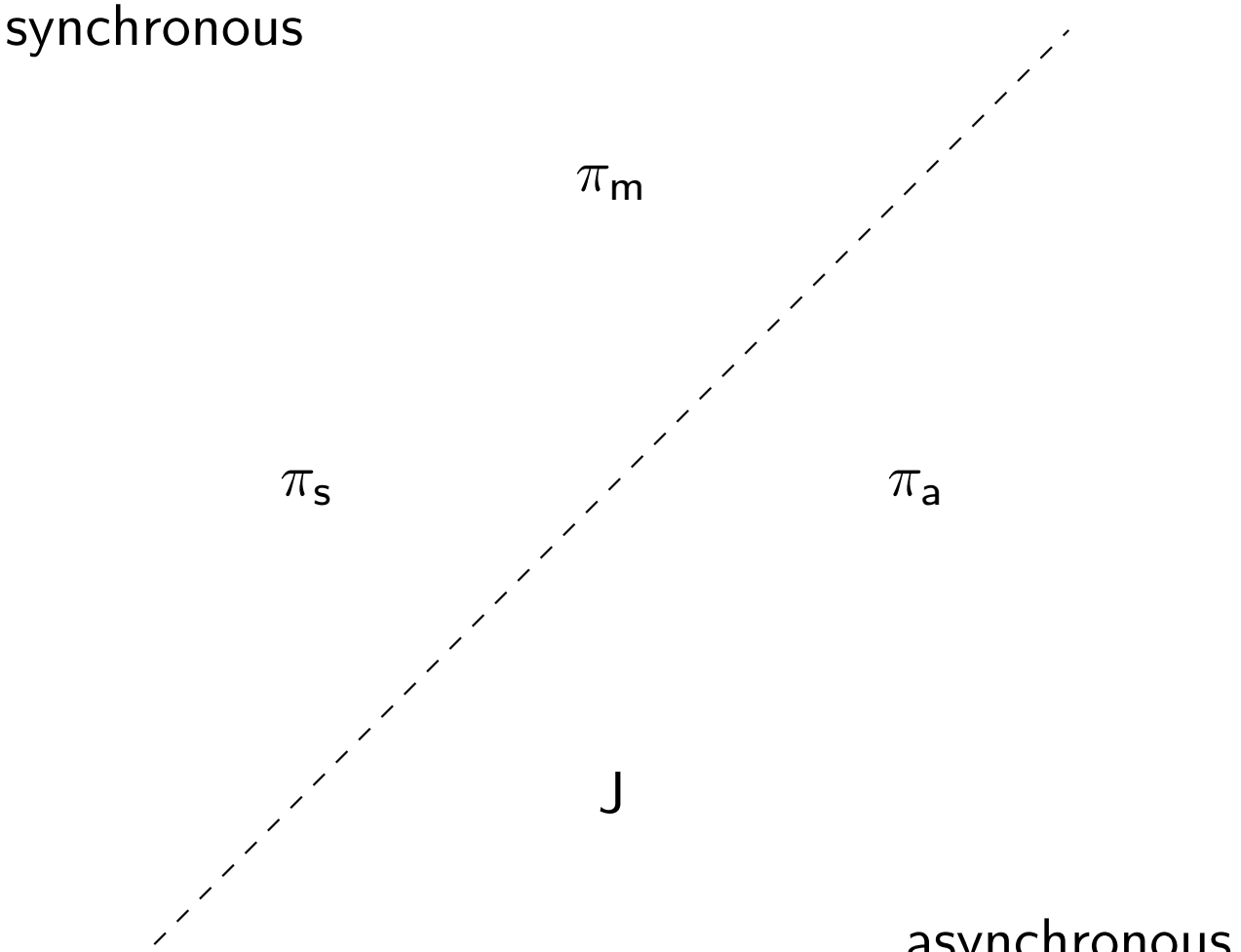
Distributability in Pi-like Calculi

π_m

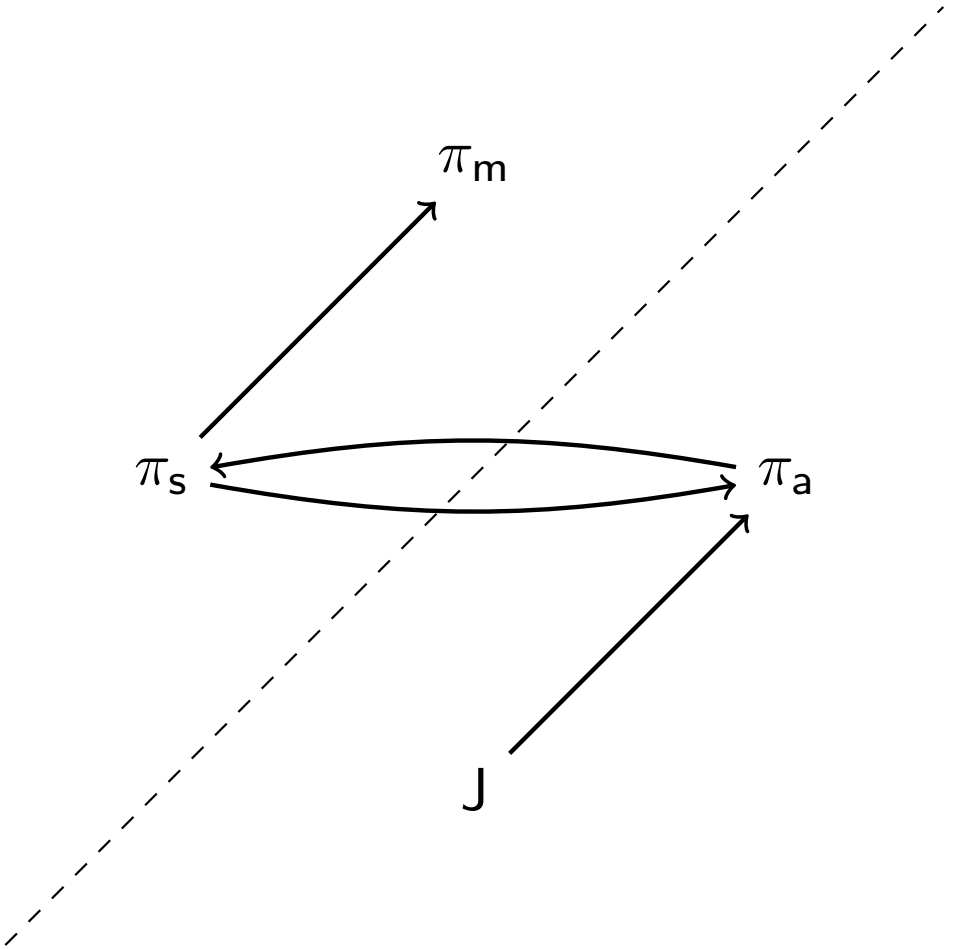
π_s π_a

J

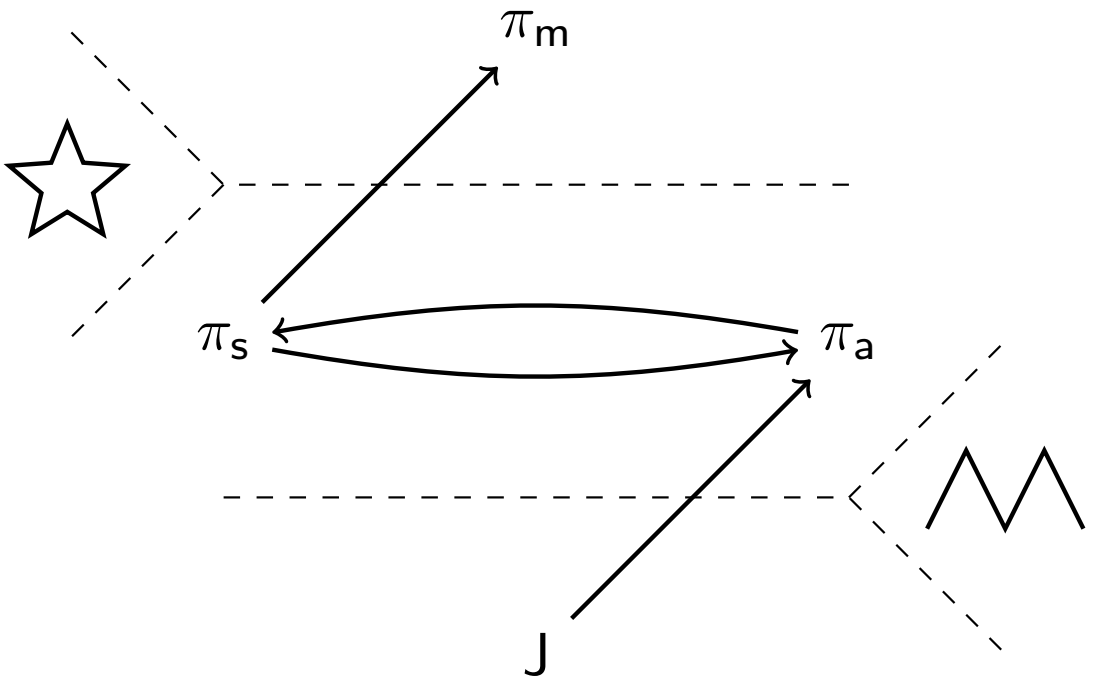
Distributability in Pi-like Calculi



Distributability in Pi-like Calculi



Distributability in Pi-like Calculi



Process Calculi—Variants of the Pi-Calculus

$$\mathcal{L} = \langle\, \mathcal{P}, \longmapsto \,\rangle$$

Process Calculi—Variants of the Pi-Calculus

$$\mathcal{L} = \langle\, \mathcal{P}, \longmapsto \,\rangle$$

Process Terms:

$$\mathcal{P}_{\mathsf{a}} ::= 0 \quad | \quad P_1 \mid P_2 \quad | \quad (\nu x)\, P \quad | \quad \overline{y}\langle z \rangle .0 \quad | \quad y(x) .P \quad | \quad y^{\star}(x) .P$$

Process Calculi—Variants of the Pi-Calculus

$$\mathcal{L} = \langle\, \mathcal{P}, \longmapsto \,\rangle$$

Process Terms:

$$\mathcal{P}_{\mathsf{a}} ::= 0 \quad | \quad P_1 \mid P_2 \quad | \quad (\nu x)\, P \quad | \quad \overline{y}\langle z \rangle .0 \quad | \quad y(x) .P \quad | \quad y^{\star}(x) .P$$

$$\mathcal{P}_{\mathsf{s}} ::= \ldots \quad | \quad \overline{y}\langle z \rangle .P \quad | \quad \ldots \quad | \quad \sum_{i \in I} \overline{y_i}\langle z_i \rangle .P_i \quad | \quad \sum_{i \in I} y_i(x_i) .P_i$$

Process Calculi—Variants of the Pi-Calculus

$$\mathcal{L} = \langle\, \mathcal{P}, \longmapsto \,\rangle$$

Process Terms:

$$\mathcal{P}_{\mathsf{a}} ::= 0 \quad | \quad P_1 \mid P_2 \quad | \quad (\nu x)\, P \quad | \quad \overline{y}\langle z \rangle .0 \quad | \quad y(x)\,.P \quad | \quad y^{\star}(x)\,.P$$

$$\mathcal{P}_{\mathsf{s}} ::= \ldots \quad | \quad \overline{y}\langle z \rangle .P \quad | \quad \ldots \quad | \quad \sum_{i \in I} \overline{y_i}\langle z_i \rangle .P_i \quad | \quad \sum_{i \in I} y_i(x_i)\,.P_i$$

$$\mathcal{P}_{\mathsf{m}} ::= \ldots \quad | \quad \sum_{i \in I} \pi_i.P_i \qquad \text{where } \pi ::= \overline{y}\langle z \rangle \quad | \quad y(x)$$

Process Calculi—Variants of the Pi-Calculus

$$\mathcal{L} = \langle\, \mathcal{P}, \longmapsto \,\rangle$$

Process Terms:

$$\mathcal{P}_{\mathsf{a}} ::= 0 \quad | \quad P_1 \mid P_2 \quad | \quad (\nu x)\, P \quad | \quad \overline{y}\langle z \rangle .0 \quad | \quad y(x)\,.P \quad | \quad y^{\star}(x)\,.P$$

$$\mathcal{P}_{\mathsf{s}} ::= \ldots \quad | \quad \overline{y}\langle z \rangle .P \quad | \quad \ldots \quad | \quad \sum_{i \in I} \overline{y_i}\langle z_i \rangle .P_i \quad | \quad \sum_{i \in I} y_i(x_i)\,.P_i$$

$$\mathcal{P}_{\mathsf{m}} ::= \ldots \quad | \quad \sum_{i \in I} \pi_i.P_i \qquad \text{where } \pi ::= \overline{y}\langle z \rangle \quad | \quad y(x)$$

Reduction Semantics:

$$y(x)\,.P \mid \overline{y}\langle z \rangle .Q + a(b)\,.R$$

Process Calculi—Variants of the Pi-Calculus

$$\mathcal{L} = \langle\, \mathcal{P}, \longmapsto \,\rangle$$

Process Terms:

$$\mathcal{P}_{\mathsf{a}} ::= 0 \quad | \quad P_1 \mid P_2 \quad | \quad (\nu x)\, P \quad | \quad \overline{y}\langle z \rangle .0 \quad | \quad y(x) .P \quad | \quad y^{\star}(x) .P$$

$$\mathcal{P}_{\mathsf{s}} ::= \ldots \quad | \quad \overline{y}\langle z \rangle .P \quad | \quad \ldots \quad | \quad \sum_{i \in I} \overline{y_i}\langle z_i \rangle .P_i \quad | \quad \sum_{i \in I} y_i(x_i) .P_i$$

$$\mathcal{P}_{\mathsf{m}} ::= \ldots \quad | \quad \sum_{i \in I} \pi_i . P_i \qquad \text{where } \pi ::= \overline{y}\langle z \rangle \quad | \quad y(x)$$

Reduction Semantics:

