A Graph Rewriting-Based Semantics and Implementation for $\rho\pi$

J. Cailler M. Vassor

University of Lorraine

July 10, 2026

Introduction

Reversible higher-order π-calculus:

- Processes that exchange messages
- Messages' content are processes

Introduction

Reversible higher-order π-calculus:

- Processes that exchange messages
- Messages' content are processes
- 2 main actions:
  - Send
  - Receive

Introduction

Reversible higher-order π-calculus:

- Processes that exchange messages
- Messages' content are processes
- 2 main actions:
  - Send
  - Receive
- 2 semantics:
  - Forward (messages are exchanged)
  - Backward (exchanges are undone)
  - Backward is *causally-consistent*

Introduction

$\mathcal{P}, \mathcal{Q}$	$::=$	$0 \mid X \mid \nu a.(P) \mid P \parallel Q$	*Process*
	$\mid$	$a\langle P \rangle \mid a(X) \triangleright (P)$	
$\mathcal{M}, \mathcal{N}$	$::=$	$0 \mid \nu u.(M) \mid M \parallel N$	*Configuration*
	$\mid$	$\kappa : P \mid [\mu; k]$	
κ	$::=$	$k \mid \langle h, \tilde{h} \rangle \cdot k$	*Tag*
μ	$::=$	$\kappa_1 : a\langle P \rangle \parallel \kappa_2 : a(X) \triangleright (Q)$	*Memory content*

Introduction

$$k_1 : \ a\,\langle P \rangle \ || \ k_2 : \ a(X) \triangleright (Q)$$

Introduction

$$k_1 : \ a\langle P\rangle \ \| \ k_2 : \ a(X) \triangleright (Q)$$

$$\rightarrow \nu k.\left(k : \ Q\{^P/_X\} \ \| \ [k_1 : \ a\langle P\rangle \ \| \ k_2 : \ a(X) \triangleright (Q); k]\right)$$

Introduction

- $\rho\pi$ is difficult to understand
  - Syntax: tags are messy
  - Semantics: structural congruence and tags

Introduction

- $\rho\pi$ is difficult to understand
  - Syntax: tags are messy
  - Semantics: structural congruence and tags
- Intuitively: tags track causal dependencies
- Should be easy

Introduction

- $\rho\pi$ is difficult to understand
  - Syntax: tags are messy
  - Semantics: structural congruence and tags
- Intuitively: tags track causal dependencies
- Should be easy

Idea: use graphs to track causal dependencies!

G$\rho\pi$F presentation

- graph rewriting-based semantics
- 1 node $=$ 1 (**HO**π) process
- edges track causal dependencies
- we do not need tags

G$\rho\pi$F presentation

- graph rewriting-based semantics
- 1 node = 1 (**HO**π) process
- edges track causal dependencies
- we do not need tags

Demo!

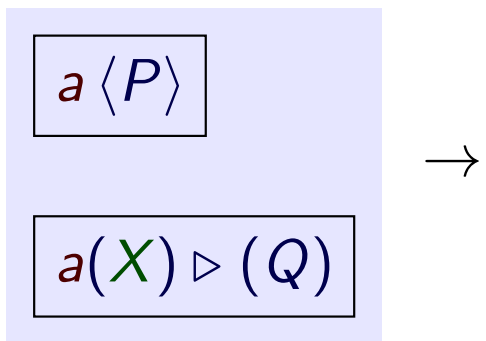
Communication rule

$$(\textsc{Comm})\ \frac{}{\kappa_1 :: a\langle P\rangle \parallel \kappa_2 :: a(X) \triangleright (Q) \rightarrow \nu k.\left(k :: Q\{^P/_X\} \parallel [\kappa_1 :: a\langle P\rangle \parallel \kappa_2 :: a(X) \triangleright (Q); k]\right)}$$

