Session-Typed Concurrent Programming
Lecture 3

Stephanie Balzer
Carnegie Mellon University

OPLSS 2021
June 25, 2021

Today’s lecture

Today's lecture

Recap

- Type system and dynamics for the intuitionistic linear session types language SILL
- Curry-Howard correspondence
- SILL readily guarantees session fidelity and deadlock-freedom

Today's lecture

Recap

- Type system and dynamics for the intuitionistic linear session types language SILL
- Curry-Howard correspondence
- SILL readily guarantees session fidelity and deadlock-freedom

Next

- Extend SILL with persistent truth (of course!)
- Then, switch gears and introduce shared session types

Follow-up on Slack

Follow-up on Slack

$$\frac{\Delta_1 \vdash P :: (x : A) \qquad \Delta_2, x : A \vdash Q :: (z : C)}{\Delta_1, \Delta_2 \vdash x \leftarrow P; Q :: (z : C)}\ Cut$$

Follow-up on Slack

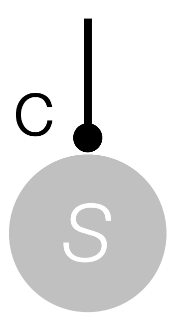
$$\frac{\Delta_1 \vdash P :: (x : A) \qquad \Delta_2, x : A \vdash Q :: (z : C)}{\Delta_1, \Delta_2 \vdash x \leftarrow P; Q :: (z : C)}\ Cut$$

$$
\begin{array}{lll}
(\text{D-}Cut) & \mathsf{proc}(c, x \leftarrow P_x ; Q_x) & \\
& \longrightarrow\ \mathsf{proc}(a, [a/x]\, P_x), \mathsf{proc}(c, [a/x]\, Q_x) & \quad (\text{a fresh})
\end{array}
$$

Follow-up on Slack

$$\frac{\Delta_1 \vdash P :: (x : A) \qquad \Delta_2, x : A \vdash Q :: (z : C)}{\Delta_1, \Delta_2 \vdash x \leftarrow P; Q :: (z : C)} \; \mathit{Cut}$$

$$\begin{array}{lll} (\text{D-}\mathit{Cut}) & \mathsf{proc}(c, x \leftarrow P_x; Q_x) & \\ & \longrightarrow \mathsf{proc}(a, [a/x]\, P_x), \mathsf{proc}(c, [a/x]\, Q_x) & (\text{a fresh}) \end{array}$$

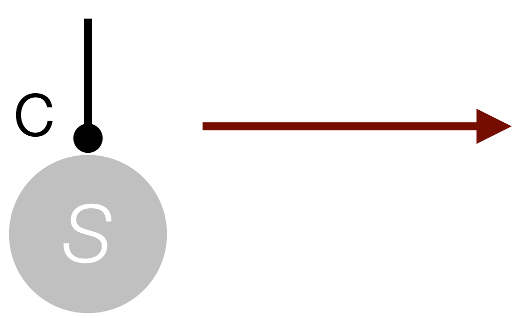


$$S = x \leftarrow P_x; Q_x$$

Follow-up on Slack

$$\frac{\Delta_1 \vdash P :: (x : A) \qquad \Delta_2, x : A \vdash Q :: (z : C)}{\Delta_1, \Delta_2 \vdash x \leftarrow P ; Q :: (z : C)}\ \mathit{Cut}$$

$$\begin{array}{ll} (\text{D-}\mathit{Cut}) & \mathsf{proc}(c, x \leftarrow P_x ; Q_x) \\ & \longrightarrow \mathsf{proc}(a, [a/x]\, P_x), \mathsf{proc}(c, [a/x]\, Q_x) \qquad (\text{a fresh}) \end{array}$$

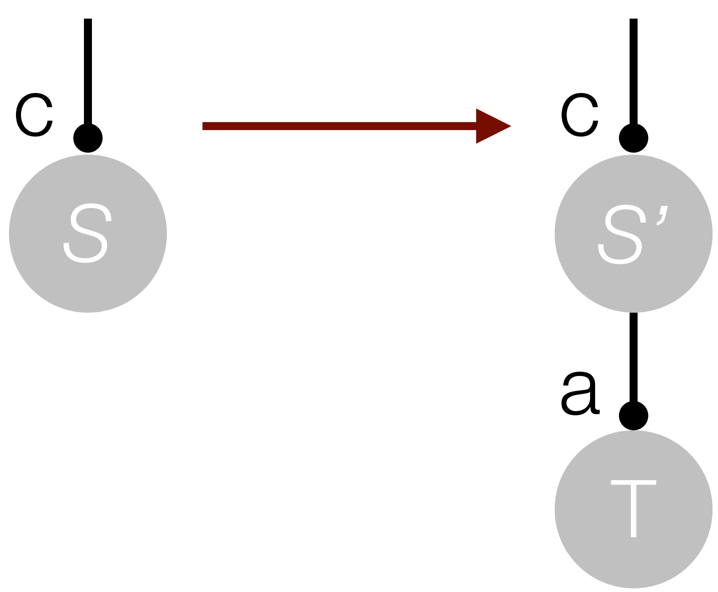


$$S = x \leftarrow P_x ; Q_x$$

Follow-up on Slack

$$\frac{\Delta_1 \vdash P :: (x : A) \qquad \Delta_2, x : A \vdash Q :: (z : C)}{\Delta_1, \Delta_2 \vdash x \leftarrow P ; Q :: (z : C)}\ \mathit{Cut}$$

$$\begin{array}{lll} (\text{D-}\mathit{Cut}) & \mathsf{proc}(c, x \leftarrow P_x ; Q_x) & \\ & \longrightarrow \mathsf{proc}(a, [a/x]\, P_x), \mathsf{proc}(c, [a/x]\, Q_x) & (\text{a fresh}) \end{array}$$



$S = x \leftarrow P_x ; Q_x$

$S' = [a/x]\, Q_x$

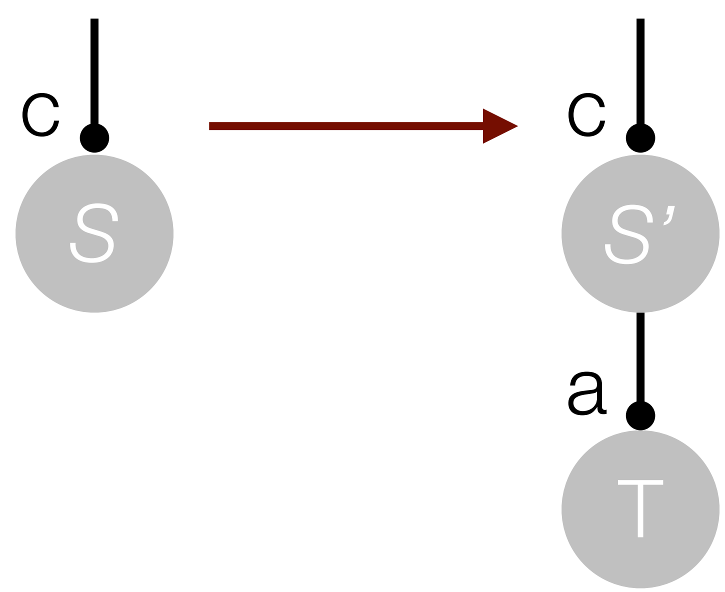
$T = [a/x]\, P_x$

Follow-up on Slack

homework:
make drawings for other
tensor and lolli!

$$\frac{\Delta_1 \vdash P :: (x : A) \qquad \Delta_2, x : A \vdash Q :: (z : C)}{\Delta_1, \Delta_2 \vdash x \leftarrow P ; Q :: (z : C)} \; Cut$$

$$\begin{array}{ll} (\text{D-}Cut) & \mathsf{proc}(c, x \leftarrow P_x ; Q_x) \\ & \longrightarrow \mathsf{proc}(a, [a/x]\, P_x), \mathsf{proc}(c, [a/x]\, Q_x) \qquad (\text{a fresh}) \end{array}$$



$$S = x \leftarrow P_x ; Q_x$$
$$S' = [a/x]\, Q_x$$
$$T = [a/x]\, P_x$$

Intuitionistic linear logic session types with !

Of course!

Of course!

one connective from linear logic still missing: persistent truth

Of course!

one connective from linear logic still missing: persistent truth

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \,\&\, B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"

Of course!

one connective from linear logic still missing: persistent truth

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \,\&\, B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"
		$!A$	"of course", persistent truth	"replication"

Of course!

one connective from linear logic still missing: persistent truth

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	“channel output”
		$A \multimap B$	multiplicative implication	“channel input”
		$A \mathbin{\&} B$	additive conjunction	“external choice”
		$A \oplus B$	additive disjunction	“internal choice”
		$\mathbf{1}$	unit for $\otimes$	“termination”
		$!A$	”of course”, persistent truth	“replication”

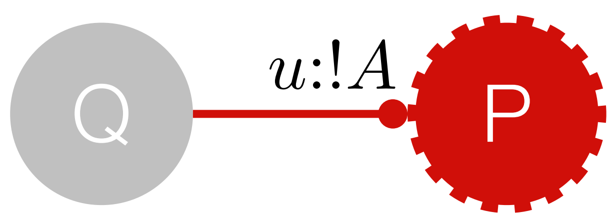
a process of type !A can be used arbitrarily often, i.e., can have any number of clients

Of course!

What is the computational meaning of “of course”?

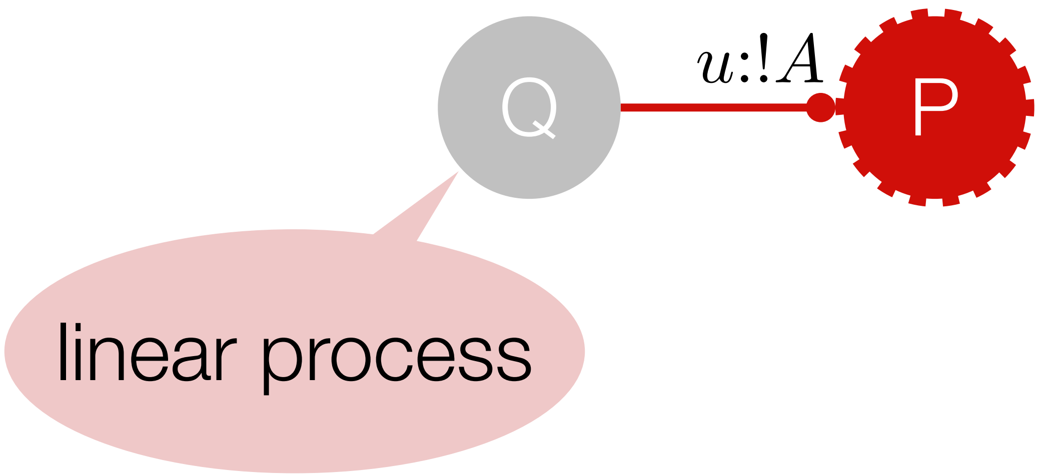
Of course!

What is the computational meaning of “of course”?



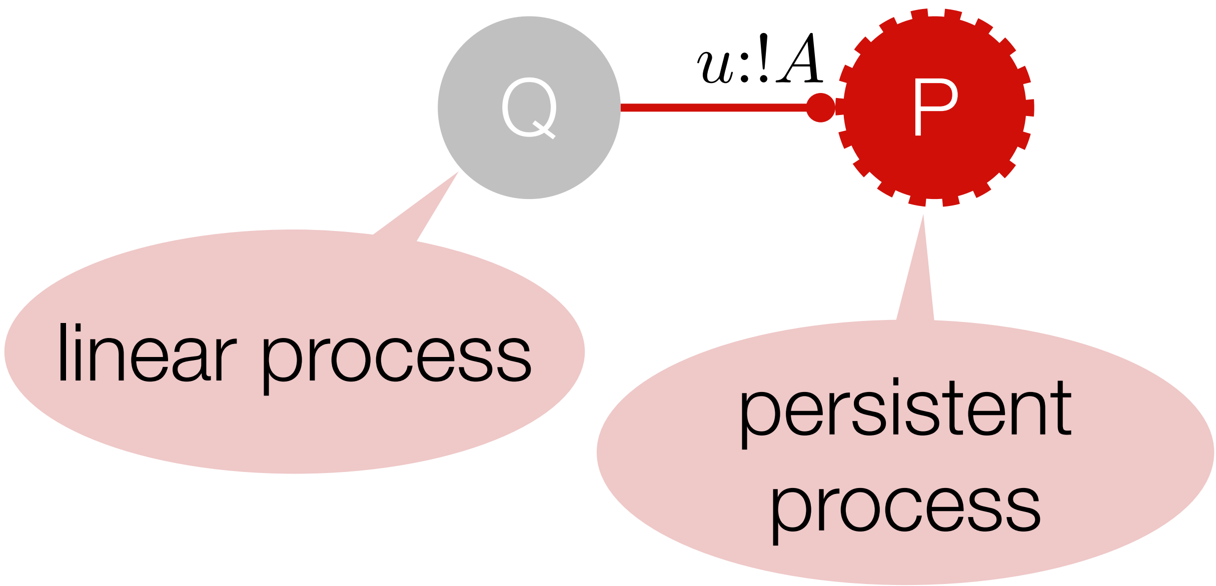
Of course!

What is the computational meaning of “of course”?



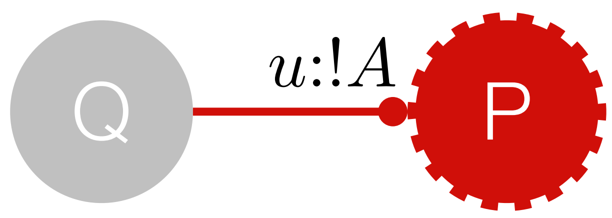
Of course!

What is the computational meaning of “of course”?



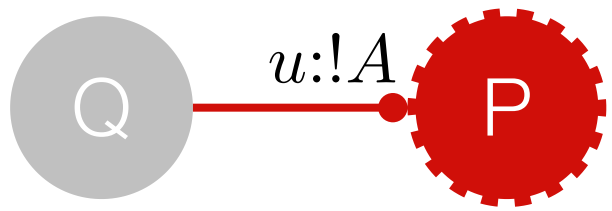
Of course!

What is the computational meaning of “of course”?



Of course!

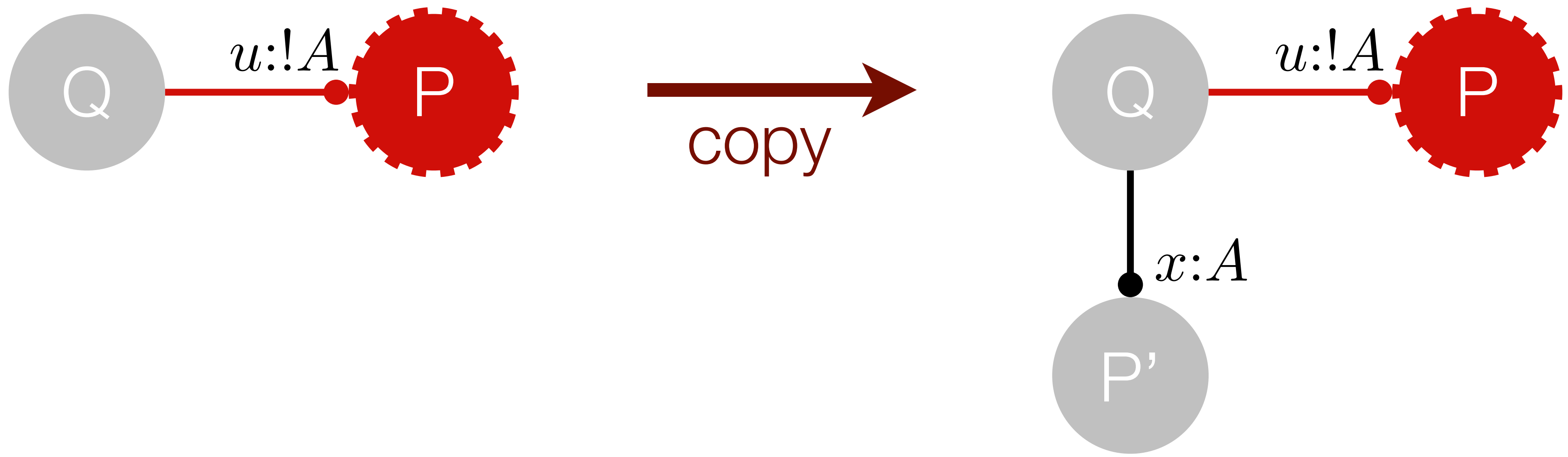
What is the computational meaning of “of course”?



obtain a linear copy P’ of unrestricted process P

Of course!

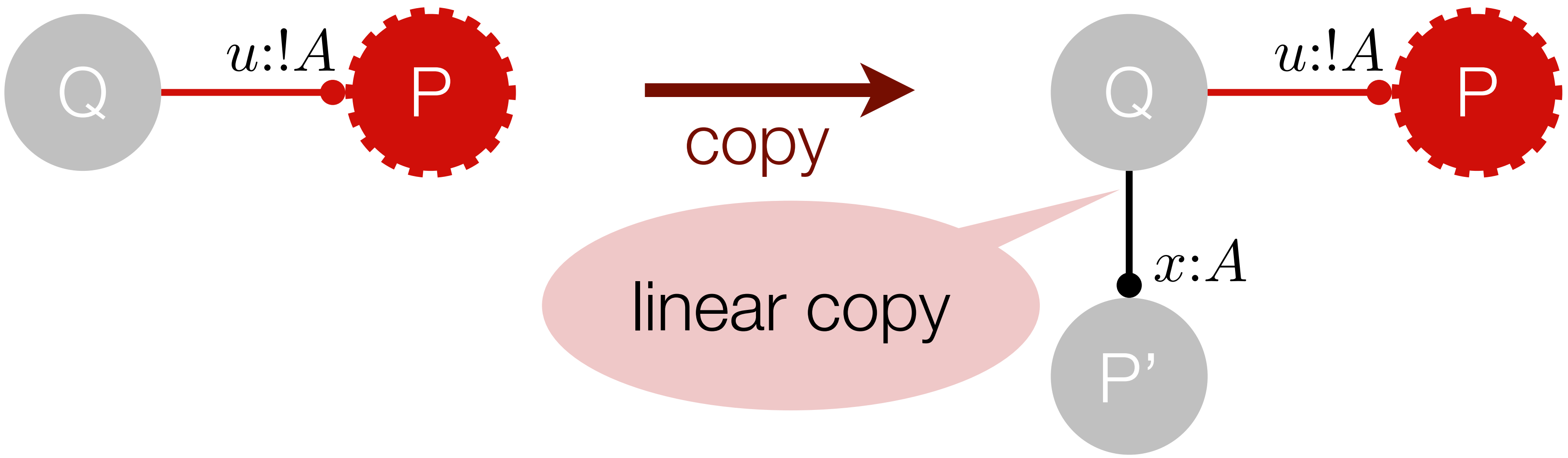
What is the computational meaning of “of course”?



obtain a linear copy P’ of unrestricted process P

Of course!

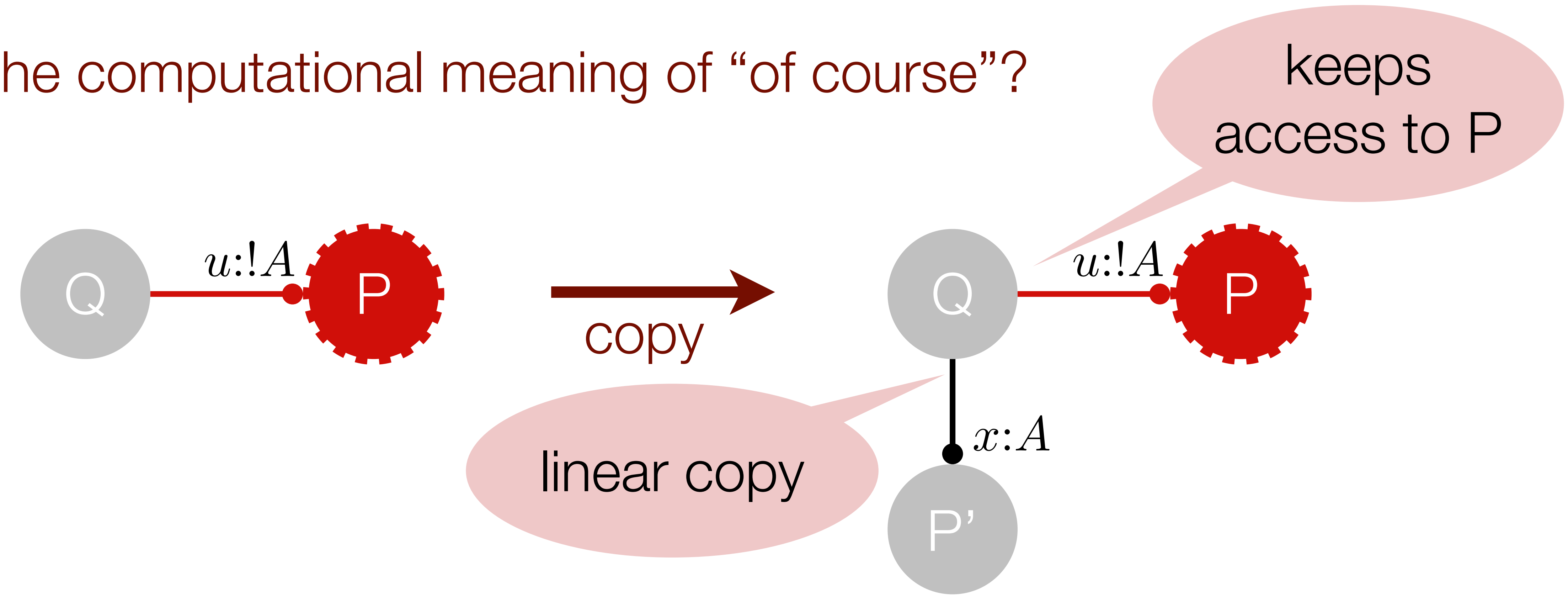
What is the computational meaning of “of course”?



obtain a linear copy P’ of unrestricted process P

Of course!