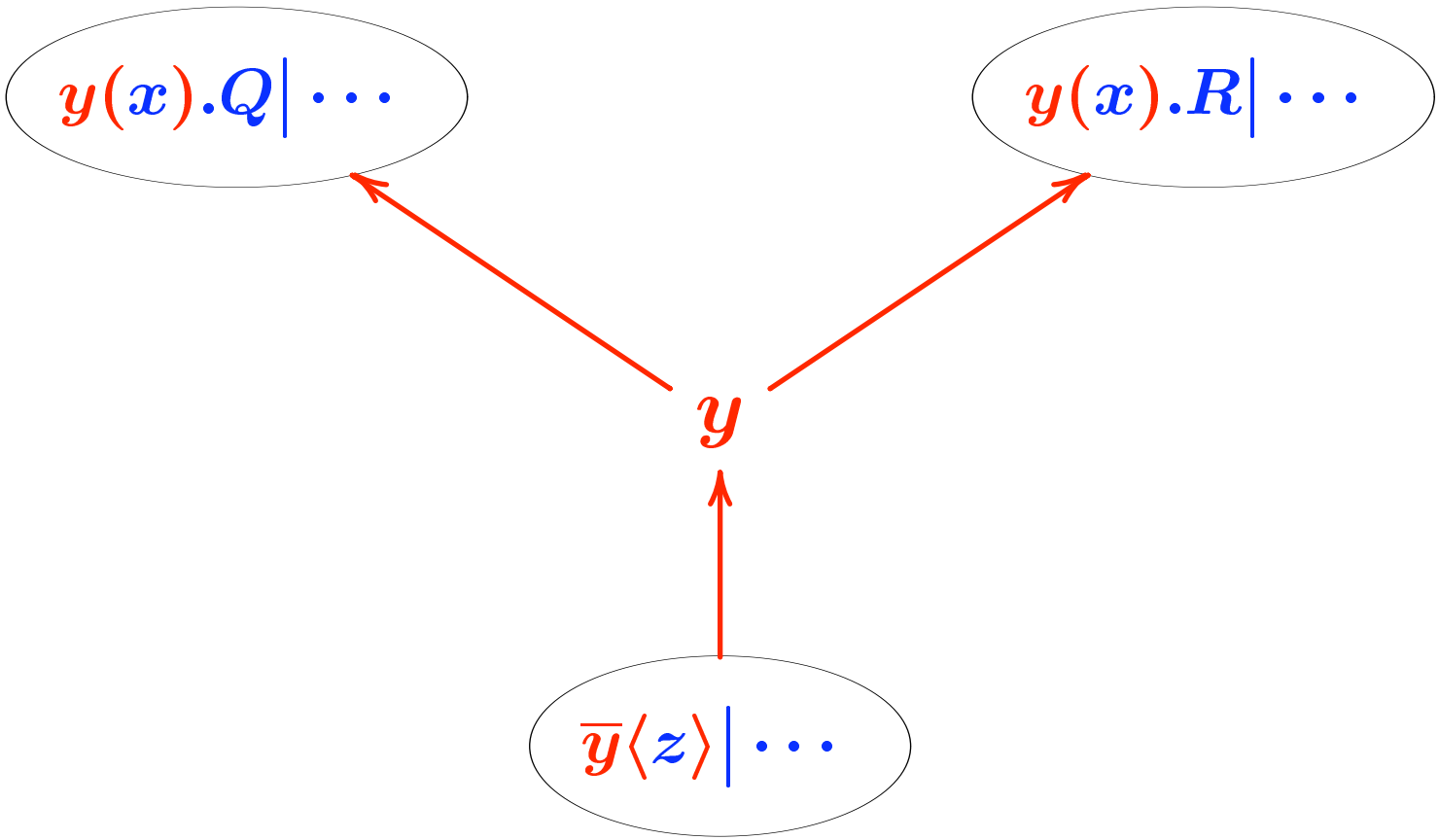
$$y(x) .P \mid \overline{y}\langle z \rangle .Q + a(b) .R \longmapsto \{\, {}^{z}/_{x} \,\}\, P \mid Q$$