Communication rule

$$(\textsc{Comm})\ \frac{}{\kappa_1 : a\langle P\rangle \parallel \kappa_2 : a(X) \triangleright (Q) \rightarrow \nu k.\left(k : Q\{^P/_X\} \parallel [\kappa_1 : a\langle P\rangle \parallel \kappa_2 : a(X) \triangleright (Q); k]\right)}$$

Rule: (Fw)

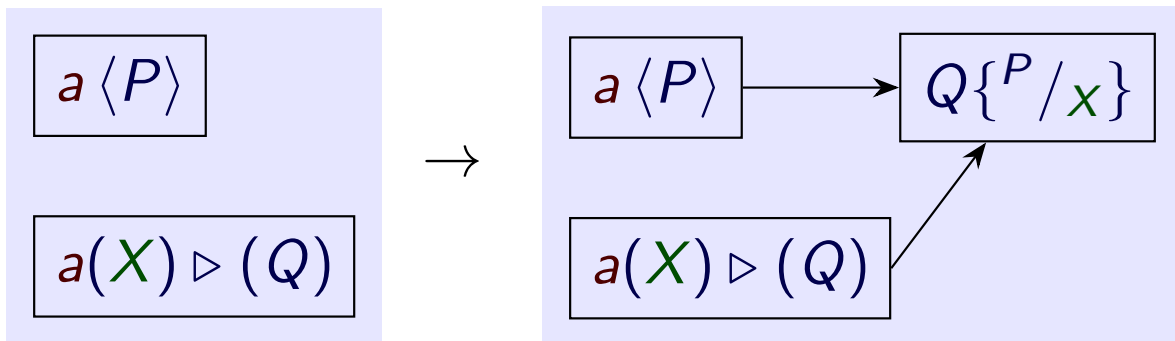


If there is no outgoing edge from the nodes in the left-hand side.

Communication rule

$$(\textsc{Comm})\ \frac{}{\kappa_1 : a\langle P\rangle \parallel \kappa_2 : a(X) \triangleright (Q) \rightarrow \nu k.\left(k : Q\{^P/_X\} \parallel [\kappa_1 : a\langle P\rangle \parallel \kappa_2 : a(X) \triangleright (Q); k]\right)}$$

Rule: (Fw)



If there is no outgoing edge from the nodes in the left-hand side.

Benefits

Many things are way simpler:

- No need for keys
- Many rules of structural congruence vanish

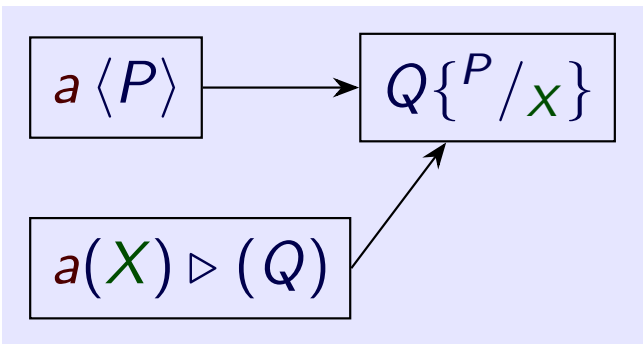
Backward rule

$$(\mathrm{Bw})\ \frac{}{\nu k.\left(k :\ Q\{^{P}/_{X}\} \parallel [\kappa_1 :\ a\langle P\rangle \parallel \kappa_2 :\ a(X) \triangleright (Q); k]\right) \to \kappa_1 :\ a\langle P\rangle \parallel \kappa_2 :\ a(X) \triangleright (Q)}$$

Backward rule

$$(\mathrm{Bw})\ \frac{}{\nu k.\left(k :\ Q\{^{P}/_{X}\} \parallel [\kappa_1 :\ a\langle P\rangle \parallel \kappa_2 :\ a(X) \triangleright (Q); k]\right) \to \kappa_1 :\ a\langle P\rangle \parallel \kappa_2 :\ a(X) \triangleright (Q)}$$

