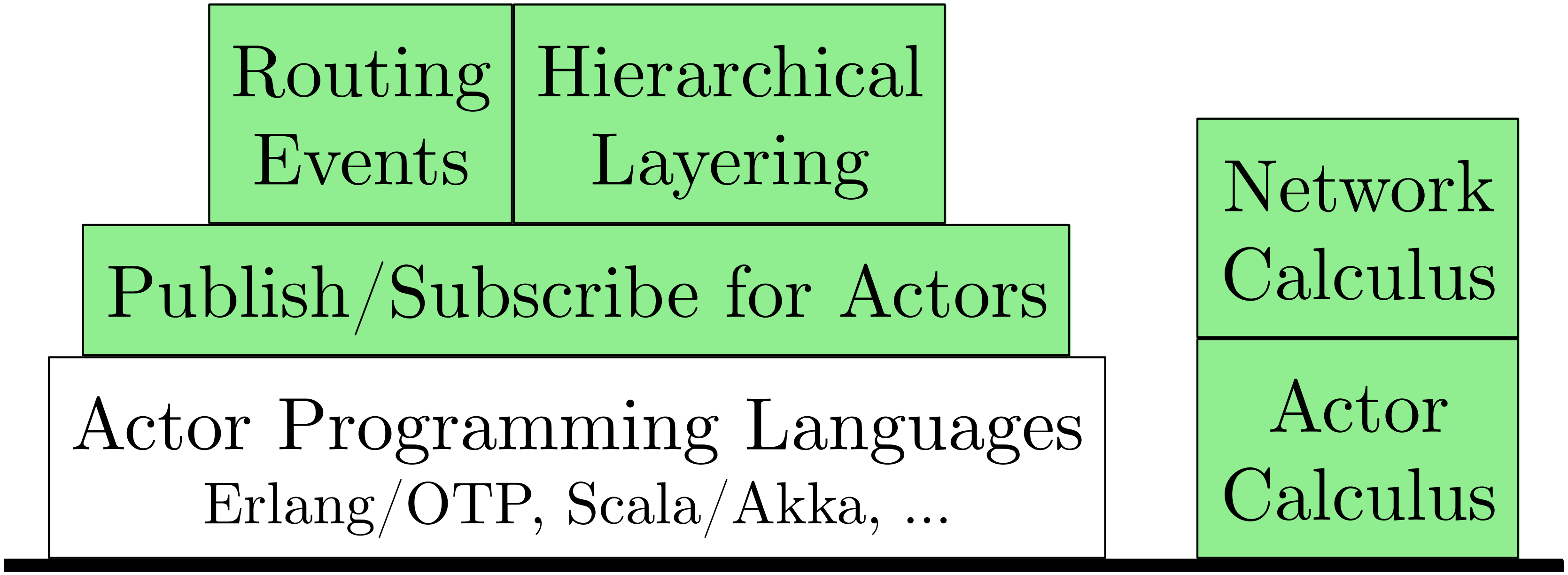
EPFL, Lausanne, Switzerland; 20140407 // ESOP 2014, Grenoble, France; 20140410

The Network as a Language Construct

Tony Garnock-Jones Sam Tobin-Hochstadt Matthias Felleisen



Routing Events

Hierarchical Layering

Publish/Subscribe for Actors

Actor Programming Languages

Erlang/OTP, Scala/Akka, ...

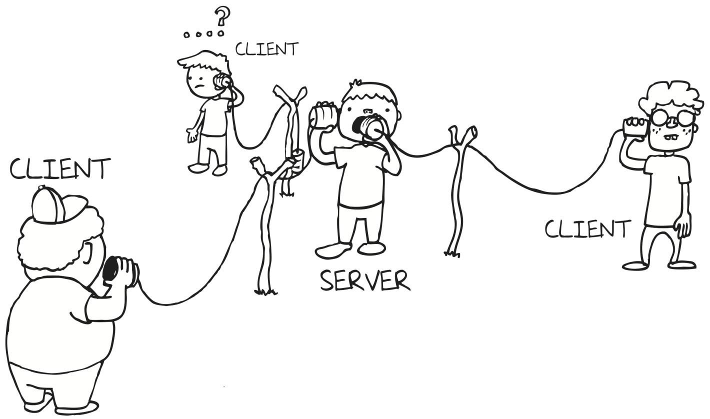
Network Calculus

Actor Calculus

PART I: The Problem

Functional I/O

Scaling up **big-bang** from domain-specific to general functional I/O

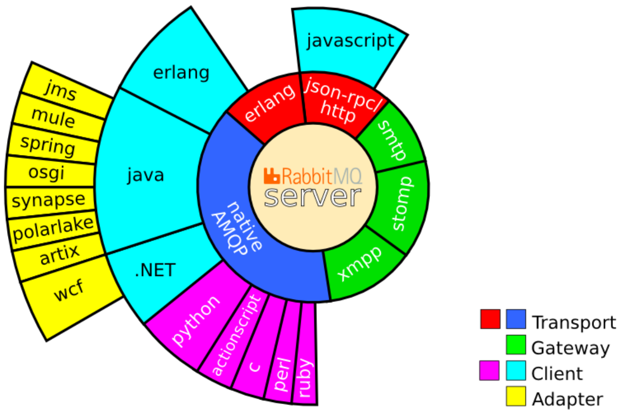


Apps in a functional I/O style:

- echo server
- multi-user chat
- DNS server
- SSH server

Distributed Systems

Implementing **RabbitMQ** and using it to build distributed systems



Investigated other paradigms:

- OO languages
- Network architecture
- CORBA services
- Erlang applications
- Modern Unix services

Ubiquitous Patterns and Problems

Event broadcasting

Naming service

Service discovery

Startup ordering

Crash/exit signalling

Conversation management

Ubiquitous Patterns and Problems

Event broadcasting

Naming service

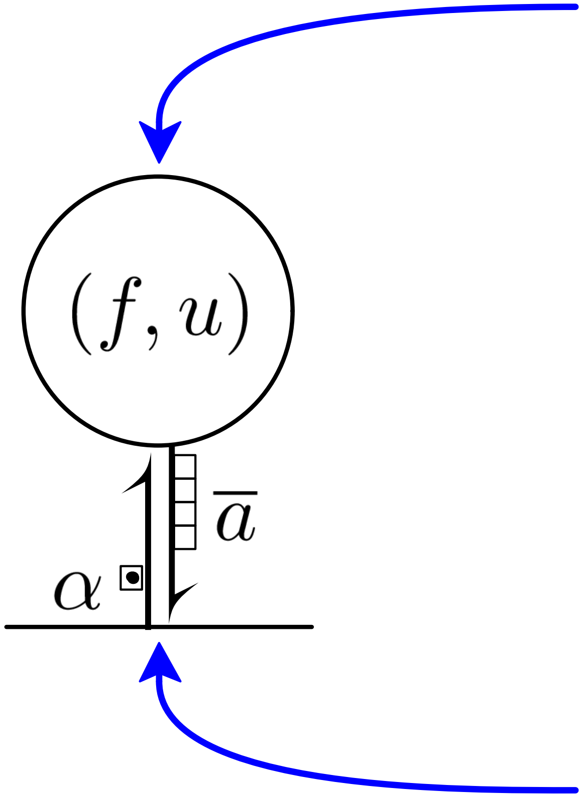
Startup ordering

Crash/exit signalling

Conversation management

Uniform Linguistic Solution

Recipe for Actor Languages



Base language actor

"Network"

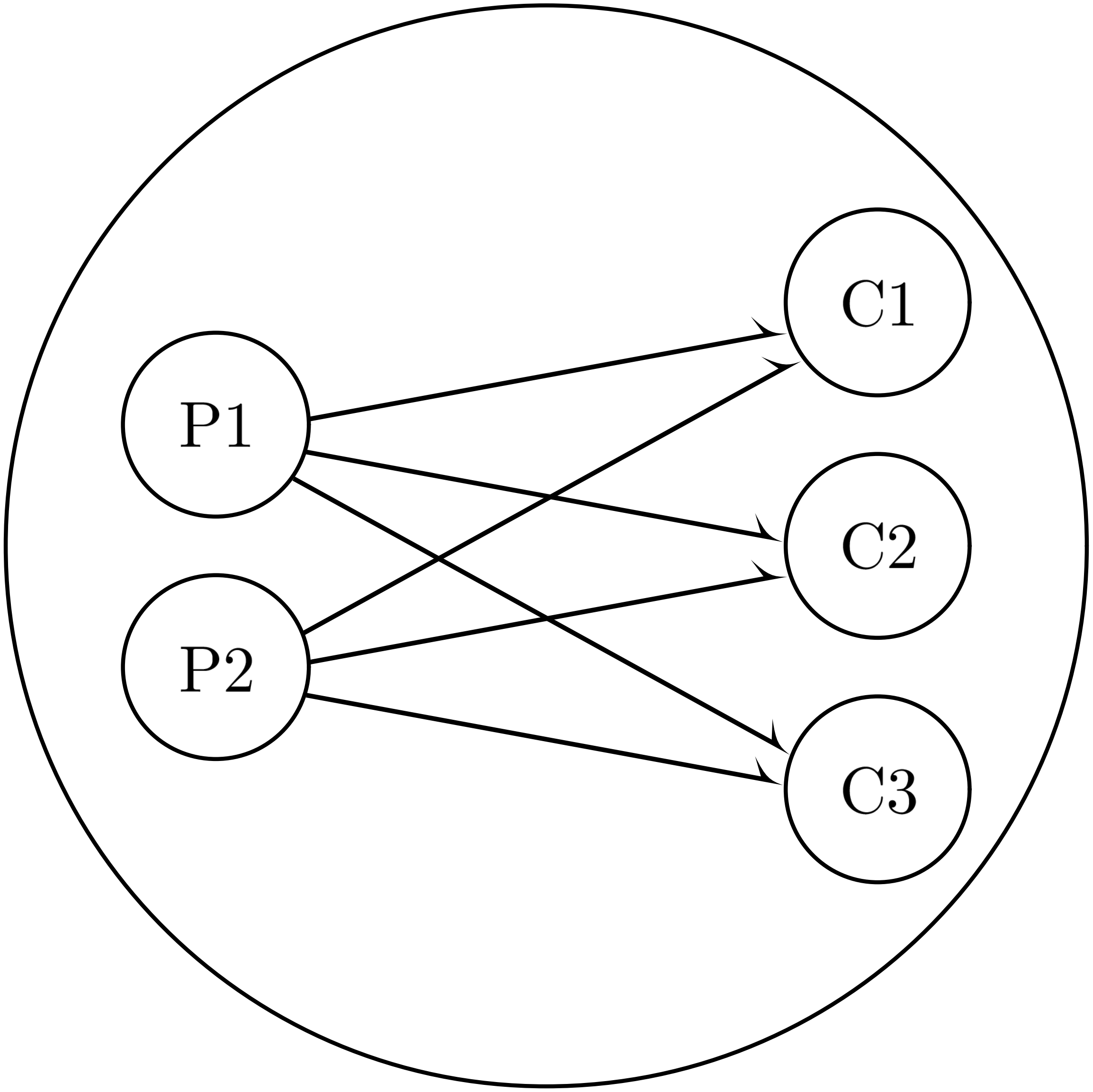
$$f(\alpha, u) = (\overline{a}, u')$$

α = event, a = action

$$\text{Log producers} \xrightarrow{\text{log messages}} \text{Log consumers}$$

$$\langle \mathsf{log}, [\mathit{subsystem}, \mathit{severity}, \mathit{data}] \rangle$$