Process Calculi—The Join-Calculus

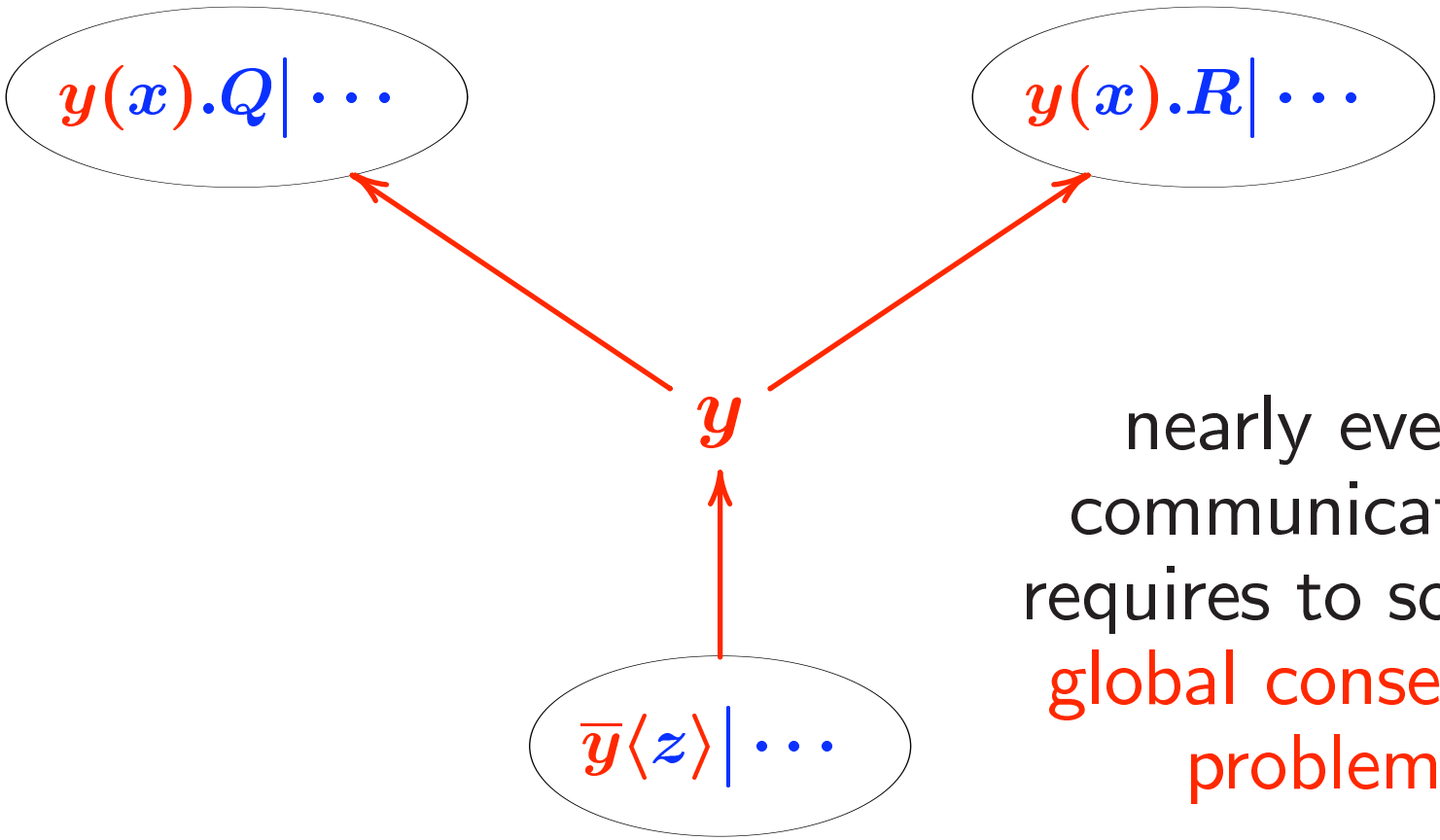
The Quest for Implementability

— Join —

Distributed Implementation

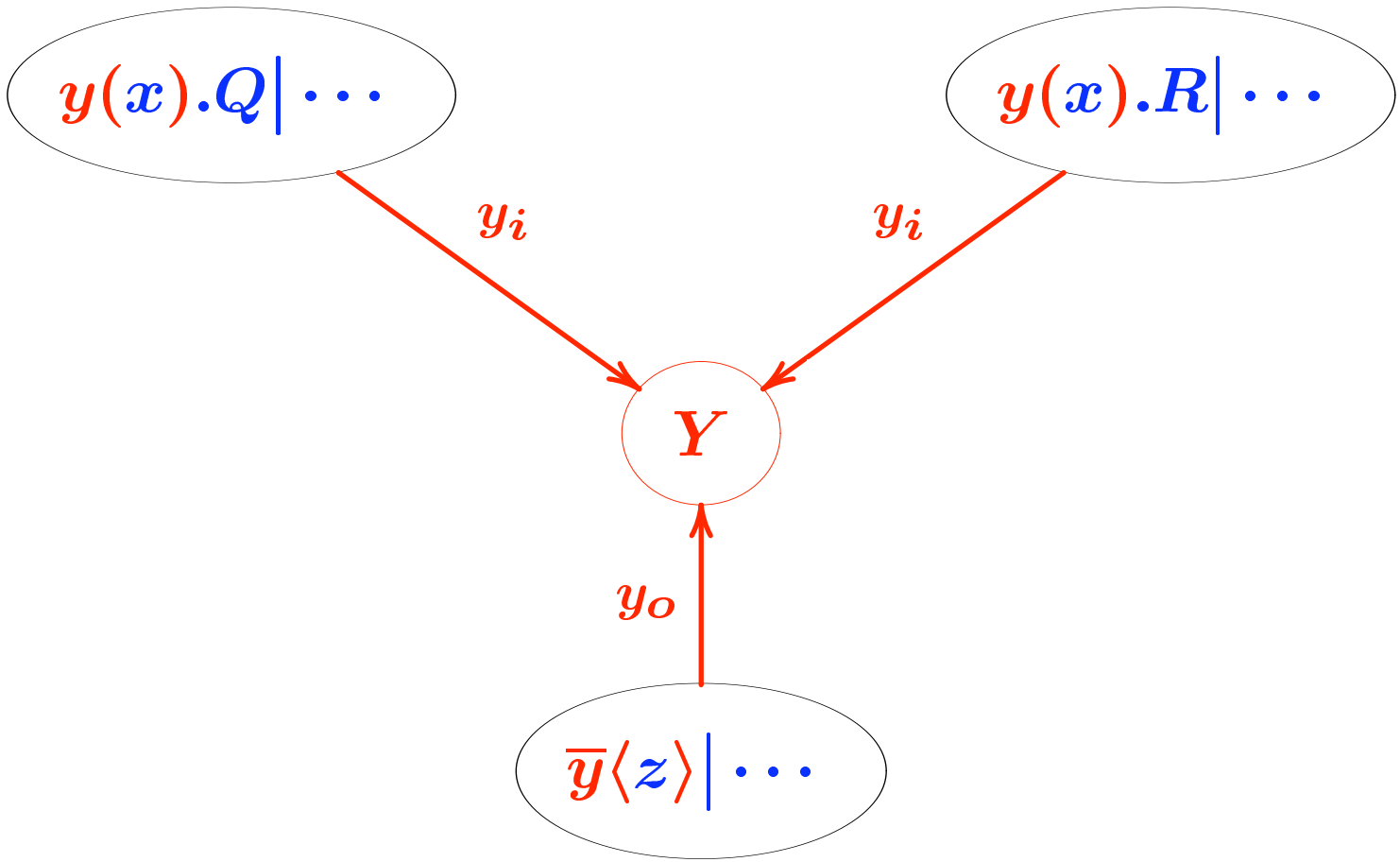


Distributed Implementation

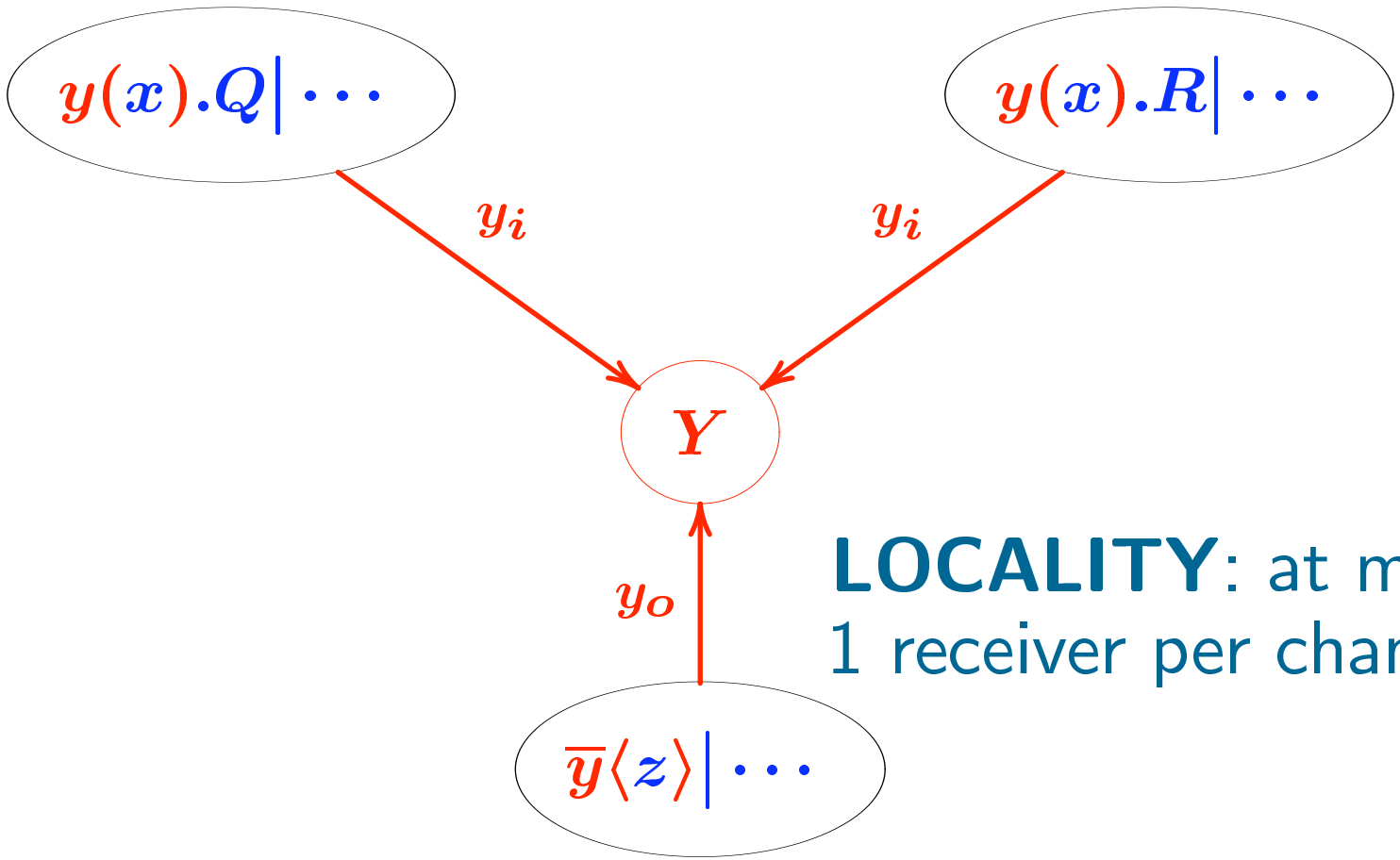


nearly every communication requires to solve a global consensus problem

Unique Channel Managers

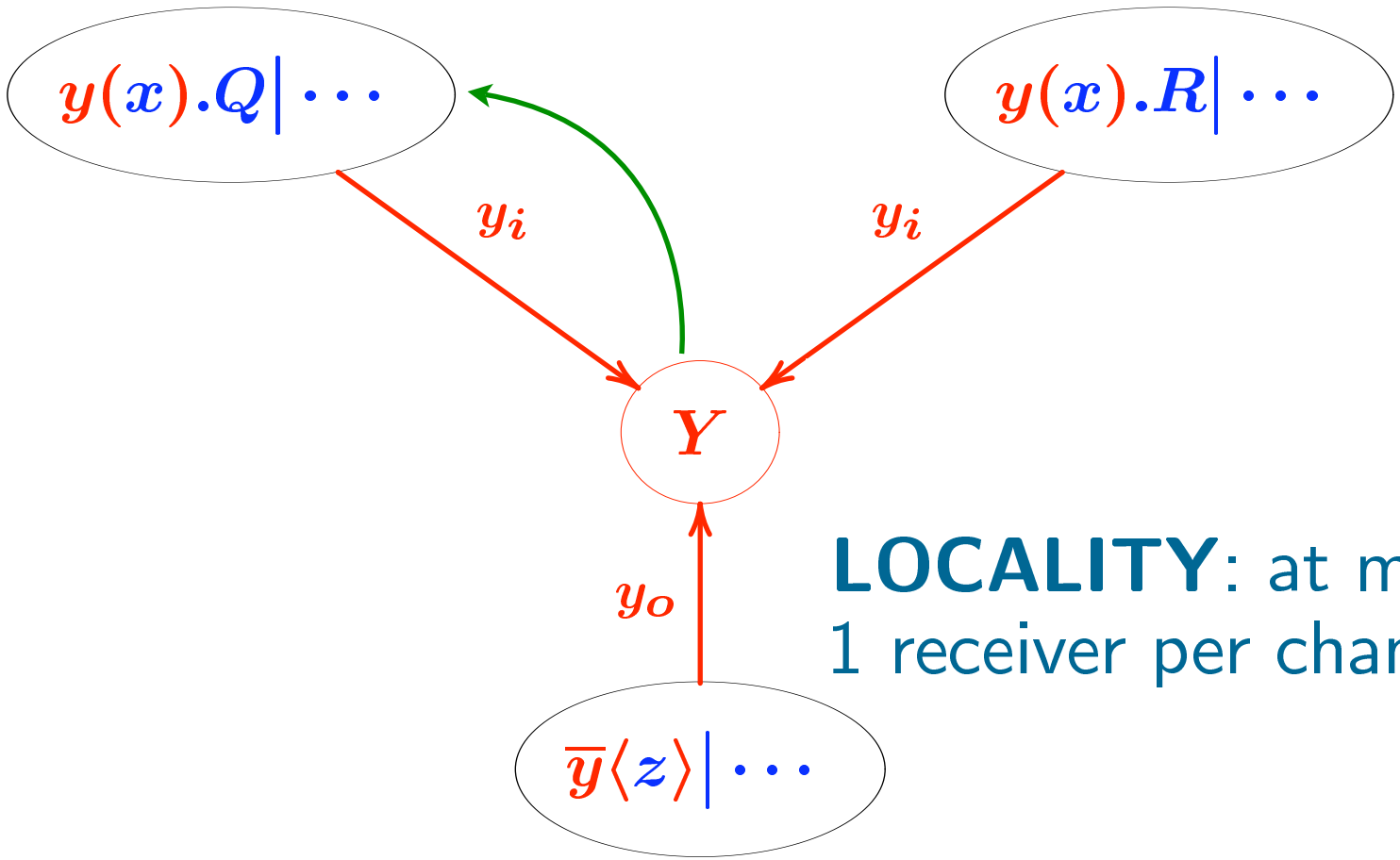


Unique Channel Managers



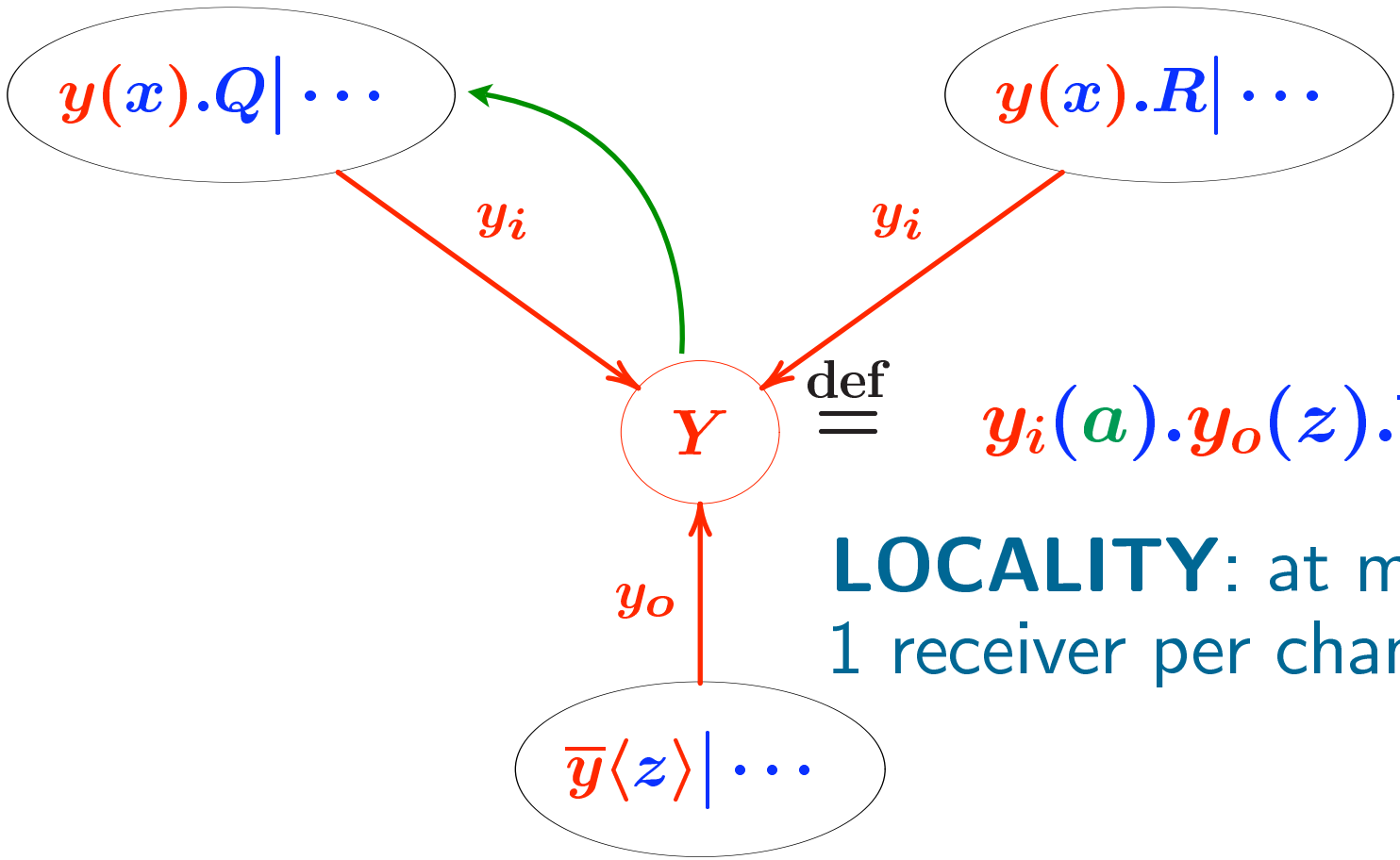
LOCALITY: at most 1 receiver per channel

Unique Channel Managers



LOCALITY: at most 1 receiver per channel

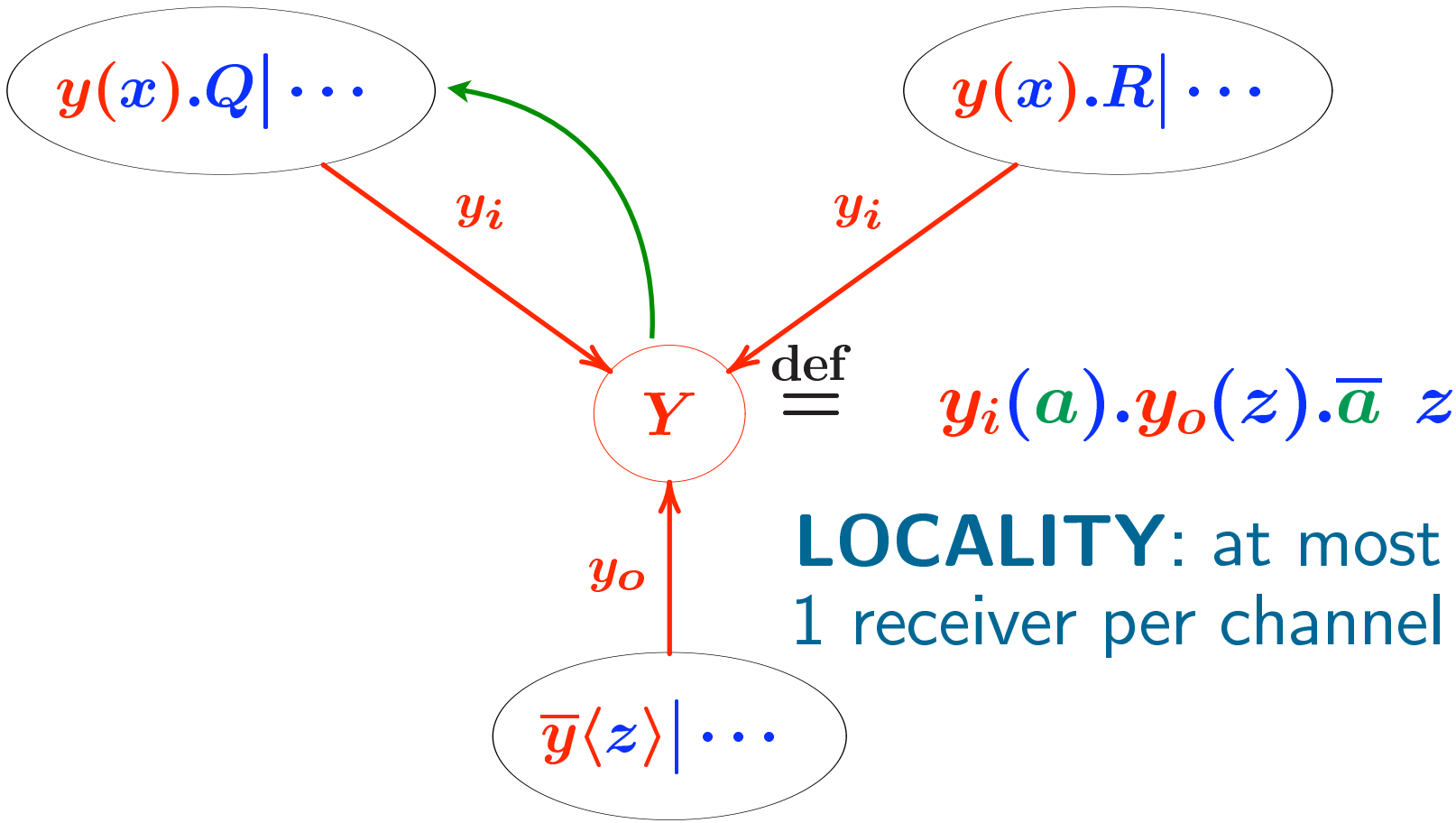
Unique Channel Managers



$y_i(a).y_o(z).\overline{a}\ z$

LOCALITY: at most
1 receiver per channel

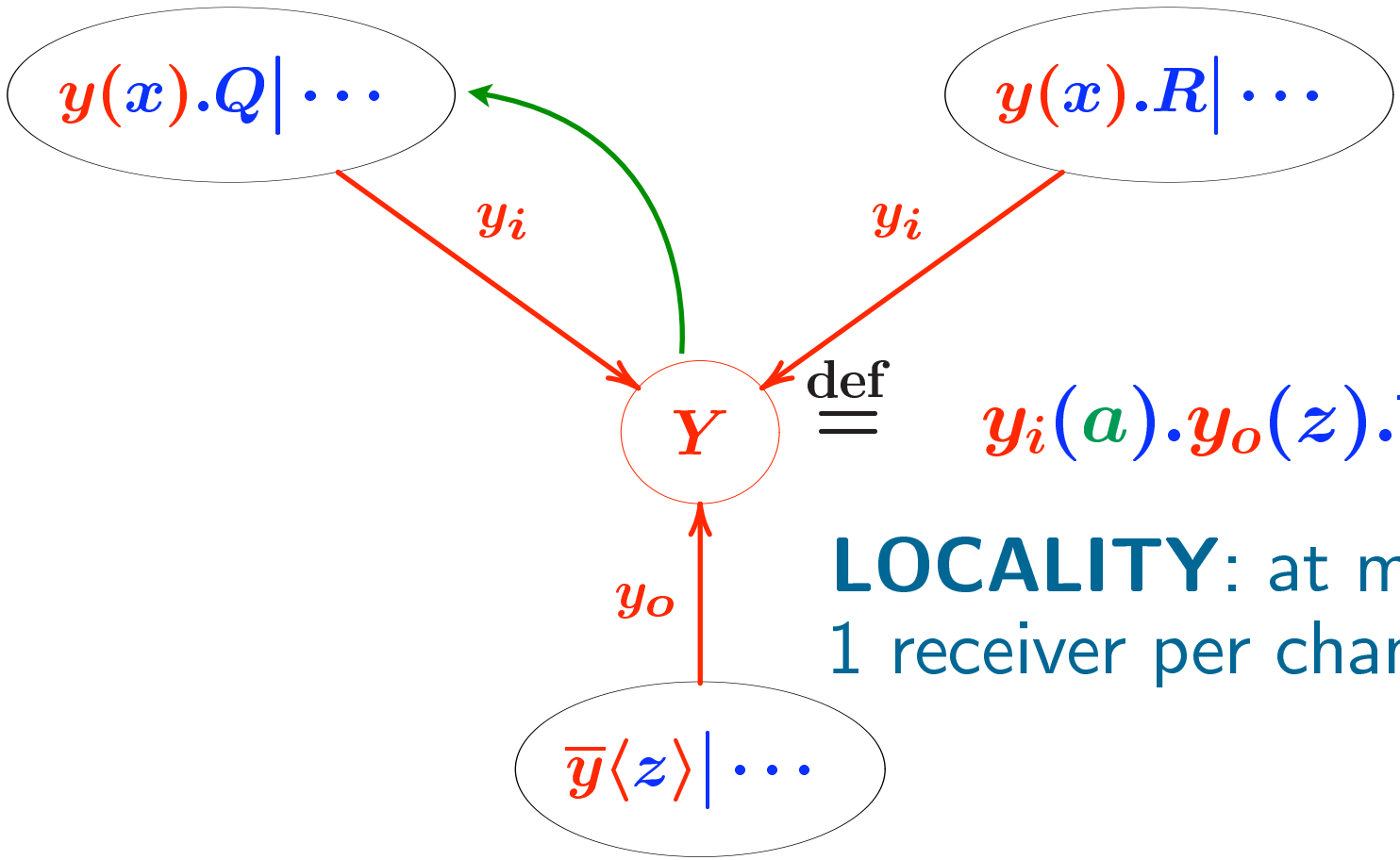
Unique Channel Managers



LOCALITY: at most
1 receiver per channel

Note: the power and convenience of name-passing ...