Rule: (Bw)



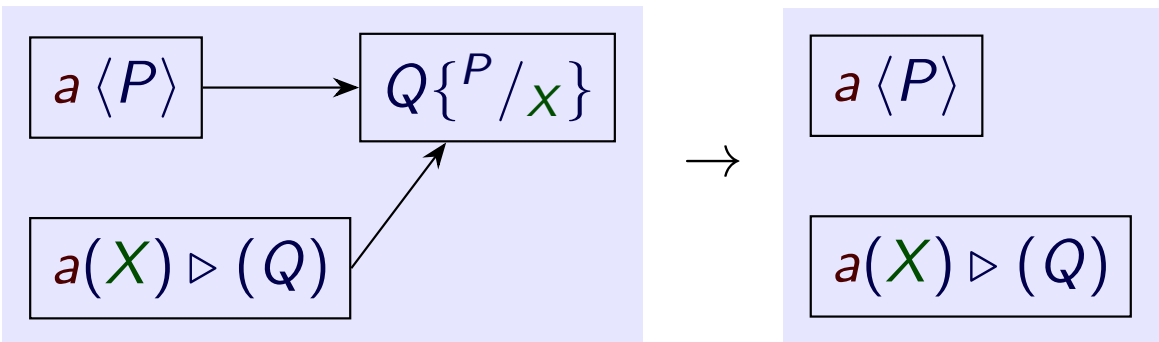
$\to$

If there is no other outgoing edge from the three vertices of the left-hand side.

Backward rule

$$(\mathrm{Bw})\ \frac{}{\nu k.\left(k :\ Q\{^{P}/_{X}\} \parallel [\kappa_1 :\ a\langle P\rangle \parallel \kappa_2 :\ a(X) \triangleright (Q); k]\right) \to \kappa_1 :\ a\langle P\rangle \parallel \kappa_2 :\ a(X) \triangleright (Q)}$$

Rule: (Bw)



If there is no other outgoing edge from the three vertices of the left-hand side.

Restrictions

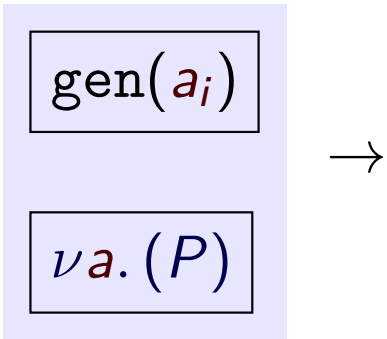
How to deal with restrictions?

$$\nu a.\,(P) \parallel Q$$

$$P\{^{f}/_{a}\} \parallel Q \text{ with fresh } f$$

Restrictions

Rule: (NuFresh)

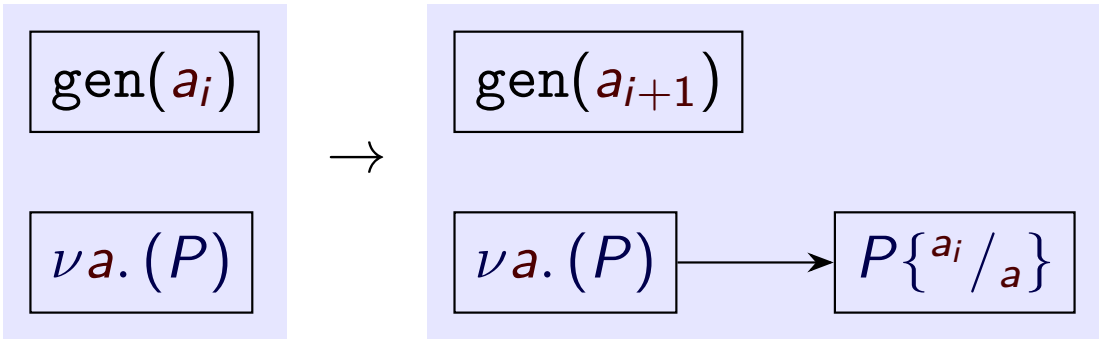


$\rightarrow$

If there is no outgoing edge from $\nu a.(P)$.