Consumers filter by subsystem, severity



P1
P2
C1
C2
C3

Logging: Requirements Scorecard

Route log entries from producers to consumers □

Consumers filter log messages □

Decouple producers from consumers □

Avoid shared-state explosion □

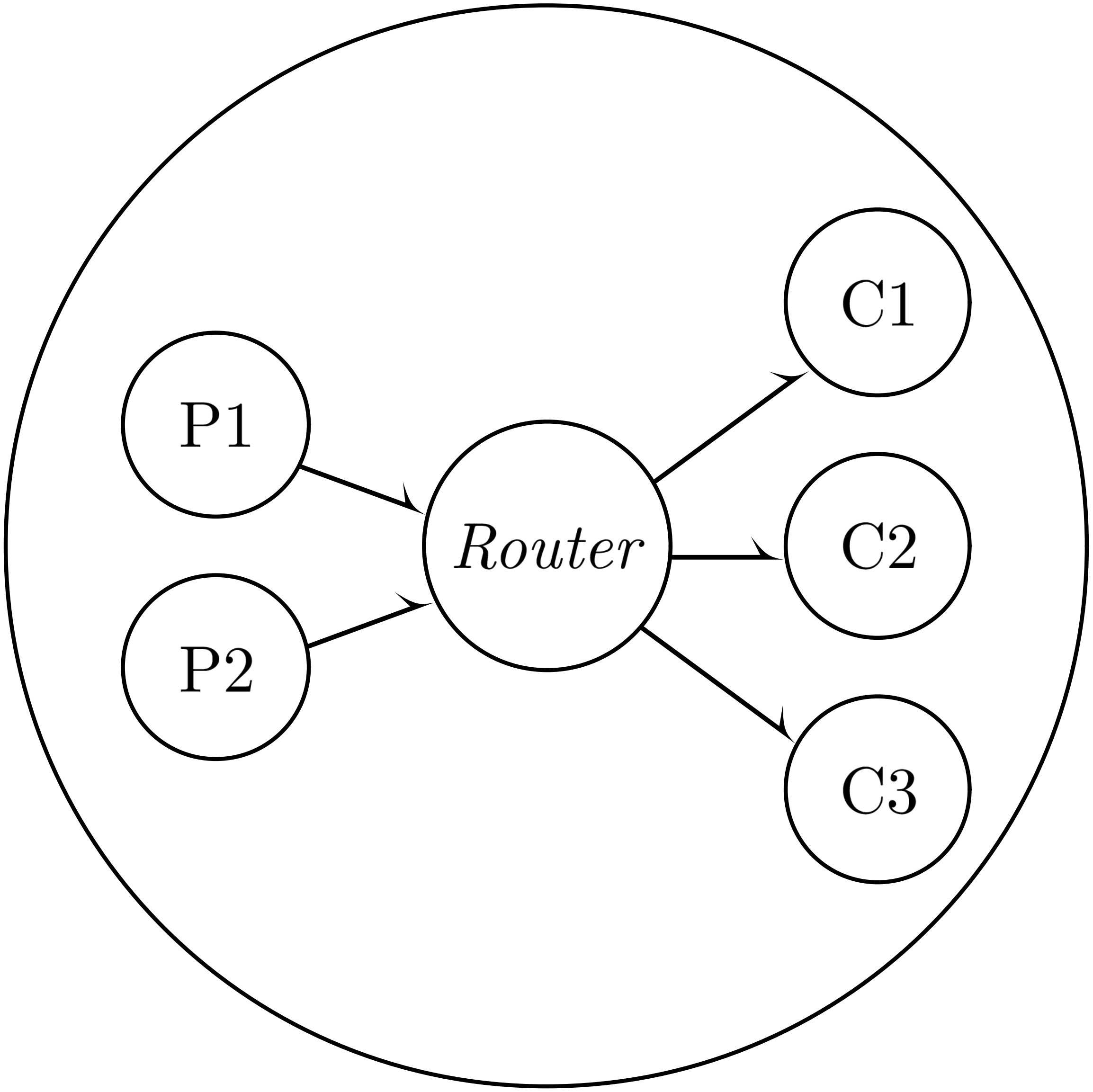
Discovery of logging service □

Only produce if someone's listening □

Alert when a producer crashes/exits □

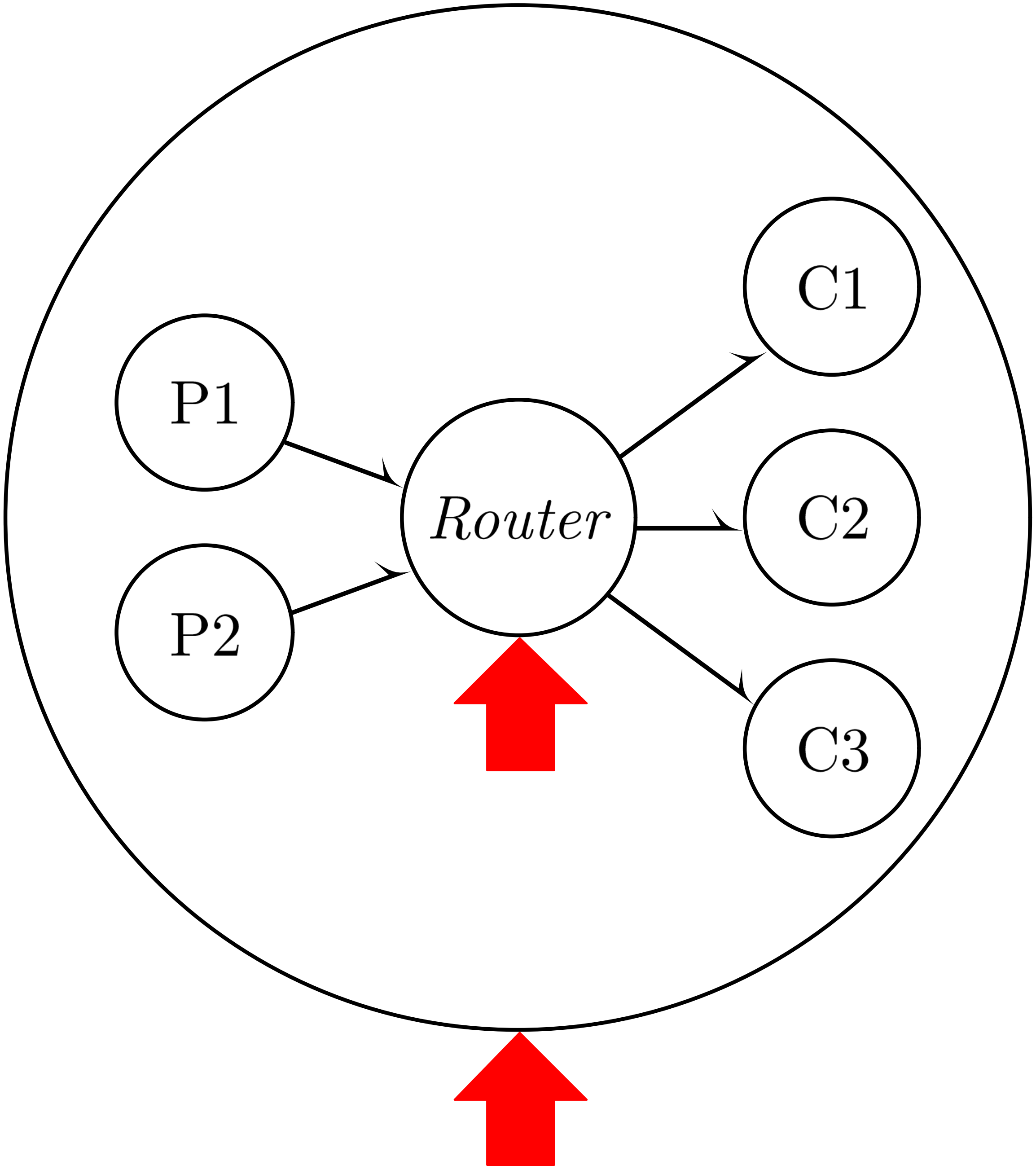
Uniform treatment of I/O □

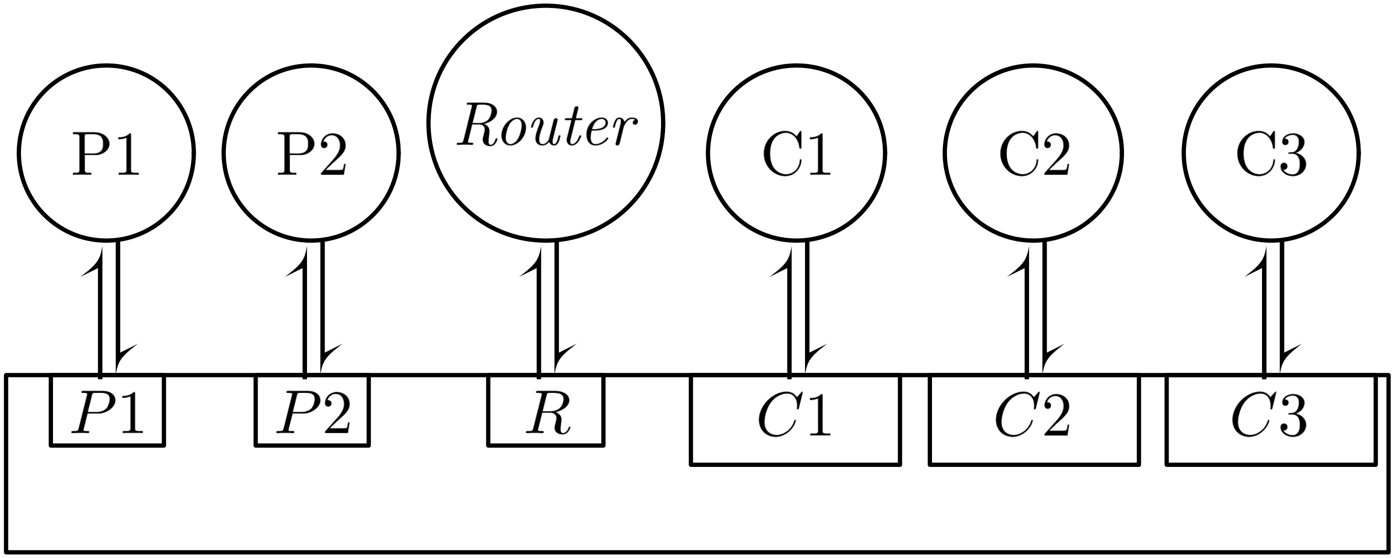
PART II: Why Publish/Subscribe? How?

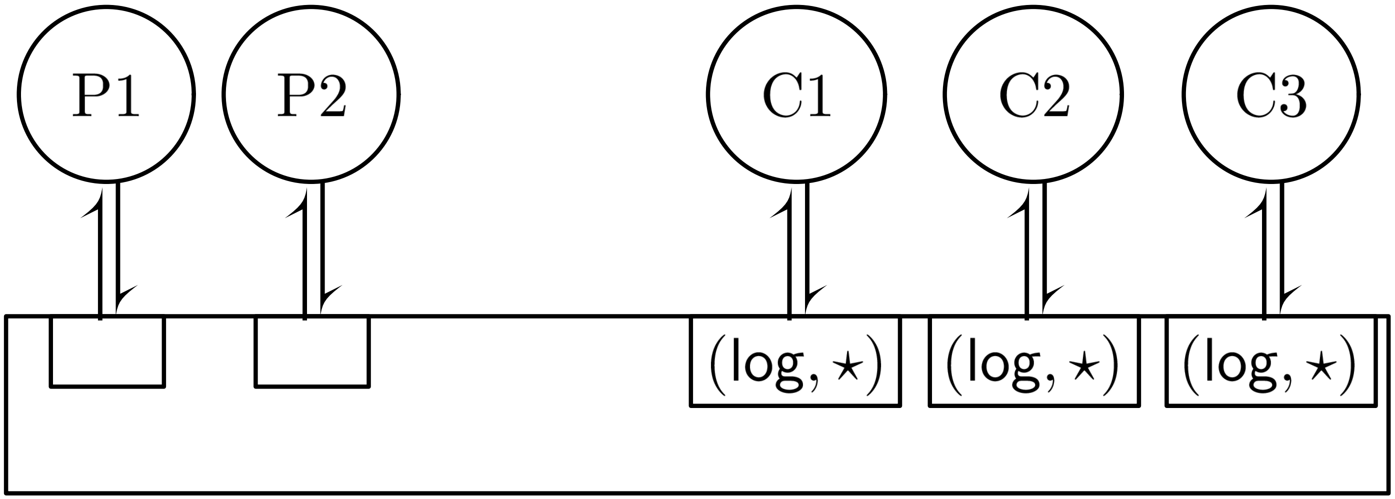

P1
P2
Router
C1
C2
C3

Logging: Requirements Scorecard

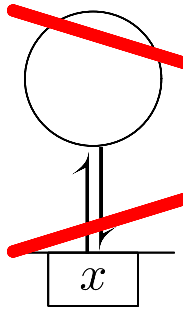
Route log entries from producers to consumers	☑	"Router" actor
Consumers filter log messages	☑	"Router" actor
Decouple producers from consumers	☑	"Router" actor
Avoid shared-state explosion	☑	"Router" actor
Discovery of logging service	□	
Only produce if someone's listening	□	
Alert when a producer crashes/exits	□	
Uniform treatment of I/O	□	


P1
P2
Router
C1
C2
C3


P1
P2
Router
C1
C2
C3
P1
P2
R
C1
C2
C3



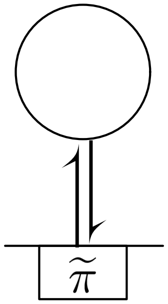
Route by address



~~Messages $m = \langle x, v \rangle$~~

~~$x \in$ Addresses~~

Route by content

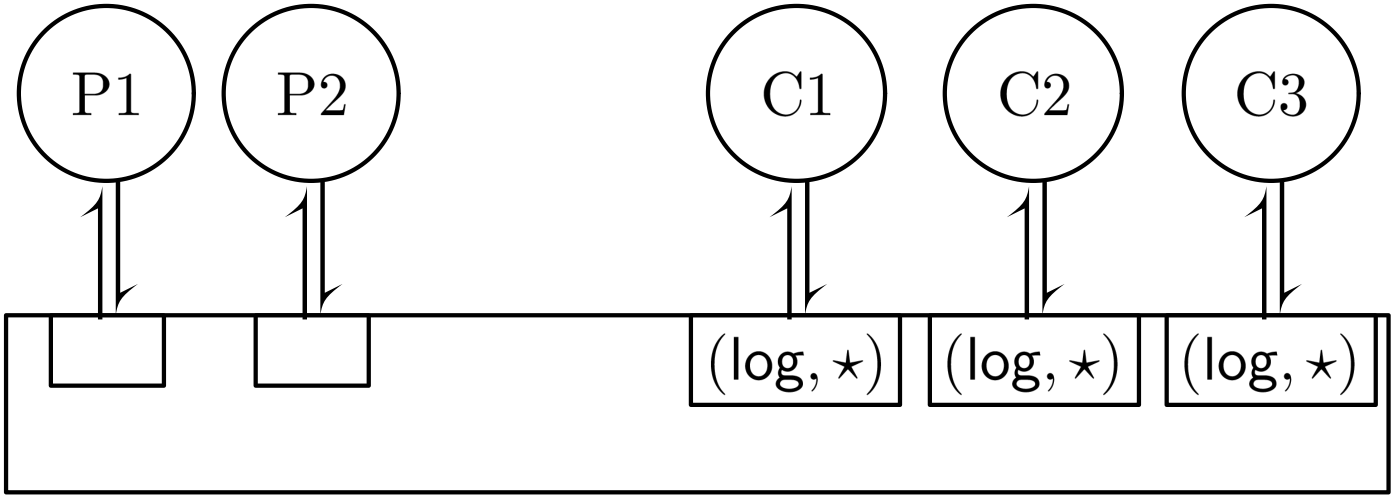


$m = \langle v \rangle$

$v = u \mid v, v$

Patterns $p = u \mid p, p \mid \star$

Interests $\pi = (p)$



Route by address

$(C1, \star)$

Route by content

$(\mathsf{log}, \star)$

or

$(\mathsf{log}, [\star, \mathsf{error}, \star])$

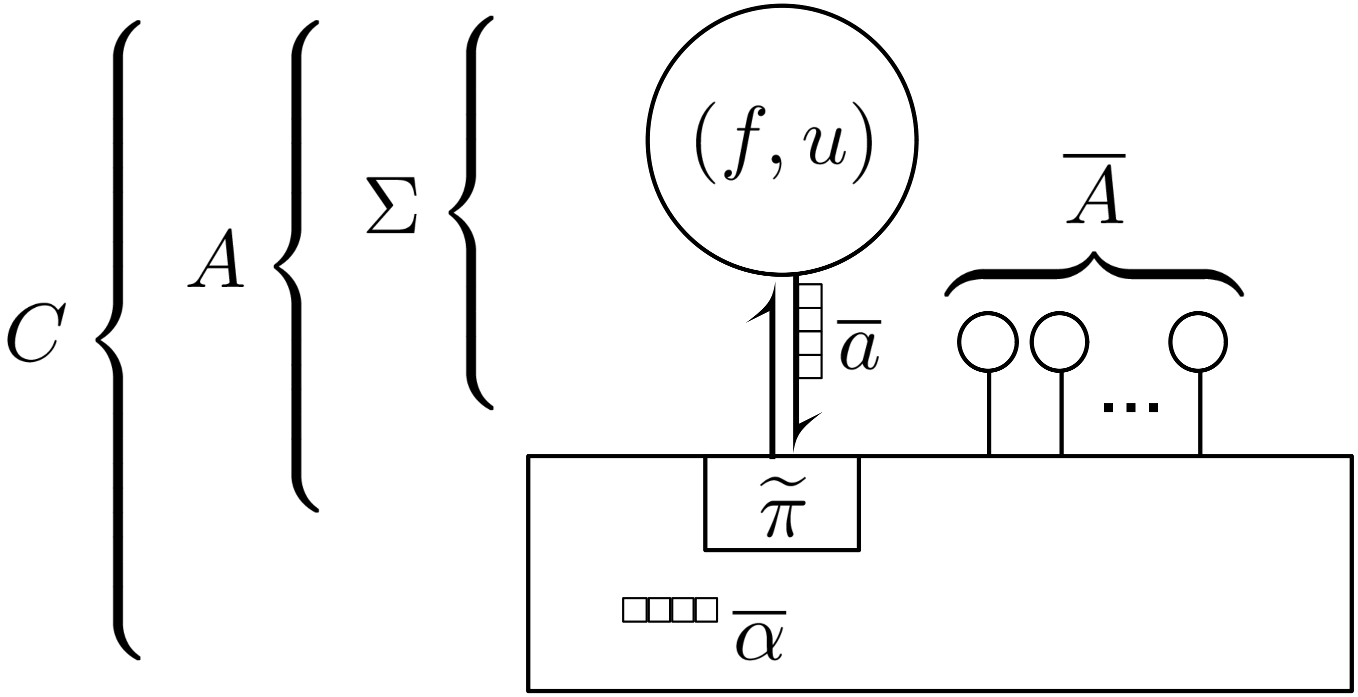
or

$(\mathsf{log}, [\mathrm{P}1, \star, \star])$

or

...

Basic Actor Model + Pub/sub



$f =$ Base language functions	
$u =$ Base language values	
$B = (f, u)$	Behaviors
$\Sigma = \overline{a} \triangleleft B$	Actor States
$A = \widetilde{\pi} : \Sigma$	Actors
$C = [\overline{\alpha}\ ;\ \overline{A}]$	Configurations

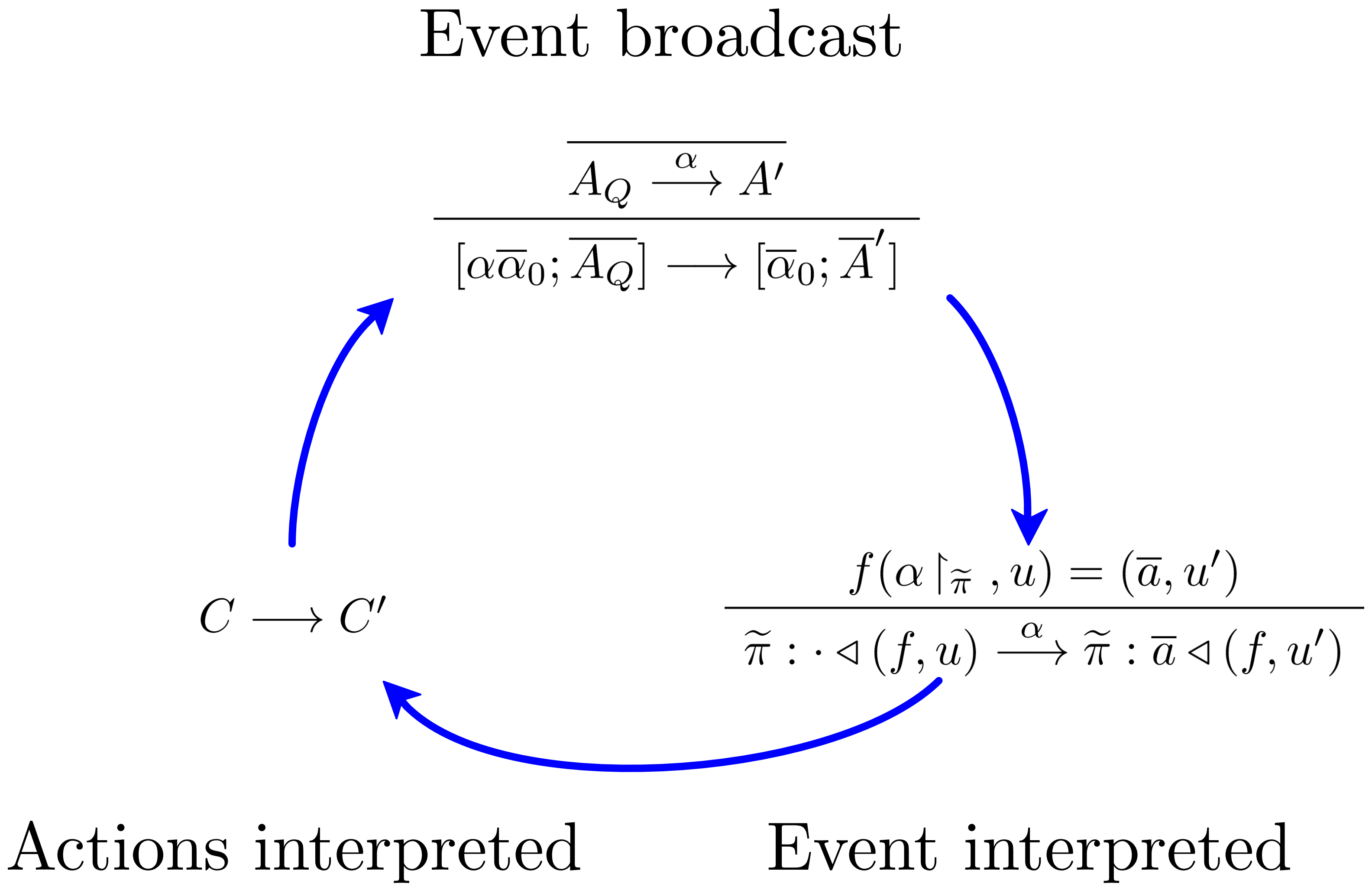
$\alpha = \langle v \rangle$	Events
$a = \alpha \mid A$	Actions
$v = u \mid v, v$	Message values
$p = u \mid p, p \mid \star$	Message patterns
$\pi = (p)$	Interests

$A = \widetilde{\pi} : \overline{a} \triangleleft B$ Actors

$A_Q = \widetilde{\pi} : \cdot \ \triangleleft B$ *Quiescent* Actors

$C = [\overline{\alpha} \, ; \ \overline{A}]$ Configurations

$C_Q = [\ \cdot \ ; \ \overline{A_Q}]$ *Quiescent* Configurations



Event broadcast

$$\frac{\overline{A_Q \xrightarrow{\alpha} A'}}{[\alpha\overline{\alpha}_0; \overline{A_Q}] \longrightarrow [\overline{\alpha}_0; \overline{A}']}$$

$$\frac{f(\alpha \restriction_{\widetilde{\pi}}, u) = (\overline{a}, u')}{\widetilde{\pi} : \cdot \lhd (f, u) \xrightarrow{\alpha} \widetilde{\pi} : \overline{a} \lhd (f, u')}$$