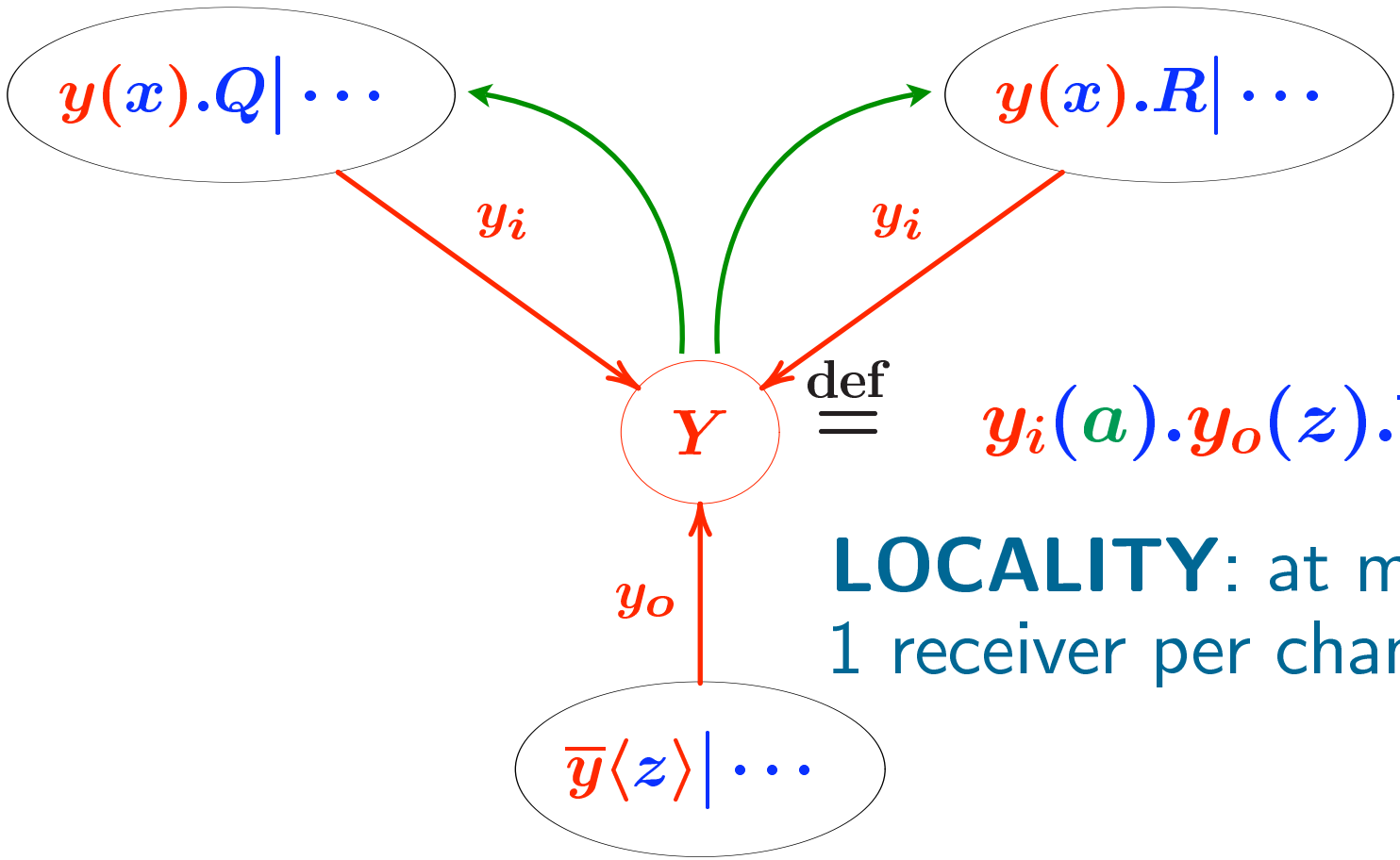
Unique Channel Managers



$y_i(a).y_o(z).\overline{a}\ z$

LOCALITY: at most
1 receiver per channel

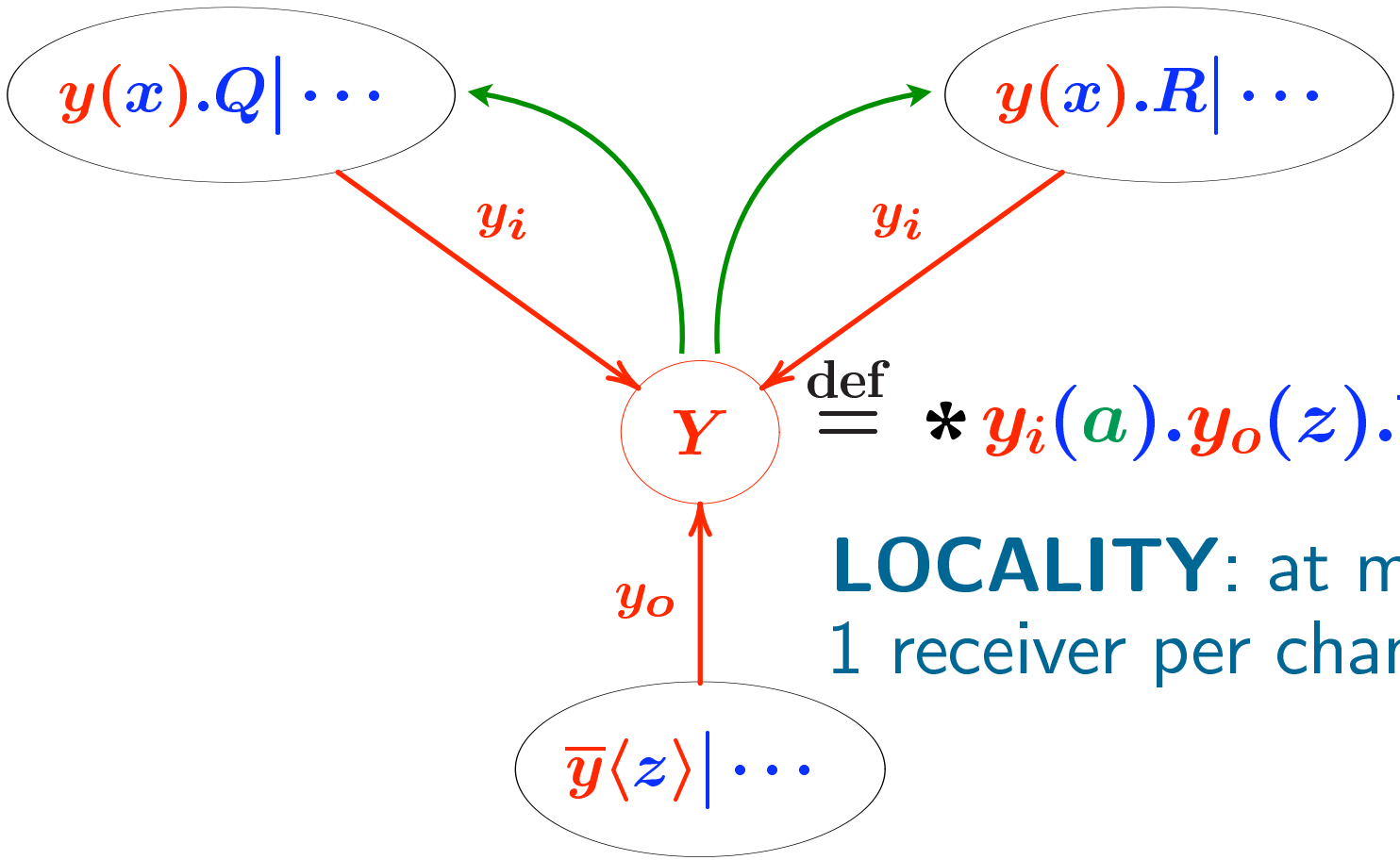
Unique Channel Managers



$y_i(a).y_o(z).\overline{a}\ z$

LOCALITY: at most 1 receiver per channel

Unique Channel Managers



$\stackrel{\text{def}}{=} * y_i(a).y_o(z).\overline{a}\ z$

LOCALITY: at most 1 receiver per channel

$\pi^{\mathbf{def}}$

residence site = creation site
avoids unnecessary global communications

$\pi^{\mathbf{def}}$

residence site = creation site
avoids unnecessary global communications

$$(\nu y)\, P$$
$$y(x).P$$
$$*P$$

$\pi^{\mathbf{def}}$

residence site = creation site
avoids unnecessary global communications

$$(\nu y)\, P$$
$$y(x).P$$
$$*P$$

$$(\nu y)\,(\; *\, y(x).P \;\mid\; Q\;)$$

$\pi^{\mathbf{def}}$

residence site = creation site
avoids unnecessary global communications

$$(\nu y)\, P$$
$$y(x).P$$
$$*P$$

$$(\nu y)\,(\; *\, y(x).P \;\mid\; Q\;)$$

$$\mathbf{def}\;\; y(x) = P \;\; \mathbf{in} \;\; Q$$

$\pi^{\mathbf{def}}$

residence site = creation site
avoids unnecessary global communications

$$(\nu y)\, P$$
$$y(x).P$$
$$*P$$

$$(\nu y)\,(\; *\, y(x).P \;\mid\; Q\;)$$

$$\mathbf{def}\;\; y(x) = P \;\; \mathbf{in} \;\; Q$$

channel managers are like function definitions
unique (replicated) receivers for the defined channels

Syntax and Reductions

Syntax and Reductions

$$
\begin{array}{rcl}
D & ::= & y(x) = P \\
P, Q & ::= & \textbf{def } D \textbf{ in } Q \mid y(x) \mid P|Q
\end{array}
$$

Syntax and Reductions

$$
\begin{array}{rcl}
D & ::= & y(x)=P \\
P,Q & ::= & \textsf{def } D \textsf{ in } Q \;\mid\; y(x) \;\mid\; P|Q
\end{array}
$$

$$\textsf{def } y(x)=P \textsf{ in } \left(Q \middle| y(z)\right)$$

Syntax and Reductions

$$
\begin{array}{rcl}
D & ::= & y(x)=P \\
P,Q & ::= & \textsf{def } D \textsf{ in } Q \;\mid\; y(x) \;\mid\; P|Q
\end{array}
$$

$$\textsf{def } y(x)=P \textsf{ in } \left(Q \middle| y(z)\right)$$

$$\longrightarrow \quad \textsf{def } y(x)=P \textsf{ in } \left(Q \middle| P\{^{z}/_{x}\}\right)$$

Expressive Power ?

$$\textbf{def } D \textbf{ in } (P \mid Q) \triangleq \textbf{def } D \textbf{ in } P \mid \textbf{def } D \textbf{ in } Q$$

Expressive Power ?

$$\textbf{def } D \textbf{ in } (P \mid Q) \triangleq \textbf{def } D \textbf{ in } P \mid \textbf{def } D \textbf{ in } Q$$

- only local/functional computations
  by sending and receiving individual messages
- no synchronization over parallel composition !

Core Join

Core Join

[Fournet, Gonthier, Lévy, ... 1995-2000]

Core Join

[Fournet, Gonthier, Lévy, ... 1995-2000]

simultaneous multi-channel reception

Core Join

[Fournet, Gonthier, Lévy, ... 1995-2000]

$$D \quad ::= \quad \overbrace{y(x)|u(w)}^{J} = P$$

$$P, Q \quad ::= \quad \textbf{def } D \textbf{ in } Q \quad | \quad y(x) \quad | \quad P|Q$$

simultaneous multi-channel reception

Core Join

[Fournet, Gonthier, Lévy, ... 1995-2000]

$$
\begin{array}{rcl}
D & ::= & \overbrace{y(x)|u(w)}^{J} = P \\
P, Q & ::= & \textbf{def } D \textbf{ in } Q \;\mid\; y(x) \;\mid\; P|Q
\end{array}
$$

simultaneous multi-channel reception

$$\textbf{def } J = P \textbf{ in } (Q|J\sigma)$$

$$\longrightarrow \quad \textbf{def } J = P \textbf{ in } (Q|P\sigma)$$

Expressive Power !

(* via mutual encodings *)

Expressive Power !

(* via mutual encodings *)

$$[\![\, \textbf{def}\ y(x)|u(w)=P\ \textbf{in}\ Q \,]\!] \stackrel{\text{def}}{=} (\nu x, u)\,(y(x).u(w).[\![\, P \,]\!] \mid [\![\, Q \,]\!])$$

$$[\![\, x(u) \,]\!] \stackrel{\text{def}}{=} \overline{x}\langle u \rangle$$

$$[\![\, P \mid Q \,]\!] \stackrel{\text{def}}{=} [\![\, P \,]\!] \mid [\![\, Q \,]\!]$$

Expressive Power !

(* via mutual encodings *)

$$[\![\, \textsf{def}\ y(x)|u(w)=P\ \textsf{in}\ Q \,]\!] \stackrel{\text{def}}{=} (\nu x, u)\,(y(x).u(w).[\![\, P \,]\!] \mid [\![\, Q \,]\!])$$

$$[\![\, x(u) \,]\!] \stackrel{\text{def}}{=} \overline{x}\langle u \rangle$$

$$[\![\, P \mid Q \,]\!] \stackrel{\text{def}}{=} [\![\, P \,]\!] \mid [\![\, Q \,]\!]$$

$$[\![\, (\nu y)\, P \,]\!] \stackrel{\text{def}}{=} \textsf{def}\ y_o(x_o, x_i)|y_i(\kappa)=\kappa(x_o, x_i)\ \textsf{in}\ [\![\, P \,]\!]$$

$$[\![\, \overline{y}\langle z \rangle \,]\!] \stackrel{\text{def}}{=} y_o(z_o, z_i)$$

$$[\![\, y(x).P \,]\!] \stackrel{\text{def}}{=} \textsf{def}\ \kappa(x_o, x_i)=[\![\, P \,]\!]\ \textsf{in}\ y_i(\kappa)$$

From Theory to Practice ...

Join patterns have been incorporated into **Polyphonic C#**, where *joins* are called *chords*, later on called **Cω**, as a .Net language.

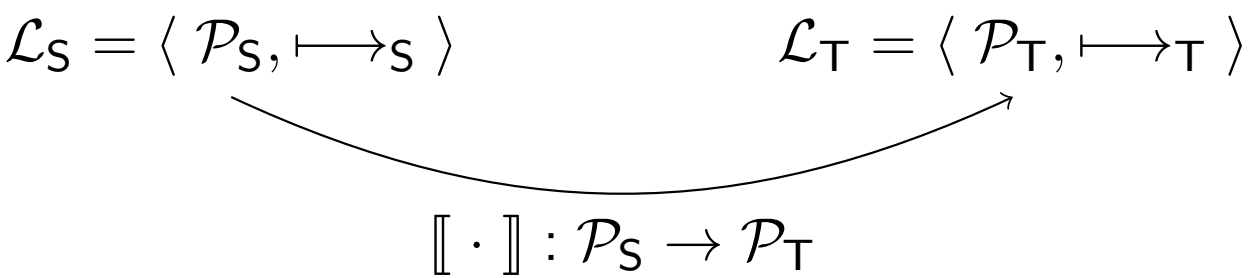
Keyword: "*modern concurrency abstractions*"

By now, there is also a **Join Java** ...