Restrictions

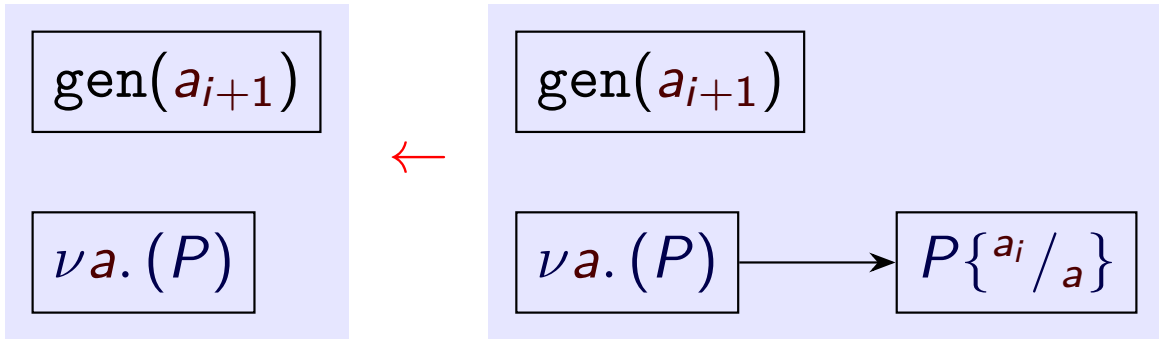
Rule: (NuFresh)



If there is no outgoing edge from $\nu a.(P)$.

Restrictions

Rule: (NuRestrict)



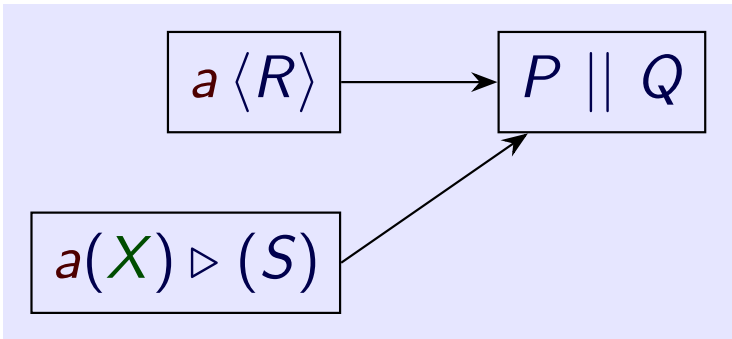
If there is no outgoing edge from $P\{^{a_i}/_{a}\}$.

Parallelism

$$k : \ P \parallel Q \equiv \nu k_1, k_2. \begin{pmatrix} \langle k_1, \{k_1, k_2\} \rangle \cdot k : \ P \\ \\ \parallel \langle k_2, \{k_1, k_2\} \rangle \cdot k : \ Q \end{pmatrix}$$