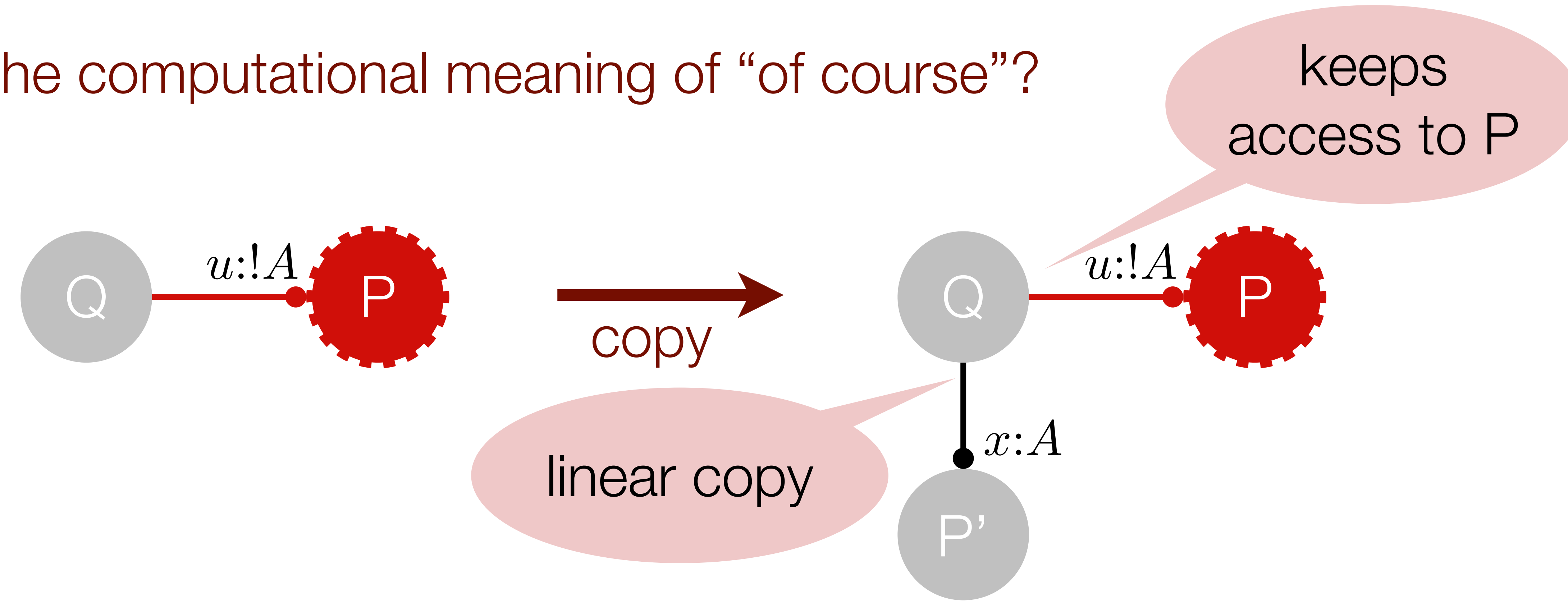
What is the computational meaning of "of course"?



obtain a linear copy P' of unrestricted process P

Of course!

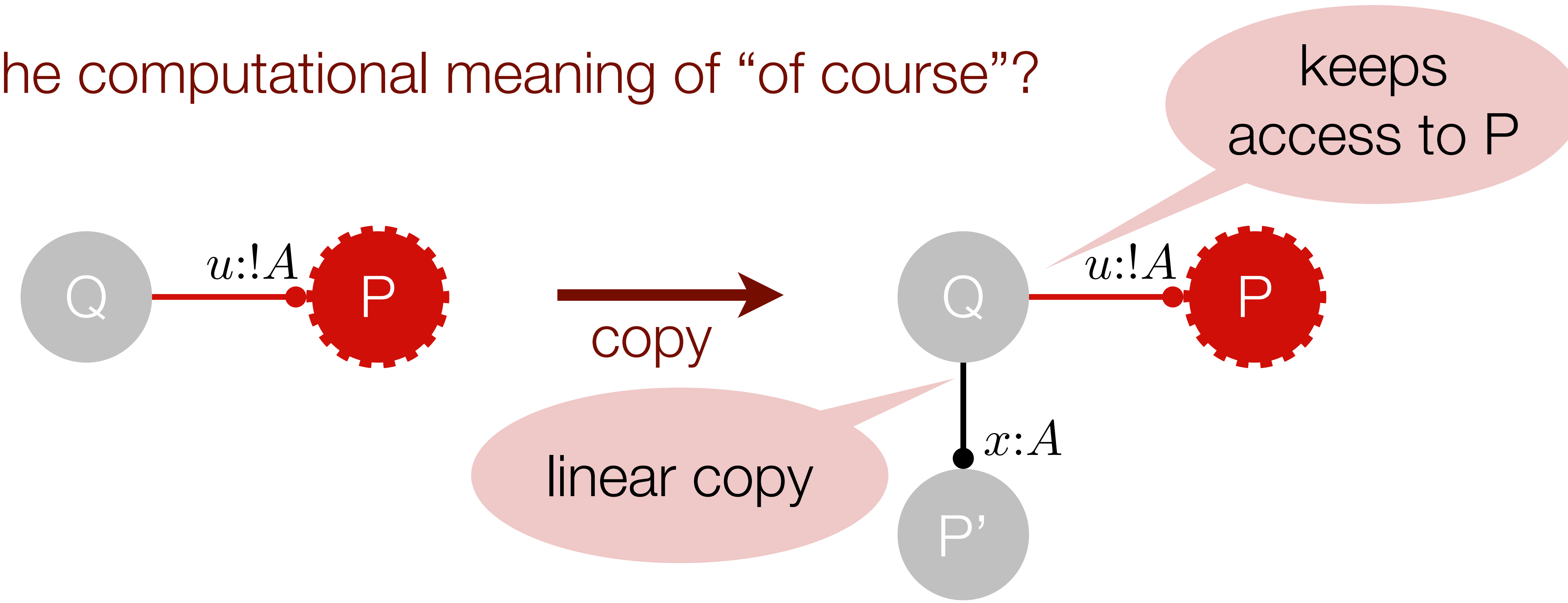
What is the computational meaning of “of course”?



- obtain a linear copy P’ of unrestricted process P
- corresponds to replication in the pi-calculus

Of course!

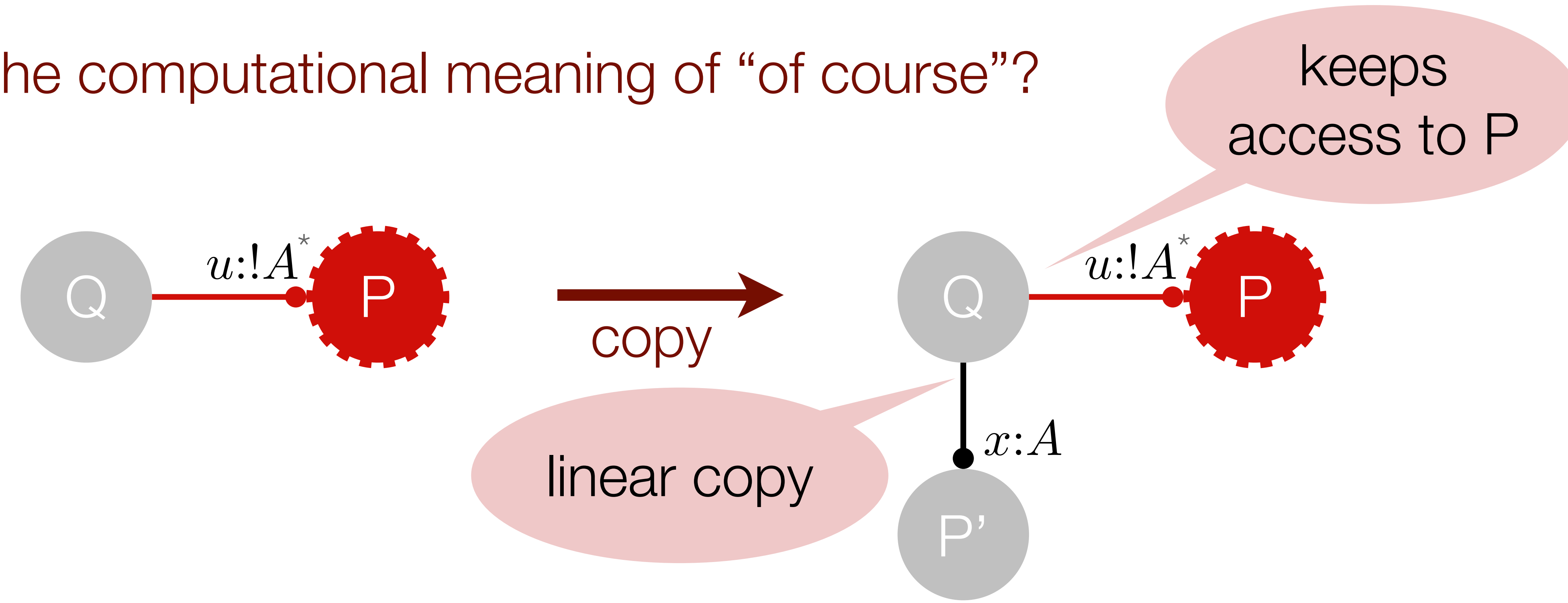
What is the computational meaning of “of course”?



- obtain a linear copy P’ of unrestricted process P
- corresponds to replication in the pi-calculus
- let’s look at typing rules and dynamics

Of course!

What is the computational meaning of “of course”?



- obtain a linear copy P’ of unrestricted process P
- corresponds to replication in the pi-calculus
- let’s look at typing rules and dynamics

(*) copy rule operates on structural context, so it should be u: A because A is judgmentally persistent

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	“channel output”
		$A \multimap B$	multiplicative implication	“channel input”
		$A \mathbin{\&} B$	additive conjunction	“external choice”
		$A \oplus B$	additive disjunction	“internal choice”
		$\mathbf{1}$	unit for $\otimes$	“termination”
		$!A$	”of course”, persistent truth	“replication”

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	“channel output”
		$A \multimap B$	multiplicative implication	“channel input”
		$A \,\&\, B$	additive conjunction	“external choice”
		$A \oplus B$	additive disjunction	“internal choice”
		$\mathbf{1}$	unit for $\otimes$	“termination”
		$!A$	”of course”, persistent truth	“replication”

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

Of course!

Types:

$$
\begin{array}{llll}
A, B \triangleq & A \otimes B & \text{multiplicative conjunction} & \text{“channel output”} \\
& A \multimap B & \text{multiplicative implication} & \text{“channel input”} \\
& A \,\&\, B & \text{additive conjunction} & \text{“external choice”} \\
& A \oplus B & \text{additive disjunction} & \text{“internal choice”} \\
& \mathbf{1} & \text{unit for } \otimes & \text{“termination”} \\
& !A & \text{”of course”, persistent truth} & \text{“replication”}
\end{array}
$$

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \,\&\, B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"
		$!A$	"of course", persistent truth	"replication"

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

persistent channels

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \,\&\, B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"
		$!A$	"of course", persistent truth	"replication"

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

persistent channels

structural context,
i.e., permits weakening and
contraction

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \mathbin{\&} B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"
		$!A$	"of course", persistent truth	"replication"

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	“channel output”
		$A \multimap B$	multiplicative implication	“channel input”
		$A \mathbin{\&} B$	additive conjunction	“external choice”
		$A \oplus B$	additive disjunction	“internal choice”
		$\mathbf{1}$	unit for $\otimes$	“termination”
		$!A$	”of course”, persistent truth	“replication”

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \mathbin{\&} B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"
		$!A$	"of course", persistent truth	"replication"

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

dyadic formulation

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	“channel output”
		$A \multimap B$	multiplicative implication	“channel input”
		$A \mathbin{\&} B$	additive conjunction	“external choice”
		$A \oplus B$	additive disjunction	“internal choice”
		$\mathbf{1}$	unit for $\otimes$	“termination”
		$!A$	”of course”, persistent truth	“replication”

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

dyadic formulation

$$\Psi = u_1 : B_1, \ldots, u_n : B_n$$

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	"channel output"
		$A \multimap B$	multiplicative implication	"channel input"
		$A \mathbin{\&} B$	additive conjunction	"external choice"
		$A \oplus B$	additive disjunction	"internal choice"
		$\mathbf{1}$	unit for $\otimes$	"termination"
		$!A$	"of course", persistent truth	"replication"

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

dyadic formulation

$$\Psi = u_1 : B_1, \ldots, u_n : B_n$$

implicitly !-typed

Of course!

Types:

A, B	$\triangleq$	$A \otimes B$	multiplicative conjunction	“channel output”
		$A \multimap B$	multiplicative implication	“channel input”
		$A \,\&\, B$	additive conjunction	“external choice”
		$A \oplus B$	additive disjunction	“internal choice”
		$\mathbf{1}$	unit for $\otimes$	“termination”
		$!A$	”of course”, persistent t	

persistent channels in Δ are of type !A

Typing judgment:

$$\Psi; \Delta \vdash P :: (x : A)$$

dyadic formulation

$$\Psi = u_1 : B_1, \ldots, u_n : B_n$$

implicitly !-typed

Judgmental rule copy

Judgmental rule copy

Typing rule:

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \mathsf{send}\, u\, (\mathsf{new}\, x); Q_x :: (z : C)}\ copy$$

Judgmental rule copy

Typing rule:

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \mathsf{send}\, u\, (\mathsf{new}\, x); Q_x :: (z : C)}\ \mathit{copy}$$

→ obtain a linear copy of a persistent server

Judgmental rule copy

Typing rule:

contraction!

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \mathsf{send}\, u\, (\mathsf{new}\, x); Q_x :: (z : C)}\ \mathit{copy}$$

obtain a linear copy of a persistent server

Judgmental rule copy

Typing rule:

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \mathsf{send}\, u\, (\mathsf{new}\, x); Q_x :: (z : C)}\ \mathit{copy}$$

obtain a linear copy of a persistent server

Judgmental rule copy

Typing rule:

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \mathsf{send}\, u\, (\mathsf{new}\, x); Q_x :: (z : C)}\ \mathit{copy}$$

obtain a linear copy of a persistent server

Dynamics:

$$\begin{array}{ll} (\text{D-}\mathit{copy}) & !\mathsf{proc}(u, x \leftarrow \mathsf{recv}\ u; P_x), \mathsf{proc}(c, \mathsf{send}\, u\, (\mathsf{new}\, x); Q_x) \\ & \longrightarrow\ \mathsf{proc}(a, [a/x]\, P_x), \mathsf{proc}(c, [a/x]\, Q_x) \qquad (\text{a fresh}) \end{array}$$

Judgmental rule copy

Typing rule:

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \textsf{send}\, u\, (\textsf{new}\, x); Q_x :: (z : C)}\ \textit{copy}$$

obtain a linear copy of a persistent server

Dynamics:

persistent!

$$(\text{D-}\textit{copy}) \quad !\textsf{proc}(u, x \leftarrow \textsf{recv}\ u; P_x), \textsf{proc}(c, \textsf{send}\, u\, (\textsf{new}\, x); Q_x)$$

$$\longrightarrow\ \textsf{proc}(a, [a/x]\, P_x), \textsf{proc}(c, [a/x]\, Q_x) \qquad (\text{a fresh})$$

Judgmental rule copy

Typing rule:

$$\frac{\Psi, u : A; \Delta, x : A \vdash Q_x :: (z : C)}{\Psi, u : A; \Delta \vdash \textsf{send}\, u\, (\textsf{new}\, x); Q_x :: (z : C)}\ \textit{copy}$$

obtain a linear copy of a persistent server

Dynamics:

persistent!

remains available in post-state

$$(\text{D-}\textit{copy}) \quad !\textsf{proc}(u, x \leftarrow \textsf{recv}\ u; P_x), \textsf{proc}(c, \textsf{send}\, u\, (\textsf{new}\, x); Q_x)$$

$$\longrightarrow \textsf{proc}(a, [a/x]\, P_x), \textsf{proc}(c, [a/x]\, Q_x) \qquad (\text{a fresh})$$

Judgmental rule cut!

Judgmental rule cut!

Typing rule:

$$\frac{\Psi; \cdot \vdash P_x :: (x : A) \qquad \Psi, u : A; \Delta \vdash Q_u :: (z : C)}{\Psi; \Delta \vdash u \leftarrow !(x \leftarrow \mathsf{recv}\ u; P_x); Q_u :: (z : C)}\ cut!$$

Judgmental rule cut!

Typing rule:

$$\frac{\Psi; \cdot \vdash P_x :: (x : A) \qquad \Psi, u : A; \Delta \vdash Q_u :: (z : C)}{\Psi; \Delta \vdash u \leftarrow !(x \leftarrow \mathsf{recv}\ u; P_x); Q_u :: (z : C)}\ cut!$$

→ spawning a persistent server

Judgmental rule cut!

Typing rule:

$$\frac{\Psi; \cdot \vdash P_x :: (x : A) \qquad \Psi, u : A; \Delta \vdash Q_u :: (z : C)}{\Psi; \Delta \vdash u \leftarrow !(x \leftarrow \textsf{recv}\ u; P_x); Q_u :: (z : C)}\ cut!$$

spawning a persistent server

Dynamics:

$$(\text{D-}cut!) \quad \textsf{proc}(c, u \leftarrow !(x \leftarrow \textsf{recv}\ u; P_x); Q_u)$$
$$\longrightarrow !\textsf{proc}(a, x \leftarrow \textsf{recv}\ a; P_x), \textsf{proc}(c, [a/u]\, Q_u) \qquad (\text{a fresh})$$

Rules for of course

Rules for of course

Typing rule:

$$\frac{\Psi; \cdot \vdash P_y :: (y : A)}{\Psi; \cdot \vdash \mathsf{send}\, x\, (\mathsf{new}\, u); !(y \leftarrow \mathsf{recv}\ u; P_y) :: (x :!A)}\ !_R$$

$$\frac{\Psi, u : A; \Delta \vdash Q_u :: (z : C)}{\Psi; \Delta, x :!A \vdash u \leftarrow \mathsf{recv}\ x; Q_u :: (z : C)}\ !_L$$

Rules for of course

Typing rule:

$$\frac{\Psi; \cdot \vdash P_y :: (y : A)}{\Psi; \cdot \vdash \mathsf{send}\, x\, (\mathsf{new}\, u); !(y \leftarrow \mathsf{recv}\ u; P_y) :: (x :!A)}\ !_R$$

$$\frac{\Psi, u : A; \Delta \vdash Q_u :: (z : C)}{\Psi; \Delta, x :!A \vdash u \leftarrow \mathsf{recv}\ x; Q_u :: (z : C)}\ !_L$$

→ spawning a persistent server

Rules for of course

Typing rule:

$$\frac{\Psi ; \cdot \vdash P_y :: (y : A)}{\Psi ; \cdot \vdash \mathsf{send}\, x\, (\mathsf{new}\, u); !(y \leftarrow \mathsf{recv}\ u; P_y) :: (x :!A)} \; !_R$$

$$\frac{\Psi, u : A; \Delta \vdash Q_u :: (z : C)}{\Psi ; \Delta, x :!A \vdash u \leftarrow \mathsf{recv}\ x; Q_u :: (z : C)} \; !_L$$

→ spawning a persistent server

Dynamics:

$$\text{(D-!)} \quad \mathsf{proc}(a, \mathsf{send}\, a\, (\mathsf{new}\, u); !(y \leftarrow \mathsf{recv}\ u; P_y)), \mathsf{proc}(c, u \leftarrow \mathsf{recv}\ a; Q_u)$$
$$\longrightarrow !\mathsf{proc}(b, y \leftarrow \mathsf{recv}\ b; P_y), \mathsf{proc}(c, [b/u]\, Q_u) \qquad (\text{b fresh})$$