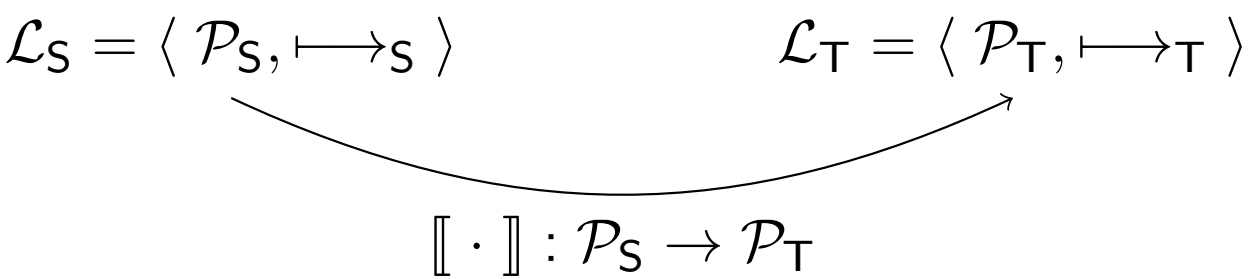
Encodings

$$\mathcal{L}_{\mathsf{S}} = \langle\, \mathcal{P}_{\mathsf{S}}, \longmapsto_{\mathsf{S}} \,\rangle \qquad \mathcal{L}_{\mathsf{T}} = \langle\, \mathcal{P}_{\mathsf{T}}, \longmapsto_{\mathsf{T}} \,\rangle$$

Encodings

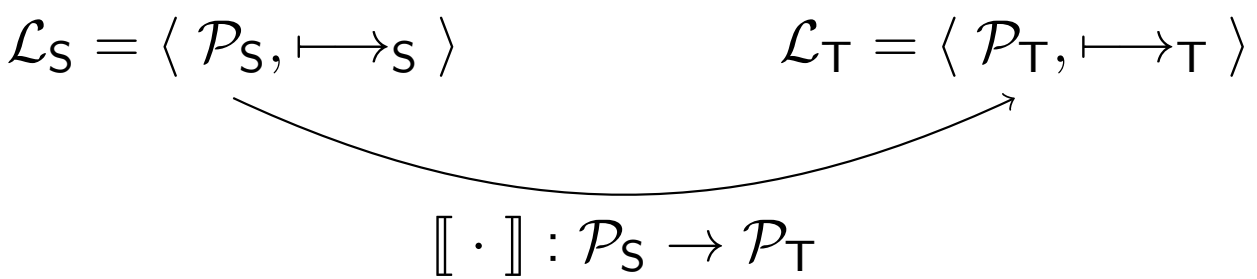

$\mathcal{L}_S = \langle \mathcal{P}_S, \longmapsto_S \rangle$
$\mathcal{L}_T = \langle \mathcal{P}_T, \longmapsto_T \rangle$
$[\![\cdot]\!] : \mathcal{P}_S \to \mathcal{P}_T$

Encodings



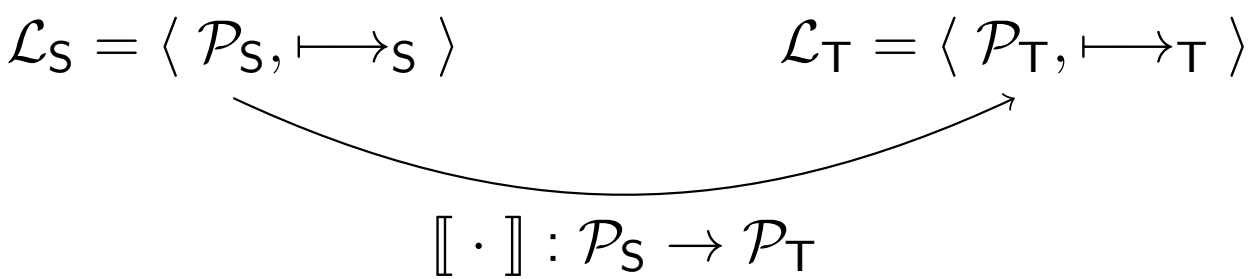
- quality criteria rule out trivial or meaningless encodings

Encodings



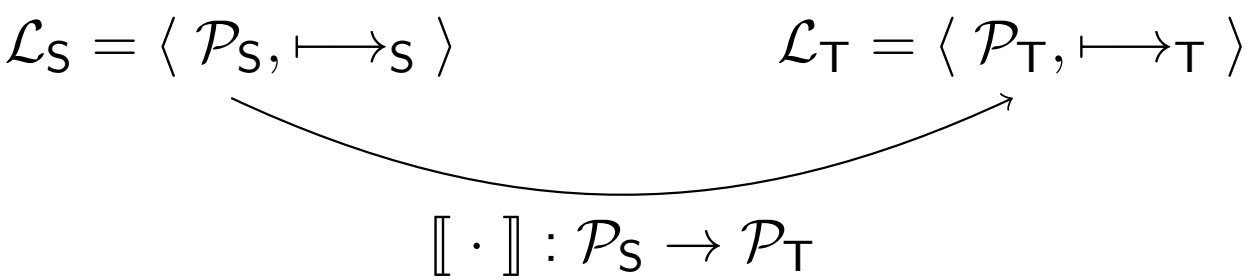
- quality criteria rule out trivial or meaningless encodings
  - preserve the "abstract" behaviour
  - reflect the "abstract" behaviour

Encodings



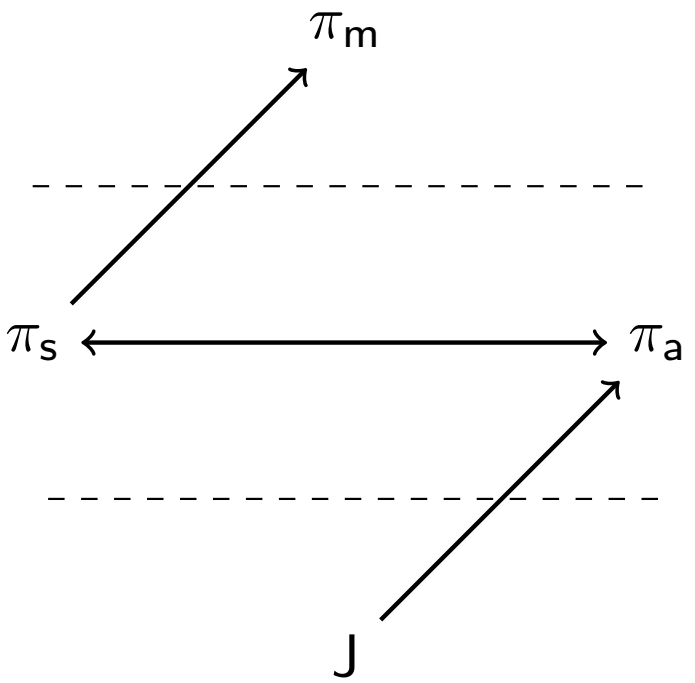
- quality criteria rule out trivial or meaningless encodings
  - preserve the "abstract" behaviour
  - reflect the "abstract" behaviour
  - do not introduce deadlock or divergence

Encodings



- quality criteria rule out trivial or meaningless encodings
  - preserve the "abstract" behaviour
  - reflect the "abstract" behaviour
  - do not introduce deadlock or divergence
- general framework [Gor10]
  - $[\![P \mid Q]\!] = C[\, [\![P]\!] \,|\, [\![Q]\!] \,]$ is allowed !

Abstract Expressive Power



Abstract Expressive Power

π_m

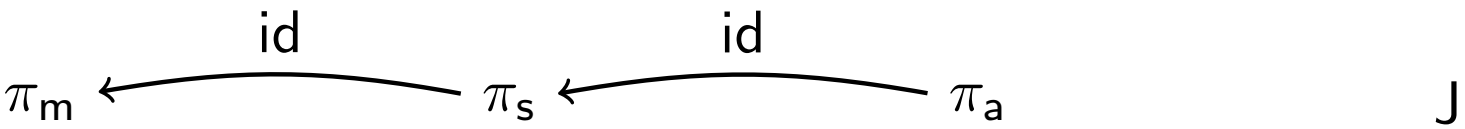
π_s π_a

J

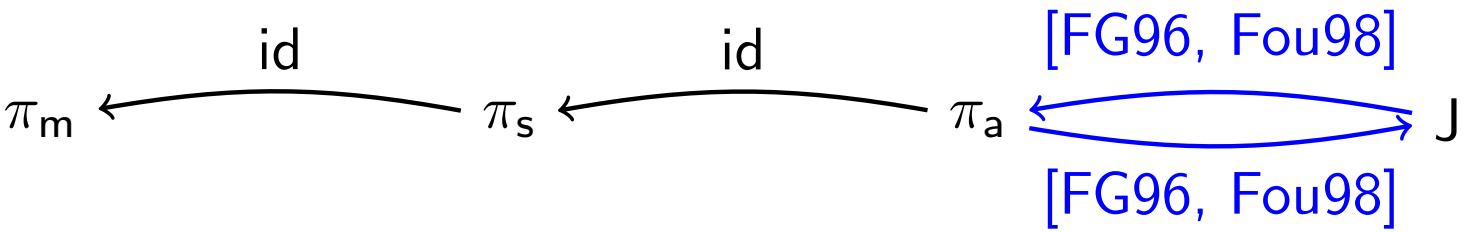
Abstract Expressive Power

π_m π_s π_a J

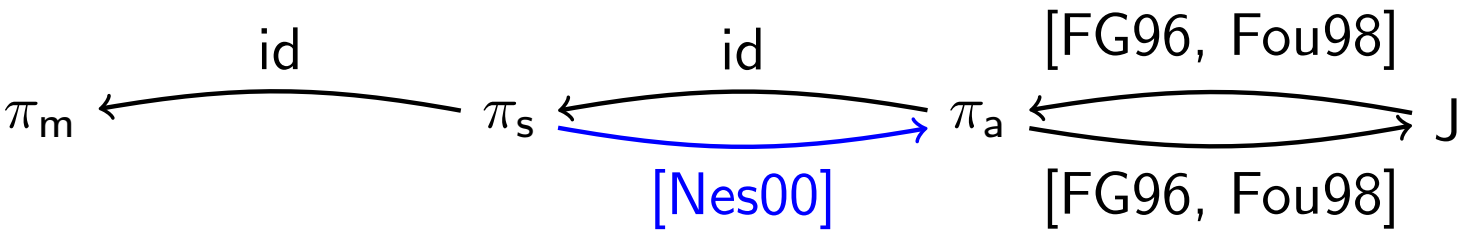
Abstract Expressive Power



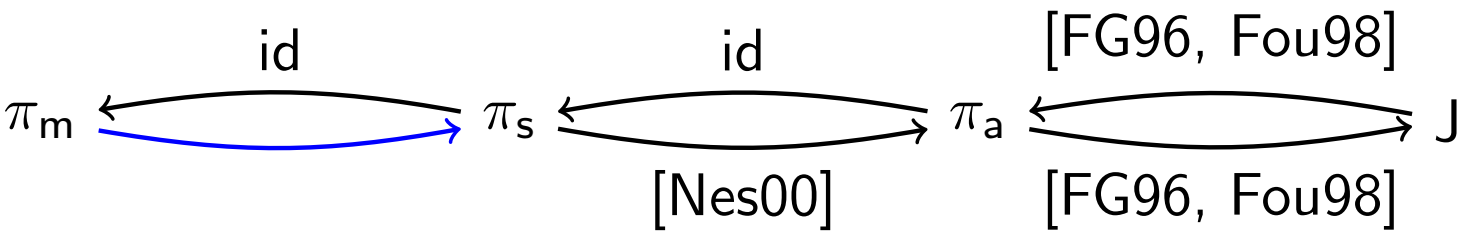
Abstract Expressive Power



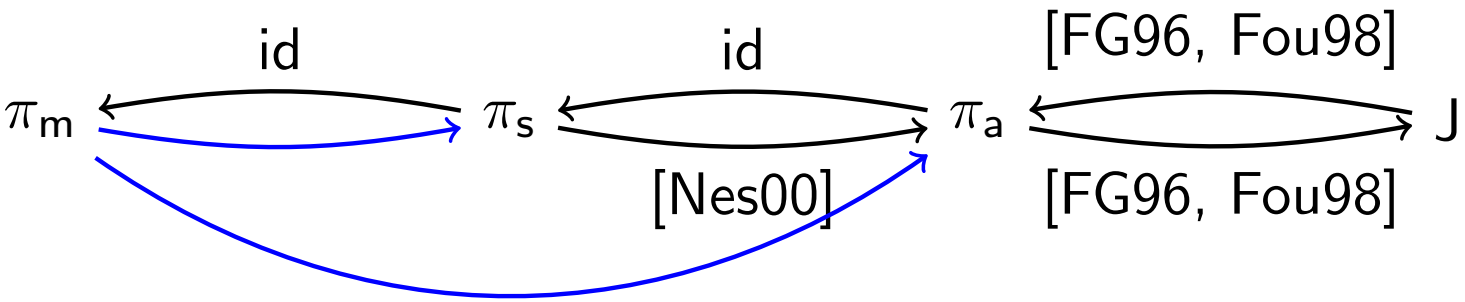
Abstract Expressive Power



Abstract Expressive Power



Abstract Expressive Power



Good Encoding from π_m into π_a

$$[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$$

Good Encoding from π_m into π_a

$[\![P \mid Q]\!] \neq [\![P]\!] \mid [\![Q]\!]$ follows from [Pal03]

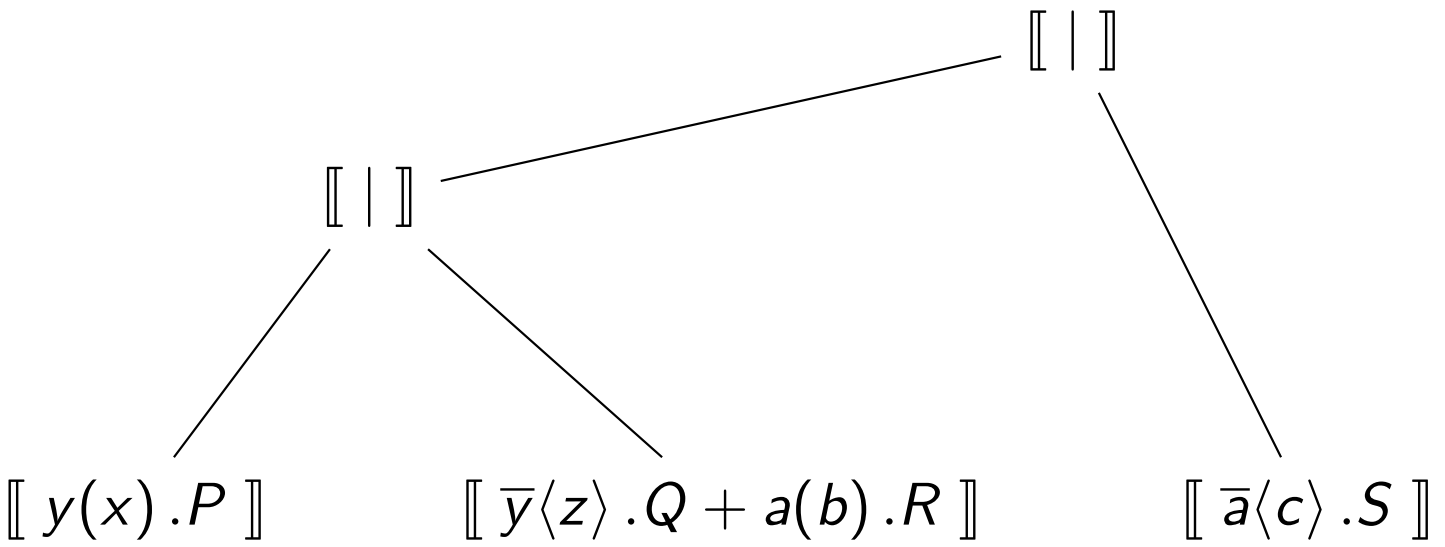
Good Encoding from π_m into π_a

$[\![P \mid Q]\!] \neq [\![P]\!] \mid [\![Q]\!]$ follows from [Pal03]

$$y(x).P \quad | \quad \overline{y}\langle z \rangle.Q + a(b).R \quad | \quad \overline{a}\langle c \rangle.S$$