Parallelism

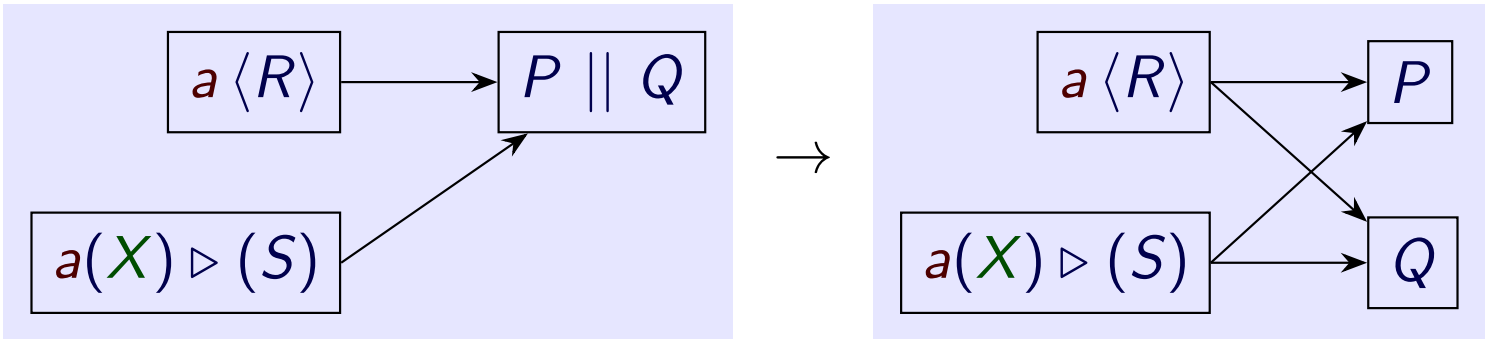
Rule: (SplitCom)



$\rightarrow$

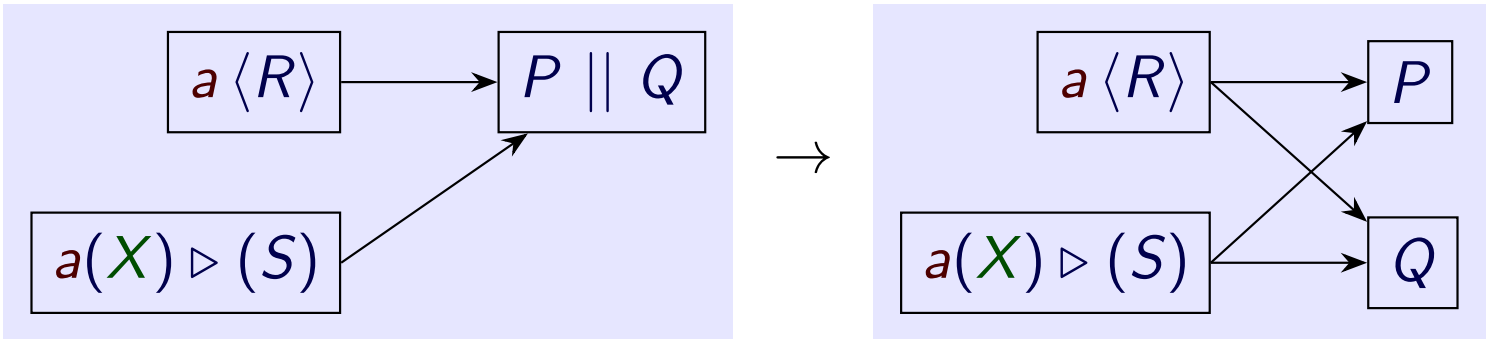
Parallelism

Rule: (SplitCom)



Parallelism

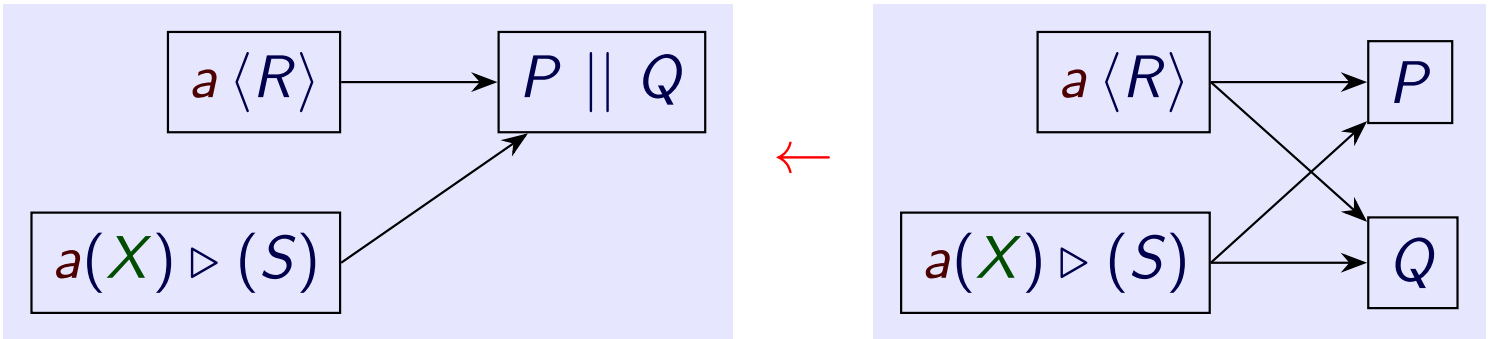
Rule: (SplitCom)