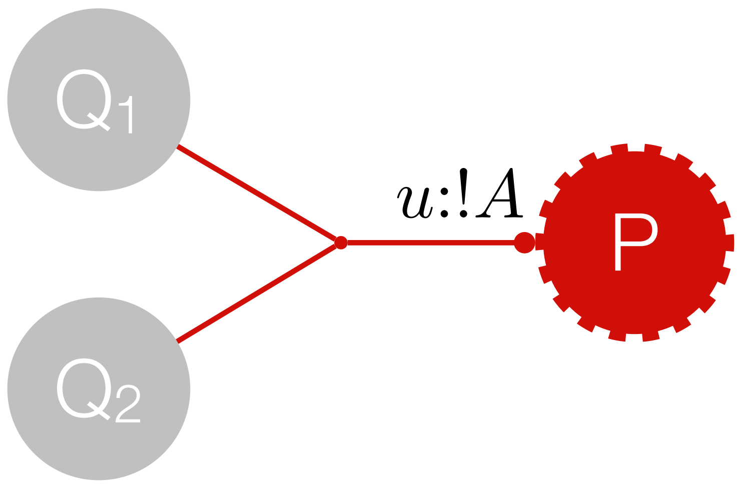
Taking stock

Taking stock

Replication — clients are shielded from each others effects

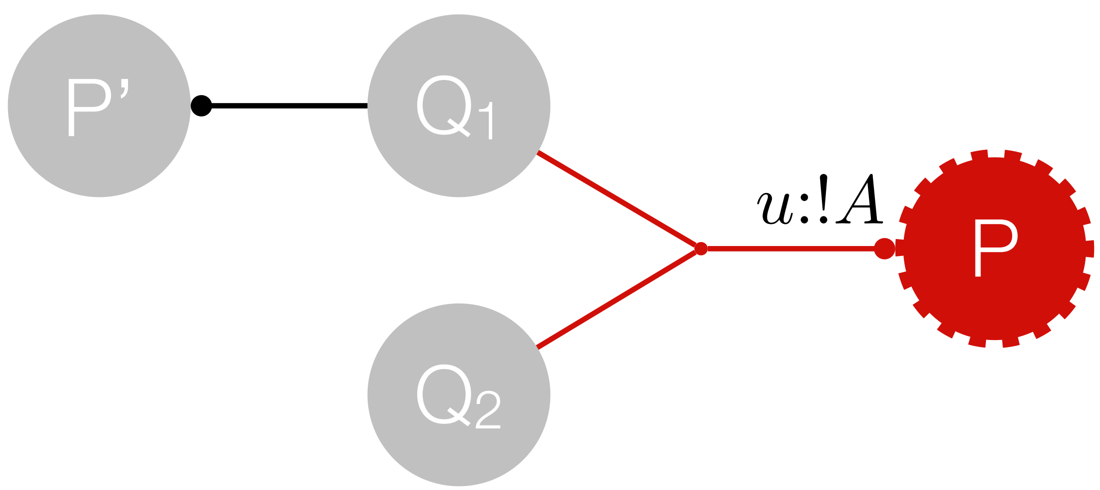
Taking stock

Replication — clients are shielded from each others effects



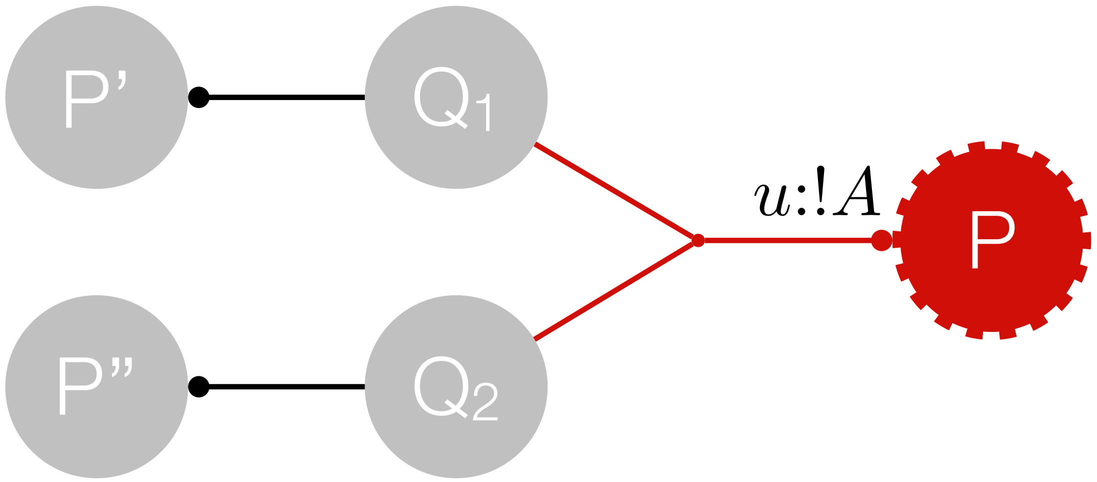
Taking stock

Replication — clients are shielded from each others effects



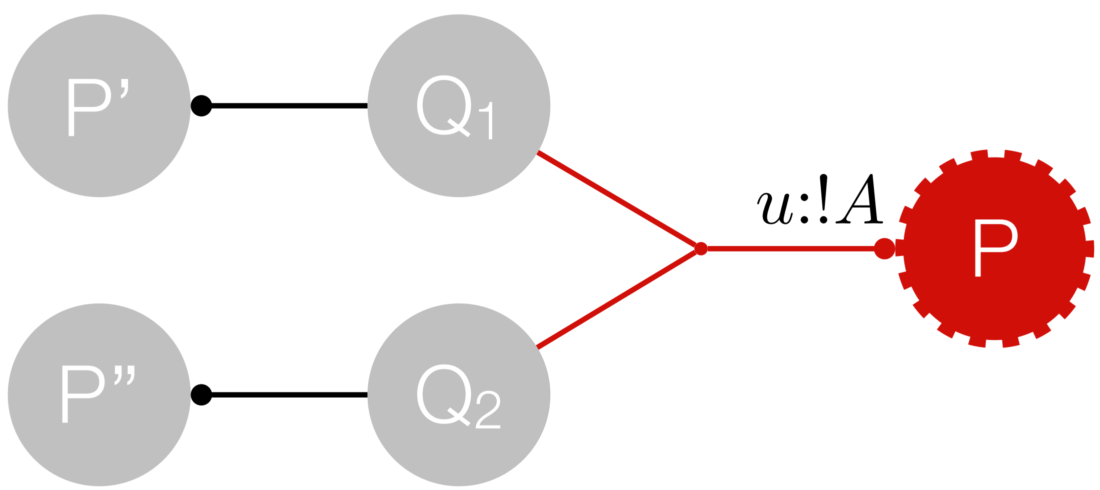
Taking stock

Replication — clients are shielded from each others effects



Taking stock

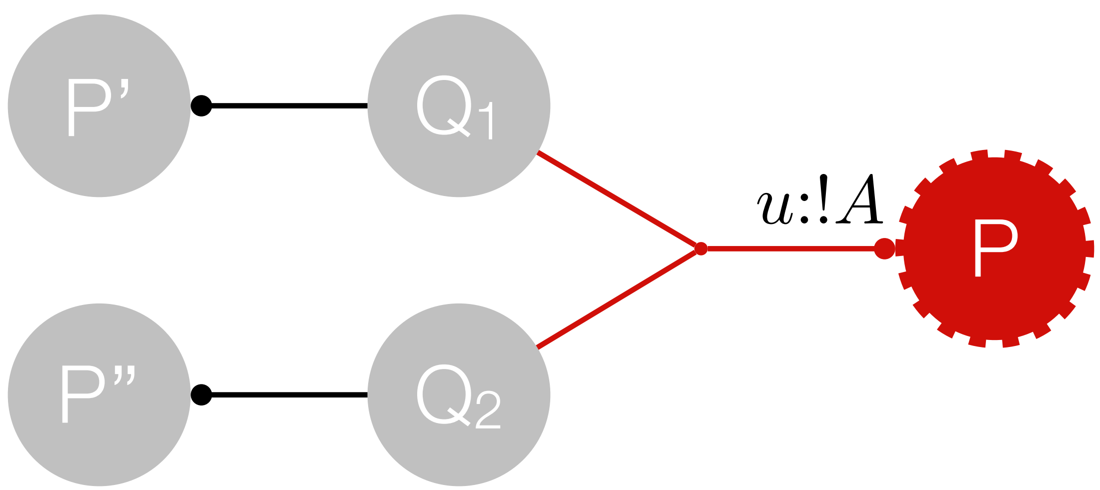
Replication — clients are shielded from each others effects



any communication of one client with its copy of P will not affect the private copies of P of other clients

Taking stock

Replication — clients are shielded from each others effects

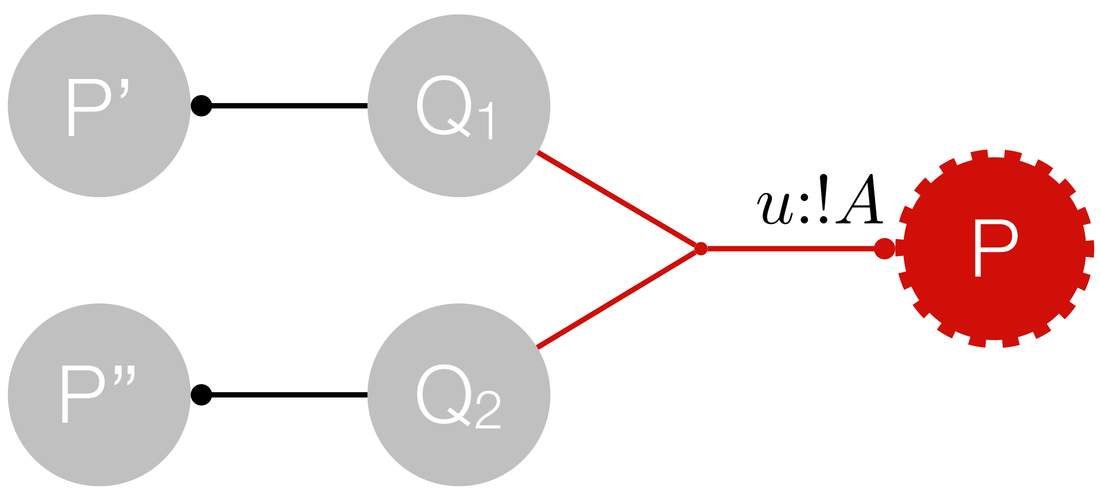


any communication of one client with its copy of P will not affect the private copies of P of other clients

for some applications this copying semantics is appropriate

Taking stock

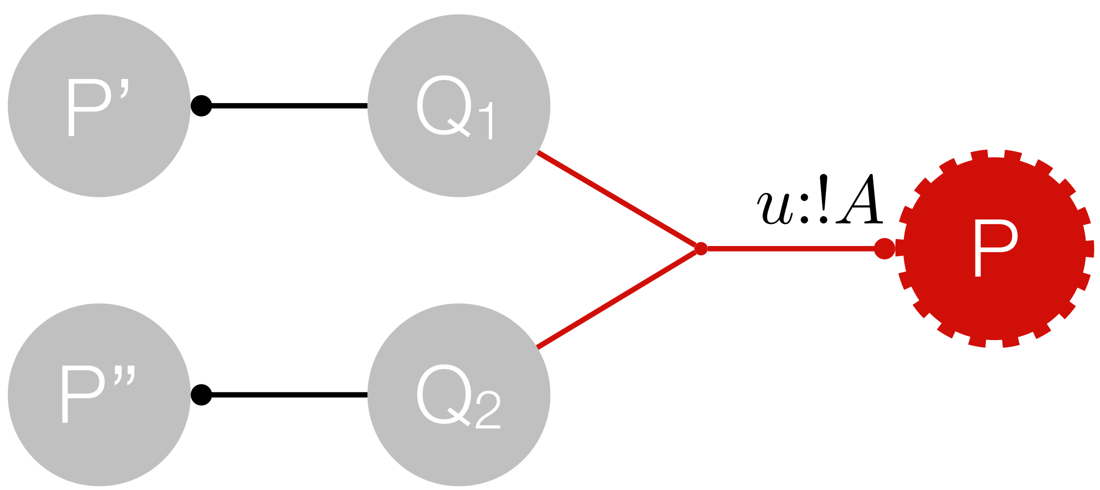
Replication — clients are shielded from each others effects



- any communication of one client with its copy of P will not affect the private copies of P of other clients
- for some applications this copying semantics is appropriate
- other applications need a true sharing semantics

Taking stock

Replication — clients are shielded from each others effects



let's explore next!

- any communication of one client with its copy of P will not affect the private copies of P of other clients
- for some applications this copying semantics is appropriate
- other applications need a true sharing semantics

Manifest sharing

Manifest sharing — key ideas

Manifest sharing — key ideas

permit aliases, rather than ruling them out

Manifest sharing — key ideas

- permit aliases, rather than ruling them out
- to guarantee preservation

Manifest sharing — key ideas

- permit aliases, rather than ruling them out
- to guarantee preservation
  - exclusive access required prior any communication

Manifest sharing — key ideas

- permit aliases, rather than ruling them out
- to guarantee preservation
  - exclusive access required prior any communication
  - relinquish exclusive access in consistent state

Manifest sharing — key ideas

- permit aliases, rather than ruling them out
- to guarantee preservation
  - exclusive access required prior any communication
  - relinquish exclusive access in consistent state
- manifest these ideas in type structure

Manifest sharing — key ideas

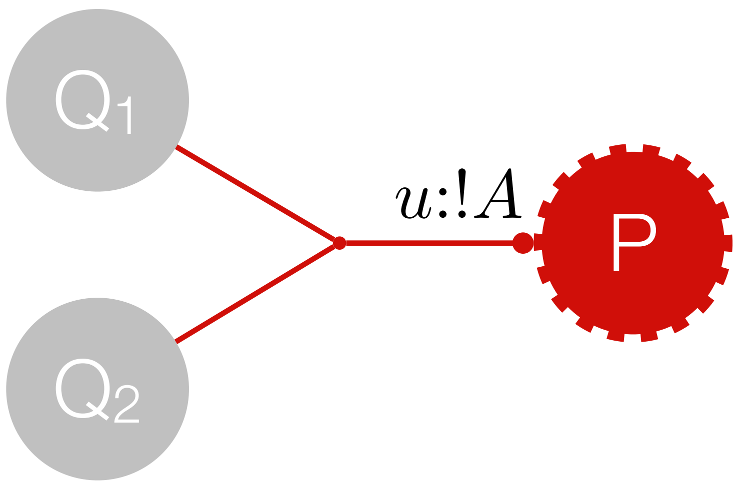
- permit aliases, rather than ruling them out
- to guarantee preservation
  - exclusive access required prior any communication
  - relinquish exclusive access in consistent state
- manifest these ideas in type structure

Stephanie Balzer and Frank Pfenning. Manifest Sharing with Session Types. ICFP, 2017.