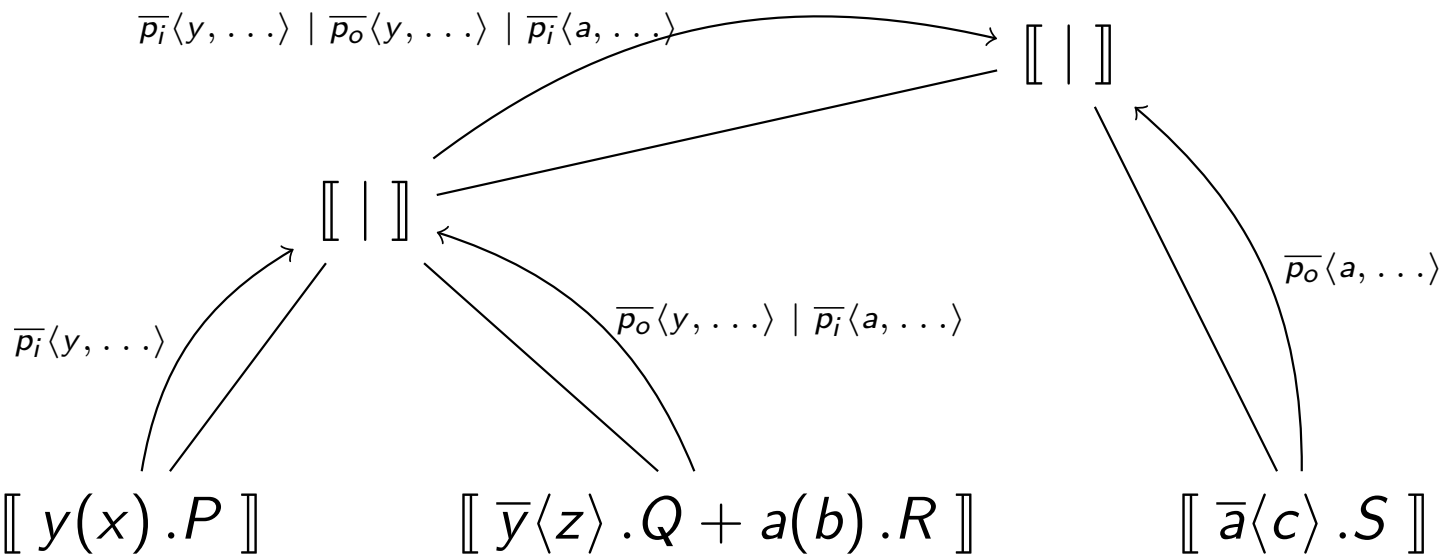
Good Encoding from π_m into π_a

$[\![P \mid Q]\!] \neq [\![P]\!] \mid [\![Q]\!]$ follows from [Pal03]



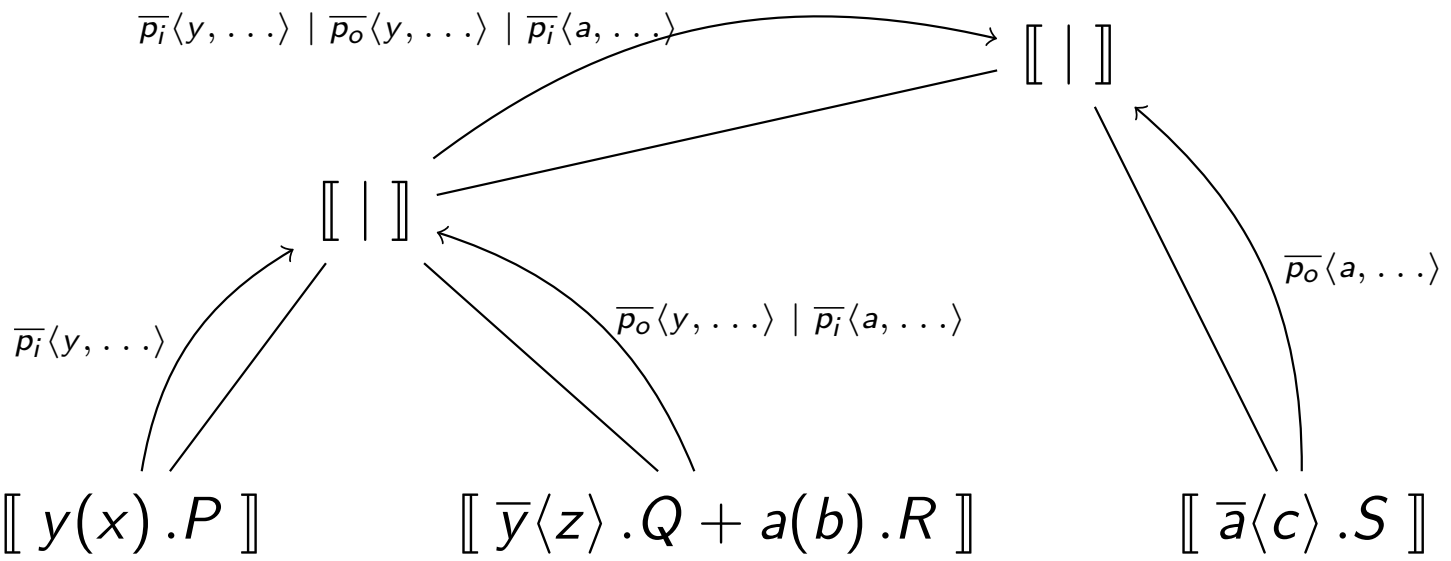
Good Encoding from π_m into π_a

$[\![P \mid Q]\!] \neq [\![P]\!] \mid [\![Q]\!]$ follows from [Pal03]



Good Encoding from π_m into π_a

$[\![P \mid Q]\!] \neq [\![P]\!] \mid [\![Q]\!]$ follows from [Pal03]



Contribution

There is a "good" encoding of mixed choice into π_a.
Formalisation of an additional criterion: *preservation of distributability*.

Abstract Expressive Power

Abstract Expressive Power

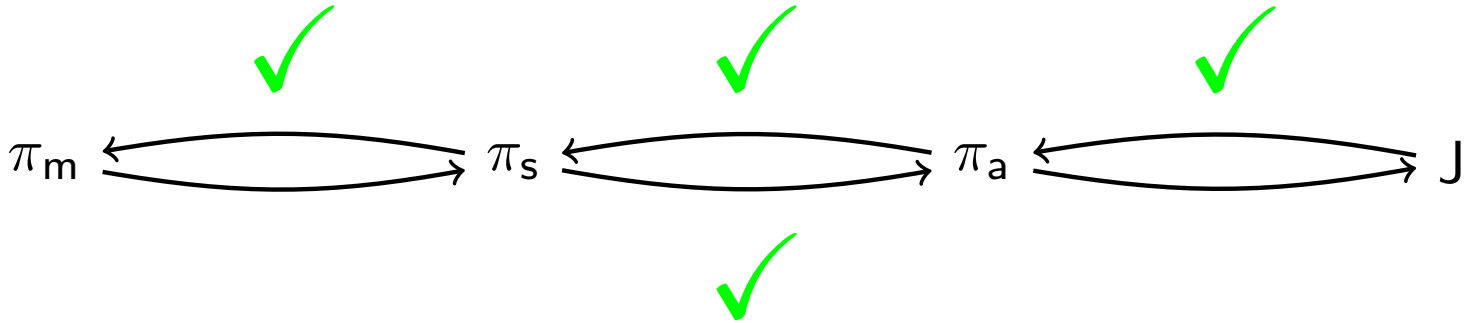
All four process calculi have the same abstract expressive power.

Abstract Expressive Power

All four process calculi have the same abstract expressive power.

However, they can be distinguished w.r.t. distributibility.
(Some encodings are not *distributability-preserving*.)

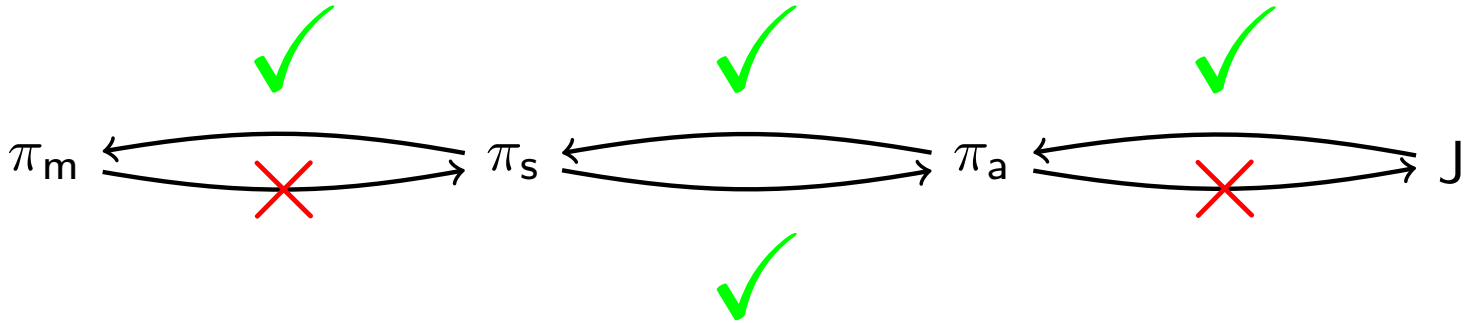
Abstract Expressive Power



All four process calculi have the same abstract expressive power.

However, they can be distinguished w.r.t. distributibility.
(Some encodings are not *distributability-preserving*.)

Abstract Expressive Power

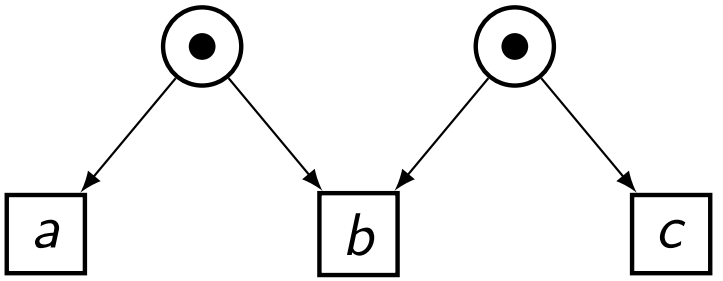


All four process calculi have the same abstract expressive power.

However, they can be distinguished w.r.t. distributibility.
(Some encodings are not *distributability-preserving*.)

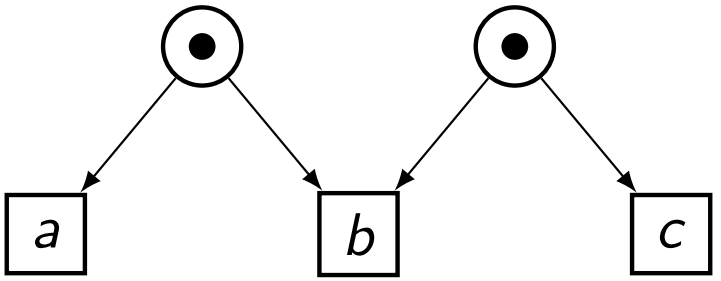
Synchronisation Pattern M

A fully reachable pure M in Petri nets [vGGS08, vGGS12]:



Synchronisation Pattern M

A fully reachable pure M in Petri nets [vGGS08, vGGS12]:

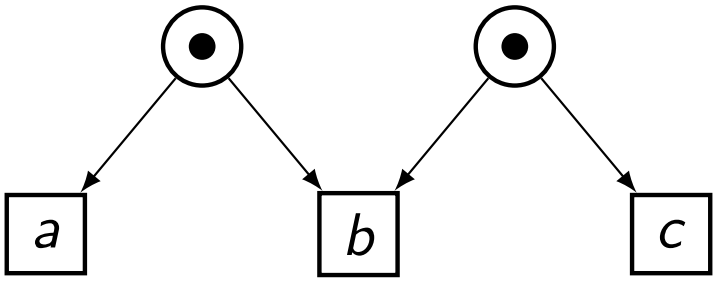


Theorem

A Petri net is distributable if it does not contain a fully reachable pure M.

Synchronisation Pattern M

A fully reachable pure M in Petri nets [vGGS08, vGGS12]:



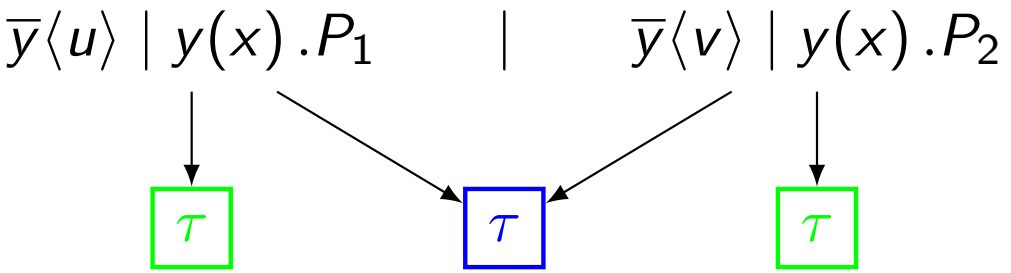
Theorem

A Petri net is distributable if it does not contain a fully reachable pure M.

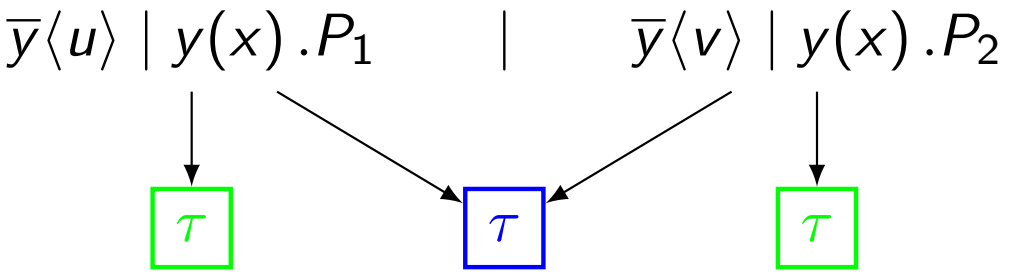
Contributions

Transferring this result on Petri nets into a result between process calculi.
A proof method for separation results based on synchronisation patterns.

Translational Separation Result: $\pi_a \not\longrightarrow J$



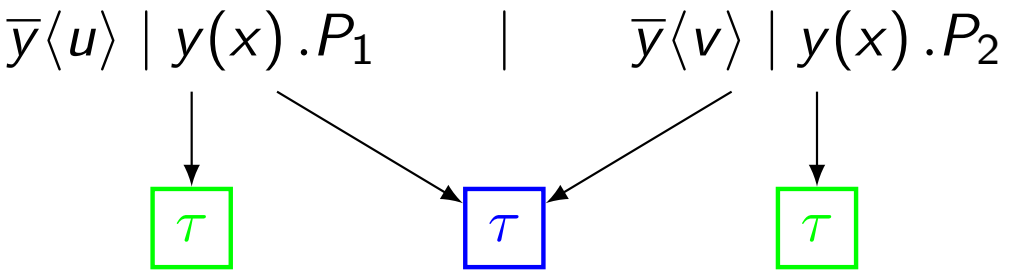
Translational Separation Result: $\pi_a \not\rightarrow J$



Example

There are M in π_a.

Translational Separation Result: $\pi_a \not\rightarrow$ J



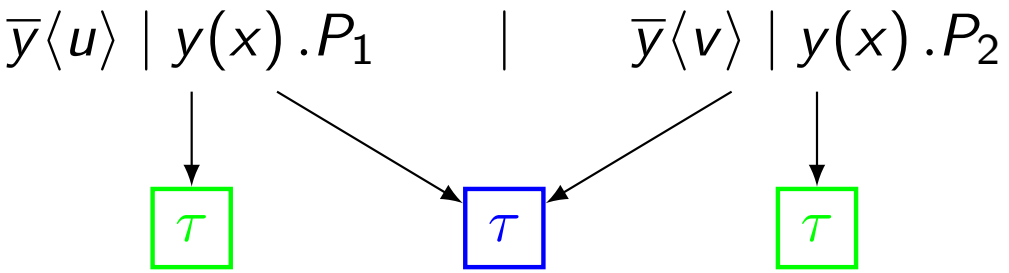
Example

There are M in π_a.

Lemma

All M in the Join-Calculus are local.

Translational Separation Result: $\pi_a \not\rightarrow J$



Example

There are M in π_a.

Lemma

All M in the Join-Calculus are local.

Contribution

The join-calculus is distributable, but the (asynchronous) pi-calculus is not.