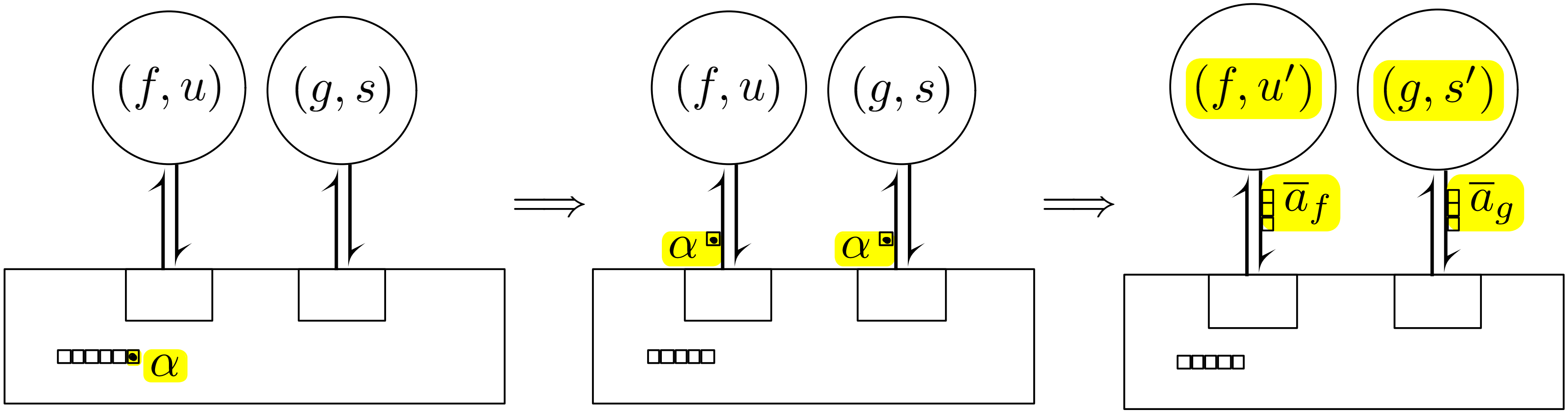
Event interpreted

$$C \longrightarrow C'$$

Actions interpreted

Event Broadcast

$$\frac{\overline{A_Q \xrightarrow{\alpha} A'}}{[\alpha\overline{\alpha}_0; \overline{A_Q}] \longrightarrow [\overline{\alpha}_0; \overline{A}']}$$



Event Filtering

$$\alpha \restriction_{\widetilde{\pi}} : \alpha \times \widetilde{\pi} \rightharpoonup \alpha$$

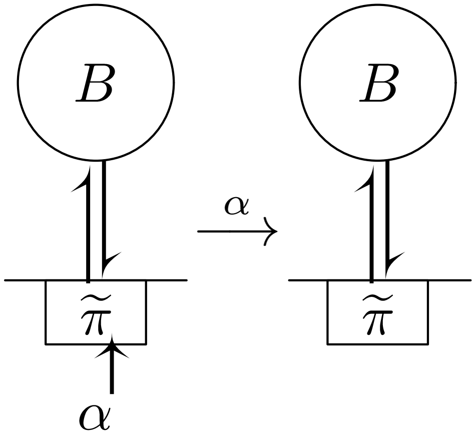
$$v\Big|_{p} : v \times p$$

$$\langle v \rangle \restriction_{\widetilde{\pi}} = \langle v \rangle, \text{ if } \exists(p) \in \widetilde{\pi} \text{ such that } v\Big|_{p}$$

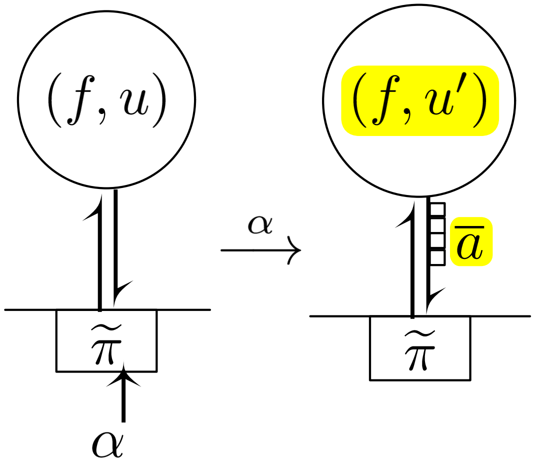
$$\alpha \restriction_{\widetilde{\pi}} \text{ otherwise undefined}$$

Event Interpretation, $A \xrightarrow{\alpha} A$

$$\frac{}{\widetilde{\pi} : \cdot \triangleleft B \xrightarrow{\alpha} \widetilde{\pi} : \cdot \triangleleft B}\ (\alpha \upharpoonright_{\widetilde{\pi}} \ \text{is undefined})$$

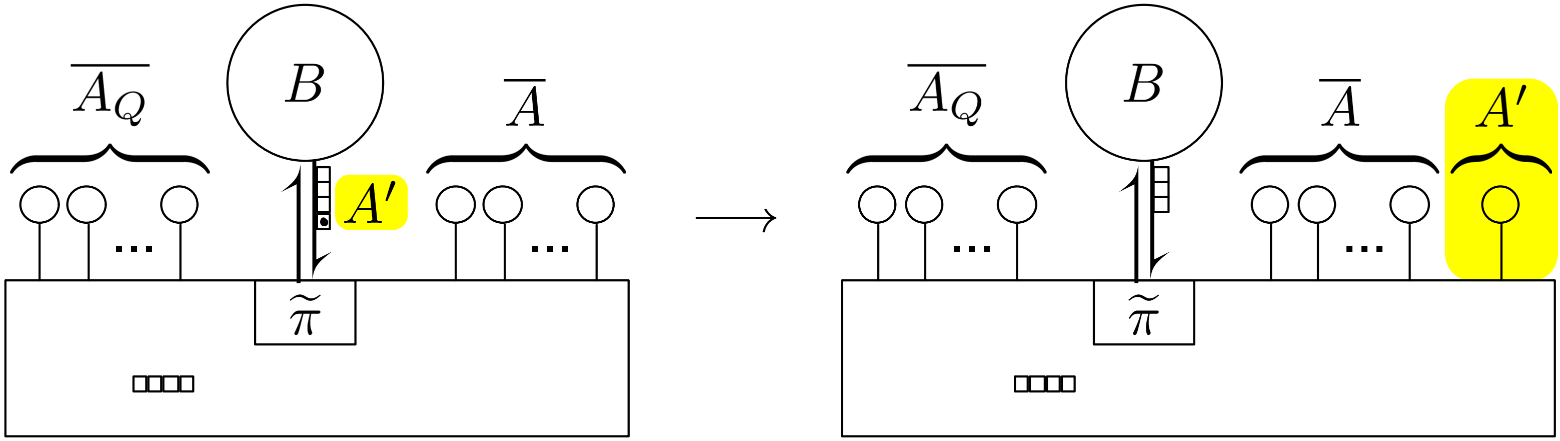


$$\frac{f(\alpha \upharpoonright_{\widetilde{\pi}} , u) = (\overline{a}, u')}{\widetilde{\pi} : \cdot \triangleleft (f, u) \xrightarrow{\alpha} \widetilde{\pi} : \overline{a} \triangleleft (f, u')}\ (\alpha \upharpoonright_{\widetilde{\pi}} \ \text{is defined})$$



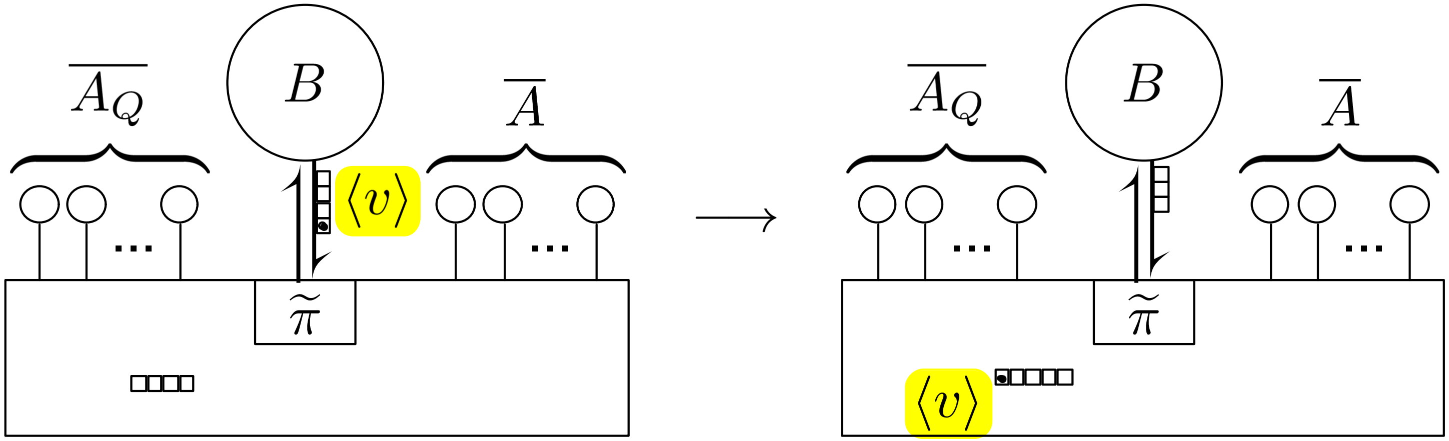
Action Interpretation: Spawn

$$[\overline{\alpha}\;;\;\overline{A_Q}(\widetilde{\pi} : A'\overline{a} \triangleleft B)\overline{A}] \longrightarrow [\overline{\alpha}\;;\;\overline{A_Q}(\widetilde{\pi} : \overline{a} \triangleleft B)\overline{A}\;A']$$



Action Interpretation: Message send

$$[\overline{\alpha}\ ;\ \overline{A_Q}(\widetilde{\pi} : \langle v\rangle\overline{a} \triangleleft B)\overline{A}] \longrightarrow [\overline{\alpha}\langle v\rangle\ ;\ \overline{A_Q}(\widetilde{\pi} : \overline{a} \triangleleft B)\overline{A}]$$



Logging: Requirements Scorecard

Route log entries from producers to consumers	☑ pub/sub
Consumers filter log messages	☑ pub/sub
Decouple producers from consumers	☑ pub/sub
Avoid shared-state explosion	☑ pub/sub
Discovery of logging service	⊟ no need!
Only produce if someone's listening	☐
Alert when a producer crashes/exits	☐
Uniform treatment of I/O	☑ pub/sub

PART III: Why Routing Events? How?

Logging: Requirements Scorecard

Route log entries from producers to consumers ☑
Consumers filter log messages ☑
Decouple producers from consumers ☑
Avoid shared-state explosion ☑
Discovery of logging service □
Only produce if someone's listening □
Alert when a producer crashes/exits □
Uniform treatment of I/O ☑

Shared Conversational Interest

$$\begin{array}{ccccc} \text{Interests} & & \text{Subscription} & & \text{Advertisement} \\ \pi & = & (p) & \mid & \langle p \rangle \end{array}$$

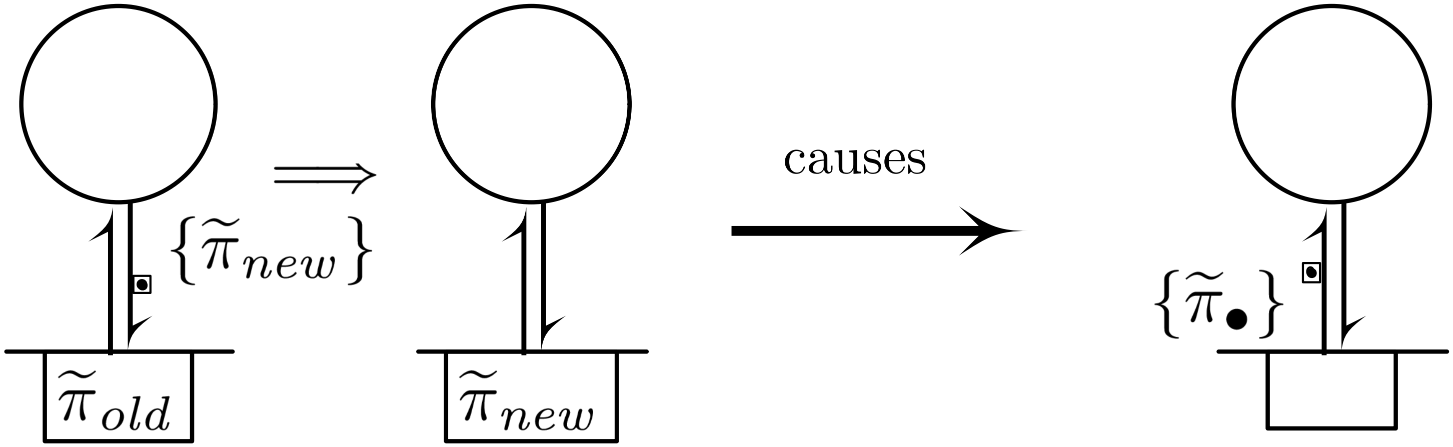
$$\langle p \rangle \not\cap \langle p' \rangle = \emptyset$$
$$(q) \not\cap (q') = \emptyset$$

$$\langle p \rangle \not\cap (q) = \langle p \cap q \rangle$$
$$(q) \not\cap \langle p \rangle = (p \cap q)$$

Any pattern language will do — if it supports ∩

What is a Routing Event?

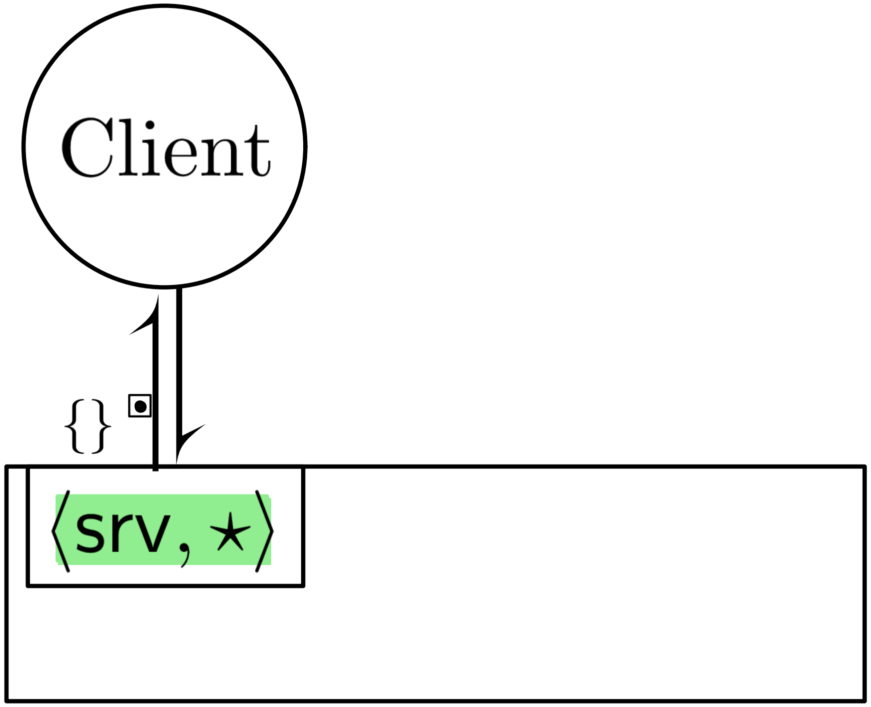
$$\{\widetilde{\pi}\}$$



From Actor to Network

From Network to Actor

Routing Events for Service Discovery

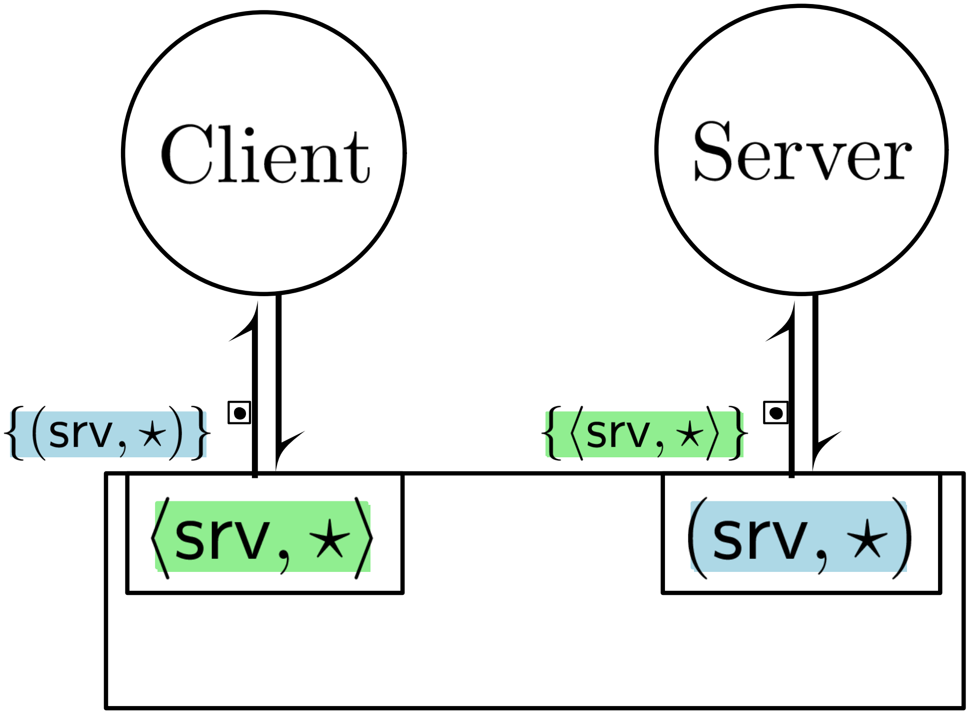


Client

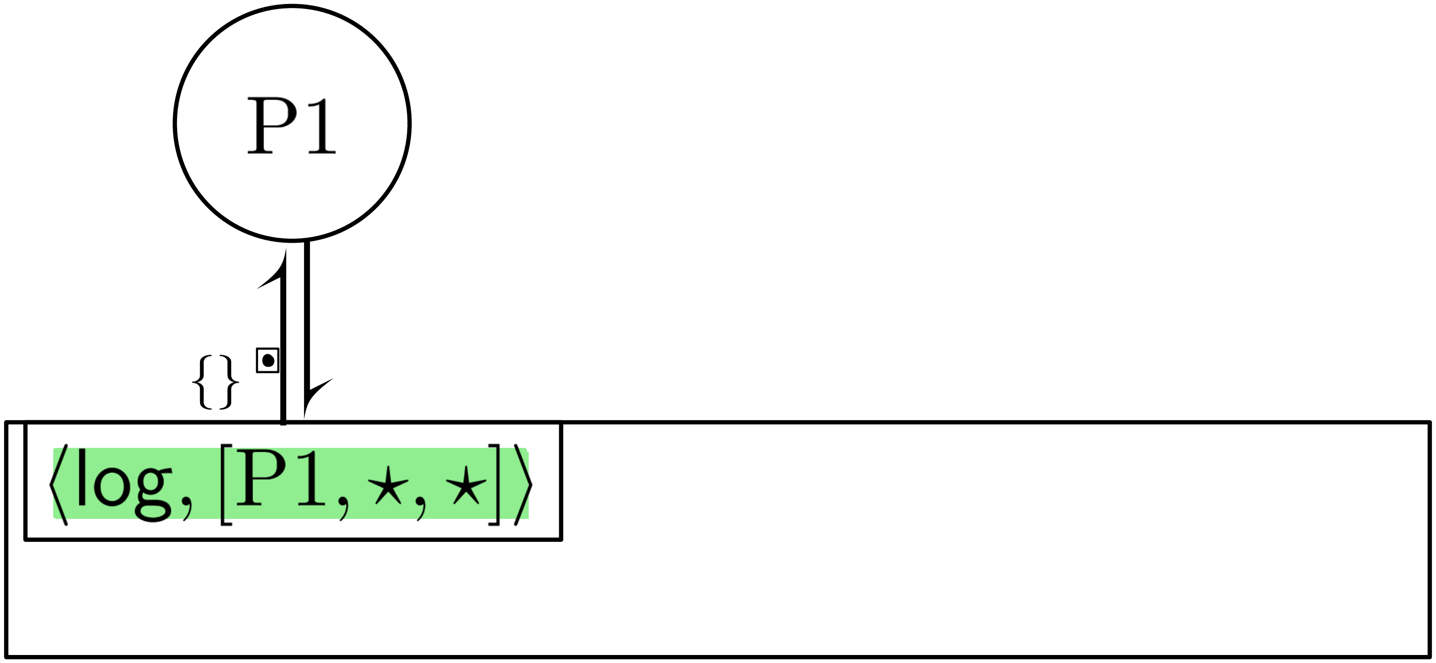
{}

⟨srv, ⋆⟩

Routing Events for Service Discovery


Client
Server
$\{(\mathsf{srv}, \star)\}$
$\boxdot$
$\{\langle \mathsf{srv}, \star \rangle\}$
$\boxdot$
$\langle \mathsf{srv}, \star \rangle$
$(\mathsf{srv}, \star)$