Copying versus sharing semantics

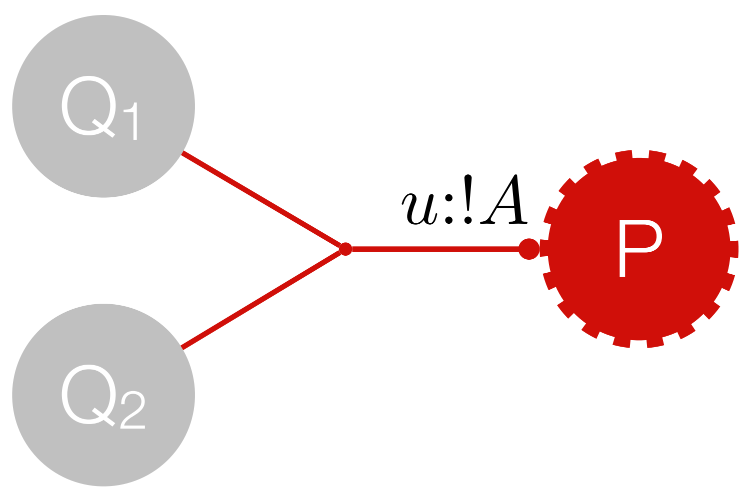
Copying versus sharing semantics

Copying semantics

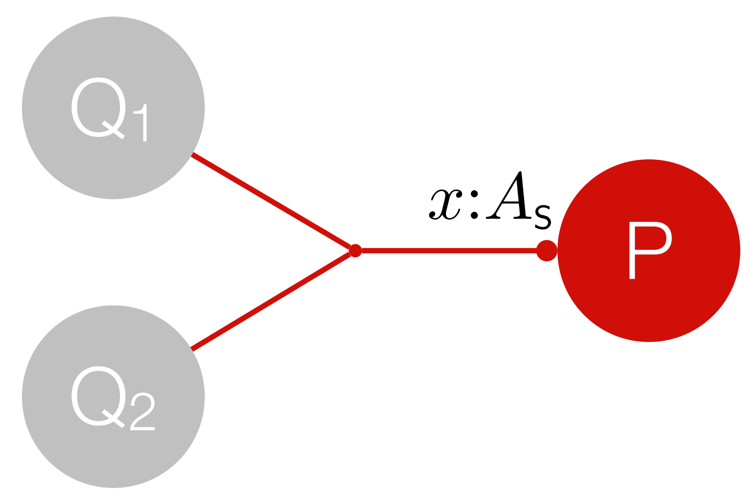


Copying versus sharing semantics

Copying semantics

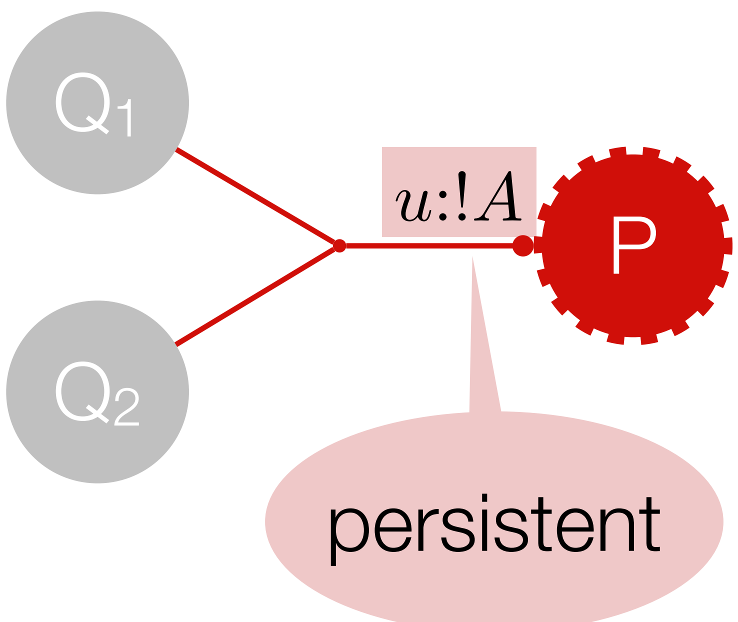


Sharing semantics

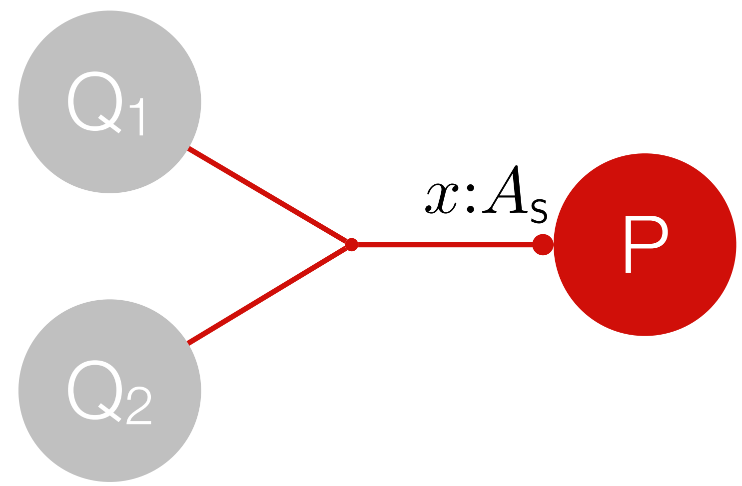


Copying versus sharing semantics

Copying semantics

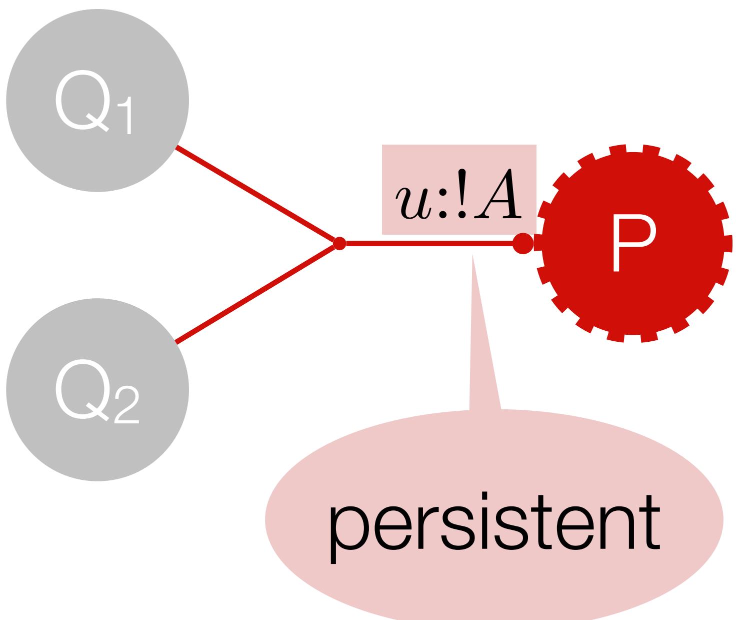


Sharing semantics

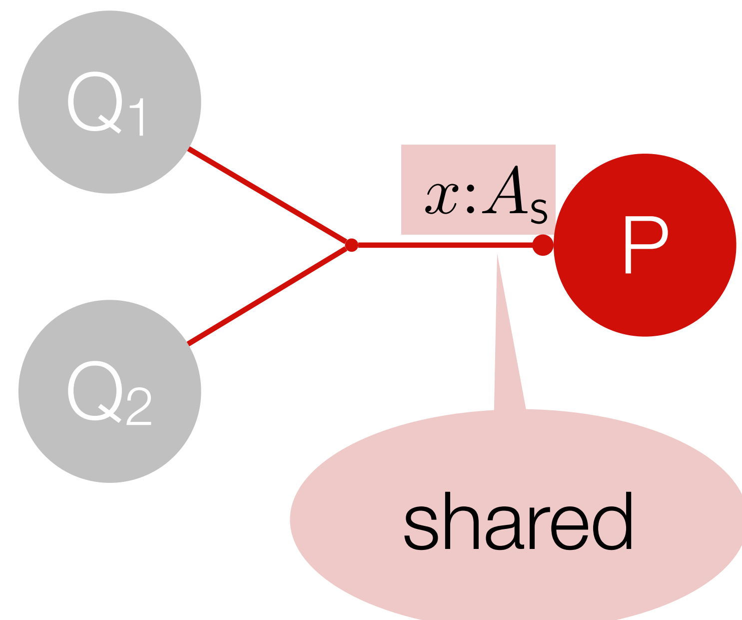


Copying versus sharing semantics

Copying semantics

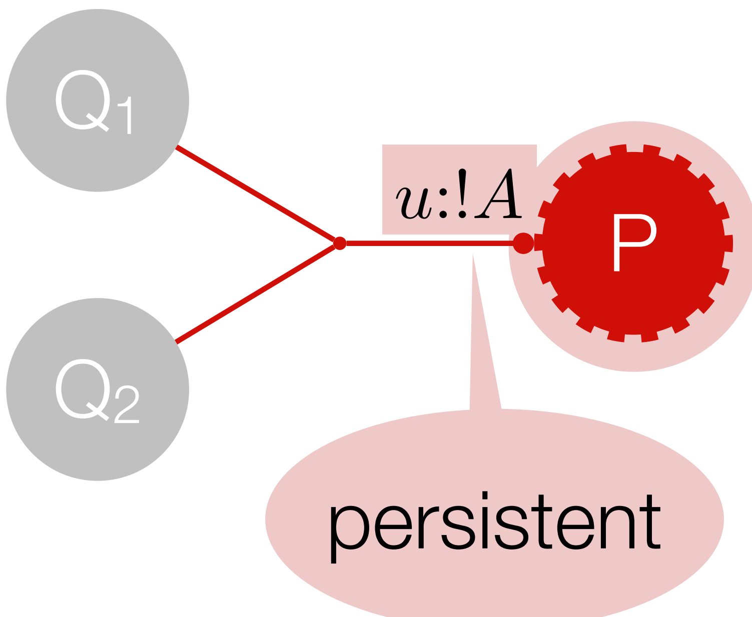


Sharing semantics

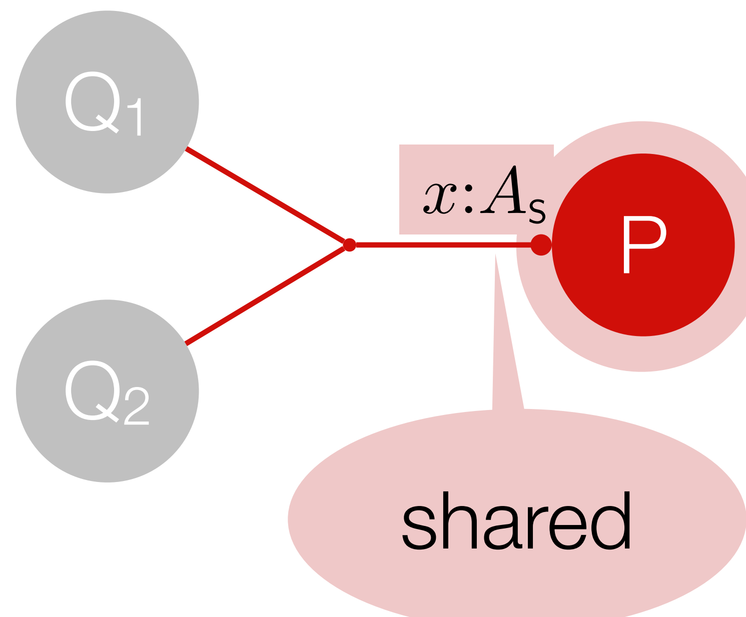


Copying versus sharing semantics

Copying semantics

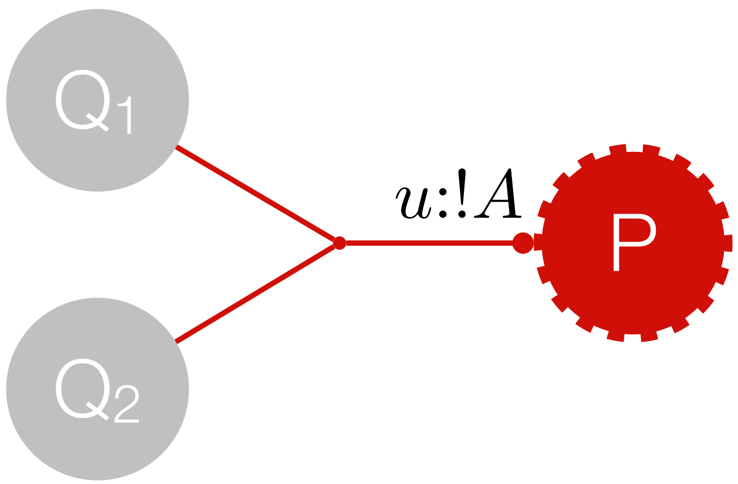


Sharing semantics

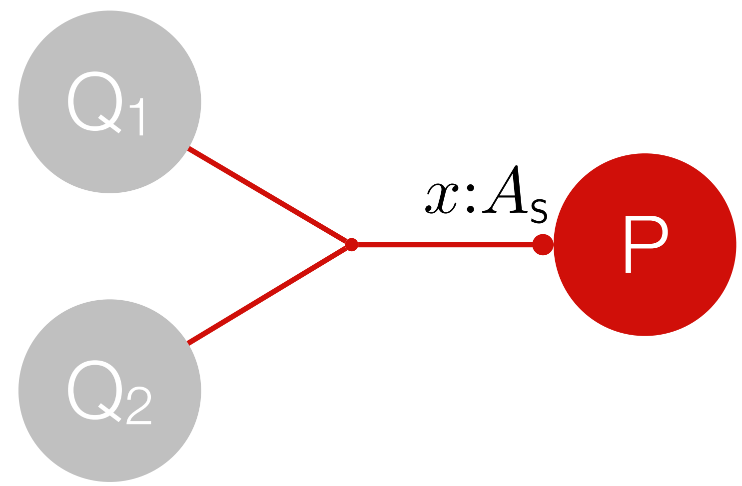


Copying versus sharing semantics

Copying semantics

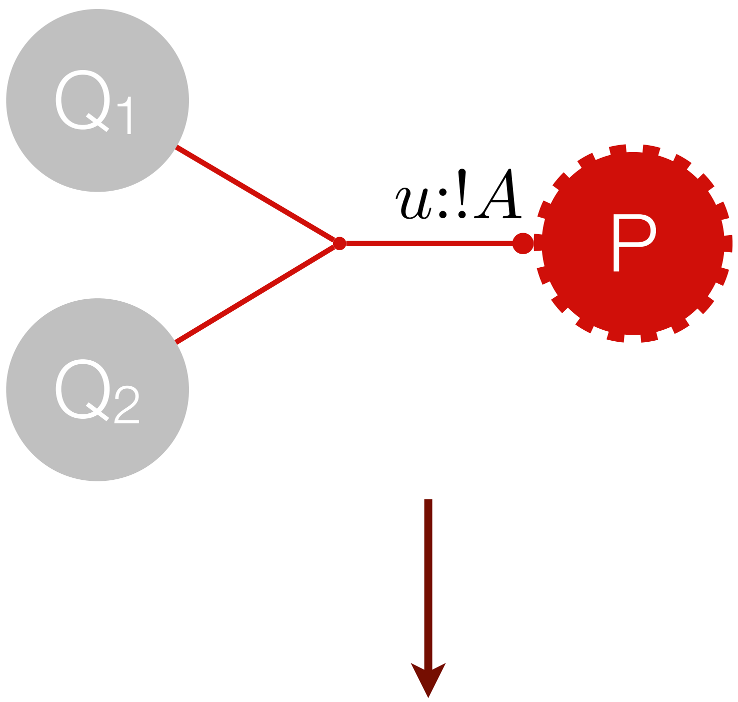


Sharing semantics