Translational Separation Result: $\pi_a \not\rightarrow J$

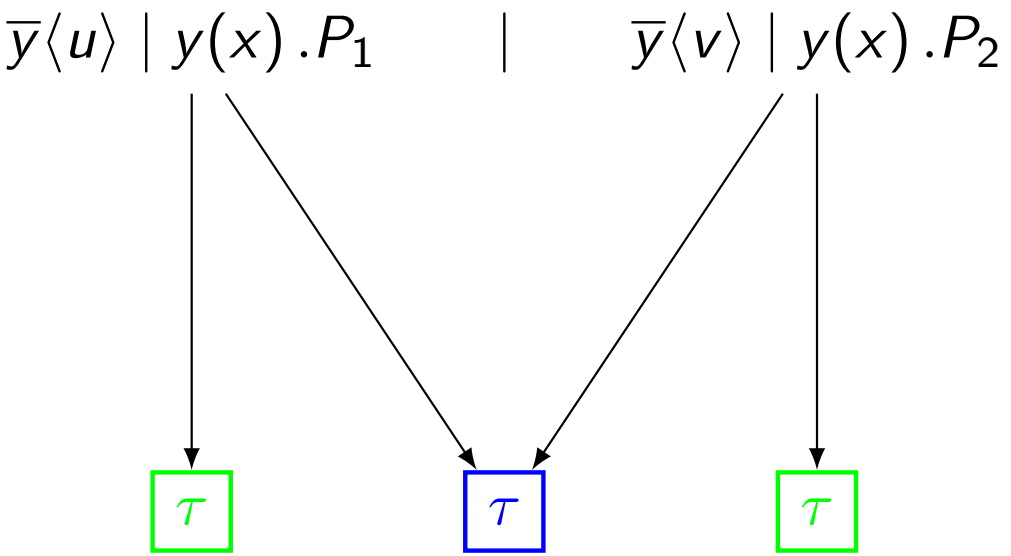
Lemma

Every good and distributability-preserving encoding from π_a into J has to split up the conflict between the blue step and the green steps.

Translational Separation Result: $\pi_a \not\rightarrow J$

Lemma

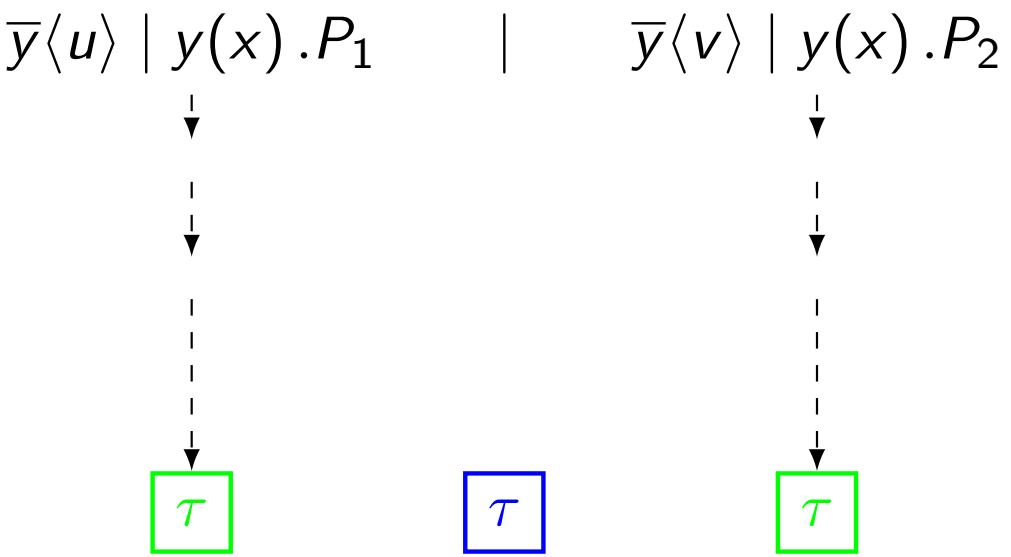
Every good and distributability-preserving encoding from π_a into J has to split up the conflict between the blue step and the green steps.



Translational Separation Result: $\pi_a \not\rightarrow J$

Lemma

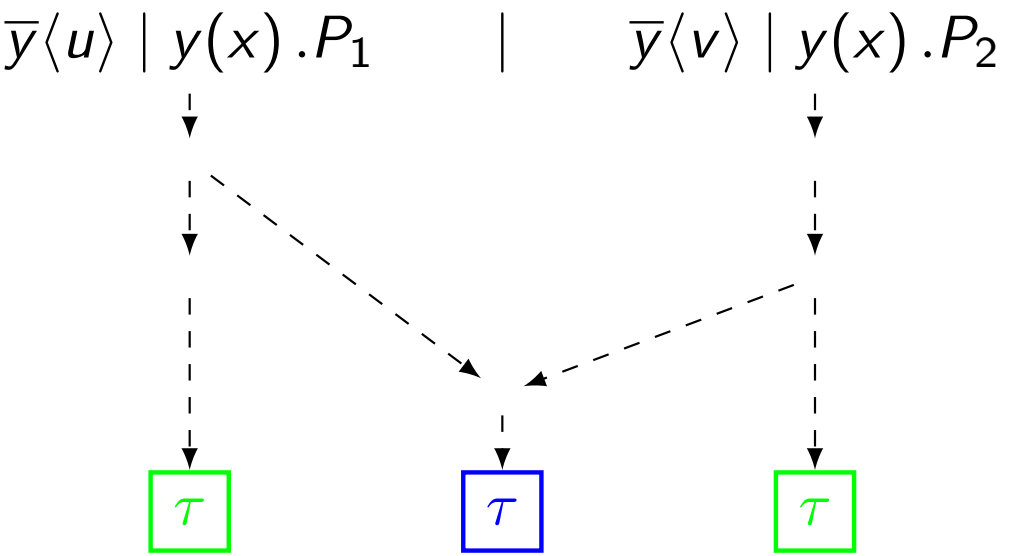
Every good and distributability-preserving encoding from π_a into J has to split up the conflict between the blue step and the green steps.



Translational Separation Result: $\pi_a \not\rightarrow J$

Lemma

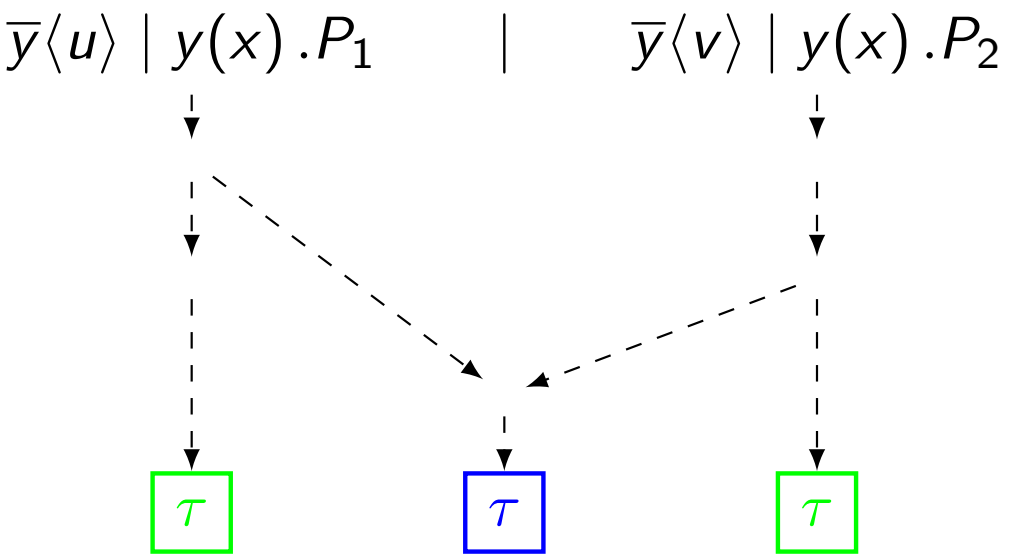
Every good and distributability-preserving encoding from π_a into J has to split up the conflict between the blue step and the green steps.



Translational Separation Result: $\pi_a \not\rightarrow J$

Lemma

Every good and distributability-preserving encoding from π_a into J has to split up the conflict between the blue step and the green steps.

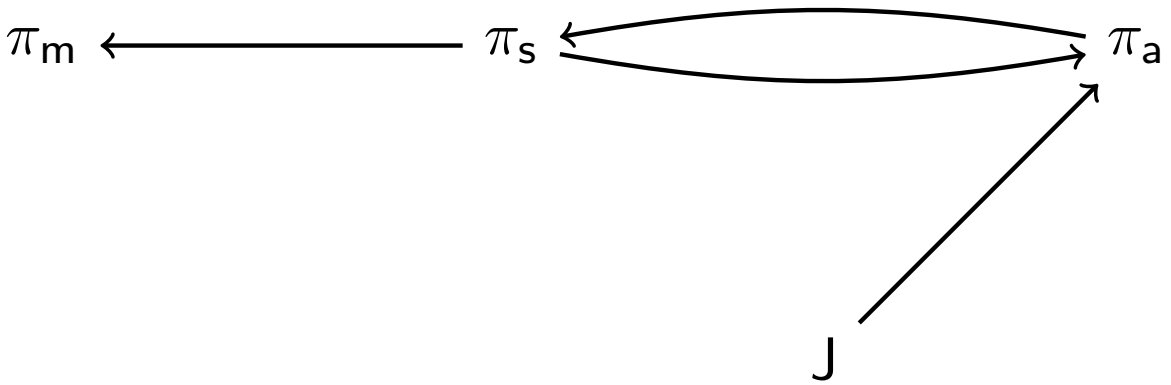


Contribution

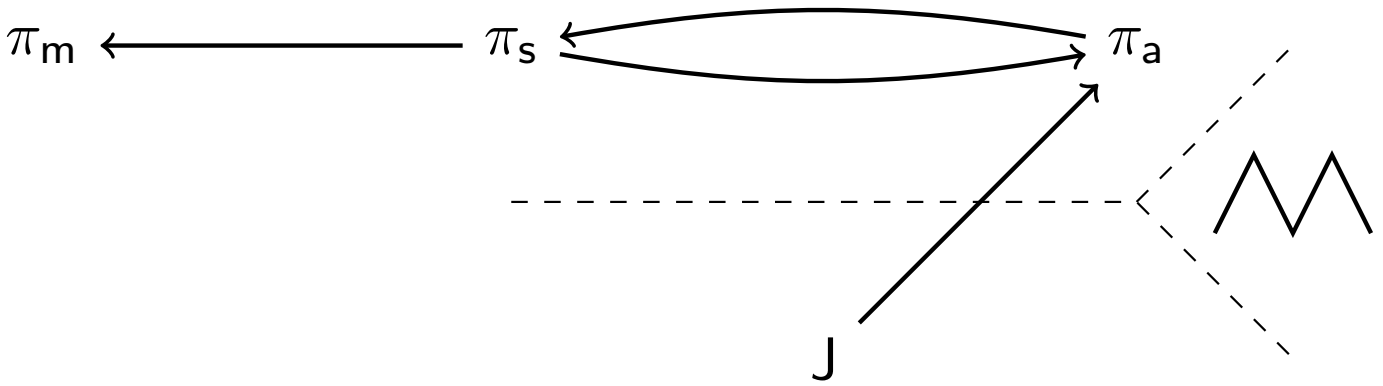
There is no good and distributability-preserving encoding from π_a into J.

Distributability in Pi-like Calculi

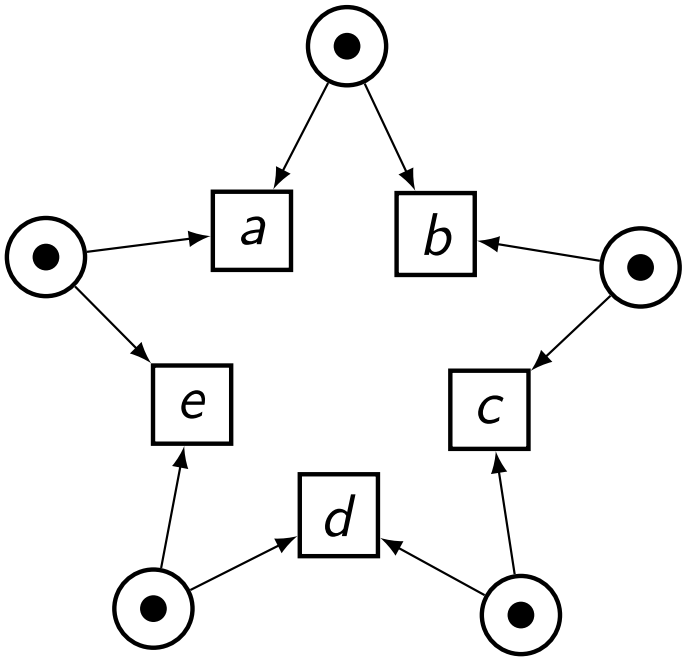
Distributability in Pi-like Calculi



Distributability in Pi-like Calculi

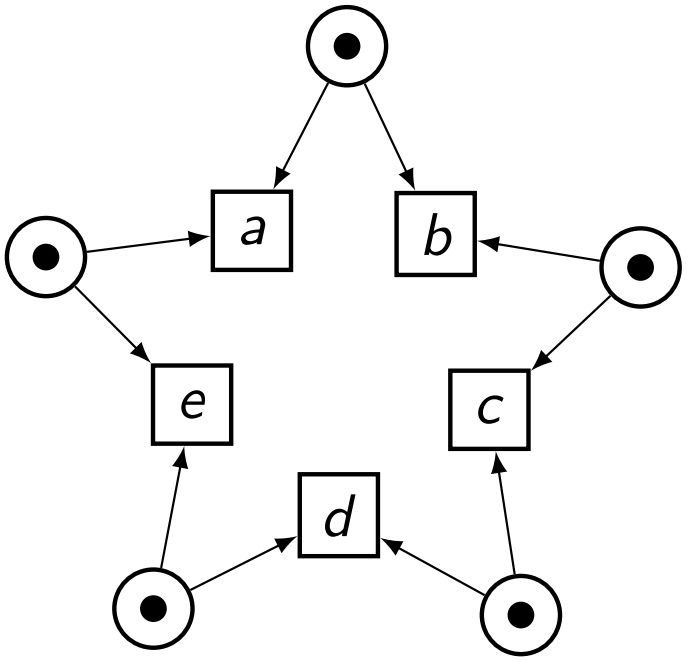


Synchronisation Pattern ⋆



- more complex synchronisation pattern
- several M that overlap in a cycle
- instance of dinning philosophers

Synchronisation Pattern ⋆

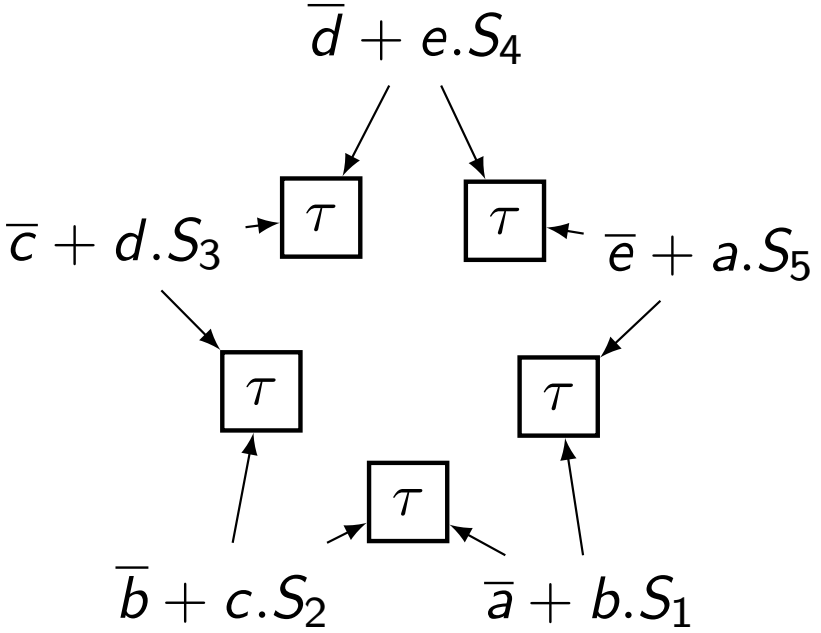


- more complex synchronisation pattern
- several M that overlap in a cycle
- instance of dinning philosophers