Routing Events for Presence Detection

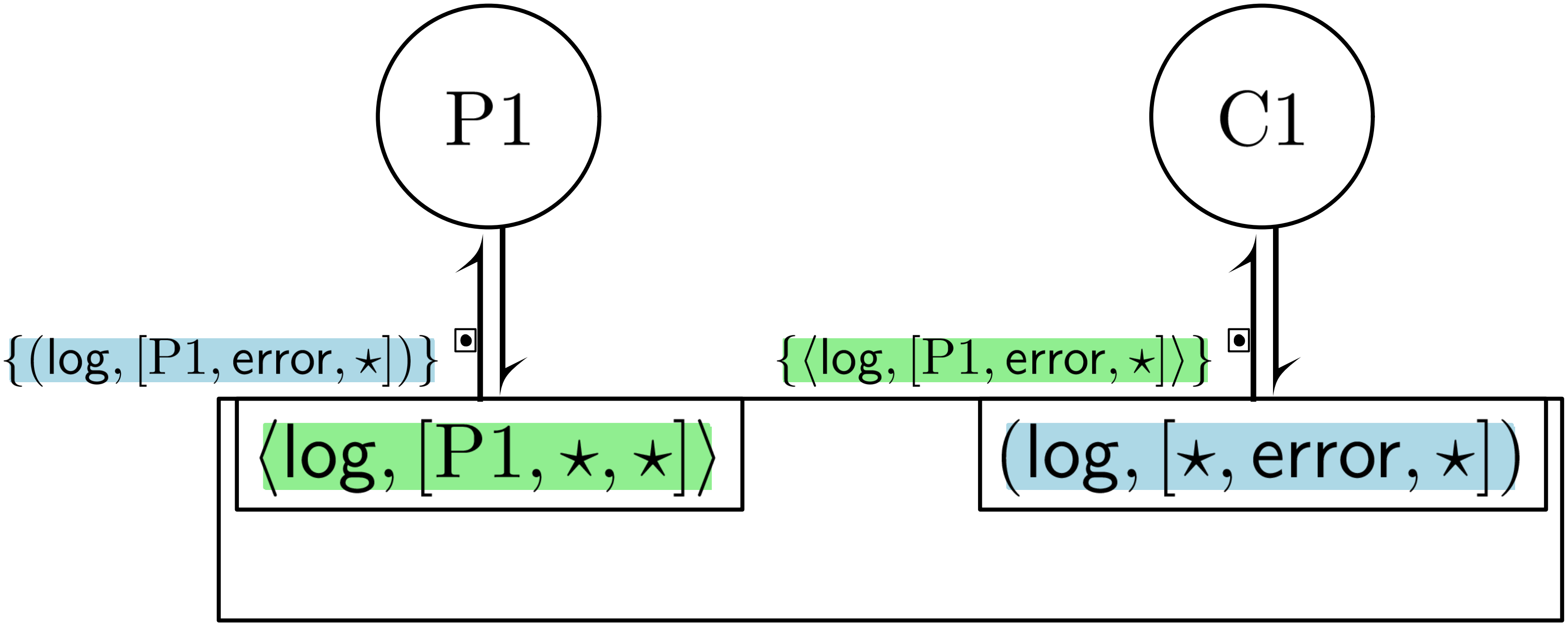


P1

$\{\}$ $\boxdot$

$\langle \mathsf{log}, [\mathrm{P1}, \star, \star] \rangle$

Routing Events for Presence Detection



$\{(\mathsf{log}, [\mathrm{P1}, \mathsf{error}, \star])\}$

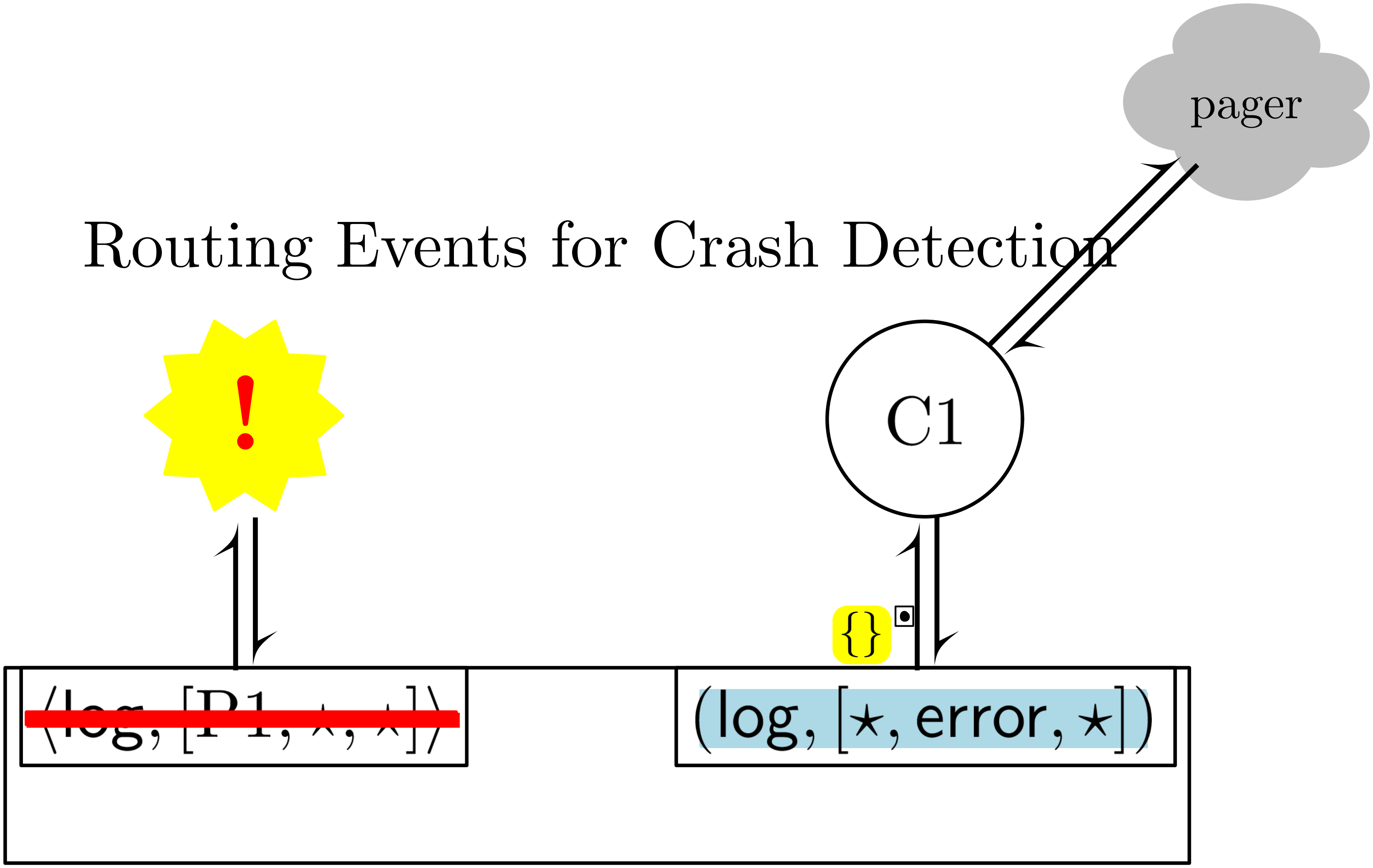
$\{\langle \mathsf{log}, [\mathrm{P1}, \mathsf{error}, \star] \rangle\}$

$\langle \mathsf{log}, [\mathrm{P1}, \star, \star] \rangle$

$(\mathsf{log}, [\star, \mathsf{error}, \star])$

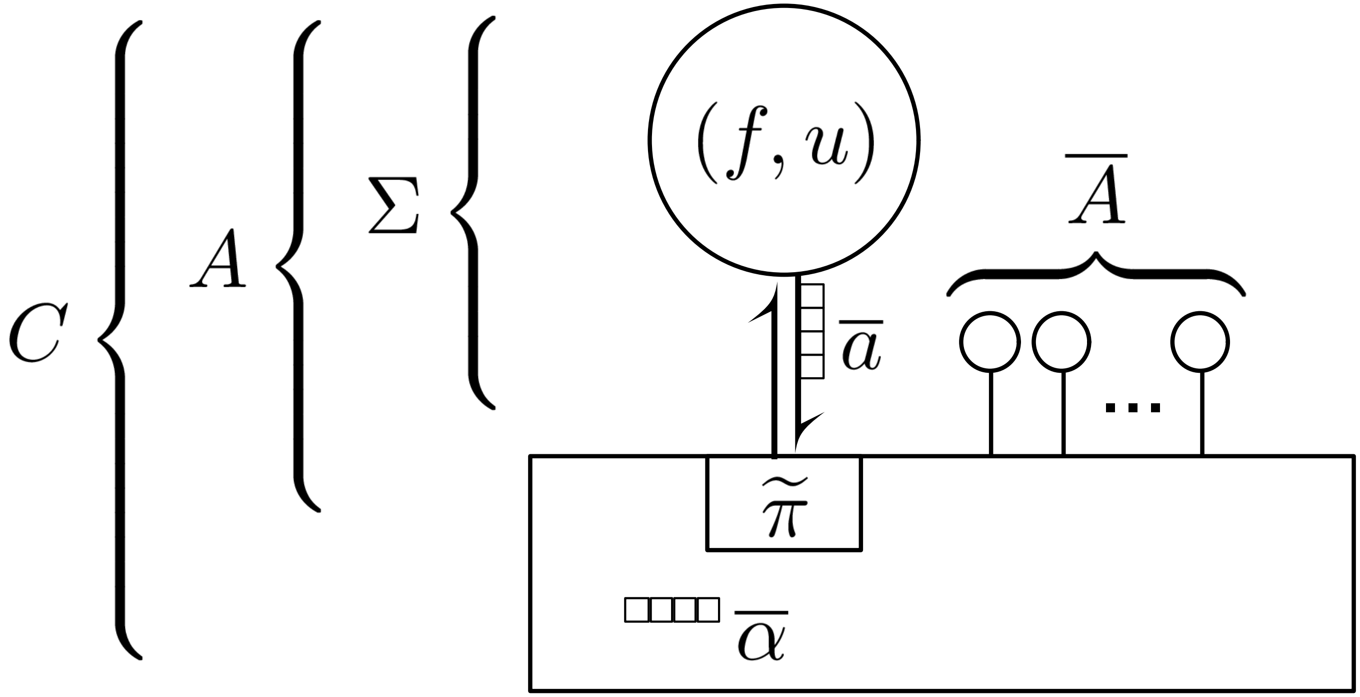
$$\mathsf{log}, [\mathrm{P1}, \star, \star] \quad \cap \quad \mathsf{log}, [\star, \mathsf{error}, \star] \quad = \quad \mathsf{log}, [\mathrm{P1}, \mathsf{error}, \star]$$

Routing Events for Crash Detection



cf. Erlang's links/monitors [Armstrong 2003]

Basic Actor Model + Pub/sub + Routing Events



$f =$ Base language functions

$u =$ Base language values

$B = (f, u)$ Behaviors

$\Sigma = \overline{a} \triangleleft B$ Actor States

$A = \widetilde{\pi} : \Sigma$ Actors

$C = [\overline{\alpha} \,;\, \overline{A}]$ Configurations

$\alpha = \langle v \rangle \;\mid\; \{\widetilde{\pi}\}$ Events

$a = \alpha \;\mid\; A$ Actions

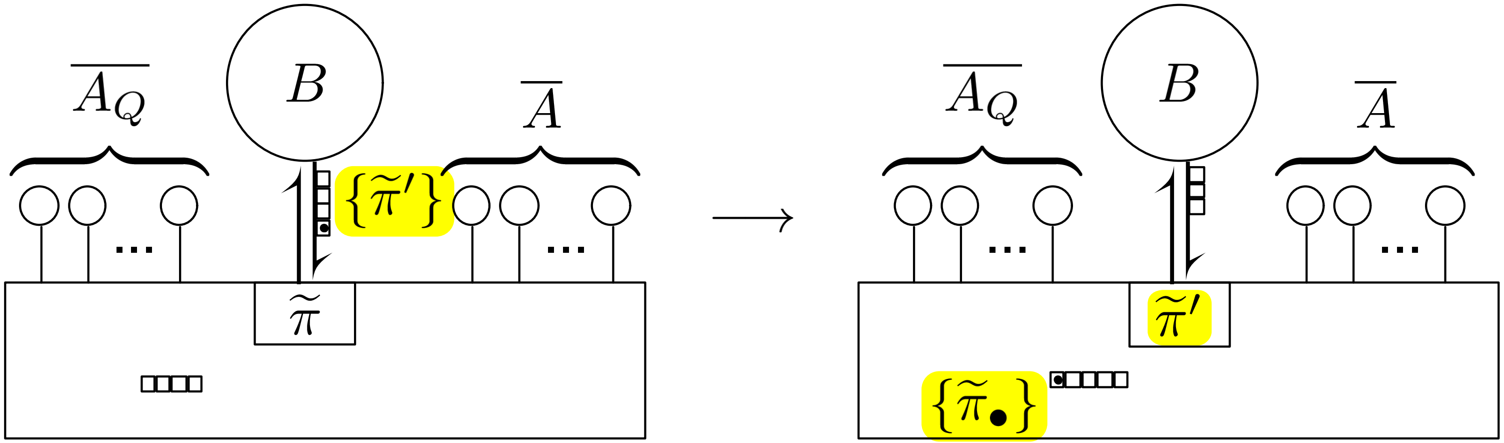
$v = u \;\mid\; v, v$ Message values

$p = u \;\mid\; p, p \;\mid\; \star$ Message patterns

$\pi = (p) \;\mid\; \langle p \rangle$ Interests

Action Interpretation: Routing event

$$\begin{array}{rl} & [\overline{\alpha} \quad ; \overline{A_Q}(\widetilde{\pi} : \{\widetilde{\pi}'\}\overline{a} \triangleleft B)\overline{A}] \\ \longrightarrow & [\overline{\alpha}\{\widetilde{\pi}_\bullet\} ; \overline{A_Q}(\widetilde{\pi}' : \quad \overline{a} \triangleleft B)\overline{A}] \end{array}$$



$$\widetilde{\pi}_\bullet = \overline{interests(A_Q)} \cup \widetilde{\pi}' \cup \overline{interests(A)}$$

Event Filtering

$$\alpha \upharpoonright_{\widetilde{\pi}} : \alpha \times \widetilde{\pi} \rightharpoonup \alpha$$

$$v\Big|_{p} : v \times p$$

$$\begin{aligned}
\langle v \rangle \upharpoonright_{\widetilde{\pi}} \; &= \langle v \rangle, \text{ if } \exists(p) \in \widetilde{\pi} \text{ such that } v\Big|_{p} \\
\{\widetilde{\pi}_1\} \upharpoonright_{\widetilde{\pi}_2} &= \{(\pi_{11} \not\cap \pi_{21}) \cup \cdots \cup (\pi_{11} \not\cap \pi_{2m}) \cup \\
&\qquad (\pi_{12} \not\cap \pi_{21}) \cup \cdots \cup (\pi_{12} \not\cap \pi_{2m}) \cup \\
&\qquad\qquad \vdots \qquad\qquad\qquad\qquad \vdots \\
&\qquad (\pi_{1n} \not\cap \pi_{21}) \cup \cdots \cup (\pi_{1n} \not\cap \pi_{2m})\}
\end{aligned}$$

$$\alpha \upharpoonright_{\widetilde{\pi}} \text{ otherwise undefined}$$

Logging: Requirements Scorecard

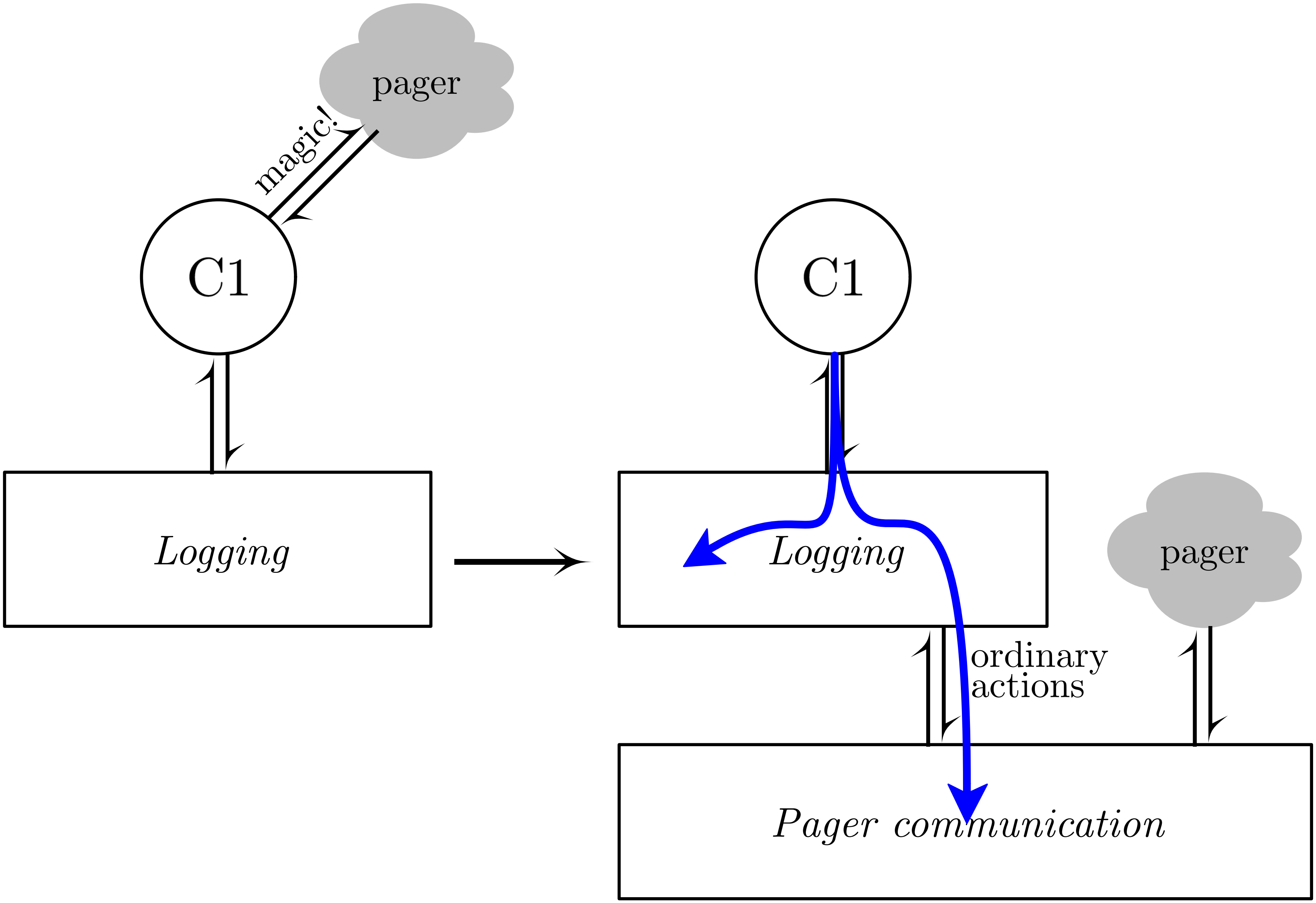
Route log entries from producers to consumers	☑	pub/sub
Consumers filter log messages	☑	pub/sub
Decouple producers from consumers	☑	pub/sub
Avoid shared-state explosion	☑	pub/sub
Discovery of logging service	☑	routing events
Only produce if someone's listening	☑	routing events
Alert when a producer crashes/exits	☑	routing events
Uniform treatment of I/O	⊟	not finished!