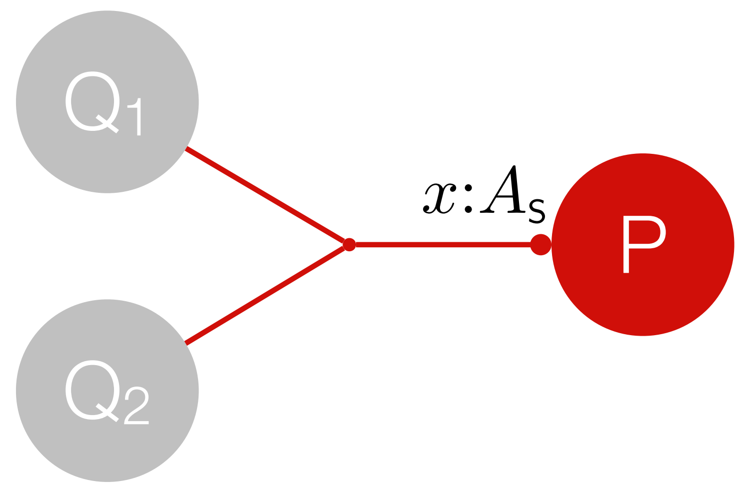


Copying versus sharing semantics

Copying semantics

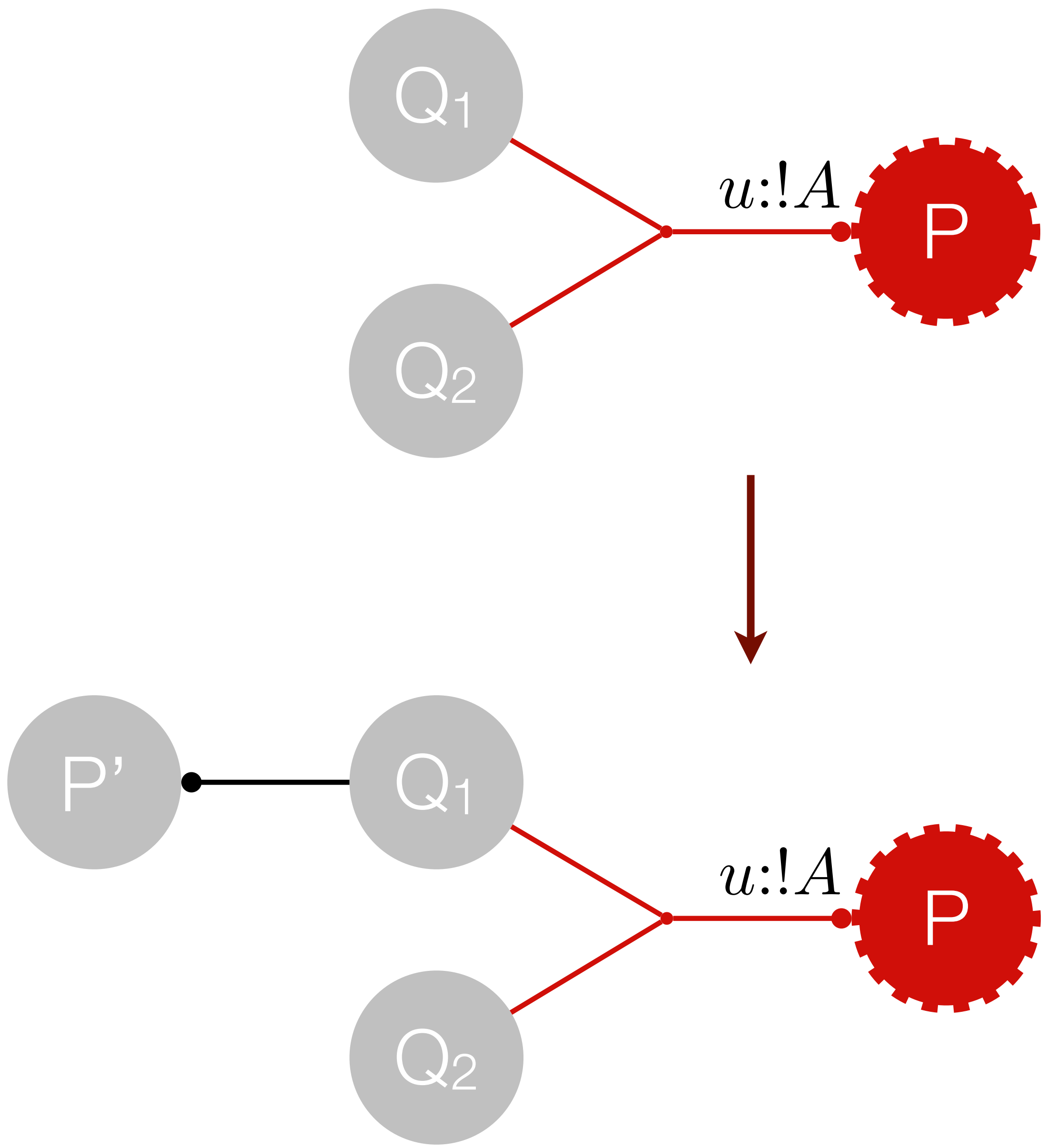


Sharing semantics

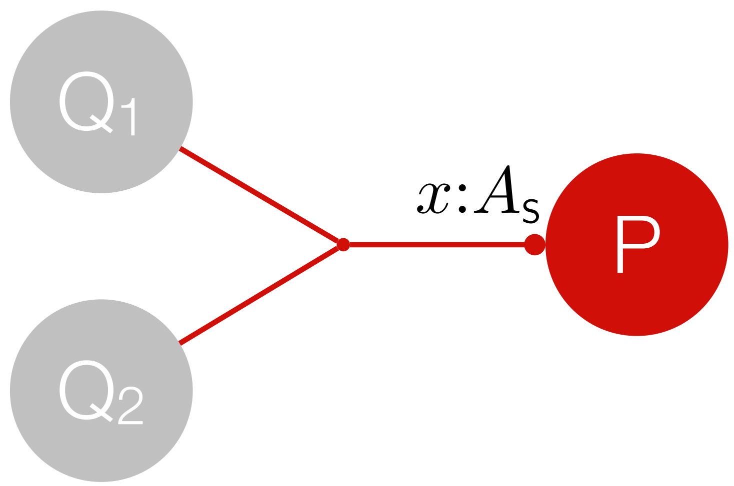


Copying versus sharing semantics

Copying semantics

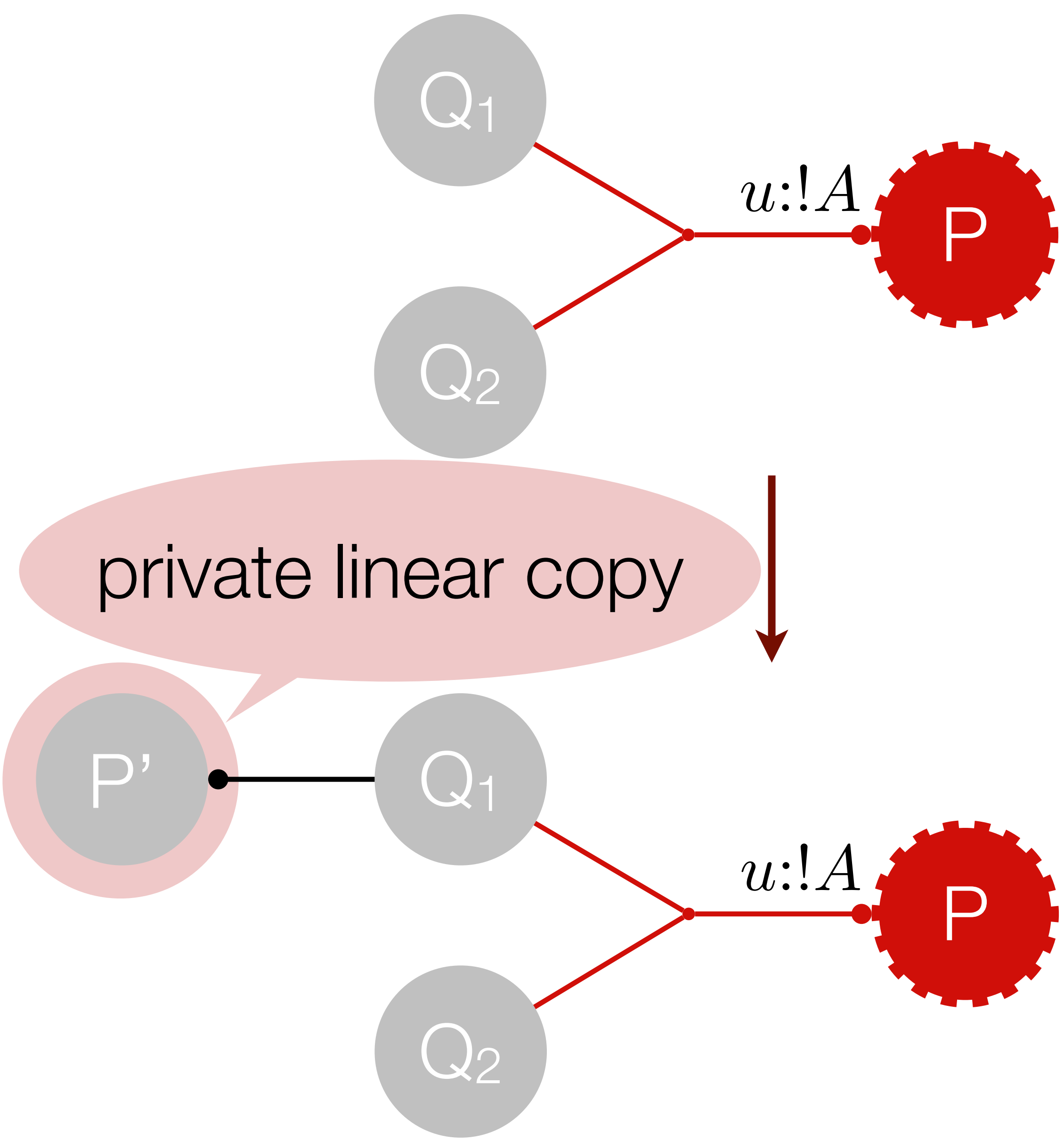


Sharing semantics

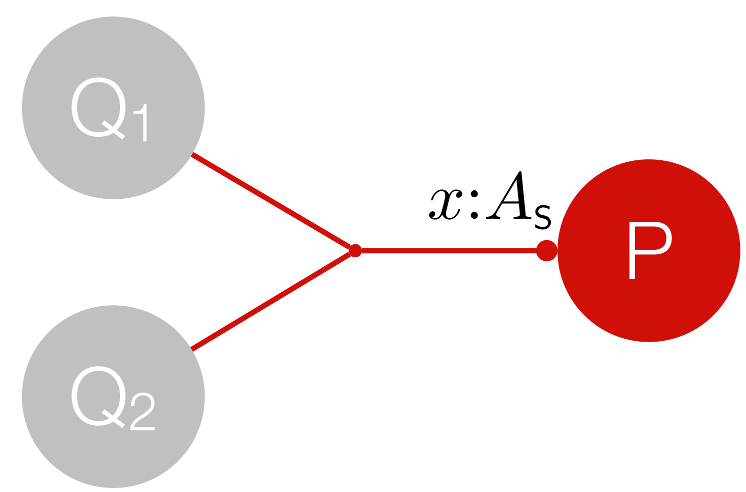


Copying versus sharing semantics

Copying semantics

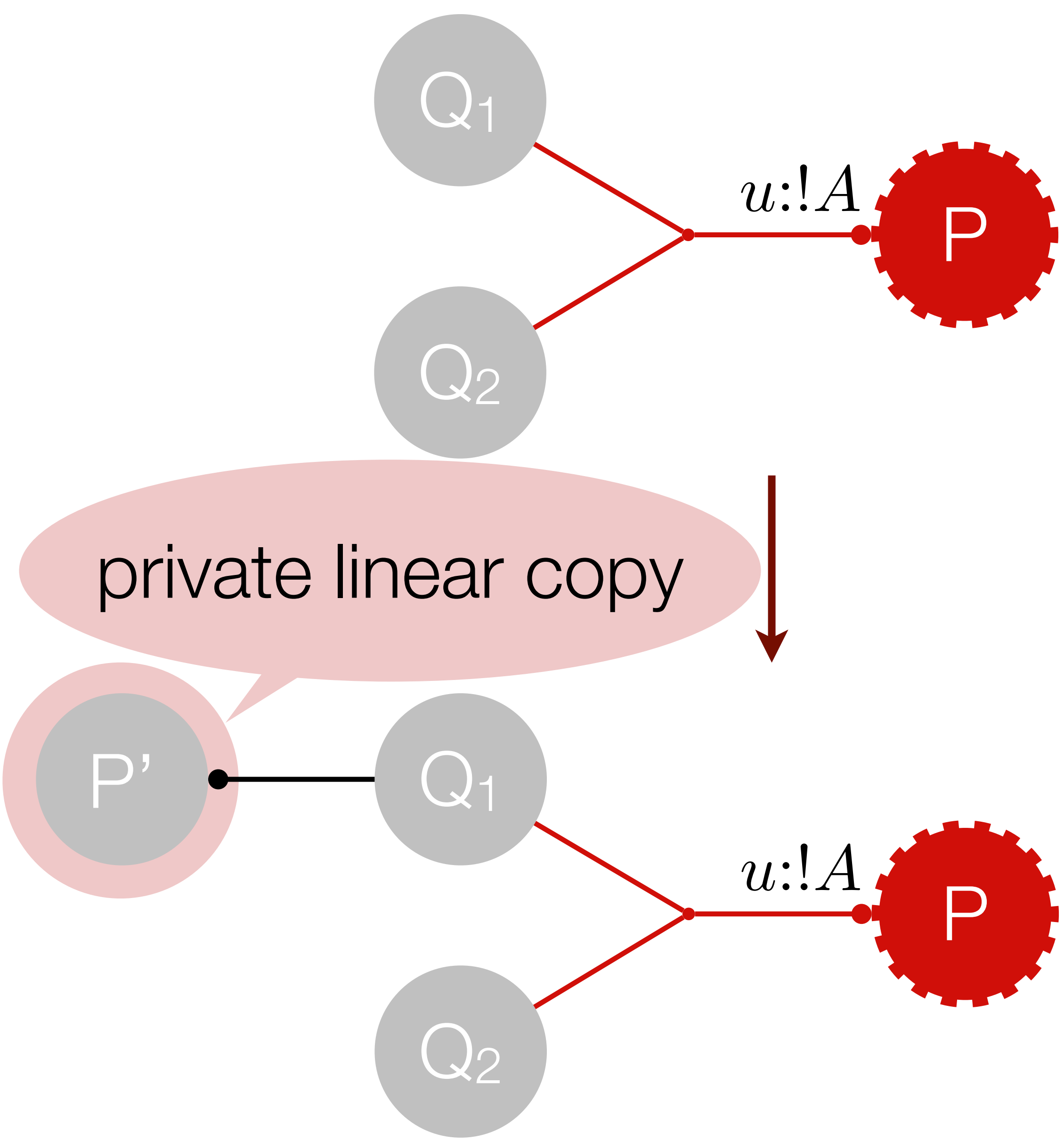


Sharing semantics

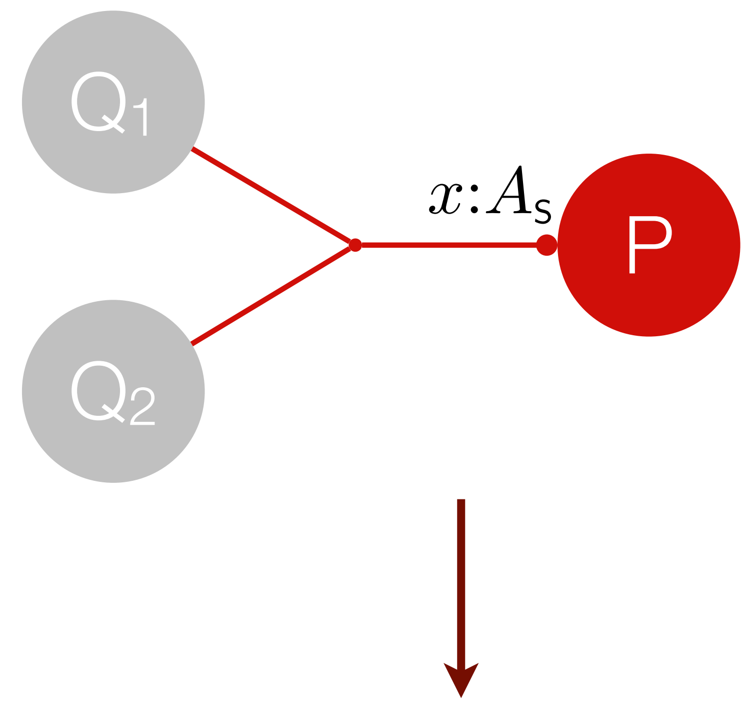


Copying versus sharing semantics

Copying semantics



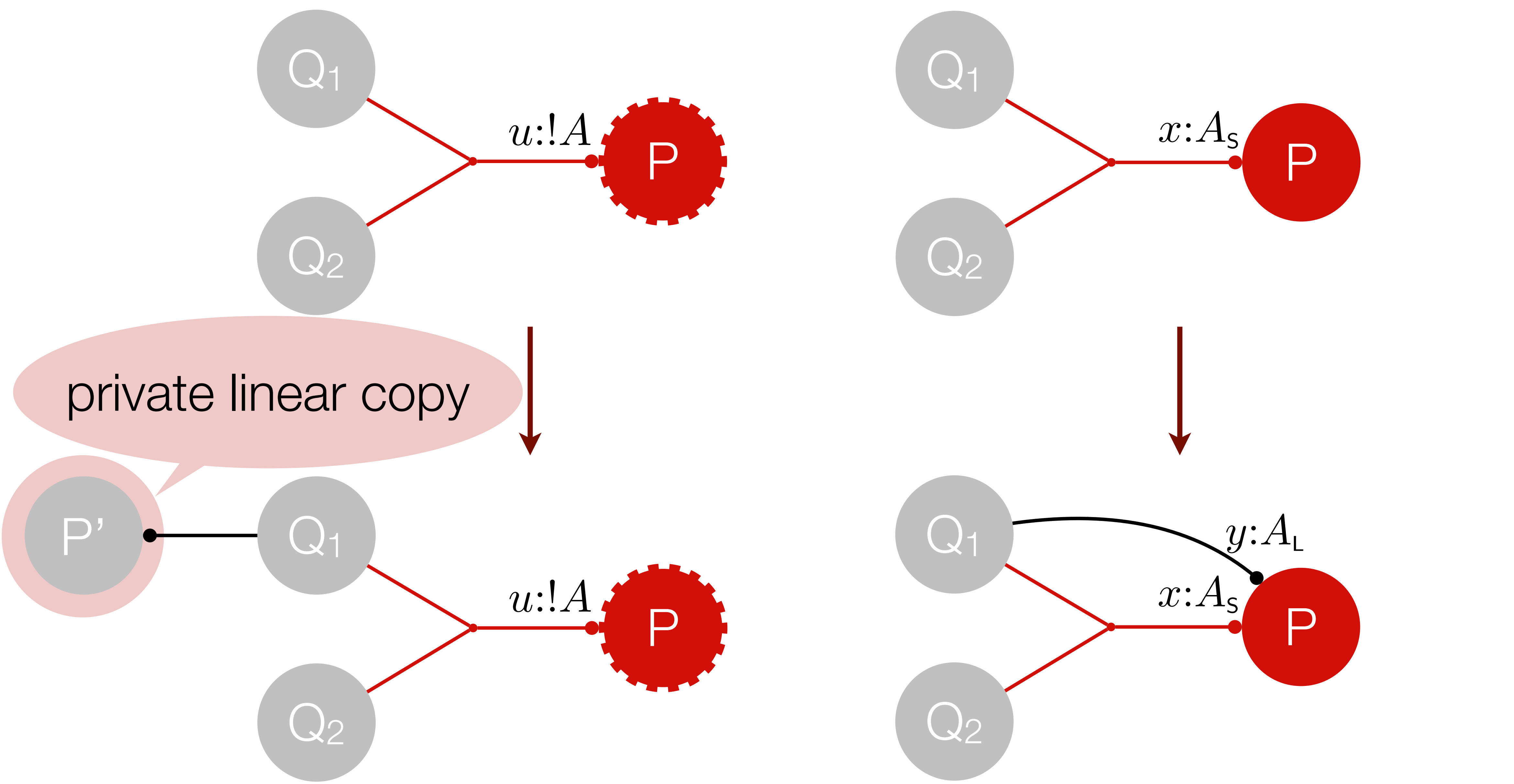
Sharing semantics



Copying versus sharing semantics

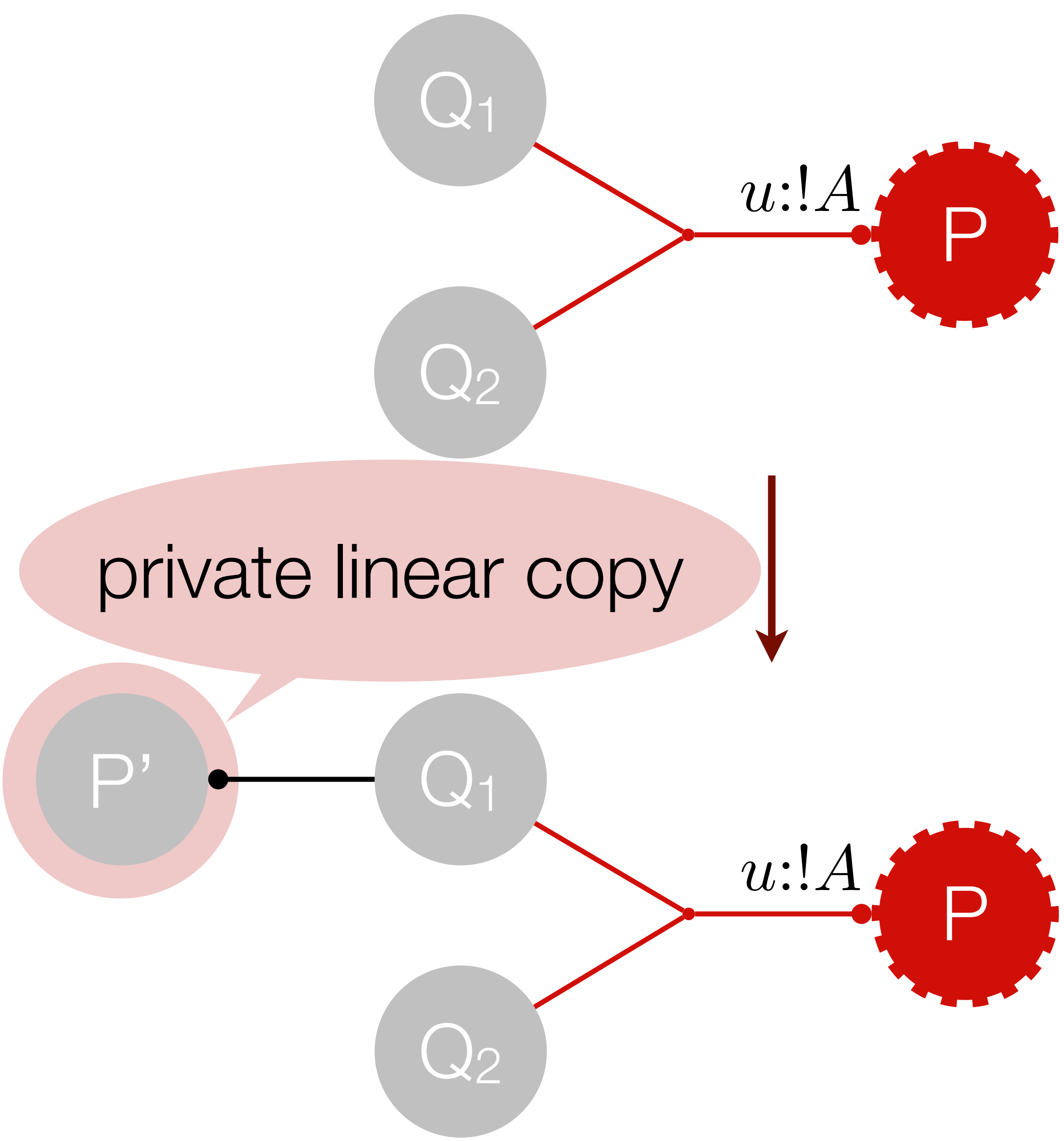
Copying semantics

Sharing semantics

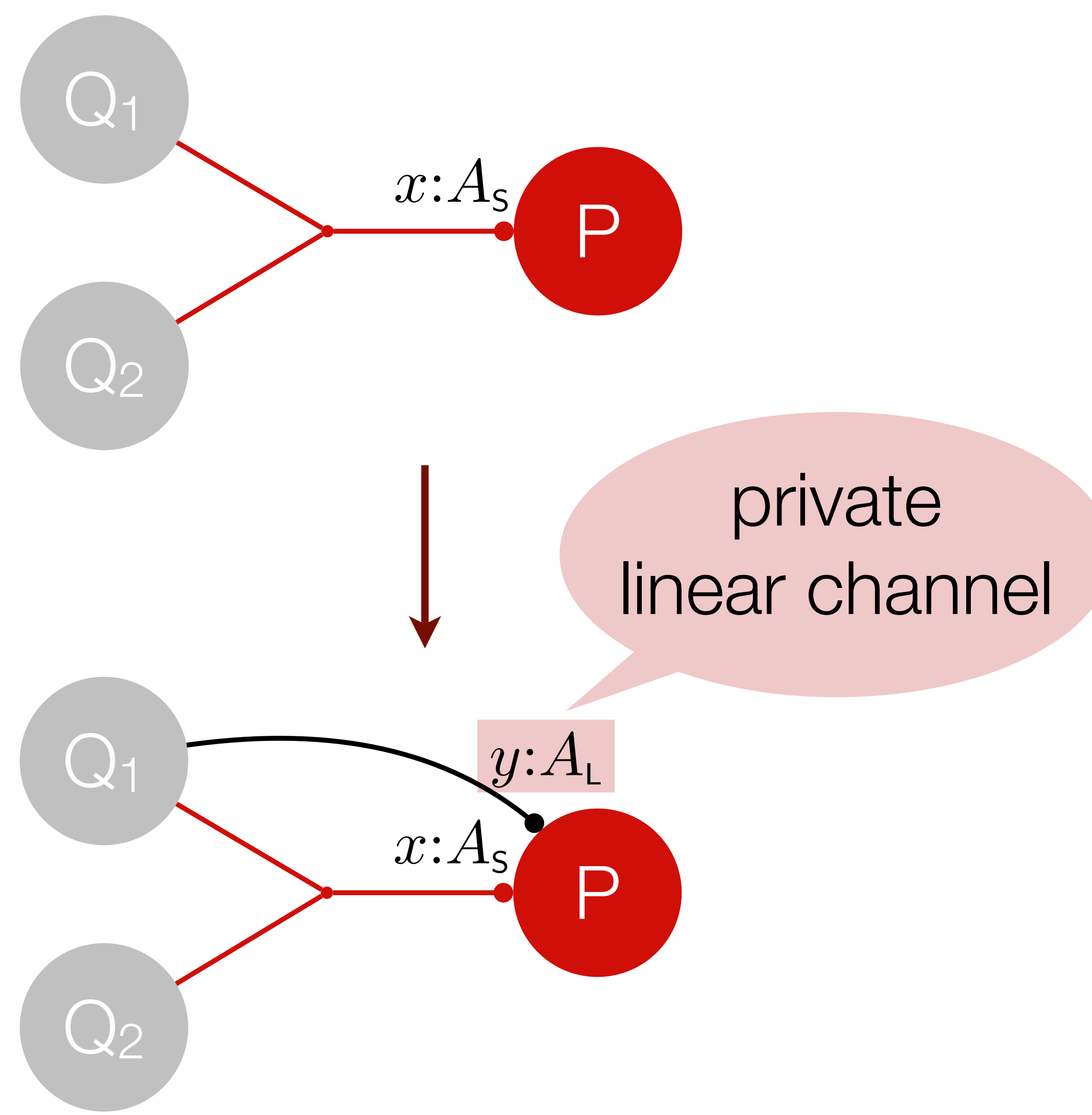


Copying versus sharing semantics

Copying semantics

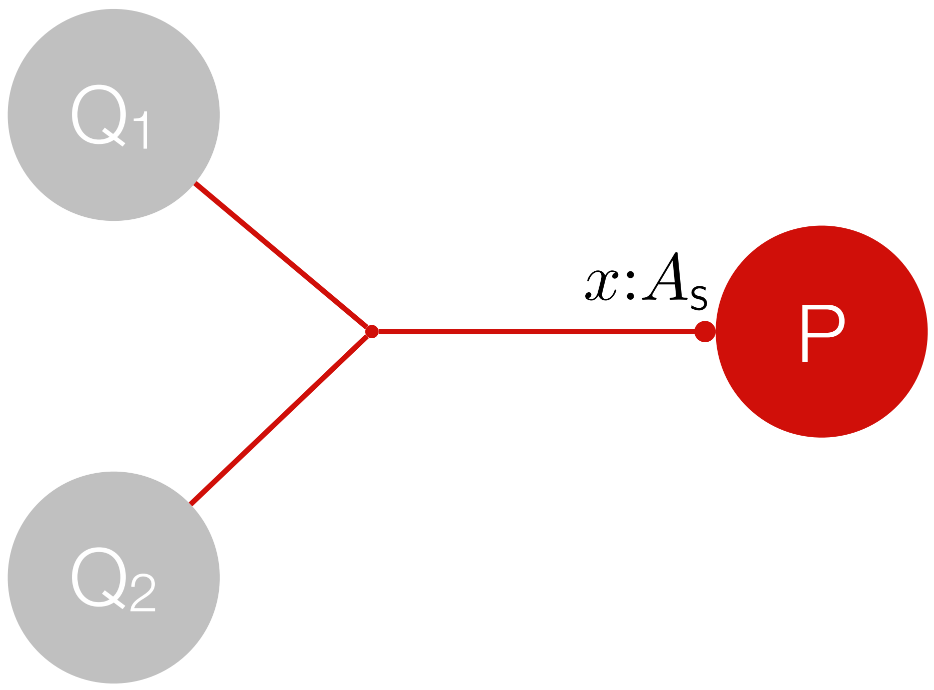


Sharing semantics

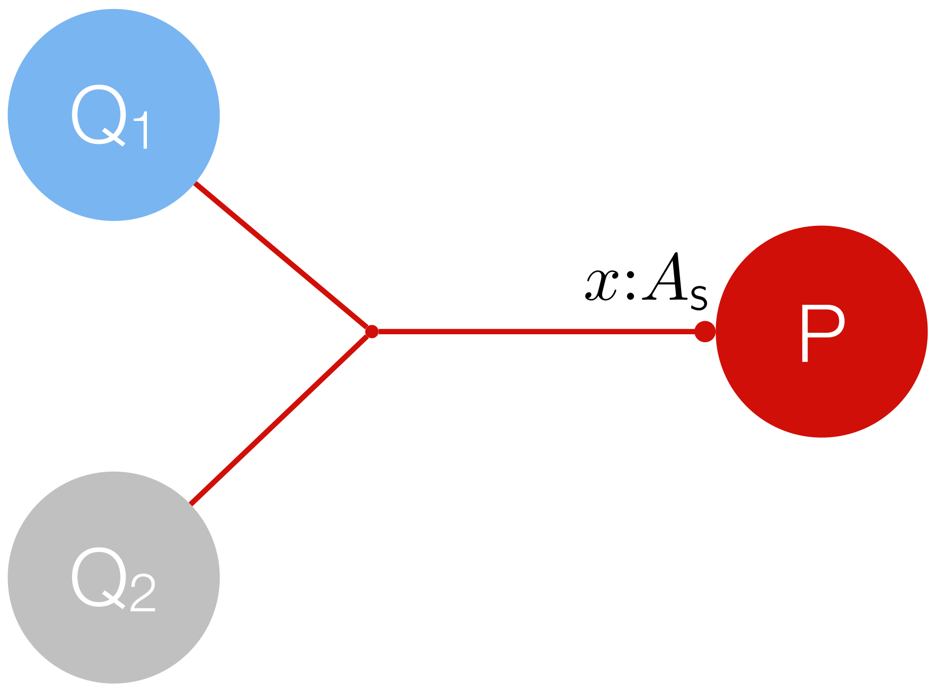