- Reduction rule vs. structural congruence
- 1 step/split

Parallelism

Rule: (MergeCom)

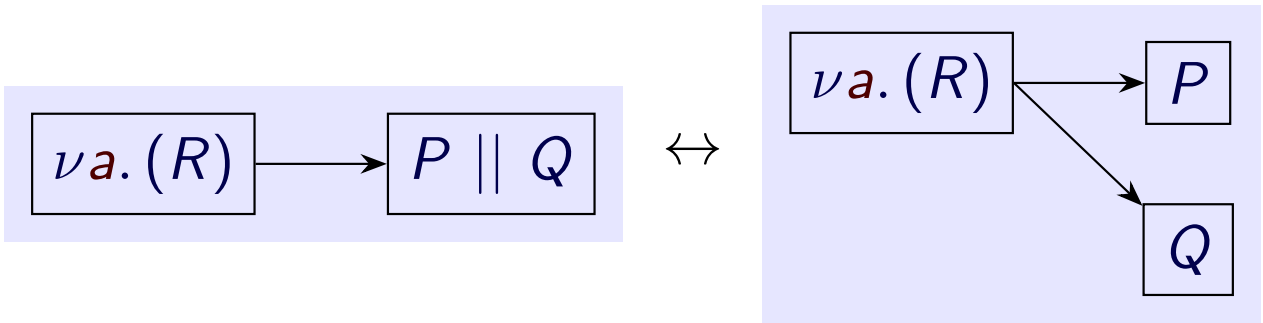


If there is no outgoing edges from P and Q.

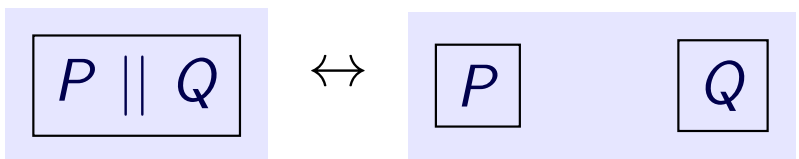
- Reduction rule vs. structural congruence
- 1 step/split

Parallelism

Rule: (SplitNu/MergeNu)



Rule: (SplitInit/MergeInit)



(With side conditions.)

Conclusion

- A graph rewriting-based semantics for $\rho\pi$
- Implemented in Python ($\sim$2000 LoC)
- Intended use:
  - Didactics (e.g. students)
  - Help identifying patterns
  - Intuition for alternative semantics in $\rho\pi$

Conclusion

- A graph rewriting-based semantics for $\rho\pi$
- Implemented in Python ($\sim$2000 LoC)
- Intended use:
  - Didactics (e.g. students)
  - Help identifying patterns
  - Intuition for alternative semantics in $\rho\pi$

Future work

- Correspondance with $\rho\pi$ (encoding)
- Automation (reachable states)

Conclusion

- A graph rewriting-based semantics for $\rho\pi$
- Implemented in Python ($\sim$2000 LoC)
- Intended use:
  - Didactics (e.g. students)
  - Help identifying patterns
  - Intuition for alternative semantics in $\rho\pi$

Future work

- Correspondance with $\rho\pi$ (encoding)
- Automation (reachable states)

Thank you for your attention!
Questions?