PART IV: Why Hierarchical Layering? How?

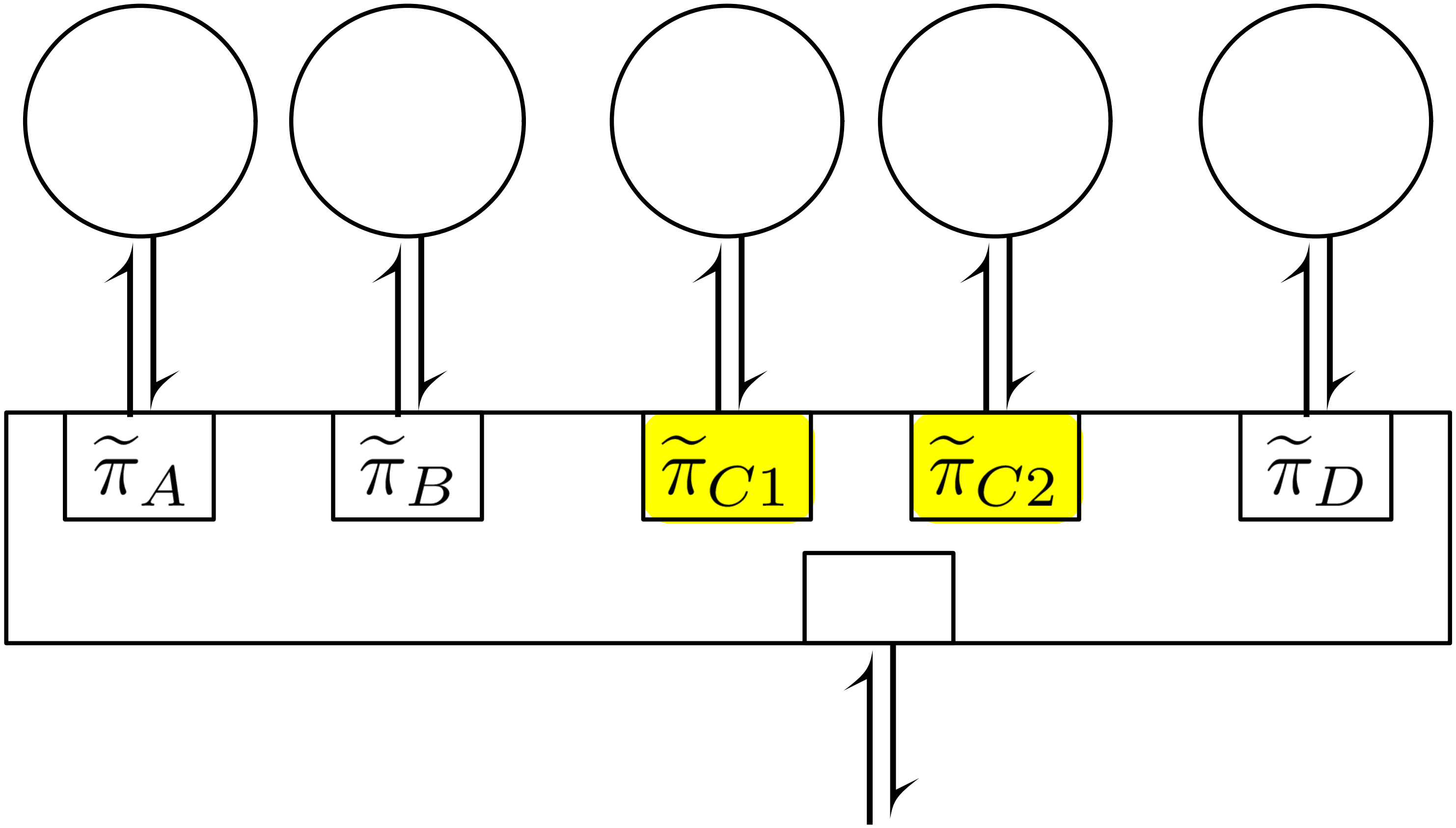
Logging: Requirements Scorecard

Route log entries from producers to consumers ☑
Consumers filter log messages ☑
Decouple producers from consumers ☑
Avoid shared-state explosion ☑
Discovery of logging service ☑
Only produce if someone's listening ☑
Alert when a producer crashes/exits ☑
Uniform treatment of I/O □ not finished!