Key idea 1: acquire-release

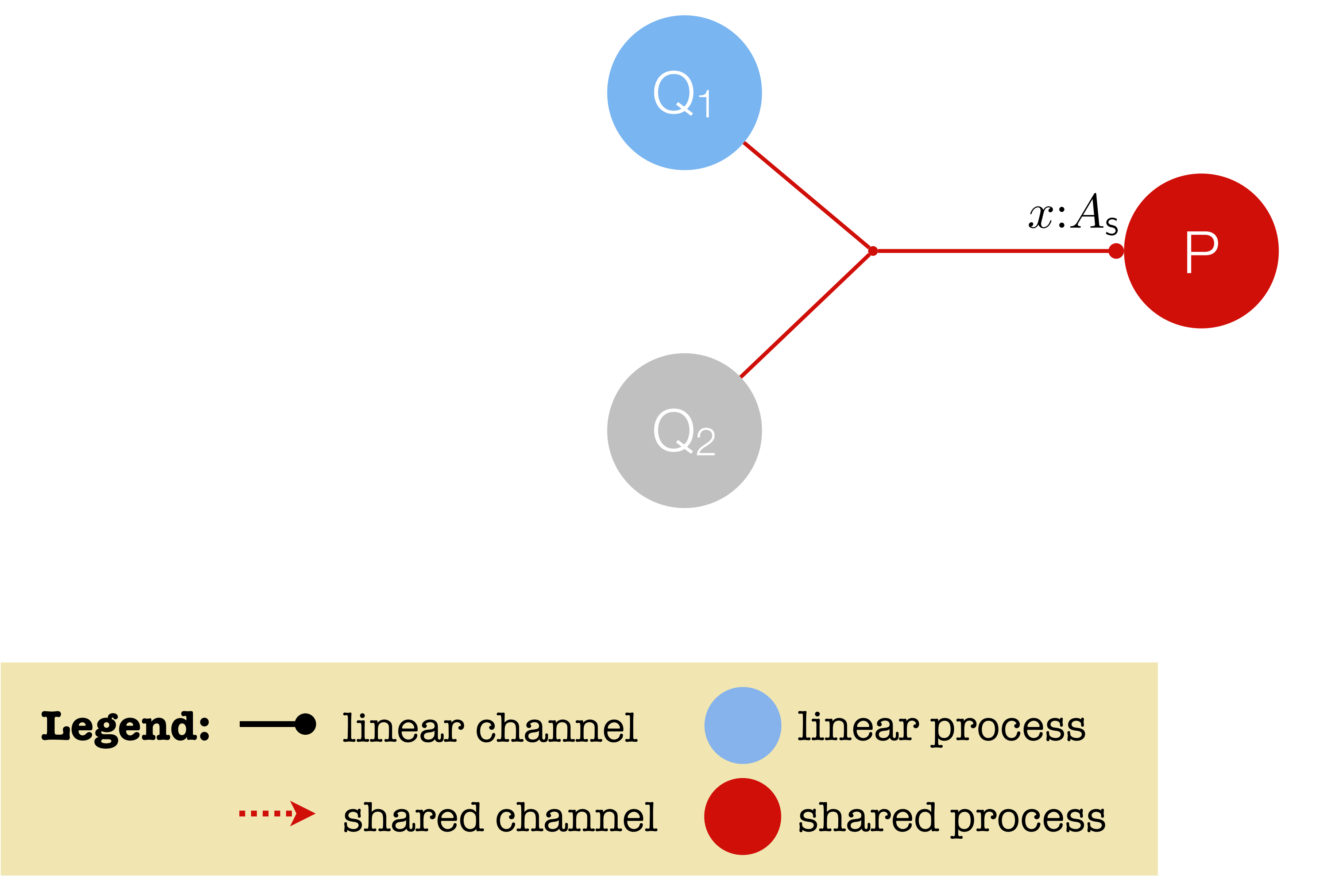
Key idea 1: acquire-release



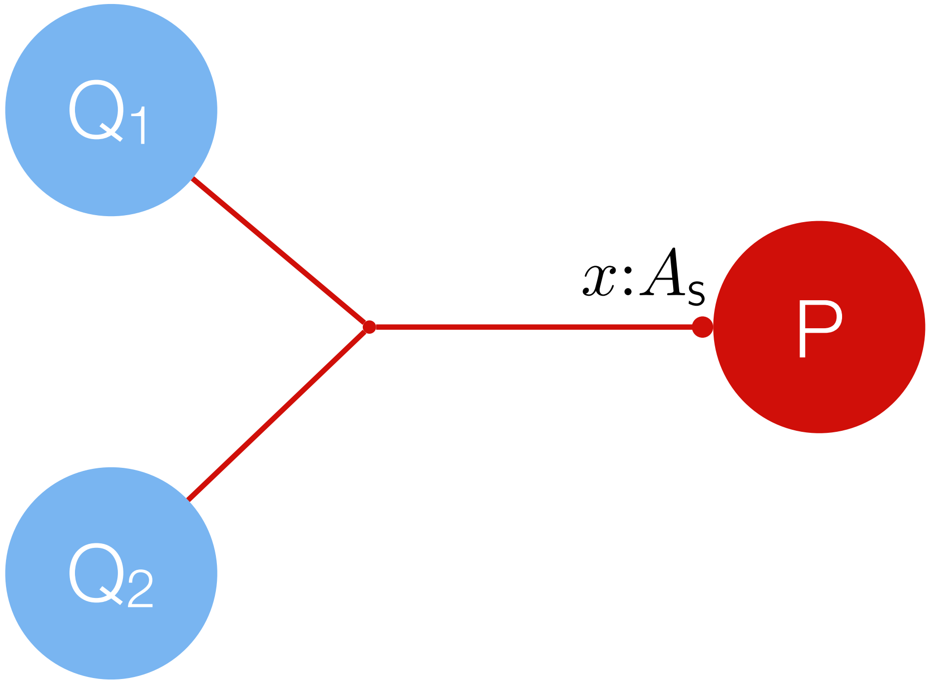
Key idea 1: acquire-release



Key idea 1: acquire-release

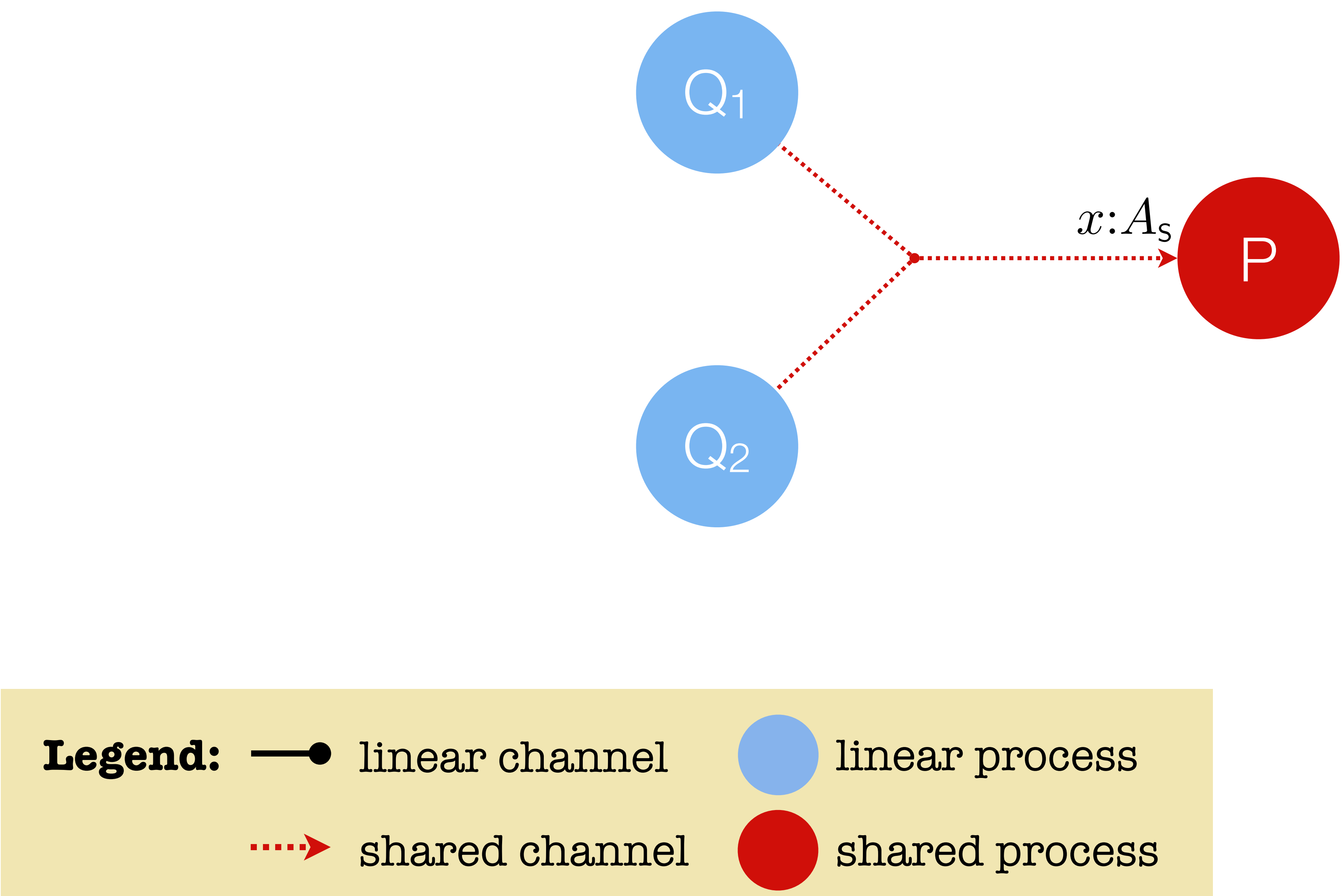


Key idea 1: acquire-release

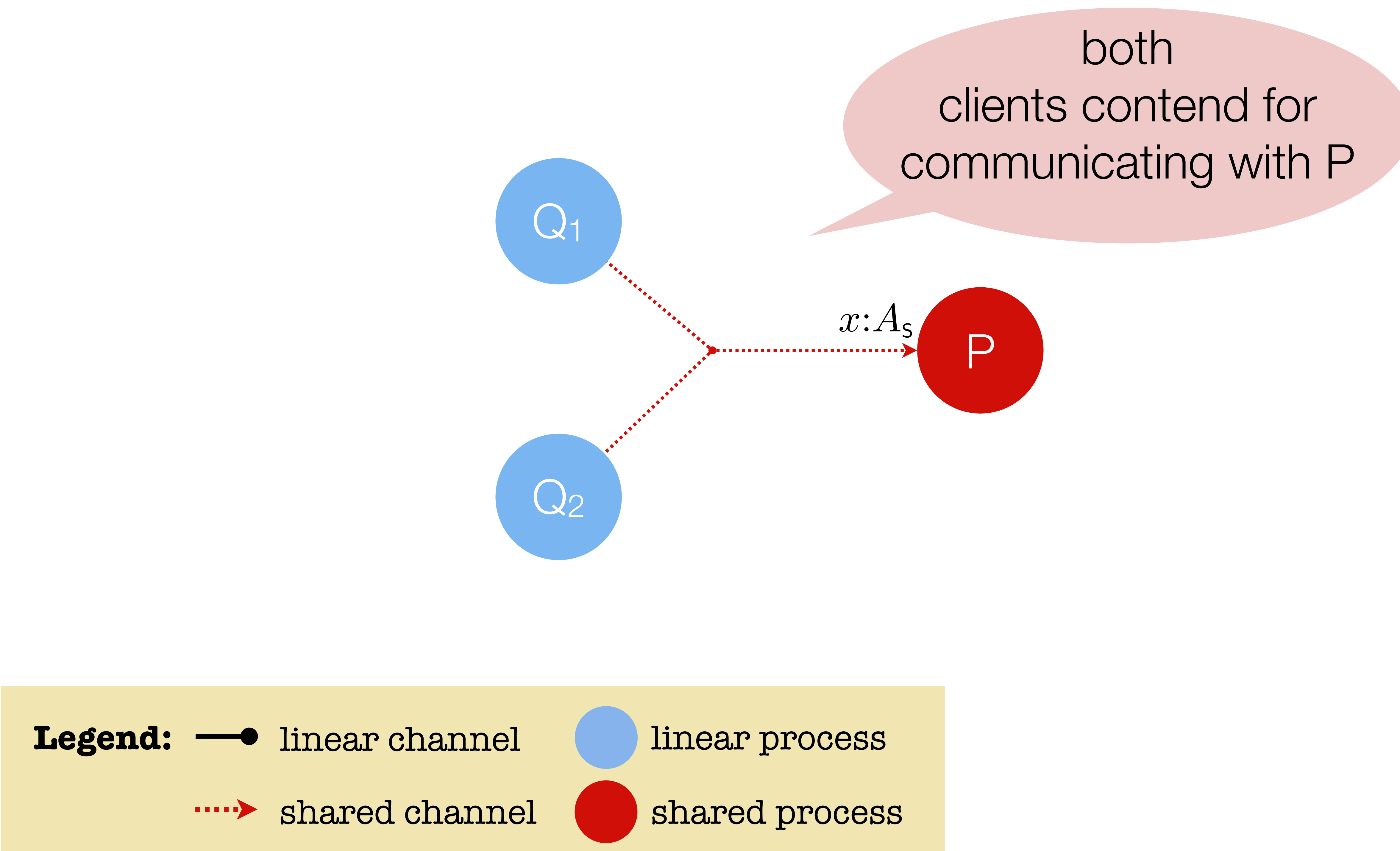


Legend: linear channel — linear process — shared channel — shared process

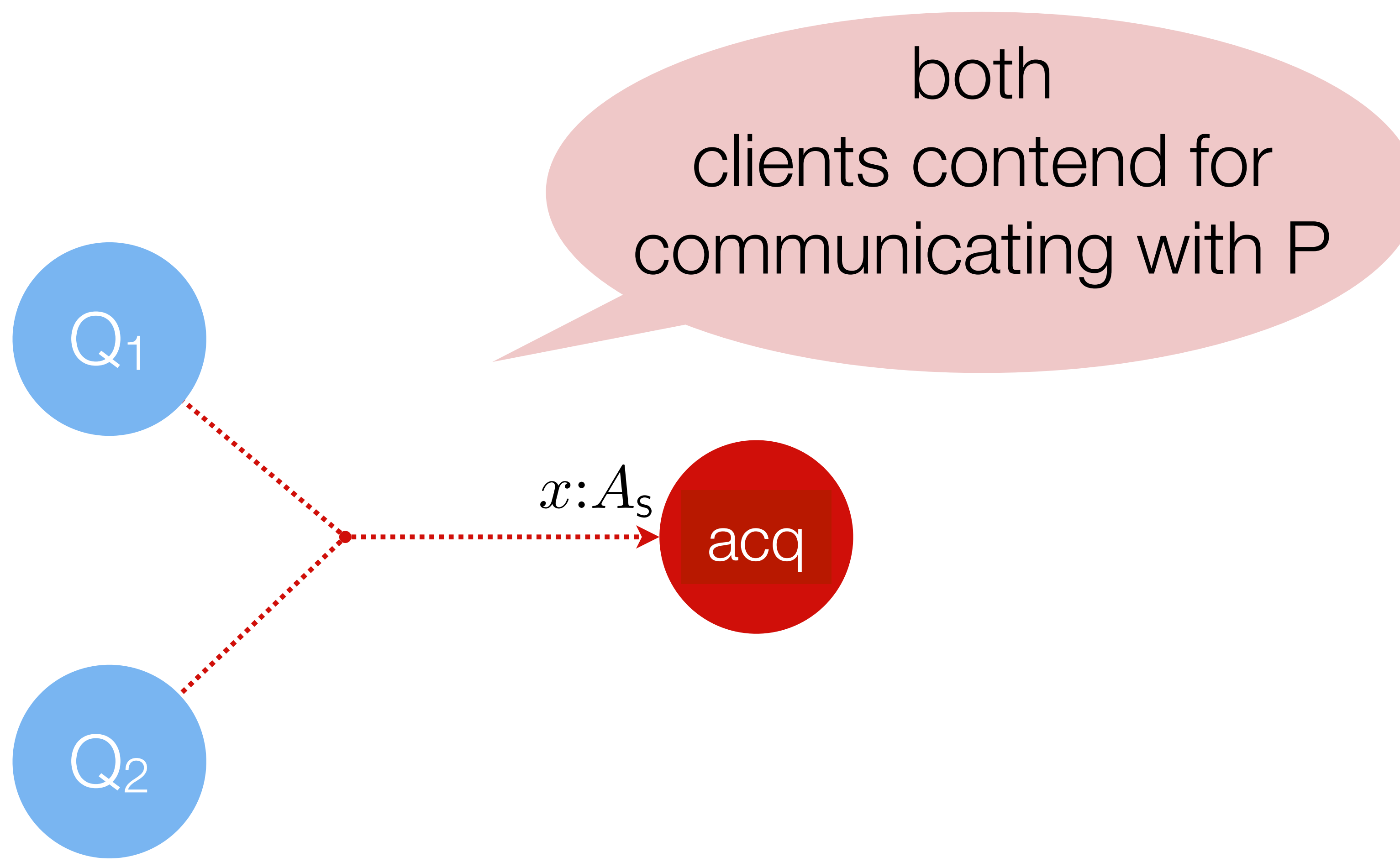
Key idea 1: acquire-release



Key idea 1: acquire-release

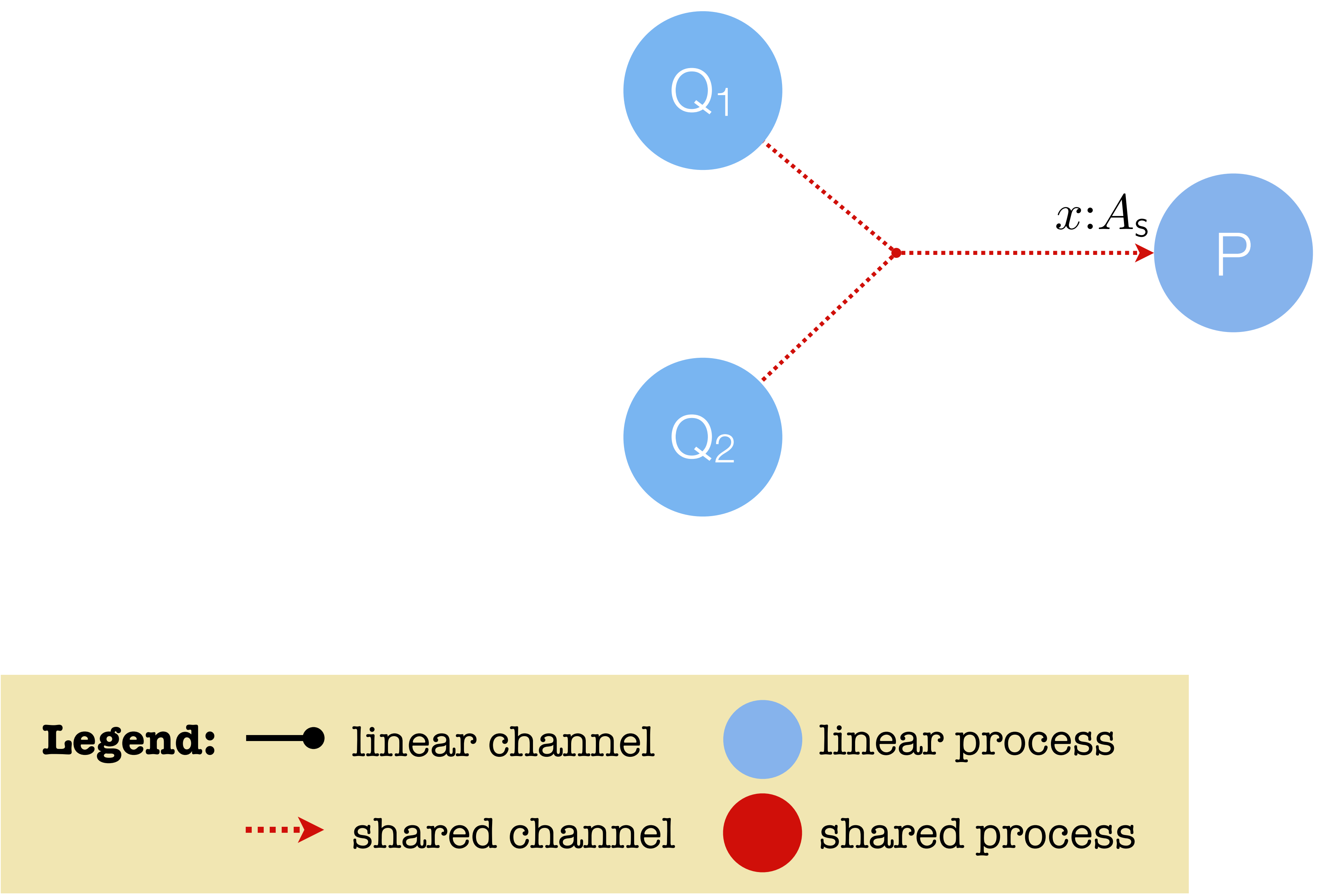


Key idea 1: acquire-release

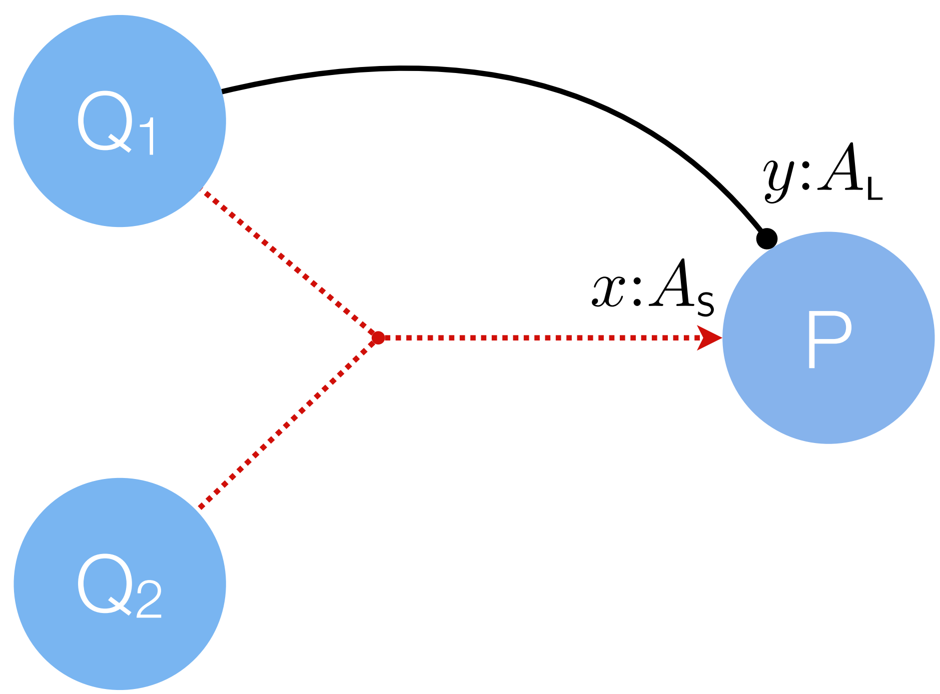


Legend: linear channel, linear process, shared channel, shared process

Key idea 1: acquire-release

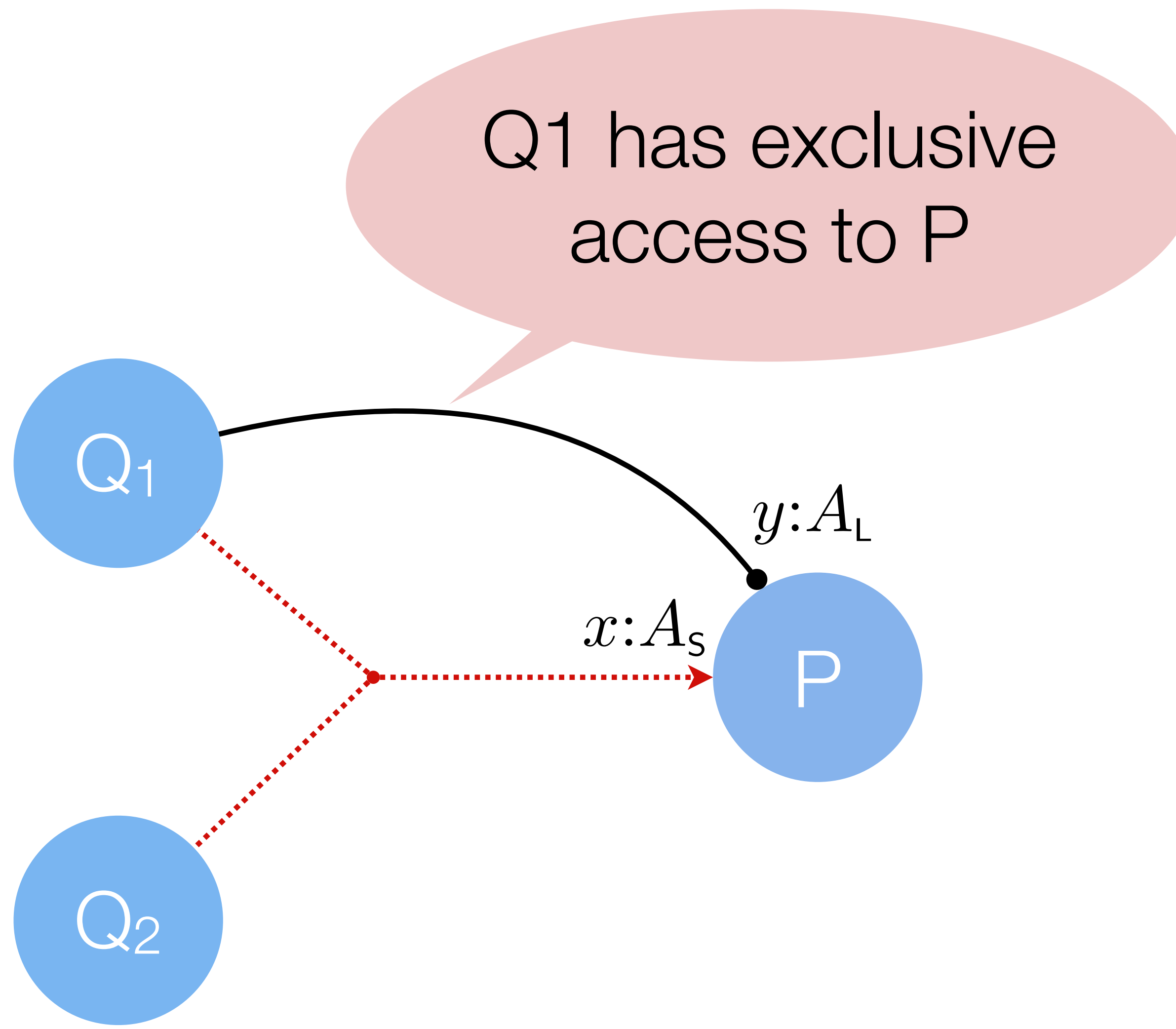


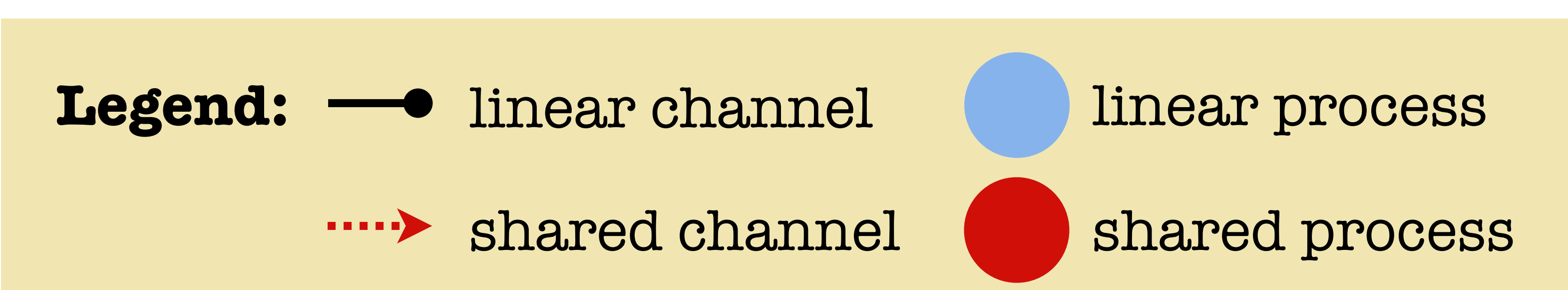
Key idea 1: acquire-release



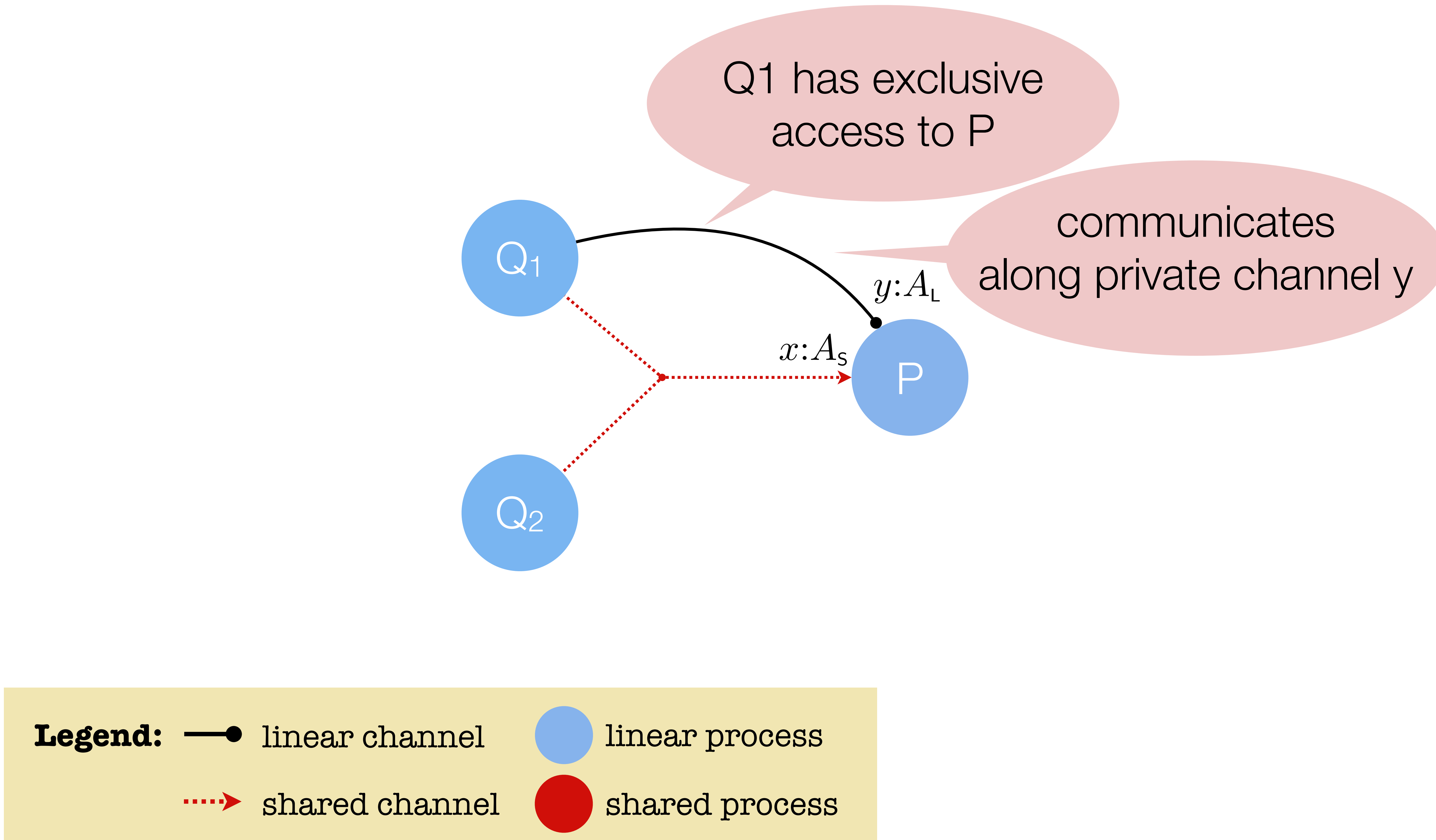
Legend: linear channel, linear process, shared channel, shared process