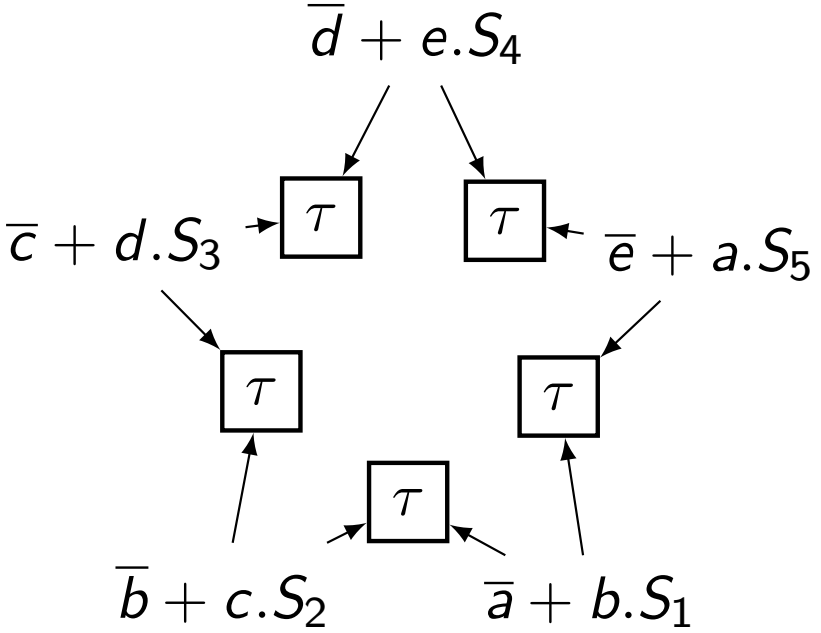
Contribution

New synchronisation pattern to capture the difference of mixed and separate choice.

Translational Separation Result: $\pi_m \not\hookrightarrow \pi_s$



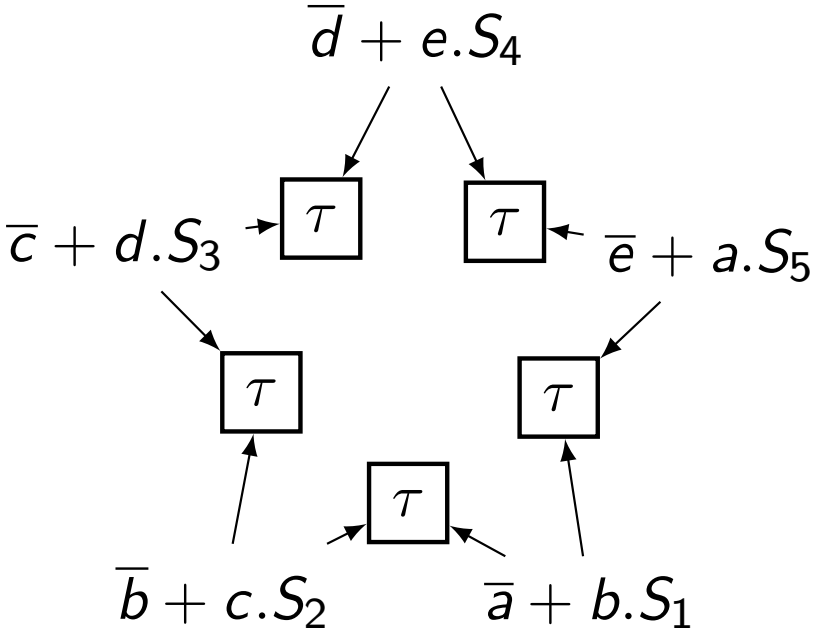
Translational Separation Result: $\pi_m \not\hookrightarrow \pi_s$



Lemma

There are no $\star$ in π_s.

Translational Separation Result: $\pi_m \not\hookrightarrow \pi_s$



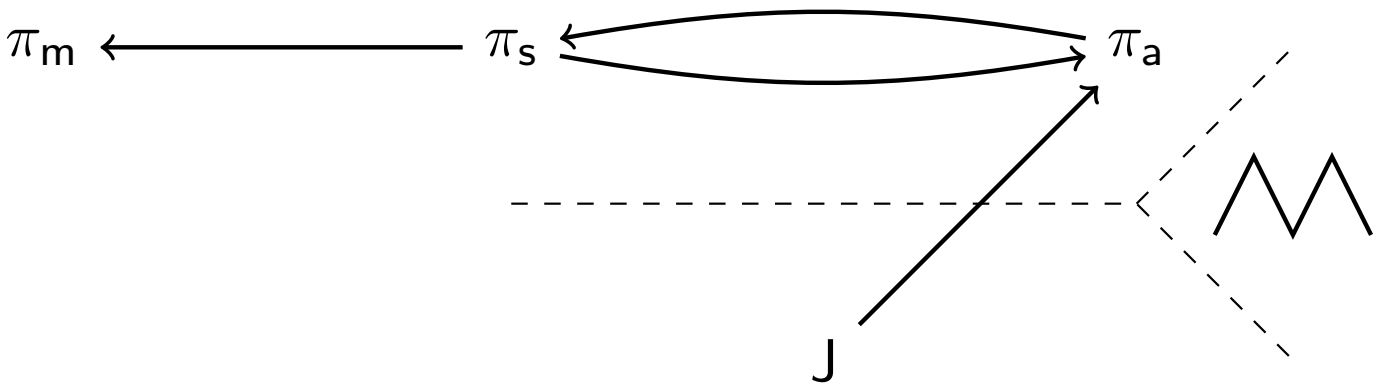
Lemma

There are no $\star$ in π_s.

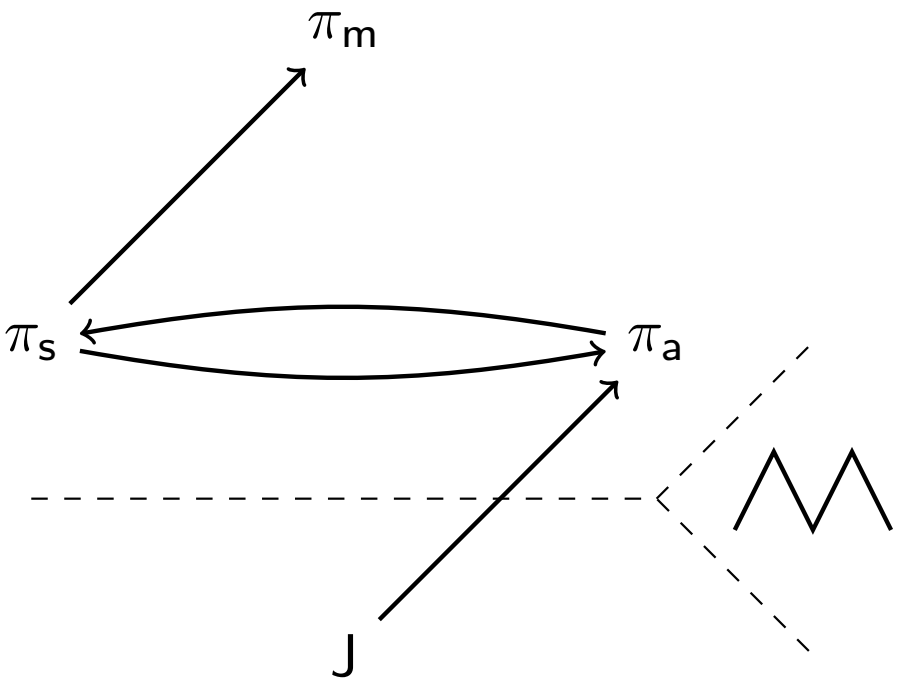
Contribution

There is no good and distributability-preserving encoding from π_m into π_s.

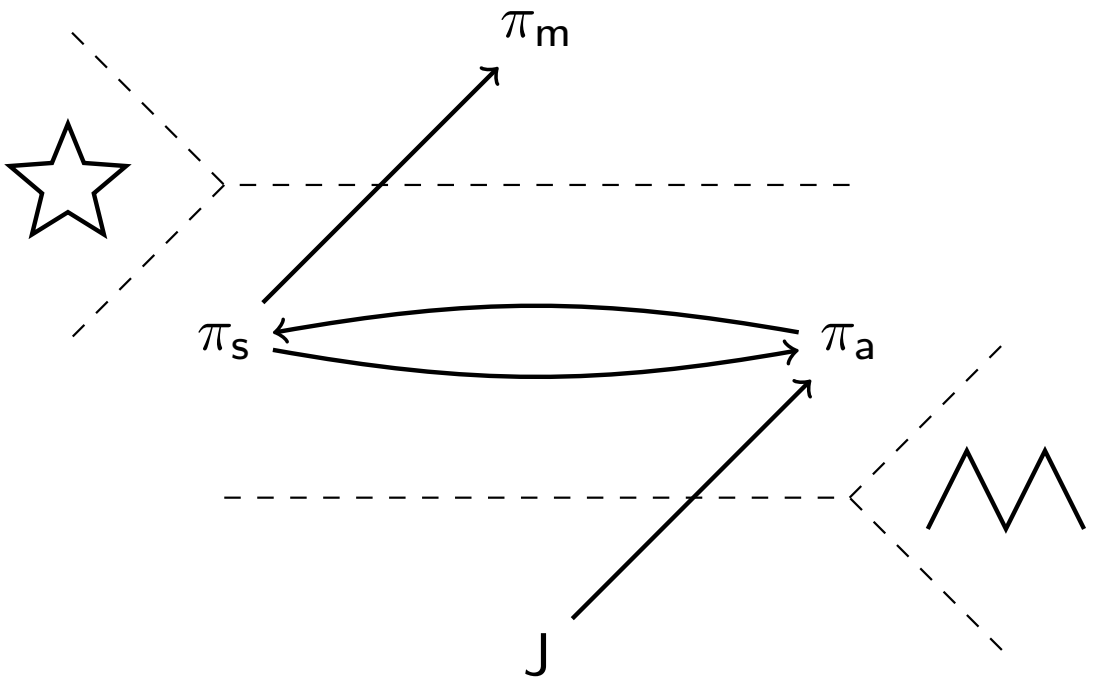
Distributability in Pi-like Calculi



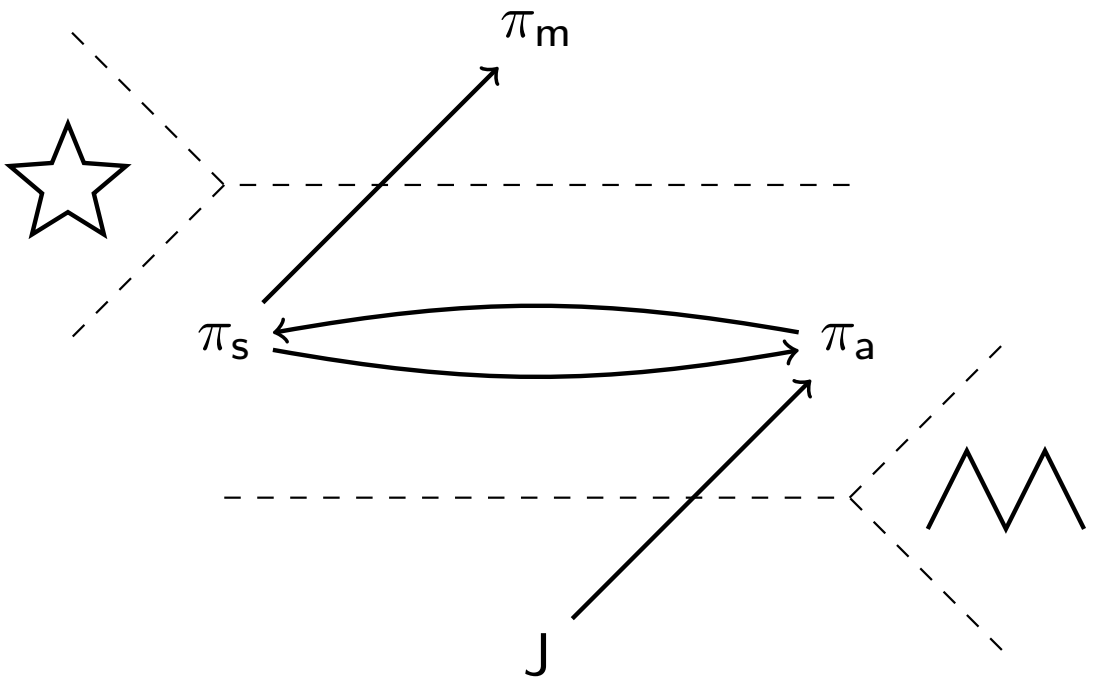
Distributability in Pi-like Calculi



Distributability in Pi-like Calculi



Distributability in Pi-like Calculi



Contribution

Three-level hierarchy.

Some Directions of Further Research

Which synchronisation pattern describes
polyadic synchronisation of Carbone+Maffeis [CM03]?

How much of the asynchronous pi-calculus is distributable?

Which system requirements are implicitly required
by the asynchronous/synchronous pi-calculus?

How can other process calculi be classified in this hierarchy?

Questions . . .


Marco Carbone and Sergio Maffeis.
On the Expressive Power of Polyadic Synchronisation in π-Calculus.
Nordic Journal of Computing, 10(2):70–98, 2003.


Cédric Fournet and Georges Gonthier.
The Reflexive CHAM and the Join-Calculus.
In *Proceedings of POPL (Principles of Programming Languages)*, SIGPLAN-SIGACT, pages 372–385. ACM, 1996.


Cédric Fournet.
The Join-Calculus: a Calculus for Distributed Mobile Programming.
PhD thesis, L'École Polytechnique, 1998.


Daniele Gorla.
Towards a Unified Approach to Encodability and Separation Results for Process Calculi.
Information and Computation, 208(9):1031–1053, 2010.


Uwe Nestmann.
What is a "Good" Encoding of Guarded Choice?
Information and Computation, 156(1-2):287–319, 2000.


Catuscia Palamidessi.
Comparing the Expressive Power of the Synchronous and the Asynchronous π-calculi.
Mathematical Structures in Computer Science, 13(5):685–719, 2003.


Rob J. van Glabbeek, Ursula Goltz, and Jens-Wolfhard Schicke.
On Synchronous and Asynchronous Interaction in Distributed Systems.
In *Proceedings of MFCS (Mathematical Foundations of Computer Science)*, volume 5162 of *Lecture Notes in Computer Science*, pages 16–35. Springer, 2008.


Rob J. van Glabbeek, Ursula Goltz, and Jens-Wolfhard Schicke-Uffmann.
On Distributability of Petri Nets.
In *Proceedings of FoSSaCS (Foundations of Software Science and Computational Structures)*, volume 7213 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2012.