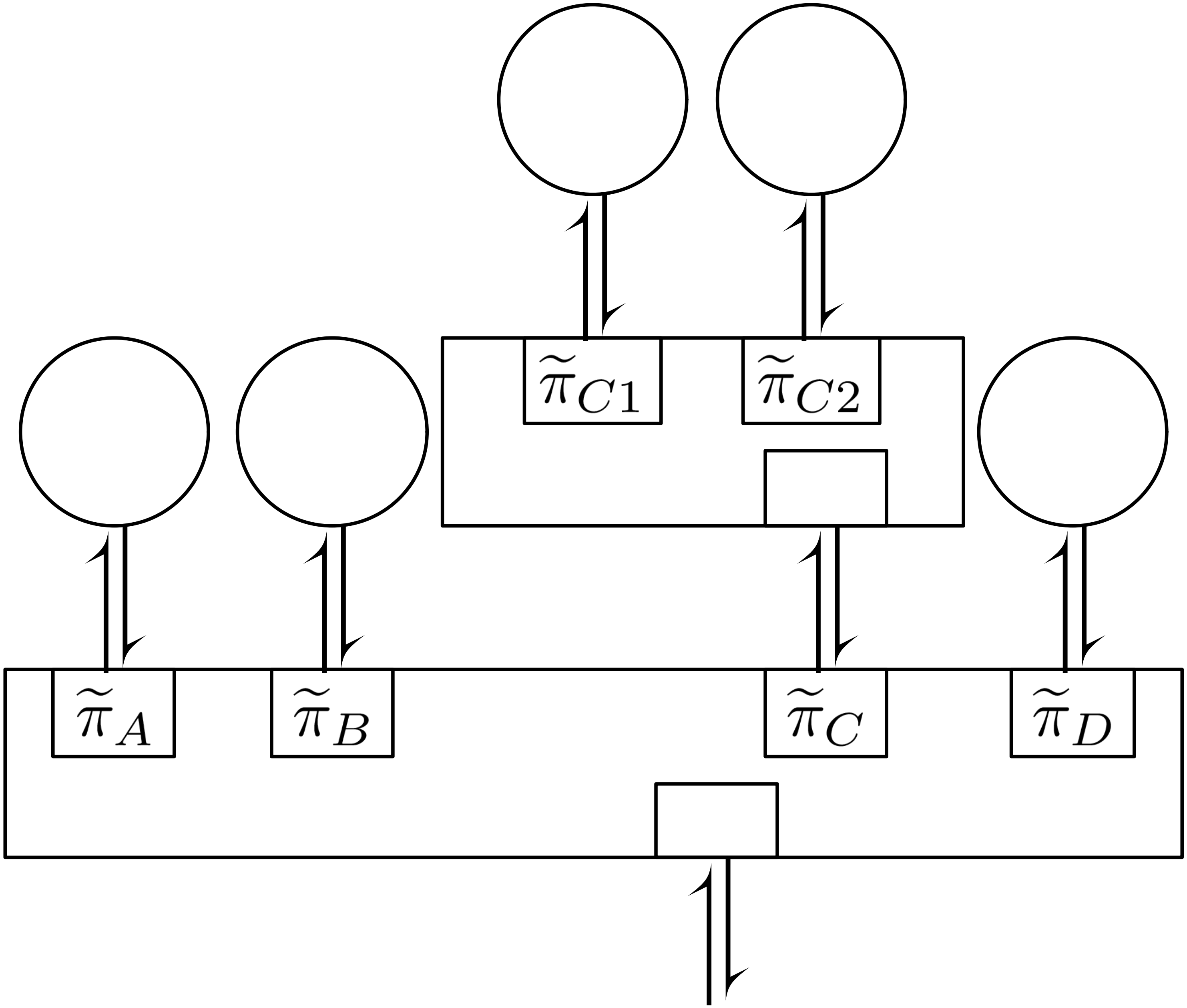
Layers make I/O Uniform



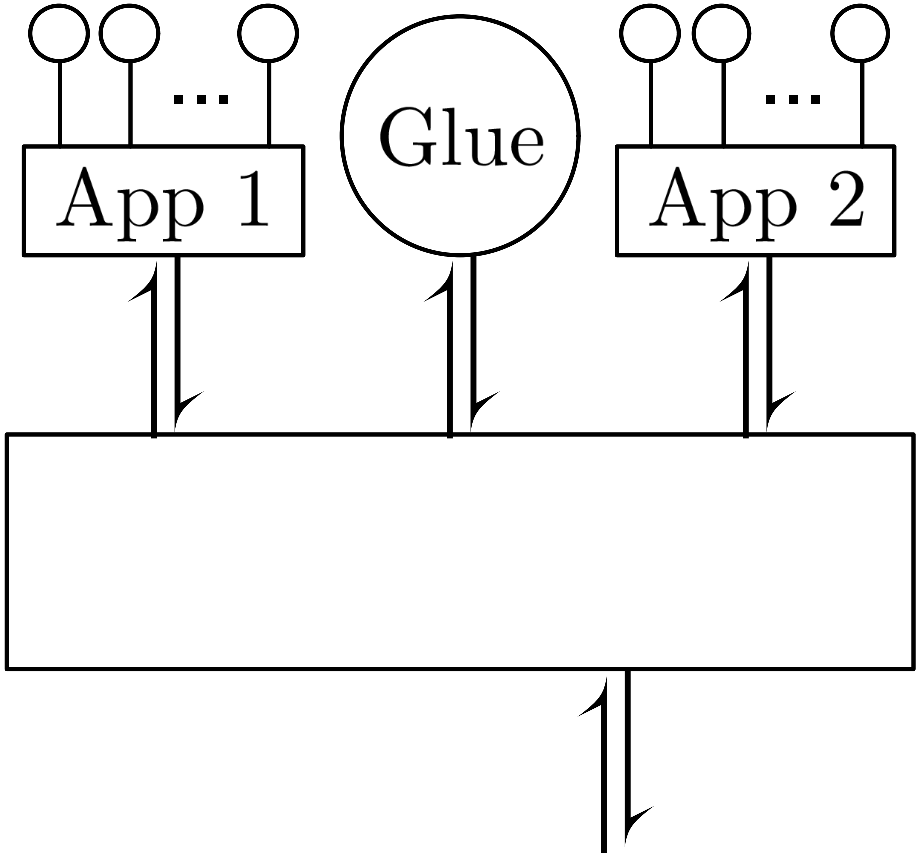
Layers Scope Conversations



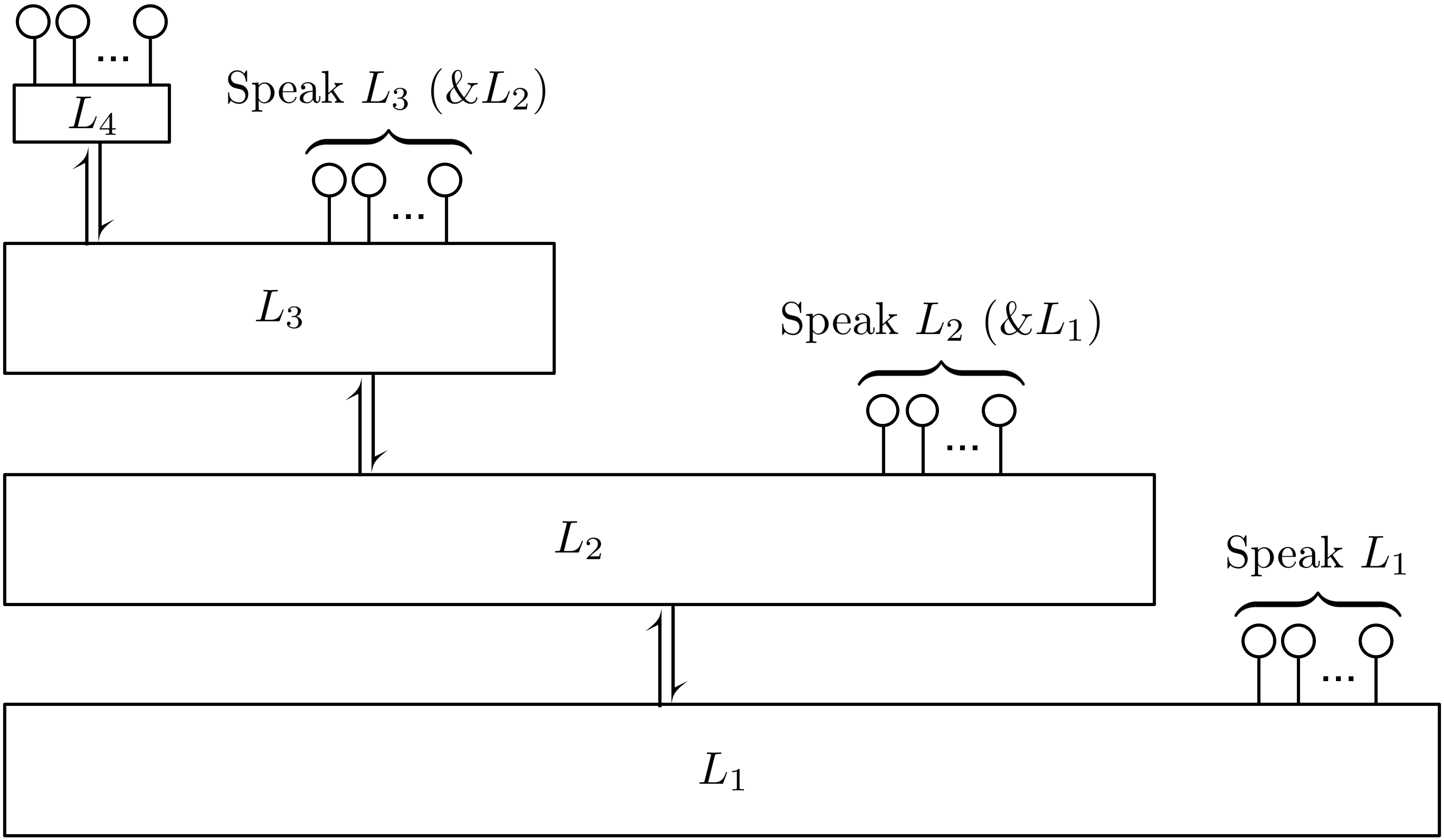
Layers Scope Conversations



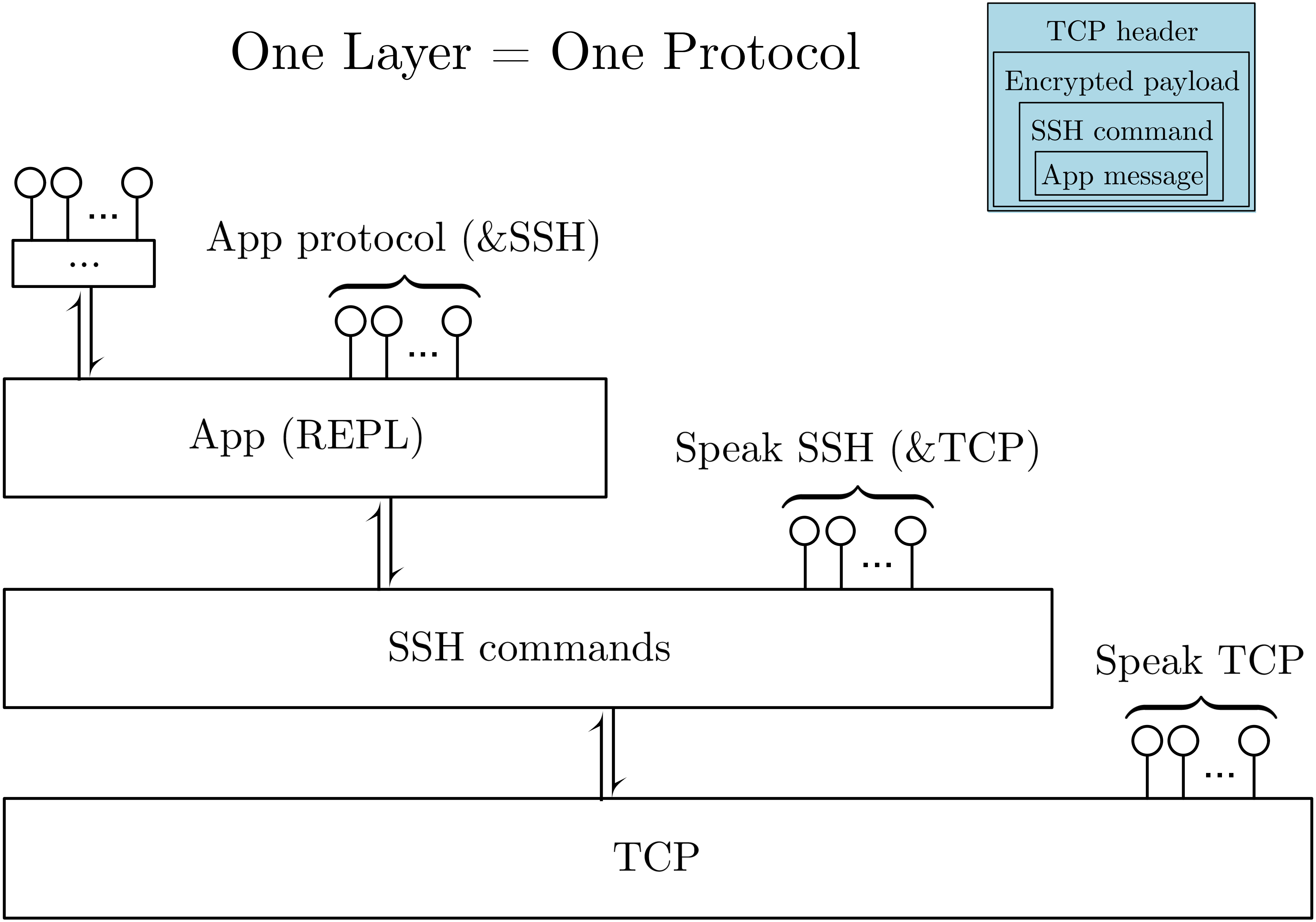
Layers Compose



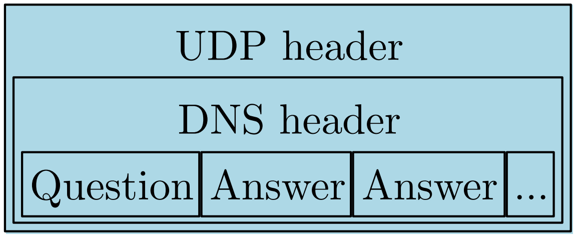
One Layer = One Protocol



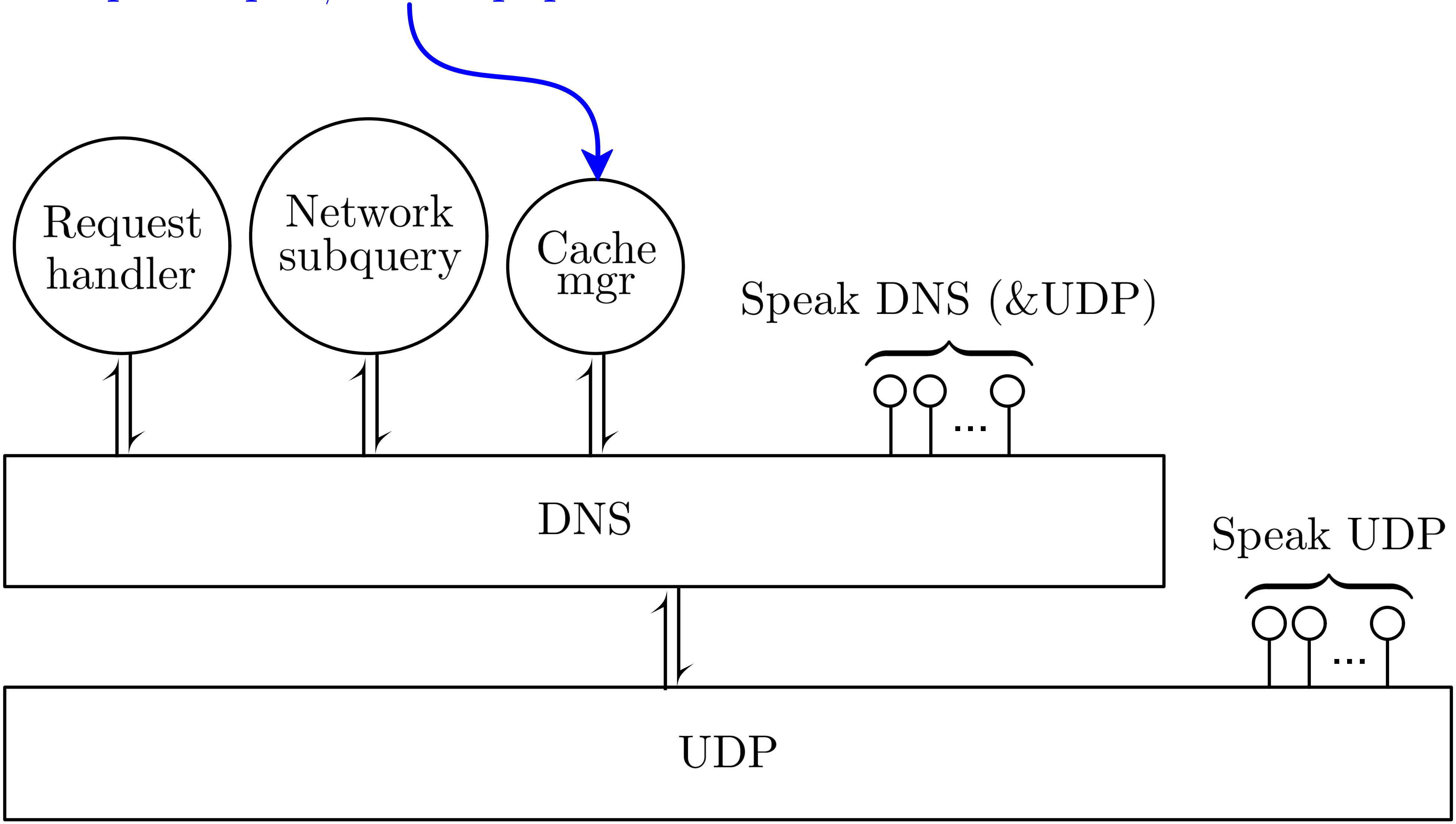
One Layer = One Protocol



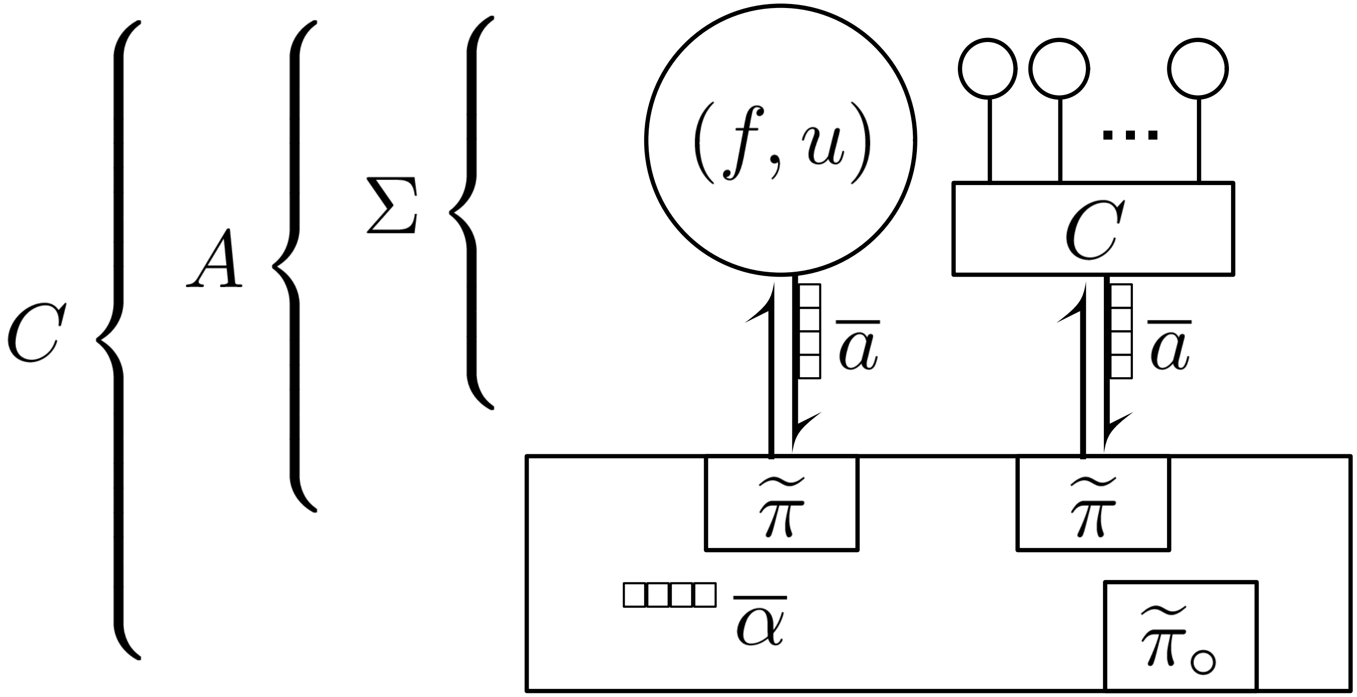
One Layer = One Protocol



Snoops via pub/sub to populate cache!



Full Network Calculus

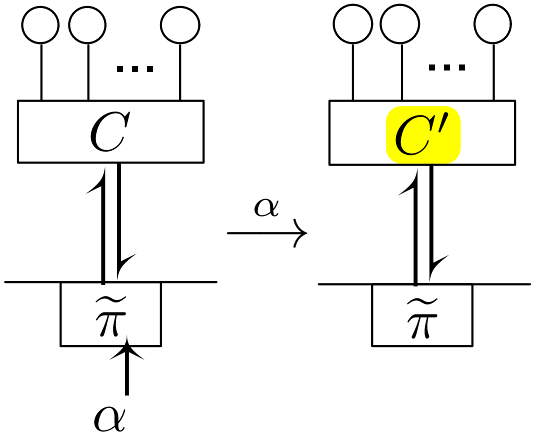


$f =$ Base language functions	
$u =$ Base language values	
$B = (f, u) \ \mid \ C$	Behaviors
$\Sigma = \overline{a} \triangleleft B$	Actor States
$A = \widetilde{\pi} : \Sigma$	Actors
$C = [\overline{\alpha}\ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A}]$	Configurations

$\alpha = m \ \mid \ \{\widetilde{\pi}\}$	Events
$a = \alpha \ \mid \ A$	Actions
$m = \langle v \rangle \ \mid \ \rfloor\pi$	Messages
$v = u \ \mid \ v, v$	Message values
$p = u \ \mid \ p, p \ \mid \ \star$	Message patterr
$\pi = (p) \ \mid \ \langle p \rangle \ \mid \ \rfloor\pi$	Interests

Event Interpretation, $A \xrightarrow{\alpha} A$

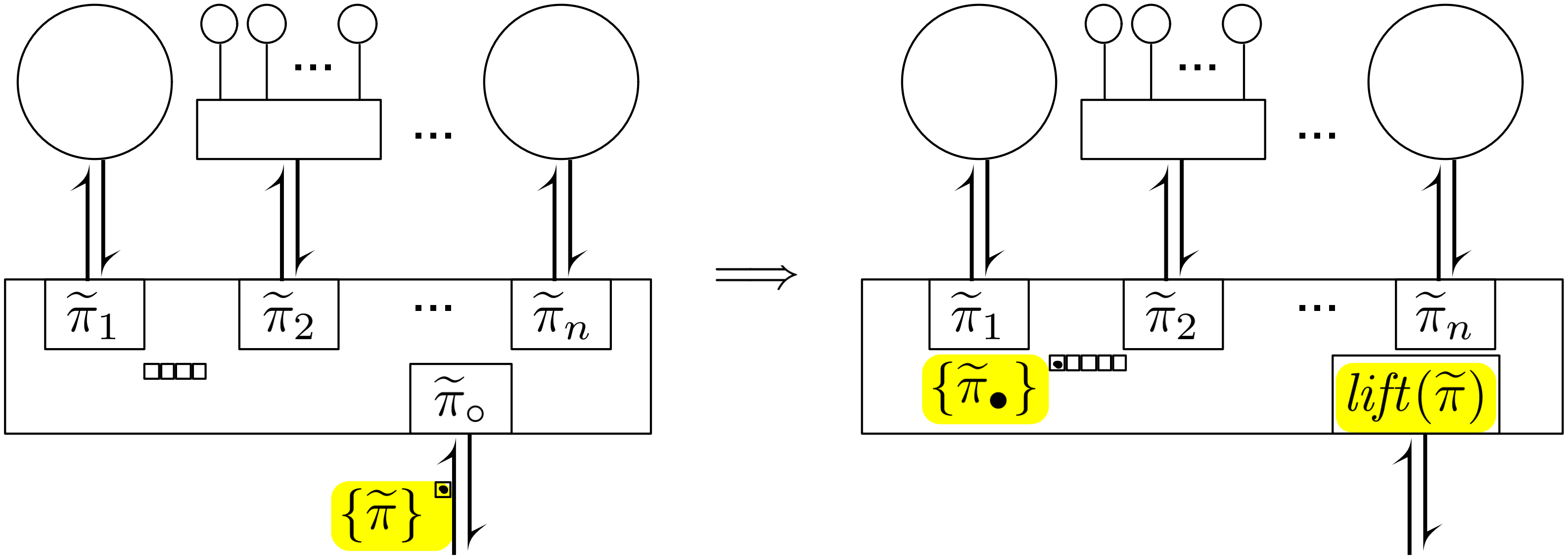
$$\frac{\mathit{inject}\,(\alpha \restriction_{\widetilde{\pi}}\ , C) = C'}{\widetilde{\pi} : \cdot \triangleleft C \xrightarrow{\alpha} \widetilde{\pi} : \cdot \triangleleft C'}\ (\alpha \restriction_{\widetilde{\pi}} \text{ is defined})$$



Event Interpretation: Routing event arrival

$$\mathit{inject} : \alpha \times C \to C$$

$$\mathit{inject}\ (\{\widetilde{\pi}\}, [\overline{\alpha}\ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A}]) = [\overline{\alpha}\ \{\widetilde{\pi}_\bullet\}\ \ ;\ \mathit{lift}(\widetilde{\pi})\ \ ;\ \overline{A}]$$

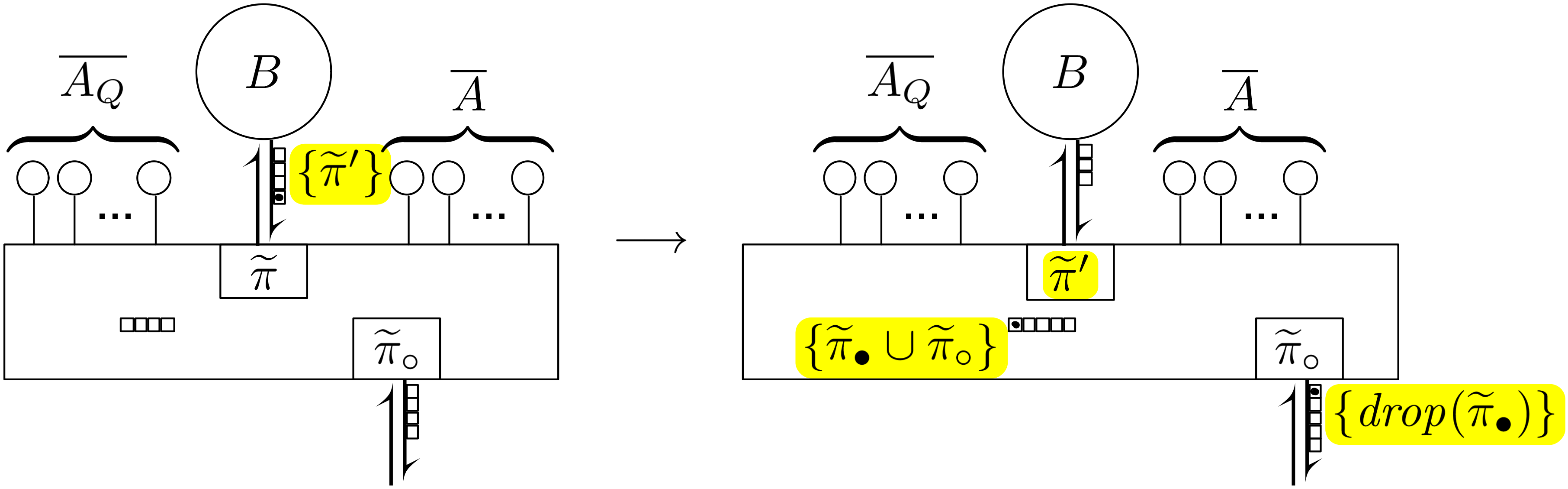


$$\widetilde{\pi}_\bullet = \widetilde{\pi}_1 \cup \widetilde{\pi}_2 \cup \cdots \cup \widetilde{\pi}_n \cup \mathit{lift}(\widetilde{\pi})$$

$$\begin{array}{c} C \longrightarrow C' \\ \Big\Downarrow \\ \overline{a} \lhd C \longrightarrow \overline{a}' \lhd C' \end{array}$$

Action Interpretation: Routing event (with layering)

$$\begin{array}{rl} & \overline{a}_0 \lhd [\overline{\alpha} \; ; \; \widetilde{\pi}_\circ \; ; \; \overline{A_Q}(\widetilde{\pi} : \; \{\widetilde{\pi}'\}\overline{a} \lhd B)\overline{A}] \\ \longrightarrow & \overline{a}_0\{drop(\widetilde{\pi}_\bullet)\} \lhd [\overline{\alpha}\{\widetilde{\pi}_\bullet\widetilde{\pi}_\circ\} \; ; \; \widetilde{\pi}_\circ \; ; \; \overline{A_Q}(\widetilde{\pi}' : \; \overline{a} \lhd B)\overline{A}] \end{array}$$



$$\widetilde{\pi}_\bullet = \overline{interests(A_Q)} \cup \widetilde{\pi}' \cup \overline{interests(A)}$$

Logging: Requirements Scorecard

Route log entries from producers to consumers	☑	pub/sub
Consumers filter log messages	☑	pub/sub
Decouple producers from consumers	☑	pub/sub
Avoid shared-state explosion	☑	pub/sub
Discovery of logging service	☑	routing events
Only produce if someone's listening	☑	routing events
Alert when a producer crashes/exits	☑	routing events
Uniform treatment of I/O	☑	layering
+ great additional benefits from layering	☑	

PART V: Conclusions

Marketplace

Typed Racket

DNS server (UDP)
SSH server (TCP)
Chat server
Echo server

Minimart

Racket

Websocket driver
Generic msg broker

JS-Marketplace

Javascript

Websocket driver
DOM driver
jQuery driver
Chat + roster
GUI composition

Details and experience report in the paper!