Key idea 1: acquire-release

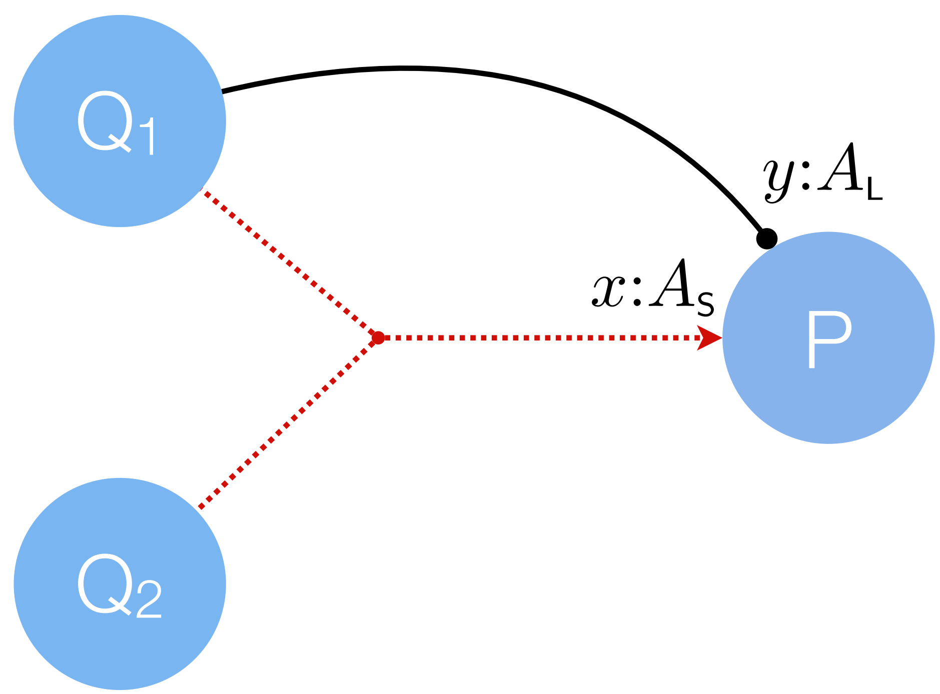




Key idea 1: acquire-release

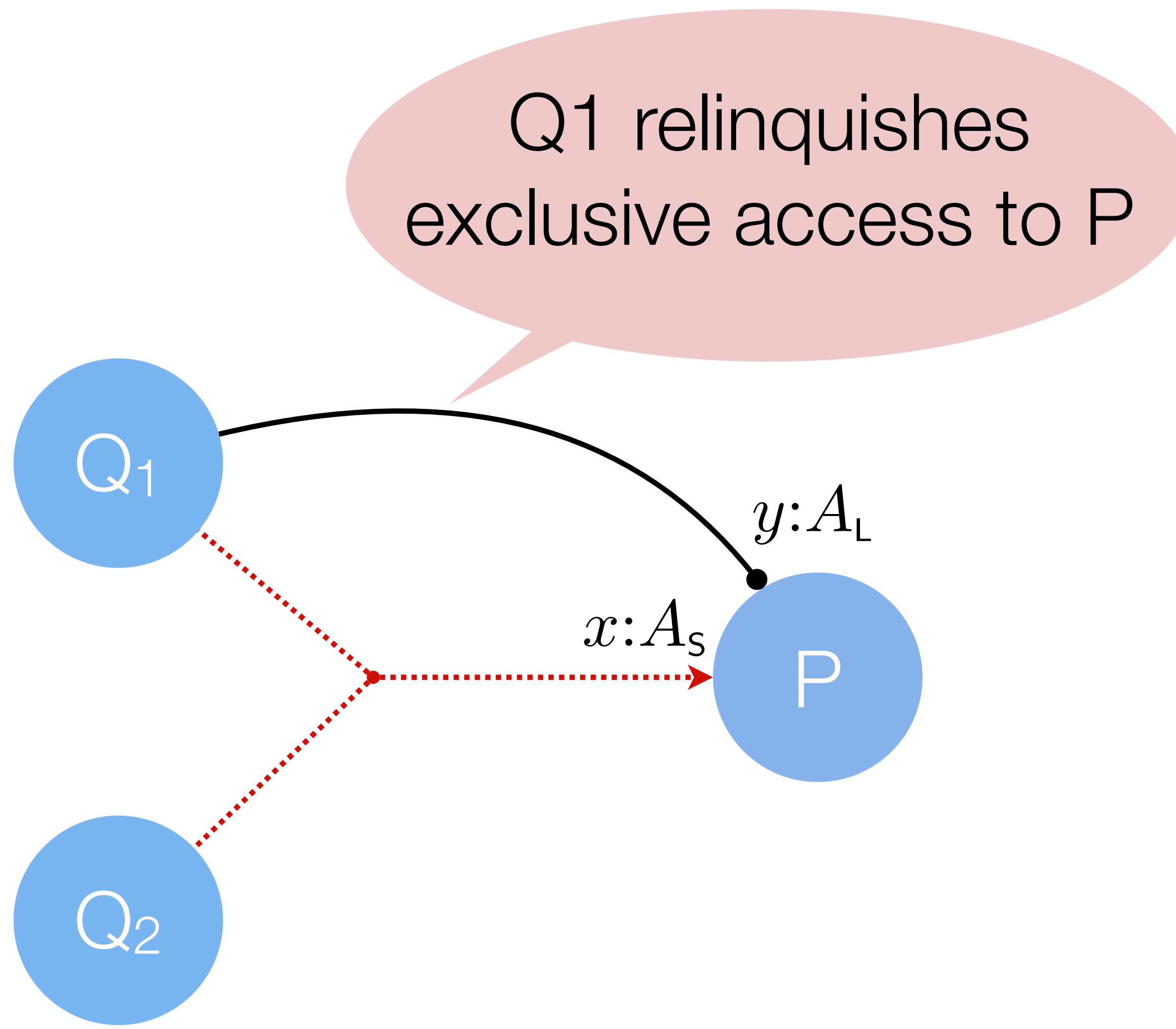


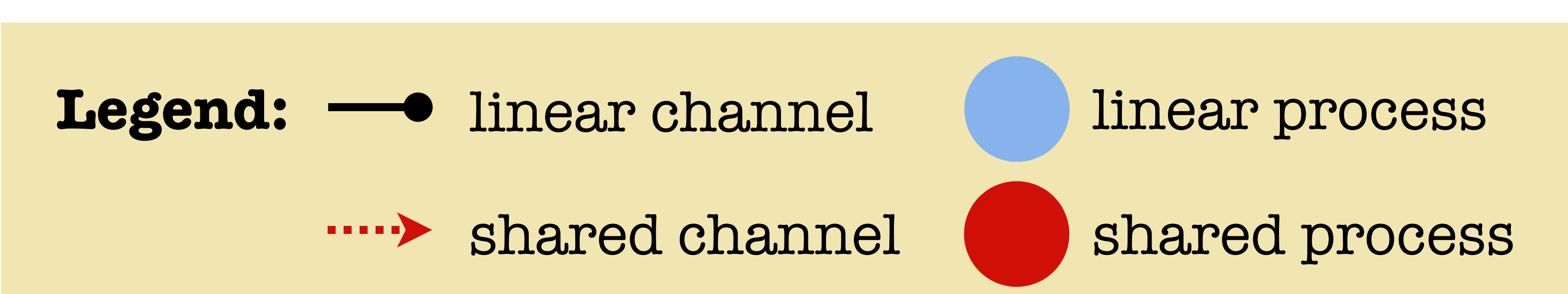
Key idea 1: acquire-release



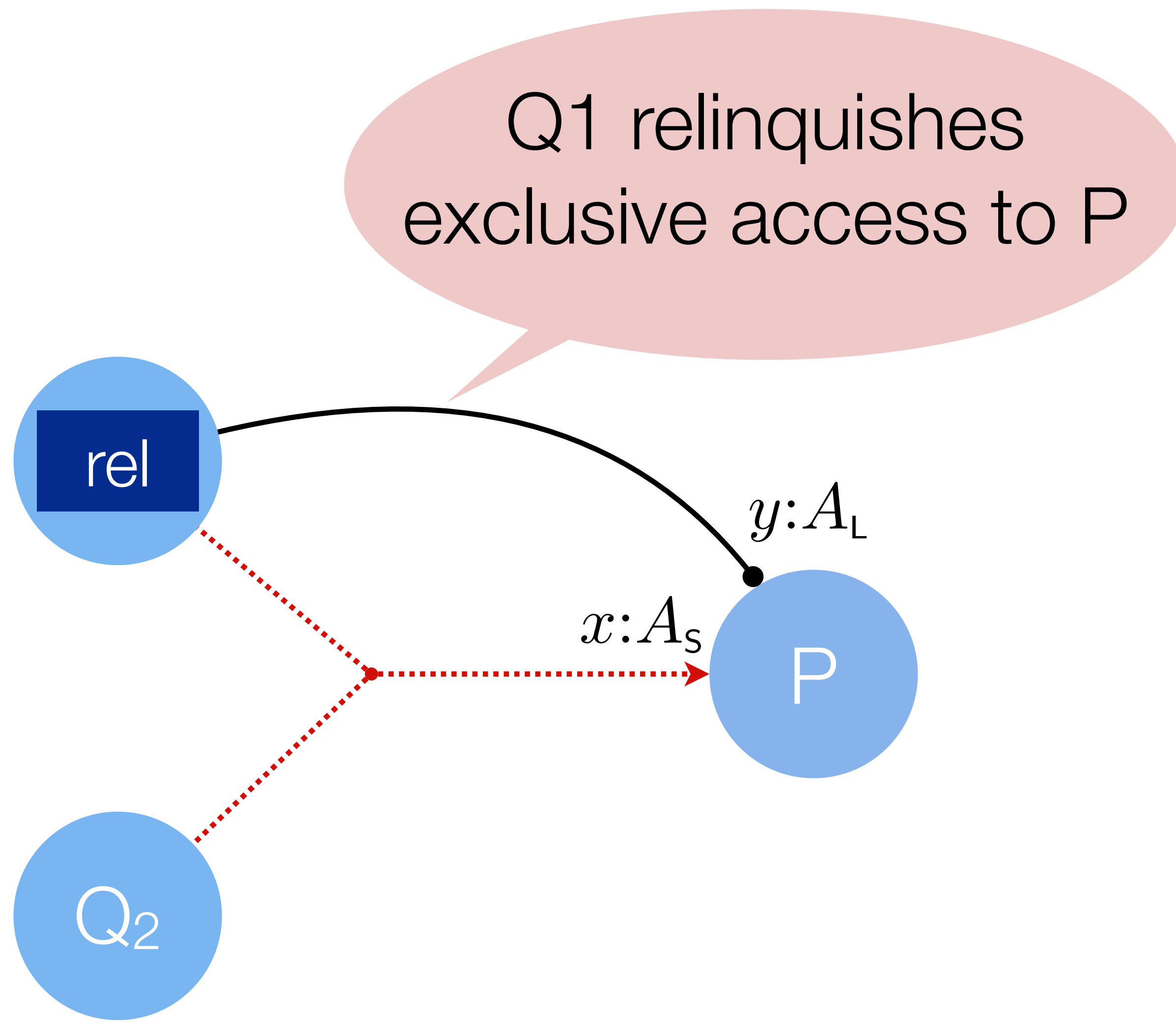
Legend: linear channel, linear process, shared channel, shared process

Key idea 1: acquire-release



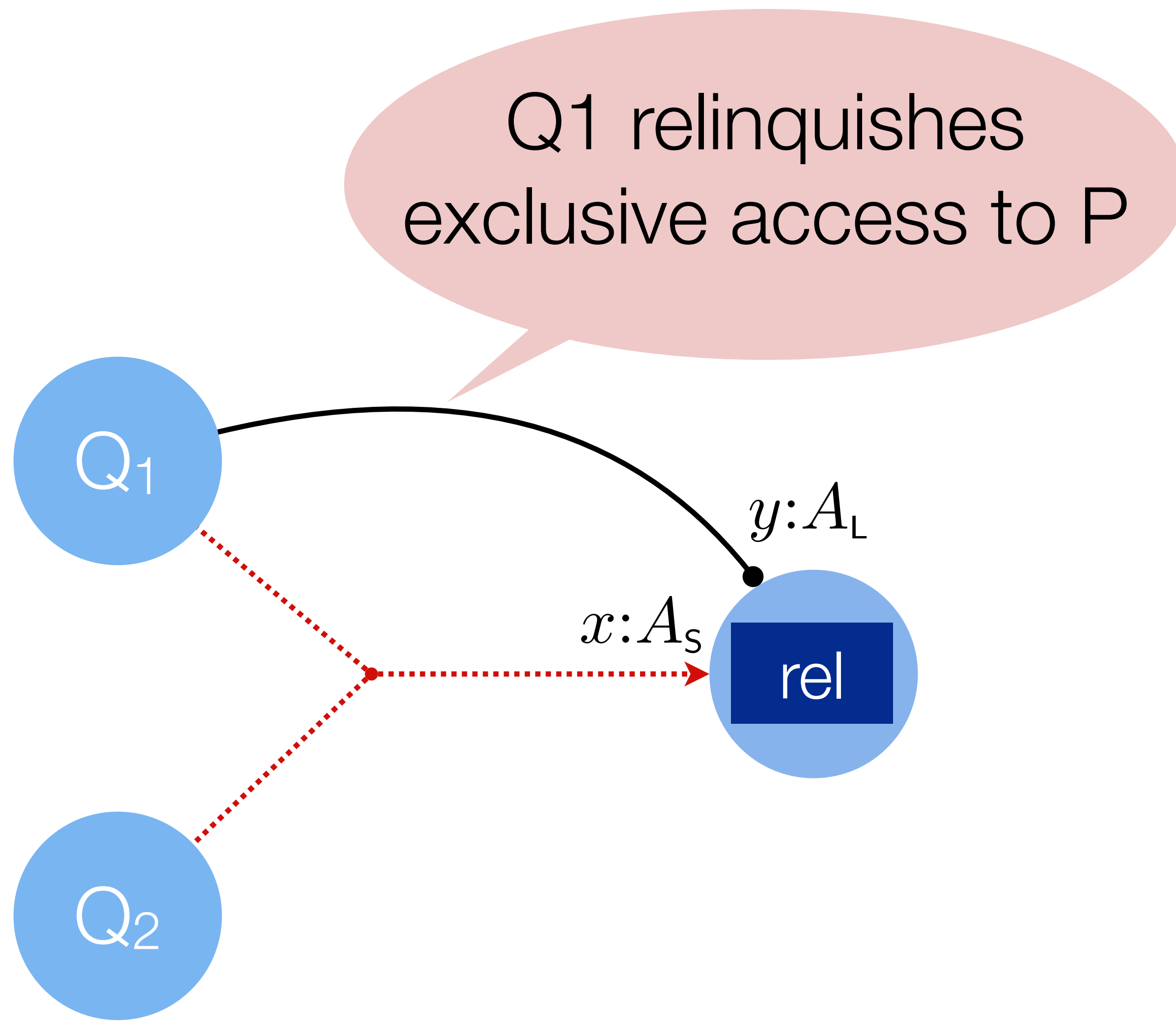


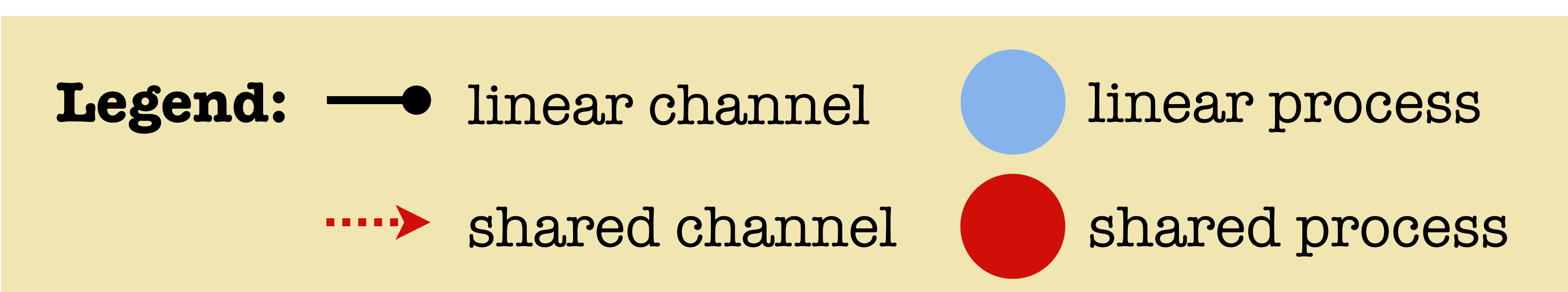
Key idea 1: acquire-release



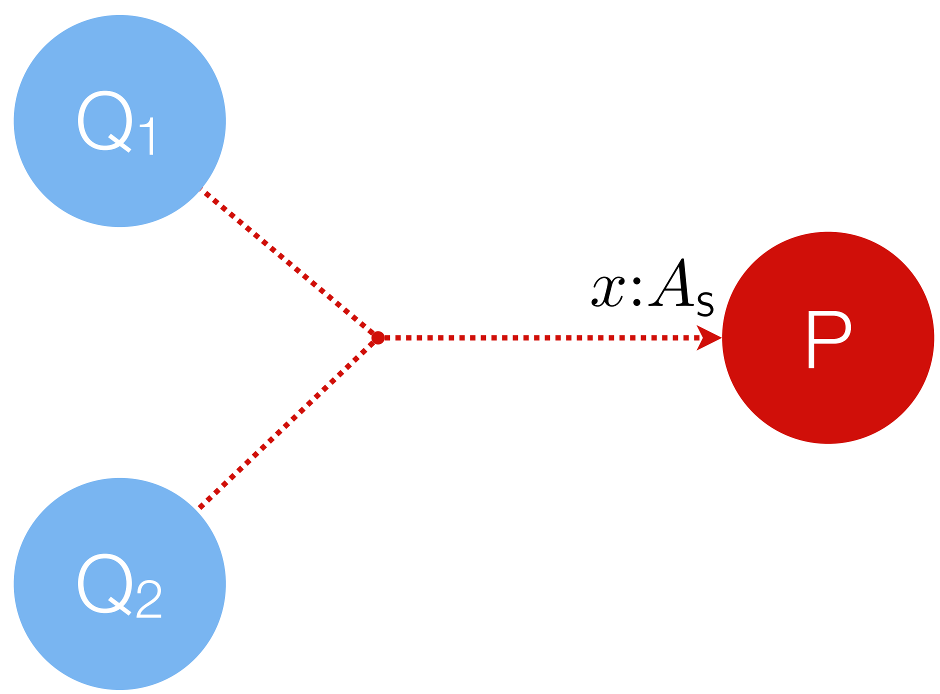
Legend: linear channel; shared channel; linear process; shared process

Key idea 1: acquire-release



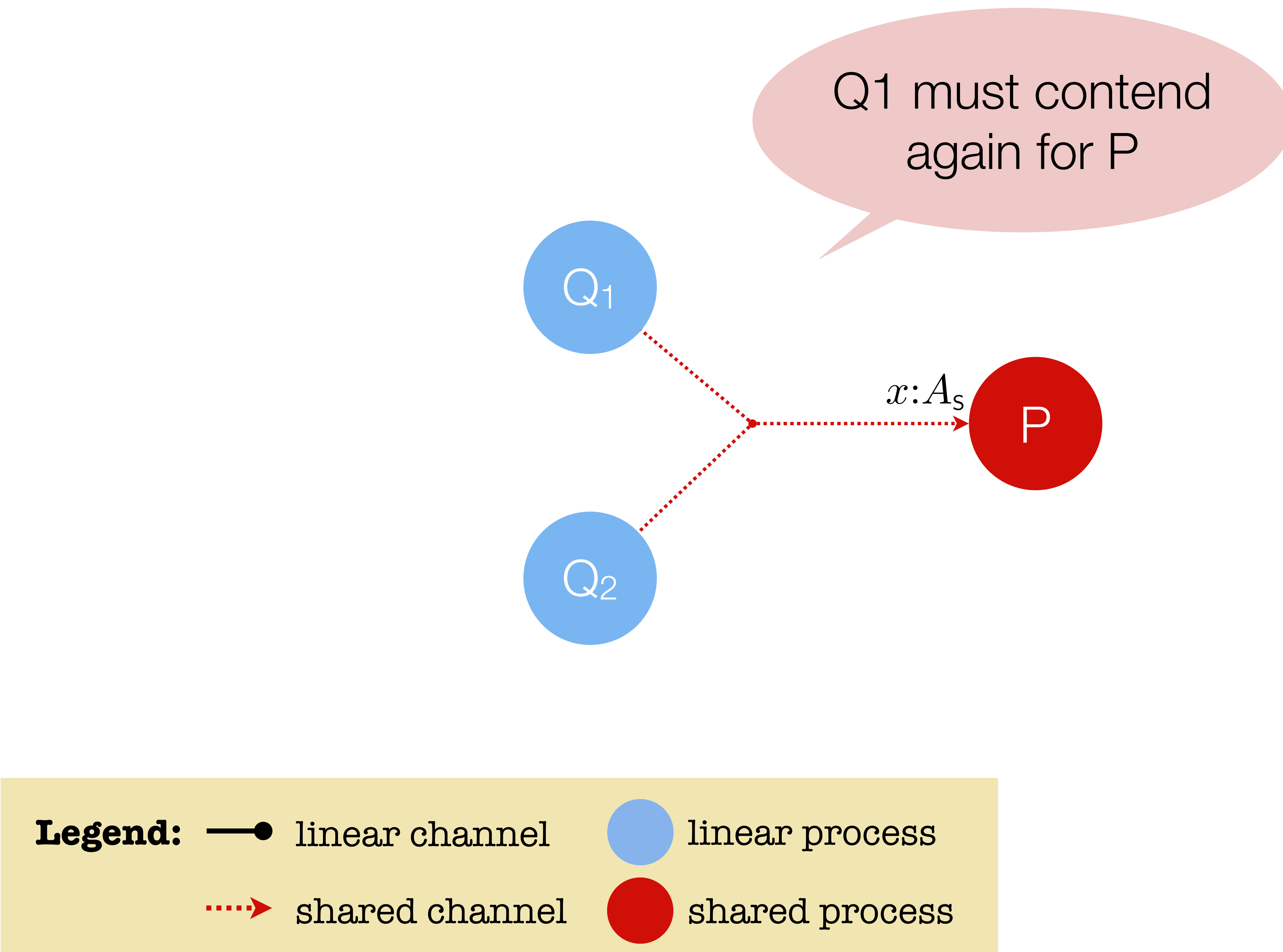


Key idea 1: acquire-release

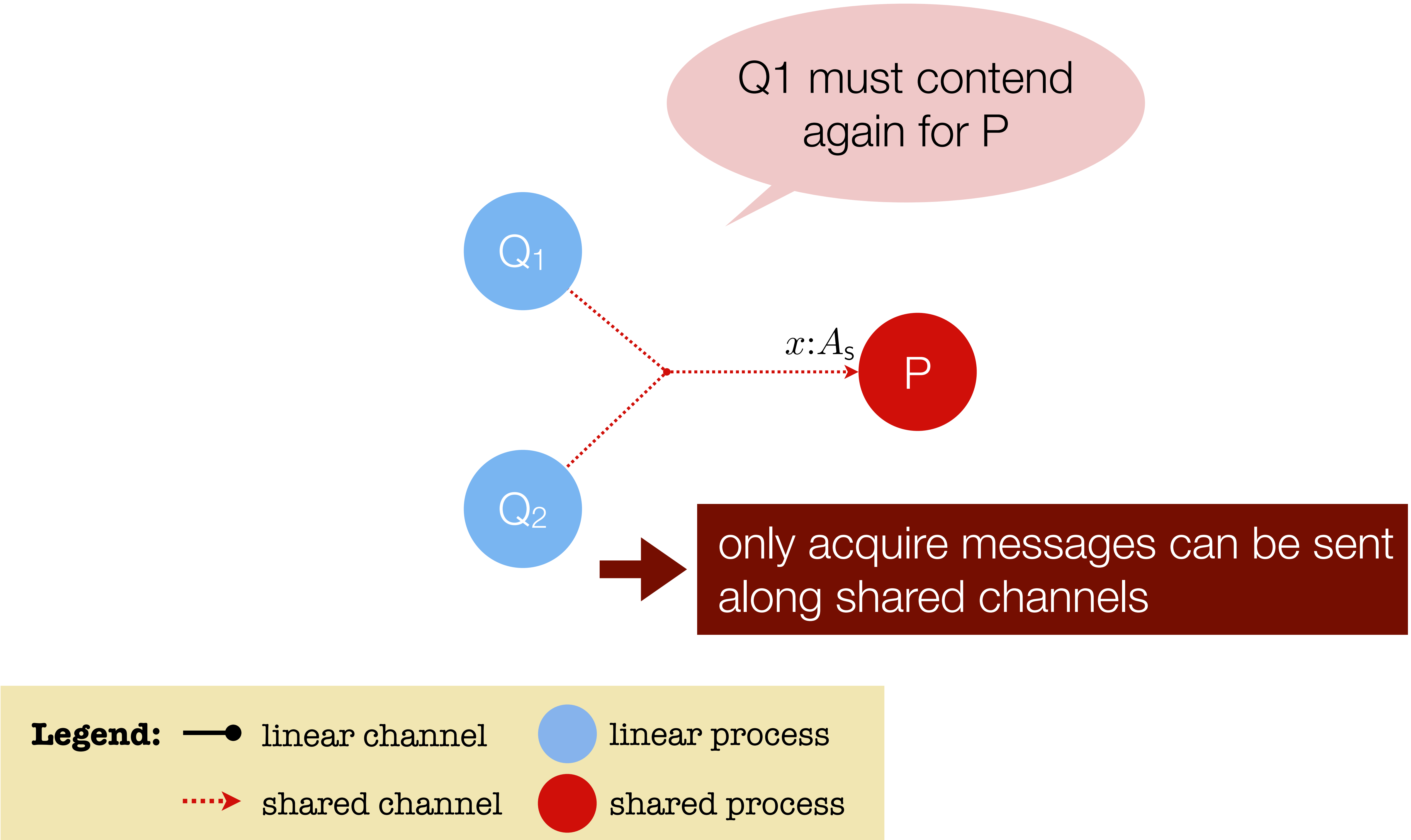


Legend: linear channel | linear process | shared channel | shared process

Key idea 1: acquire-release



Key idea 1: acquire-release



Key idea 2: manifest acquire-release in types

Key idea 2: manifest acquire-release in types

Observation:

Key idea 2: manifest acquire-release in types

Observation:

→ **processes are at one of two modes: either linear or shared**

Key idea 2: manifest acquire-release in types

Observation:

→ **processes are at one of two modes: either linear or shared**

Adjoint stratification of session types:

Key idea 2: manifest acquire-release in types

Observation:

- processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

- stratify session types into a linear and shared layer, s.t. S > L

Key idea 2: manifest acquire-release in types

Observation:

→ processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

→ stratify session types into a linear and shared layer, s.t. S > L

$$A_{\mathsf{S}} \triangleq$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}}$$

Key idea 2: manifest acquire-release in types

Observation:

➡ processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

➡ stratify session types into a linear and shared layer, s.t. S > L

weakening
contraction

+ ↑ -

$$A_{\mathsf{S}} \triangleq$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}}$$

Key idea 2: manifest acquire-release in types

Observation:

- processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

- stratify session types into a linear and shared layer, s.t. S > L
- connect layers with modalities going back and forth

weakening
contraction

+ (↑) −

$$A_{\mathsf{S}} \triangleq$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}}$$

Key idea 2: manifest acquire-release in types

Observation:

- processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

- stratify session types into a linear and shared layer, s.t. S > L
- connect layers with modalities going back and forth

weakening
contraction

+

-

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}}$$

Key idea 2: manifest acquire-release in types

Observation:

- processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

- stratify session types into a linear and shared layer, s.t. S > L
- connect layers with modalities going back and forth

weakening
contraction

+ ↑ -

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S$$

- support of sending shared channels along linear channels

Key idea 2: manifest acquire-release in types

Observation:

- processes are at one of two modes: either linear or shared

Adjoint stratification of session types:

- stratify session types into a linear and shared layer, s.t. S > L
- connect layers with modalities going back and forth

weakening
contraction

+

-

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid \& \{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}}$$

- support of sending shared channels along linear channels

Example: shared queue

What should be the type of a shared queue?