Thank you!

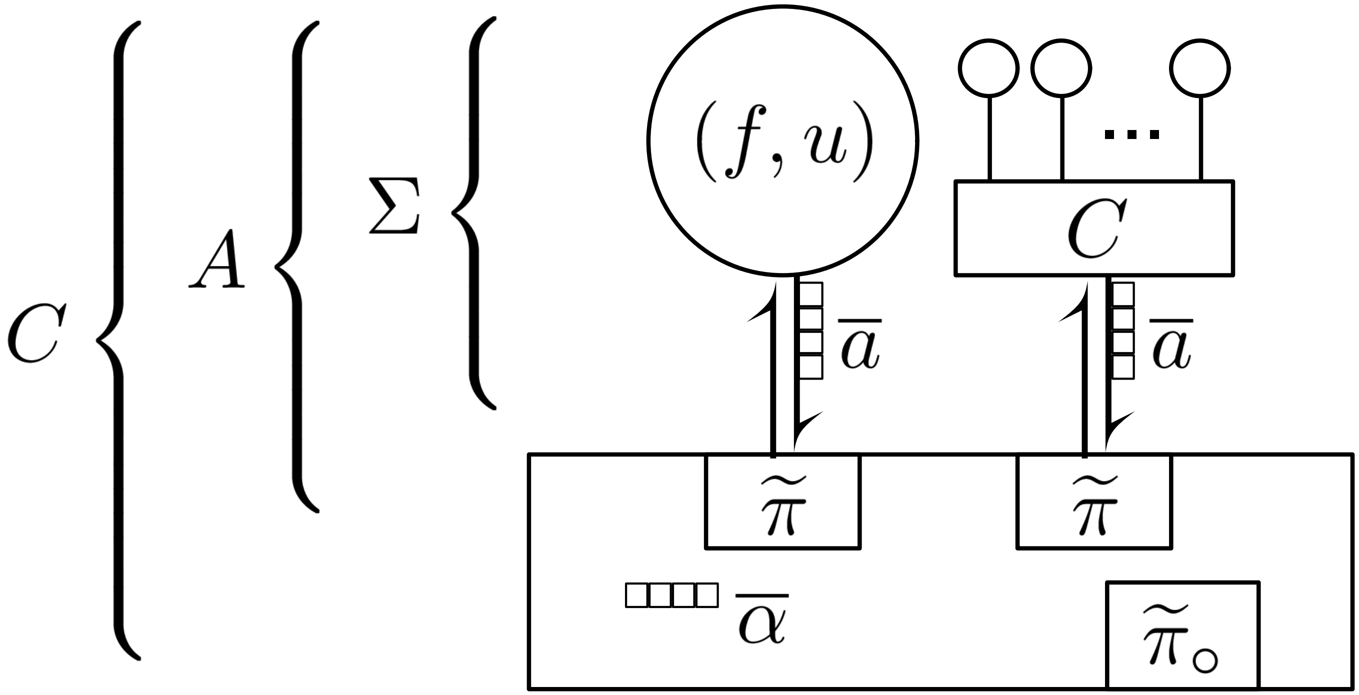
Actor Programming Language
+ Publish/Subscribe
+ Routing Events
+ Hierarchical Layering

Network Calculus
Actor Calculus
(see paper)

Experience reports
(see paper)

http://www.ccs.neu.edu/home/tonyg/marketplace/

Network Calculus Summary



$f =$ Base language functions	
$u =$ Base language values	
$B = (f, u) \mid C$	Behaviors
$\Sigma = \overline{a} \triangleleft B$	Actor States
$A = \widetilde{\pi} : \Sigma$	Actors
$C = [\overline{\alpha}\ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A}]$	Configurations

$\alpha = m \mid \{\widetilde{\pi}\}$	Events
$a = \alpha \mid A$	Actions
$v = u \mid v, v$	Message values
$p = u \mid p, p \mid \star$	Message patterns
$\pi = (p) \mid \langle p \rangle \mid \downharpoonleft\pi$	Subscriptions
$m = \quad \langle v \rangle \mid \downharpoonleft m$	Messages

$A = \widetilde{\pi} : \overline{a} \lhd B$	Actors
$A_Q = \widetilde{\pi} : \cdot \lhd B$	*Quiescent* Actors
$A_I = \widetilde{\pi} : \cdot \lhd B_I$	*Inert* Actors
$C = [\overline{\alpha} \,;\, \widetilde{\pi} \,;\, \overline{A}]$	Configurations
$C_Q = [\,\cdot\,;\, \widetilde{\pi} \,;\, \overline{A_Q}]$	*Quiescent* Configurations
$C_I = [\,\cdot\,;\, \widetilde{\pi} \,;\, \overline{A_I}]$	*Inert* Configurations
$B_I = (f, u) \mid C_I$	Inert Behaviors

$$
\begin{aligned}
interests &: A \to \widetilde{\pi} \\
interests(\widetilde{\pi} : \Sigma) &= \widetilde{\pi}
\end{aligned}
$$

$$
\begin{aligned}
lift &: \widetilde{\pi} \to \widetilde{\pi} \\
lift(\widetilde{\pi}) &= \overline{\rfloor \pi}
\end{aligned}
$$

$$
\begin{aligned}
drop &: \widetilde{\pi} \to \widetilde{\pi} \\
drop(\widetilde{\pi}) &= \overline{drop'(\pi)} \\
drop'(\pi) &= \begin{cases} \pi' & \text{if } \pi = \rfloor \pi' \\ \cdot & \text{otherwise} \end{cases}
\end{aligned}
$$

$$\alpha\restriction_{\widetilde{\pi}} : \alpha \times \widetilde{\pi} \rightharpoonup \alpha$$

$$v\Big|_{p} : v \times p$$

$$\begin{array}{rl}
\langle v \rangle \restriction_{\widetilde{\pi}} & = \langle v \rangle, \text{ if } \exists (p) \in \widetilde{\pi} \text{ such that } v\Big|_{p} \\
\downharpoonleft m \restriction_{\widetilde{\pi}} & = \downharpoonleft m, \text{ if } m \restriction_{drop(\widetilde{\pi})} \\
\{\widetilde{\pi}_1\} \restriction_{\widetilde{\pi}_2} & = \{(\pi_{11} \not\cap \pi_{21}) \cup \cdots \cup (\pi_{11} \not\cap \pi_{2m}) \cup \\
& \quad (\pi_{12} \not\cap \pi_{21}) \cup \cdots \cup (\pi_{12} \not\cap \pi_{2m}) \cup \\
& \qquad\qquad \vdots \qquad\qquad\qquad\qquad\qquad \vdots \\
& \quad (\pi_{1n} \not\cap \pi_{21}) \cup \cdots \cup (\pi_{1n} \not\cap \pi_{2m})\} \\
\alpha \restriction_{\widetilde{\pi}} & \text{ otherwise undefined}
\end{array}$$

$$\pi \not\cap \pi : \pi \times \pi \rightarrow \pi$$

$$\langle p \rangle \not\cap \langle p' \rangle = \emptyset$$

$$(q) \not\cap (q') = \emptyset$$

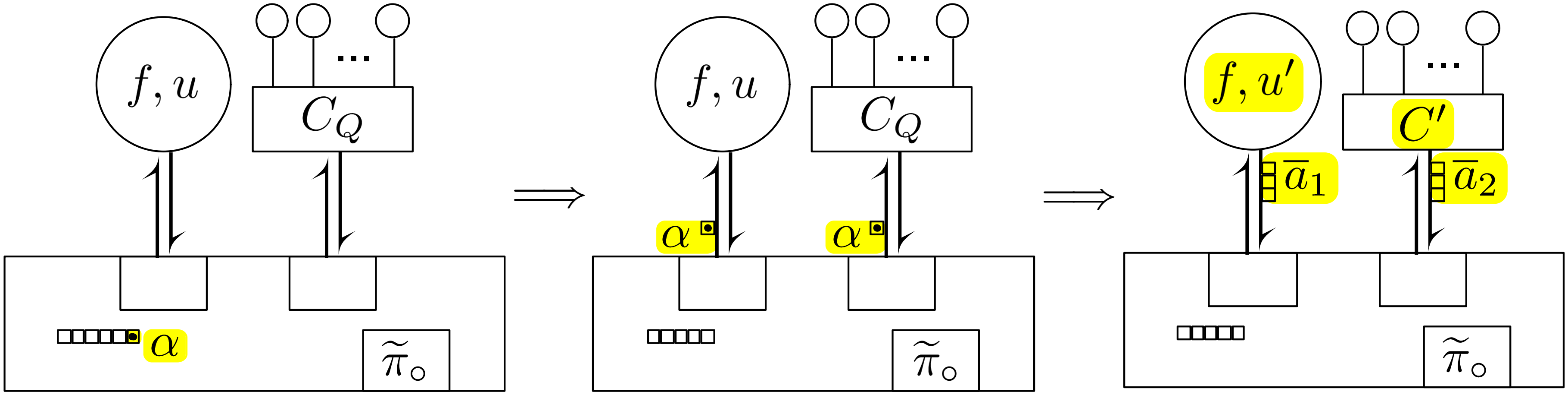
$$\langle p \rangle \not\cap (q) = \langle p \cap q \rangle$$

$$(q) \not\cap \langle p \rangle = (p \cap q)$$

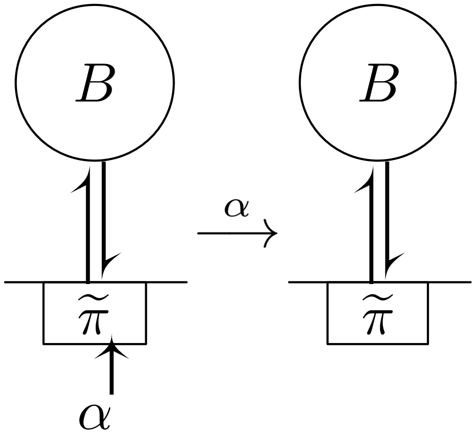
$$
\begin{aligned}
p \cap p &: p \times p \rightharpoonup p \\
u \cap u &= u \\
p_{11}, p_{12} \cap p_{21}, p_{22} &= (p_{11} \cap p_{21}) \ , \ (p_{12} \cap p_{22}) \\
p \cap \star &= p \\
\star \cap p &= p
\end{aligned}
$$

$$
\frac{}{u\Big|_{u}} \qquad \frac{v_1\Big|_{p_1} \qquad v_2\Big|_{p_2}}{v_1, v_2\Big|_{p_1, p_2}} \qquad \frac{}{v\Big|_{\star}}
$$

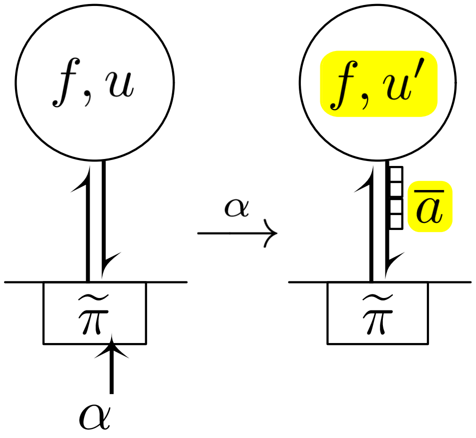
$$\frac{\overline{A_Q \xrightarrow{\alpha} A'}}{\overline{a} \triangleleft [\alpha\overline{\alpha}_0 \; ; \; \widetilde{\pi}_\circ \; ; \; \overline{A_Q}] \longrightarrow \overline{a} \triangleleft [\overline{\alpha}_0 \; ; \; \widetilde{\pi}_\circ \; ; \; \overline{A}']}$$



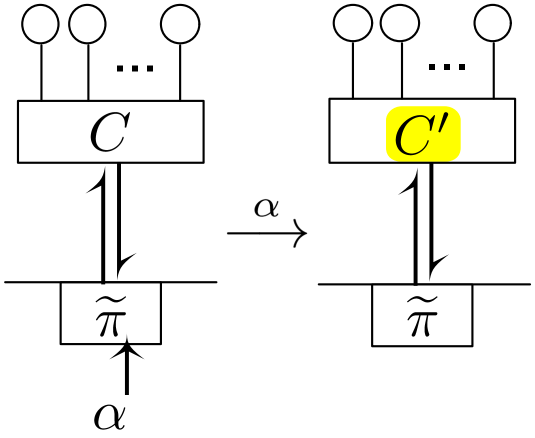
$$\frac{}{\widetilde{\pi} : \cdot \triangleleft B \xrightarrow{\alpha} \widetilde{\pi} : \cdot \triangleleft B}\ (\alpha \restriction_{\widetilde{\pi}} \text{ is undefined})$$



$$\frac{f(\alpha \restriction_{\widetilde{\pi}}, u) = (\overline{a}, u')}{\widetilde{\pi} : \cdot \triangleleft (f, u) \xrightarrow{\alpha} \widetilde{\pi} : \overline{a} \triangleleft (f, u')}\ (\alpha \restriction_{\widetilde{\pi}} \text{ is defined})$$

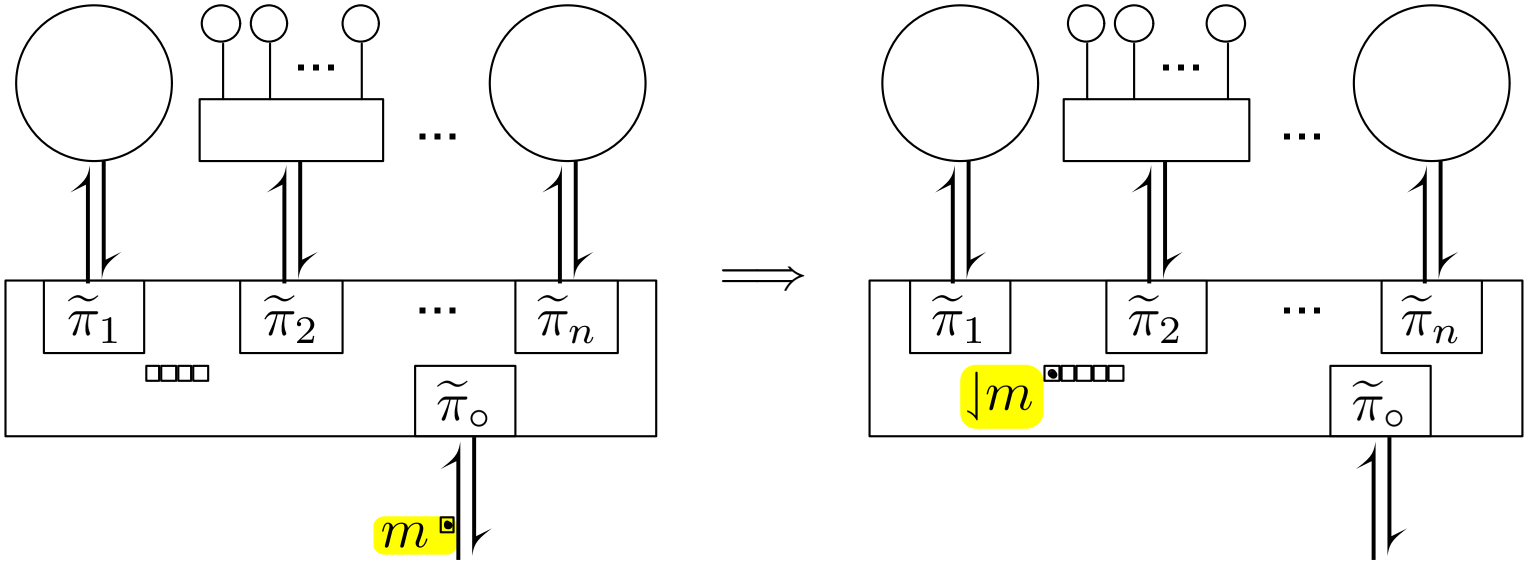


$$\frac{\mathit{inject}\ (\alpha \restriction_{\widetilde{\pi}}, C) = C'}{\widetilde{\pi} : \cdot \triangleleft C \xrightarrow{\alpha} \widetilde{\pi} : \cdot \triangleleft C'}\ (\alpha \restriction_{\widetilde{\pi}} \text{ is defined})$$



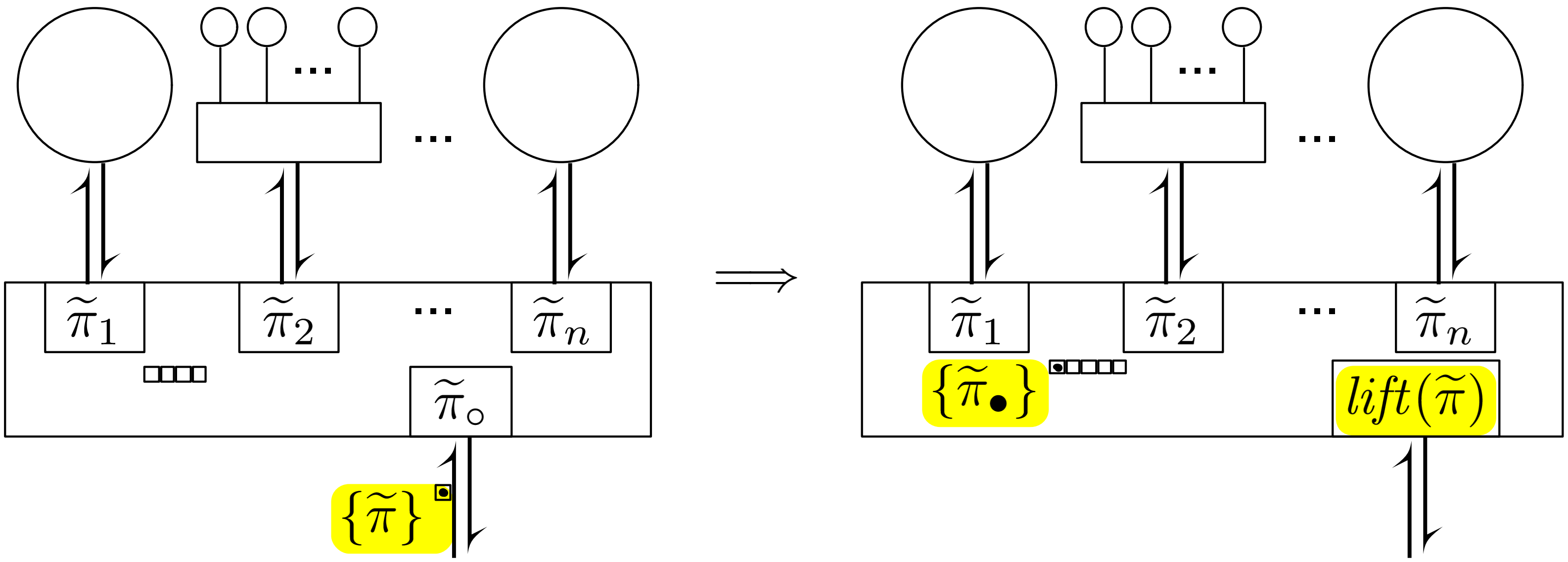
$$\mathit{inject} : \alpha \times C \to C$$

$$\mathit{inject}\ (m, [\overline{\alpha}\ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A}]) = [\overline{\alpha}\downharpoonleft m\ \ ;\ \widetilde{\pi}_\circ\ \ ;\ \overline{A}]$$



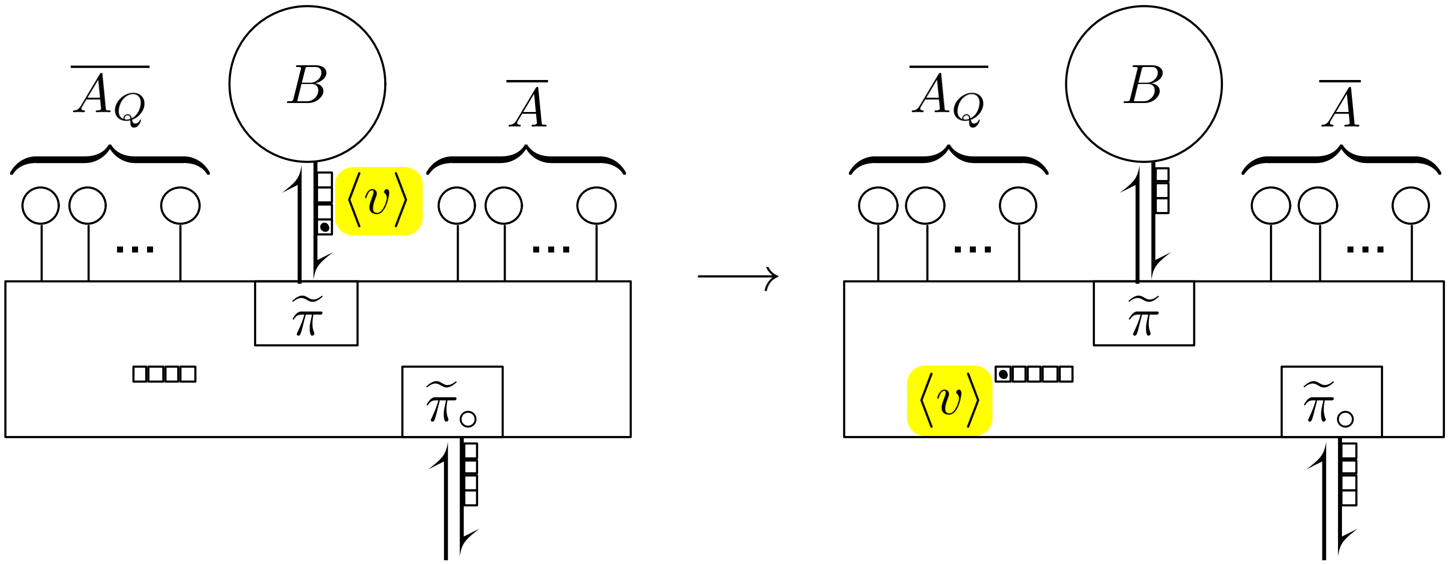
$$\mathit{inject} : \alpha \times C \to C$$

$$\mathit{inject}\ (\{\widetilde{\pi}\}, [\overline{\alpha}\ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A}]) = [\overline{\alpha}\ \{\widetilde{\pi}_\bullet\}\ \ ;\ \mathit{lift}(\widetilde{\pi})\ \ ;\ \overline{A}]$$

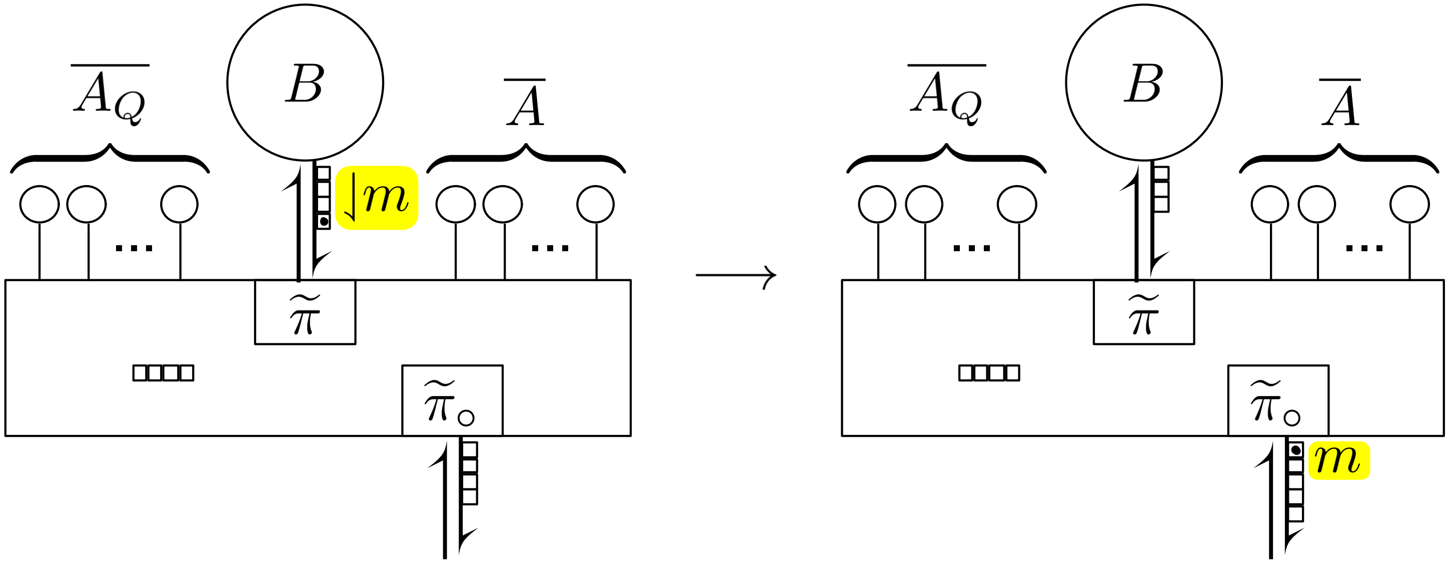


$$\widetilde{\pi}_\bullet = \widetilde{\pi}_1 \cup \widetilde{\pi}_2 \cup \cdots \cup \widetilde{\pi}_n \cup \mathit{lift}(\widetilde{\pi})$$

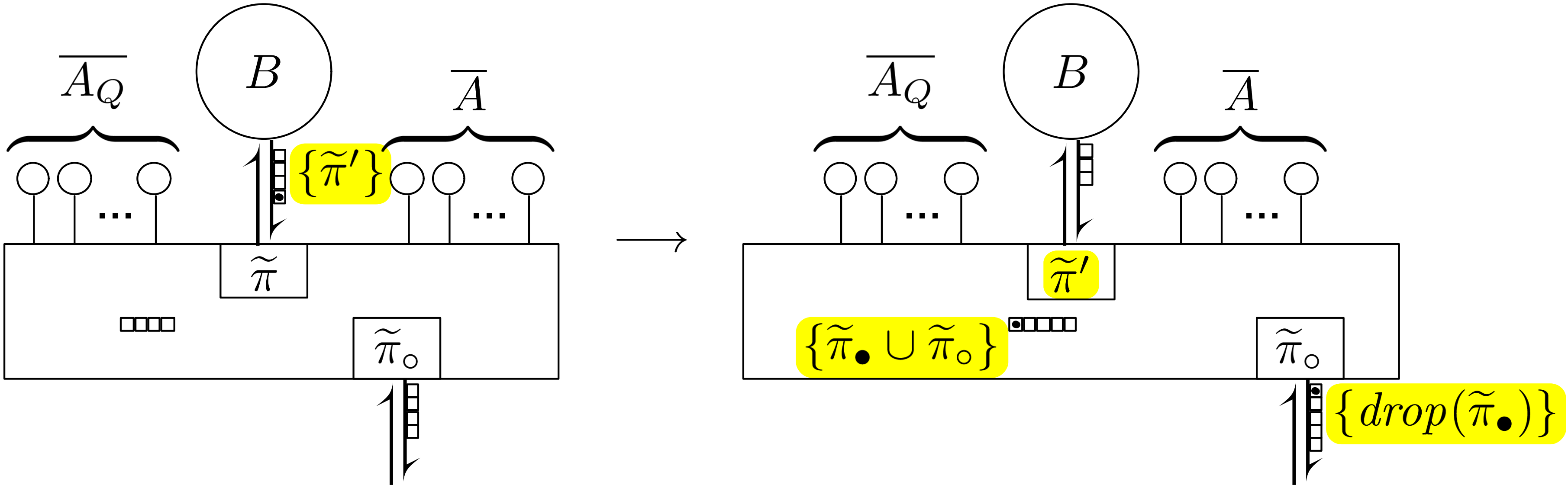
$$\overline{a}_0 \triangleleft [\overline{\alpha}\ \ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} : \langle v\rangle\overline{a} \triangleleft B)\overline{A}] \longrightarrow \overline{a}_0 \triangleleft [\overline{\alpha}\langle v\rangle\ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} : \overline{a} \triangleleft B)\overline{A}]$$



$$\overline{a}_0 \triangleleft [\overline{\alpha}\ \ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} : \downharpoonleft m\overline{a} \triangleleft B)\overline{A}] \longrightarrow \overline{a}_0 m \triangleleft [\overline{\alpha}\ \ ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} : \overline{a} \triangleleft B)\overline{A}]$$

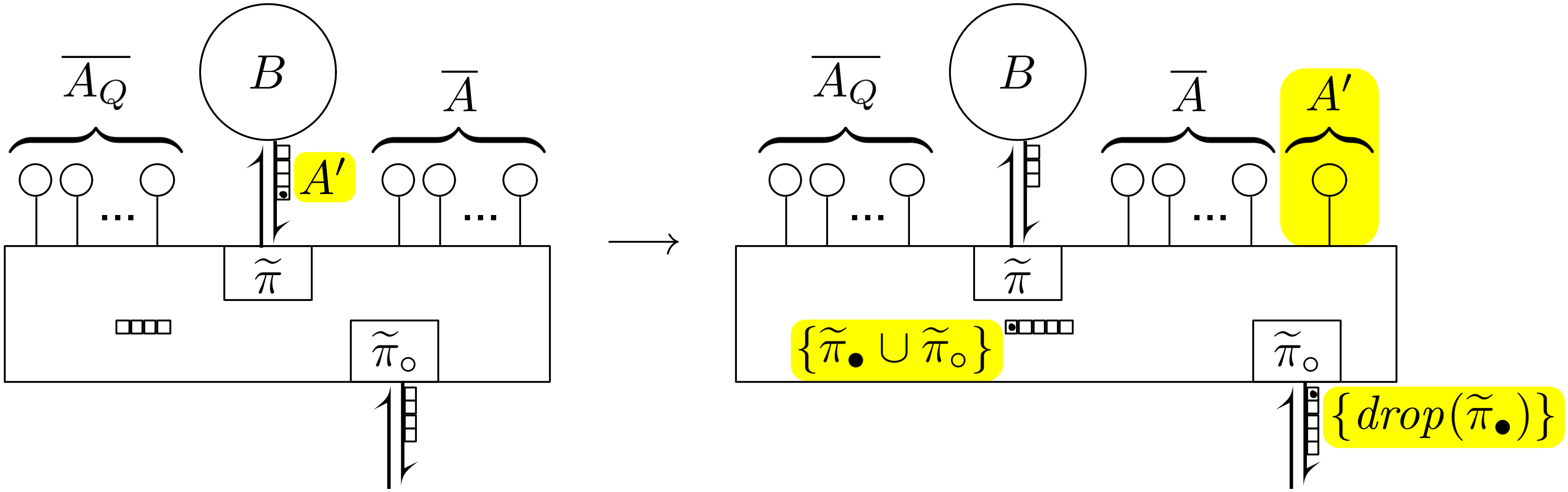


$$\begin{array}{lllll} & \overline{a}_0 & \triangleleft\, [\overline{\alpha} & ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} :\ \{\widetilde{\pi}'\}\overline{a} \triangleleft B)\overline{A}] \\ \longrightarrow & \overline{a}_0\{drop(\widetilde{\pi}_\bullet)\} & \triangleleft\, [\overline{\alpha}\{\widetilde{\pi}_\bullet\widetilde{\pi}_\circ\} & ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi}' :\ \overline{a} \triangleleft B)\overline{A}] \end{array}$$



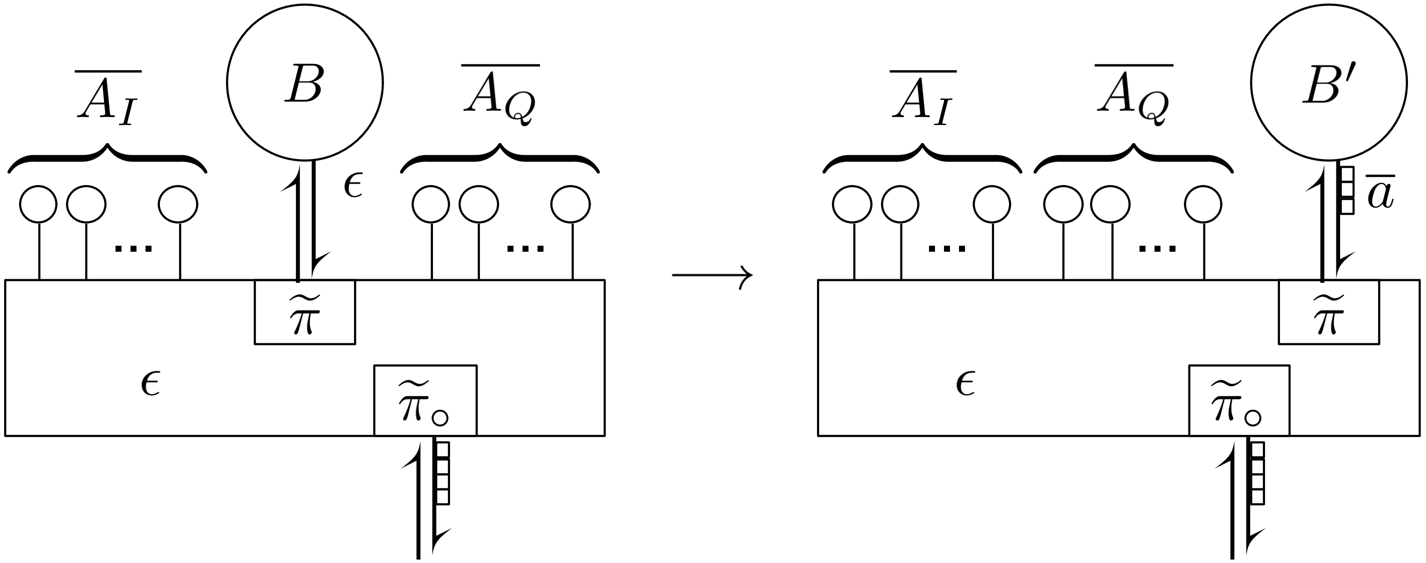
$$\widetilde{\pi}_\bullet = \overline{interests(A_Q)} \cup \widetilde{\pi}' \cup \overline{interests(A)}$$

$$\begin{array}{lllll} & \overline{a}_0 & \lhd [\overline{\alpha} & ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} : A'\overline{a} \lhd B)\overline{A}] \\ \longrightarrow & \overline{a}_0\{drop(\widetilde{\pi}_\bullet)\} & \lhd [\overline{\alpha}\{\widetilde{\pi}_\bullet\widetilde{\pi}_\circ\} & ;\ \widetilde{\pi}_\circ\ ;\ \overline{A_Q}(\widetilde{\pi} :\ \ \overline{a} \lhd B)\overline{A}\ A'] \end{array}$$

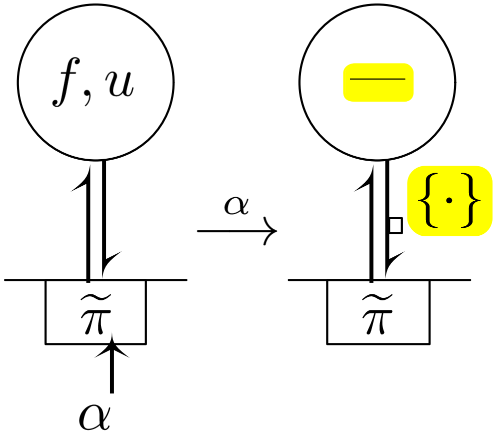


$$\widetilde{\pi}_\bullet = \overline{interests(A_Q)} \cup \widetilde{\pi} \cup \overline{interests(A)} \cup interests(A')$$

$$\frac{\cdot \triangleleft B \longrightarrow \overline{a} \triangleleft B'}{\overline{a}_0 \triangleleft [\cdot\,;\, \widetilde{\pi}_\circ\,;\, \overline{A_I}(\widetilde{\pi} : \cdot \triangleleft B)\overline{A_Q}] \longrightarrow \overline{a}_0 \triangleleft [\cdot\,;\, \widetilde{\pi}_\circ\,;\, \overline{A_Q}\;\overline{A_I}(\widetilde{\pi} : \overline{a} \triangleleft B')]}$$



$$\frac{f(\alpha\restriction_{\widetilde{\pi}}\;, u) = \mathit{exception}}{\widetilde{\pi} : \cdot \triangleleft (f,u) \xrightarrow{\alpha} \widetilde{\pi} : \{\cdot\} \triangleleft (\cdot,\cdot)}\;(\alpha\restriction_{\widetilde{\pi}}\; \text{is defined})$$


f, u
—
$\{\cdot\}$
α
$\widetilde{\pi}$
$\widetilde{\pi}$
α