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+ ↑ -

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid$$
$$\&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+ (weakening, contraction) ↑ −

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid \\ \&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

$$\begin{aligned} \text{queue } A_S = \quad & \&\{\text{enq} : \Pi x{:}A_S.\quad \text{queue } A_S, \\ & \quad \text{deq} : \oplus\{\text{none} : \quad \text{queue } A_S,\ \text{some} : \exists x{:}A_S.\quad \text{queue } A_S\}\} \end{aligned}$$

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+ ↑ −

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid$$
$$\&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

$$\mathsf{queue}\ A_S = \&\{\mathsf{enq} : \Pi x{:}A_S.\ \mathsf{queue}\ A_S,$$
$$\mathsf{deq} : \oplus\{\mathsf{none} : \mathsf{queue}\ A_S,\ \mathsf{some} : \exists x{:}A_S.\ \mathsf{queue}\ A_S\}\}$$

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+ (↑) −

$$A_{\mathsf{S}} \triangleq {\uparrow}_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid$$
$$\&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid {\downarrow}_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}}$$

$$\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \mathsf{queue}\ A_{\mathsf{S}},$$
$$\mathsf{deq} : \oplus\{\mathsf{none} : \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

\+ (↑) −

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$\begin{aligned} A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq\ & \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\ B_{\mathsf{L}} \mid \\ & \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\ B_{\mathsf{L}} \end{aligned}$$

$$\begin{aligned} \mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{ & \mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}, \\ & \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\} \end{aligned}$$

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+ ↑ -

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$\begin{aligned} A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq\ & \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid \\ & \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \end{aligned}$$

$$\begin{aligned} \mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{ & \mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\, \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}, \\ & \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\, \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\} \end{aligned}$$

Takeaway:

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+

-

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\ B_L \mid$$
$$\&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S \mid \Pi x{:}A_S.\ B_L$$

$$\text{queue } A_S = \uparrow_L^S \&\{\text{enq} : \Pi x{:}A_S.\ \downarrow_L^S \text{queue } A_S,$$
$$\text{deq} : \oplus\{\text{none} : \downarrow_L^S \text{queue } A_S,\ \text{some} : \exists x{:}A_S.\ \downarrow_L^S \text{queue } A_S\}\}$$

Takeaway:

up-shift is an acquire

Example: shared queue

What should be the type of a shared queue?

weakening
contraction

+ (up) / - (down)

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}}$$

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Takeaway:

- up-shift is an acquire
- down-shift is a release

Key idea 3: equi-synchronizing

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ \mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}^{\mathsf{S}}_{\mathsf{L}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}^{\mathsf{S}}_{\mathsf{L}}\, \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\, \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\, \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\, \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}^{\mathsf{S}}_{\mathsf{L}}\ \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}^{\mathsf{S}}_{\mathsf{L}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

process is released
back to same type previously
acquired

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}{\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}{\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \uparrow_{\mathsf{L}}^{\mathsf{S}} \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}}\uparrow_{\mathsf{L}}^{\mathsf{S}} \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \&\{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \uparrow_{\mathsf{L}}^{\mathsf{S}} \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_\mathsf{S} = \uparrow_\mathsf{L}^\mathsf{S} \&\{\mathsf{enq} : \Pi x{:}A_\mathsf{S}.\ \downarrow_\mathsf{L}^\mathsf{S} \mathsf{queue}\ A_\mathsf{S},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_\mathsf{L}^\mathsf{S} \mathsf{queue}\ A_\mathsf{S},\ \mathsf{some} : \exists x{:}A_\mathsf{S}.\ \downarrow_\mathsf{L}^\mathsf{S} \mathsf{queue}\ A_\mathsf{S}\}\}$$

$$\mathsf{queue}\ A_\mathsf{S} = \uparrow_\mathsf{L}^\mathsf{S} \&\{\mathsf{enq} : \Pi x{:}A_\mathsf{S}.\ \downarrow_\mathsf{L}^\mathsf{S} \mathsf{queue}\ A_\mathsf{S},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_\mathsf{L}^\mathsf{S}\uparrow_\mathsf{L}^\mathsf{S} \mathbf{1},\ \mathsf{some} : \exists x{:}A_\mathsf{S}.\ \downarrow_\mathsf{L}^\mathsf{S} \mathsf{queue}\ A_\mathsf{S}\}\}$$

process is released back to different type

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}^{\mathsf{S}}_{\mathsf{L}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}^{\mathsf{S}}_{\mathsf{L}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}^{\mathsf{S}}_{\mathsf{L}}{\uparrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}^{\mathsf{S}}_{\mathsf{L}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

process is released back to different type

next client to acquire encounters protocol violation!

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

$$\begin{aligned}\mathsf{queue}\ A_{\mathsf{S}} = {\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \&\{&\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}},\\ &\mathsf{deq} : \oplus\{\mathsf{none} : {\downarrow}_{\mathsf{L}}^{\mathsf{S}}{\uparrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ {\downarrow}_{\mathsf{L}}^{\mathsf{S}}\ \mathsf{queue}\ A_{\mathsf{S}}\}\}\end{aligned}$$

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \& \{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

$$\mathsf{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \& \{\mathsf{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}},\ \mathsf{deq} : \oplus\{\mathsf{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \uparrow_{\mathsf{L}}^{\mathsf{S}} \mathbf{1},\ \mathsf{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \mathsf{queue}\ A_{\mathsf{S}}\}\}$$

→ equi-synchronizing: type wellformedness condition guaranteeing that any release is back to type at which previously acquired

Key idea 3: equi-synchronizing

Is mutual exclusion enough for restoring preservation?

$$\text{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \& \{\text{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \text{queue}\ A_{\mathsf{S}},\ \text{deq} : \oplus\{\text{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \text{queue}\ A_{\mathsf{S}},\ \text{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \text{queue}\ A_{\mathsf{S}}\}\}$$

$$\text{queue}\ A_{\mathsf{S}} = \uparrow_{\mathsf{L}}^{\mathsf{S}} \& \{\text{enq} : \Pi x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \text{queue}\ A_{\mathsf{S}},\ \text{deq} : \oplus\{\text{none} : \downarrow_{\mathsf{L}}^{\mathsf{S}} \uparrow_{\mathsf{L}}^{\mathsf{S}} \mathbf{1},\ \text{some} : \exists x{:}A_{\mathsf{S}}.\ \downarrow_{\mathsf{L}}^{\mathsf{S}} \text{queue}\ A_{\mathsf{S}}\}\}$$

→ equi-synchronizing: type wellformedness condition guaranteeing that any release is back to type at which previously acquired

→ acquire-release and equi-synchronizing guarantee preservation

Typing judgments

Typing judgments

weakening
contraction

+

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid$$
$$\&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}}$$

-

Typing judgments

weakening
contraction

+

$$
\begin{aligned}
A_{\mathsf{S}} &\triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}} \\
A_{\mathsf{L}}, B_{\mathsf{L}} &\triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid \\
&\quad\ \&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}}
\end{aligned}
$$

-

$\Gamma \vdash_{\Sigma} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})$	shared process P, providing session of type A_{S} along x_{S}, using channels in Γ
$\Gamma;\ \Delta \vdash_{\Sigma} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})$	linear process P, providing session of type A_{L} along x_{L}, using channels in Γ and Δ
Γ	shared (structural) context
Δ	linear context

Typing judgments

weakening
contraction

\+ ↑ -

$$A_S \triangleq {\uparrow}_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid \\ \&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid {\downarrow}_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

$\Gamma \vdash_\Sigma P :: (x_S : A_S)$	shared process P, providing session of type A_S along x_S, using channels in Γ
$\Gamma;\ \Delta \vdash_\Sigma P :: (x_L : A_L)$	linear process P, providing session of type A_L along x_L, using channels in Γ and Δ
Γ	shared (structural) context
Δ	linear context

Typing judgments

weakening
contraction

$+$ / $-$

$$A_{\mathsf{S}} \triangleq \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}}$$

$$A_{\mathsf{L}}, B_{\mathsf{L}} \triangleq \oplus\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \otimes B_{\mathsf{L}} \mid \mathbf{1} \mid \exists x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}} \mid$$
$$\&\{\overline{l : A_{\mathsf{L}}}\} \mid A_{\mathsf{L}} \multimap B_{\mathsf{L}} \mid \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \mid \Pi x{:}A_{\mathsf{S}}.\, B_{\mathsf{L}}$$

$\Gamma \vdash_{\Sigma} P :: (x_{\mathsf{S}} : A_{\mathsf{S}})$	shared process P, providing session of type A_{S} along x_{S}, using channels in Γ
$\Gamma;\ \Delta \vdash_{\Sigma} P :: (x_{\mathsf{L}} : A_{\mathsf{L}})$	linear process P, providing session of type A_{L} along x_{L}, using channels in Γ and Δ
Γ	shared (structural) context
Δ	linear context

Typing judgments

weakening
contraction

+ ↑ −

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid$$
$$\&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

$\Gamma \vdash_\Sigma P :: (x_S : A_S)$	shared process P, providing session of type A_S along x_S, using channels in Γ
$\Gamma;\ \Delta \vdash_\Sigma P :: (x_L : A_L)$	linear process P, providing session of type A_L along x_L, using channels in Γ and Δ
Γ	shared (structural) context
Δ	linear context

Typing judgments

weakening
contraction

+

-

$$A_S \triangleq \uparrow_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid$$
$$\&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid \downarrow_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

$\Gamma \vdash_\Sigma P :: (x_S : A_S)$	shared process P, providing session of type A_S along x_S, using channels in Γ
$\Gamma;\ \Delta \vdash_\Sigma P :: (x_L : A_L)$	linear process P, providing session of type A_L along x_L, using channels in Γ and Δ
Γ	shared (structural) context
Δ	linear context

Typing judgments

weakening
contraction

+ ↑ −

$$A_S \triangleq {\uparrow}_L^S A_L$$

$$A_L, B_L \triangleq \oplus\{\overline{l : A_L}\} \mid A_L \otimes B_L \mid \mathbf{1} \mid \exists x{:}A_S.\, B_L \mid$$
$$\&\{\overline{l : A_L}\} \mid A_L \multimap B_L \mid {\downarrow}_L^S A_S \mid \Pi x{:}A_S.\, B_L$$

$\Gamma \vdash_\Sigma P :: (x_S : A_S)$	shared process P, providing session of type A_S along x_S, using channels in Γ
$\Gamma;\ \Delta \vdash_\Sigma P :: (x_L : A_L)$	linear process P, providing session of type A_L along x_L, using channels in Γ and Δ
Γ	shared (structural) context
Δ	linear context

Acquire

Acquire

$$\frac{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta, x_{\mathsf{L}} : A_{\mathsf{L}} \vdash_{\Sigma} Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ; Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\mathrm{T}\text{-}\uparrow_{\mathsf{L}}^{\mathsf{S}}\mathrm{L})$$

Acquire

$$\frac{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}} ;\ \Delta, x_{\mathsf{L}} : A_{\mathsf{L}} \vdash_{\Sigma} Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}} ;\ \Delta \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ; Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\mathrm{T}\text{-}\uparrow_{\mathsf{L}}^{\mathsf{S}}\mathrm{L})$$

Acquire

$$\frac{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta, x_{\mathsf{L}} : A_{\mathsf{L}} \vdash_{\Sigma} Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ; Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})} \ (\text{T-}\uparrow_{\mathsf{L}}^{\mathsf{S}}\text{L})$$

$$\frac{\Gamma;\ \cdot \vdash_{\Sigma} P_{x_{\mathsf{L}}} :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Gamma \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{accept}\ x_{\mathsf{S}}\ ; P_{x_{\mathsf{L}}} :: (x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})} \ (\text{T-}\uparrow_{\mathsf{L}}^{\mathsf{S}}\text{R})$$

Acquire

$$\frac{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta, x_{\mathsf{L}} : A_{\mathsf{L}} \vdash_{\Sigma} Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ; Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\mathrm{T}\text{-}\uparrow_{\mathsf{L}}^{\mathsf{S}}\mathrm{L})$$

$$\frac{\Gamma;\ \cdot \vdash_{\Sigma} P_{x_{\mathsf{L}}} :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Gamma \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{accept}\ x_{\mathsf{S}}\ ; P_{x_{\mathsf{L}}} :: (x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})}\ (\mathrm{T}\text{-}\uparrow_{\mathsf{L}}^{\mathsf{S}}\mathrm{R})$$

Acquire

$$\frac{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta, x_{\mathsf{L}} : A_{\mathsf{L}} \vdash_{\Sigma} Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma, x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}};\ \Delta \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ x_{\mathsf{S}}\ ; Q_{x_{\mathsf{L}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})} \ (\mathrm{T}\text{-}{\uparrow_{\mathsf{L}}^{\mathsf{S}}}\mathrm{L})$$

$$\frac{\Gamma;\ \cdot \vdash_{\Sigma} P_{x_{\mathsf{L}}} :: (x_{\mathsf{L}} : A_{\mathsf{L}})}{\Gamma \vdash_{\Sigma} x_{\mathsf{L}} \leftarrow \mathsf{accept}\ x_{\mathsf{S}}\ ; P_{x_{\mathsf{L}}} :: (x_{\mathsf{S}} : \uparrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{L}})} \ (\mathrm{T}\text{-}{\uparrow_{\mathsf{L}}^{\mathsf{S}}}\mathrm{R})$$

$$(\mathrm{D}\text{-}{\uparrow_{\mathsf{L}}^{\mathsf{S}}}) \quad \mathsf{proc}(a_{\mathsf{S}}, x_{\mathsf{L}} \leftarrow \mathsf{accept}\ a_{\mathsf{S}}\ ; P_{x_{\mathsf{L}}}), \mathsf{proc}(c_{\mathsf{L}}, x_{\mathsf{L}} \leftarrow \mathsf{acquire}\ a_{\mathsf{S}}\ ; Q_{x_{\mathsf{L}}})$$
$$\longrightarrow \mathsf{unvail}(a_{\mathsf{S}}), \mathsf{proc}(a_{\mathsf{L}}, [a_{\mathsf{L}}/x_{\mathsf{L}}]\, P_{x_{\mathsf{L}}}), \mathsf{proc}(c_{\mathsf{L}}, [a_{\mathsf{L}}/x_{\mathsf{L}}]\, Q_{x_{\mathsf{L}}})$$

Release

Release

$$\frac{\Gamma, x_{\mathsf{S}} : A_{\mathsf{S}};\ \Delta \vdash_{\Sigma} Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma;\ \Delta, x_{\mathsf{L}} : \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{release}\ x_{\mathsf{L}}\ ; Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\mathrm{T}\text{-}\downarrow_{\mathsf{L}}^{\mathsf{S}}\mathrm{L})$$

Release

$$\frac{\Gamma, x_{\mathsf{S}} : A_{\mathsf{S}};\ \Delta \vdash_{\Sigma} Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma;\ \Delta, x_{\mathsf{L}} :\downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{release}\ x_{\mathsf{L}}\ ; Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\mathrm{T}\text{-}{\downarrow_{\mathsf{L}}^{\mathsf{S}}}\mathrm{L})$$

Release

$$\frac{\Gamma, x_{\mathsf{S}} : A_{\mathsf{S}};\ \Delta \vdash_{\Sigma} Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma;\ \Delta, x_{\mathsf{L}} : \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}} \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{release}\ x_{\mathsf{L}}\ ; Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})} \ (\text{T-}\downarrow_{\mathsf{L}}^{\mathsf{S}}\text{L})$$

$$\frac{\Gamma \vdash_{\Sigma} P_{x_{\mathsf{S}}} :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\Gamma;\ \cdot \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{detach}\ x_{\mathsf{L}}\ ; P_{x_{\mathsf{S}}} :: (x_{\mathsf{L}} : \downarrow_{\mathsf{L}}^{\mathsf{S}} A_{\mathsf{S}})} \ (\text{T-}\downarrow_{\mathsf{L}}^{\mathsf{S}}\text{R})$$

Release

$$\frac{\Gamma, x_{\mathsf{S}} : A_{\mathsf{S}};\ \Delta \vdash_{\Sigma} Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma;\ \Delta, x_{\mathsf{L}} : \downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}} \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{release}\ x_{\mathsf{L}}\ ; Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\text{T-}\downarrow^{\mathsf{S}}_{\mathsf{L}}\text{L})$$

$$\frac{\Gamma \vdash_{\Sigma} P_{x_{\mathsf{S}}} :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\Gamma;\ \cdot \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{detach}\ x_{\mathsf{L}}\ ; P_{x_{\mathsf{S}}} :: (x_{\mathsf{L}} : \downarrow^{\mathsf{S}}_{\mathsf{L}} A_{\mathsf{S}})}\ (\text{T-}\downarrow^{\mathsf{S}}_{\mathsf{L}}\text{R})$$

Release

$$\frac{\Gamma, x_{\mathsf{S}} : A_{\mathsf{S}};\ \Delta \vdash_{\Sigma} Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}{\Gamma;\ \Delta, x_{\mathsf{L}} : {\downarrow_{\mathsf{L}}^{\mathsf{S}}} A_{\mathsf{S}} \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{release}\ x_{\mathsf{L}}\ ; Q_{x_{\mathsf{S}}} :: (z_{\mathsf{L}} : C_{\mathsf{L}})}\ (\text{T-}{\downarrow_{\mathsf{L}}^{\mathsf{S}}}\text{L})$$

$$\frac{\Gamma \vdash_{\Sigma} P_{x_{\mathsf{S}}} :: (x_{\mathsf{S}} : A_{\mathsf{S}})}{\Gamma;\ \cdot \vdash_{\Sigma} x_{\mathsf{S}} \leftarrow \mathsf{detach}\ x_{\mathsf{L}}\ ; P_{x_{\mathsf{S}}} :: (x_{\mathsf{L}} : {\downarrow_{\mathsf{L}}^{\mathsf{S}}} A_{\mathsf{S}})}\ (\text{T-}{\downarrow_{\mathsf{L}}^{\mathsf{S}}}\text{R})$$

$$\begin{array}{ll} (\text{D-}{\downarrow_{\mathsf{L}}^{\mathsf{S}}}) & \mathsf{proc}(a_{\mathsf{L}}, x_{\mathsf{S}} \leftarrow \mathsf{detach}\ a_{\mathsf{L}}\ ; P_{x_{\mathsf{S}}}), \mathsf{proc}(c_{\mathsf{L}}, x_{\mathsf{S}} \leftarrow \mathsf{release}\ a_{\mathsf{L}}\ ; Q_{x_{\mathsf{S}}}), \\ & \mathsf{unvail}(a_{\mathsf{S}}) \\ & \longrightarrow\ \mathsf{proc}(a_{\mathsf{S}}, [a_{\mathsf{S}}/x_{\mathsf{S}}]\, P_{x_{\mathsf{S}}}), \mathsf{proc}(c_{\mathsf{L}}, [a_{\mathsf{S}}/x_{\mathsf{S}}]\, Q_{x_{\mathsf{S}}}) \end{array}$$

Let's implement a shared queue in SILL$_S$

Taking stock

Taking stock

We have a session type system that allows shared and linear channels to coexist and guarantees:

Taking stock

- We have a session type system that allows shared and linear channels to coexist and guarantees:
  - data-race-freedom (low-level and high-level)

Taking stock

- We have a session type system that allows shared and linear channels to coexist and guarantees:
  - data-race-freedom (low-level and high-level)
  - protocol adherence

Taking stock

- We have a session type system that allows shared and linear channels to coexist and guarantees:
  - data-race-freedom (low-level and high-level)
  - protocol adherence
- What about deadlock-freedom?

Taking stock

- We have a session type system that allows shared and linear channels to coexist and guarantees:
  - data-race-freedom (low-level and high-level)
  - protocol adherence
- What about deadlock-freedom?
  - unfortunately we have lost deadlock-freedom

Taking stock

- We have a session type system that allows shared and linear channels to coexist and guarantees:
  - data-race-freedom (low-level and high-level)
  - protocol adherence
- What about deadlock-freedom?
  - unfortunately we have lost deadlock-freedom

next time!