An Aspect-Oriented Paradigm for Component-based Systems and its Application to RV

Antoine El-Hokayem[1], Yliès Falcone[1], Mohamad Jaber[2]

Univ. Grenoble Alpes, INRIA, LIG, Grenoble, France
first.last@imag.fr

American University of Beirut, Beirut, Lebanon
mj54@aub.edu.lb

COST Meeting

Outline

Component-Based Design

- Separation between computation and coordination:
  1. Design of individual components: components are white-boxes;
  2. Coordination between various components: components are black-boxes.

Component-Based Design

- Separation between computation and coordination:
  1. Design of individual components: components are white-boxes;
  2. Coordination between various components: components are black-boxes.
- Advantages:
  1. Re-usability of components
  2. Ability to make internal changes to components without affecting system
  3. Compositional construction, testing and verification

Aspect-Oriented Programming

- [Czarnecki et al., 1997] A concern is defined as:

  "*domain used as a decomposition criterion for a system or another domain with that concern*"

- Example concerns: logging, persistence, policies and reconfiguration

Aspect-Oriented Programming

- [Czarnecki et al., 1997] A concern is defined as:

  "*domain used as a decomposition criterion for a system or another domain with that concern*"

- Example concerns: logging, persistence, policies and reconfiguration

- Concerns can be:

  1. Dependant on/independent from other concerns;
  2. Localized or spread out.

Aspect-Oriented Programming

- [Czarnecki et al., 1997] A concern is defined as:

  "*domain used as a decomposition criterion for a system or another domain with that concern*"

- Example concerns: logging, persistence, policies and reconfiguration
- Concerns can be:
  1. Dependant on/independent from other concerns;
  2. Localized or spread out.
- Concerns that are spread out or overlap are called crosscutting.

Crosscutting Concerns

- Crosscutting concerns lead to two typical problems:

1. Tangling: occurs when concerns overlap in one region of the system.
2. Scattering: occurs when a single concern is spread across multiple regions in the system.

Crosscutting Concerns

- Crosscutting concerns lead to two typical problems:
  1. Tangling: occurs when concerns overlap in one region of the system.
  2. Scattering: occurs when a single concern is spread across multiple regions in the system.
- Aspect-Oriented Programming seeks to modularize crosscutting concerns.

AOP Concepts

1. Where is the concern?
   - Joinpoint: location of the concern.
   - Pointcut: matching mechanism for the joinpoints.

AOP Concepts

1. Where is the concern?
   - Joinpoint: location of the concern.
   - Pointcut: matching mechanism for the joinpoints.
2. What to do at the point of execution?
   - Advice: code executed at the joinpoint.

AOP Concepts

1. Where is the concern?
   - Joinpoint: location of the concern.
   - Pointcut: matching mechanism for the joinpoints.
2. What to do at the point of execution?
   - Advice: code executed at the joinpoint.
3. How is it coordinated?
   - Weaving: the process that adds the advice to a joinpoint.

AOP Concepts

1. Where is the concern?
   - Joinpoint: location of the concern.
   - Pointcut: matching mechanism for the joinpoints.
2. What to do at the point of execution?
   - Advice: code executed at the joinpoint.
3. How is it coordinated?
   - Weaving: the process that adds the advice to a joinpoint.

Aspect

The module that wraps a set of pointcuts, their associated advices and extra context information.

Problem Definition

- [Duclos et al., 2002, Lieberherr et al., 2003] In CBS, crosscutting concerns arise at the level of components (building blocks) and architectures (communications).

Problem Definition

- [Duclos et al., 2002, Lieberherr et al., 2003] In CBS, crosscutting concerns arise at the level of components (building blocks) and architectures (communications).
- Our goal is to integrate the notion of crosscutting concerns in CBSs.
  1. Formalizing identification of concerns
  2. Formalizing a description of concerns
  3. Determining the rules that govern the integration of the concerns onto CBSs
  4. Examining the composition of concerns

Problem Definition

- [Duclos et al., 2002, Lieberherr et al., 2003] In CBS, crosscutting concerns arise at the level of components (building blocks) and architectures (communications).
- Our goal is to integrate the notion of crosscutting concerns in CBSs.
  1. Formalizing identification of concerns
  2. Formalizing a description of concerns
  3. Determining the rules that govern the integration of the concerns onto CBSs
  4. Examining the composition of concerns
- Motivation
  1. Reliability: correct-by-construction, verification, validation
  2. Maintenance: encapsulation of concerns
  3. Refinement: progressive building of complex systems

The BIP Framework

We target component-based systems (CBSs) expressed in the **BIP** framework

The Behavior Interaction Priority (BIP) framework is a component-based framework for modeling complex heterogeneous systems.

The BIP Framework

We target component-based systems (CBSs) expressed in the **BIP** framework

The Behavior Interaction Priority (BIP) framework is a component-based framework for modeling complex heterogeneous systems.

Layered Component Model

- Behavior - automata extended with data and communication ports
- Interactions - synchronization and data transfer between components
- Priorities - partial order on interactions

The BIP Framework

We target component-based systems (CBSs) expressed in the **BIP** framework

The Behavior Interaction Priority (BIP) framework is a component-based framework for modeling complex heterogeneous systems.

Layered Component Model

- Behavior - automata extended with data and communication ports
- Interactions - synchronization and data transfer between components
- Priorities - partial order on interactions

Advantages

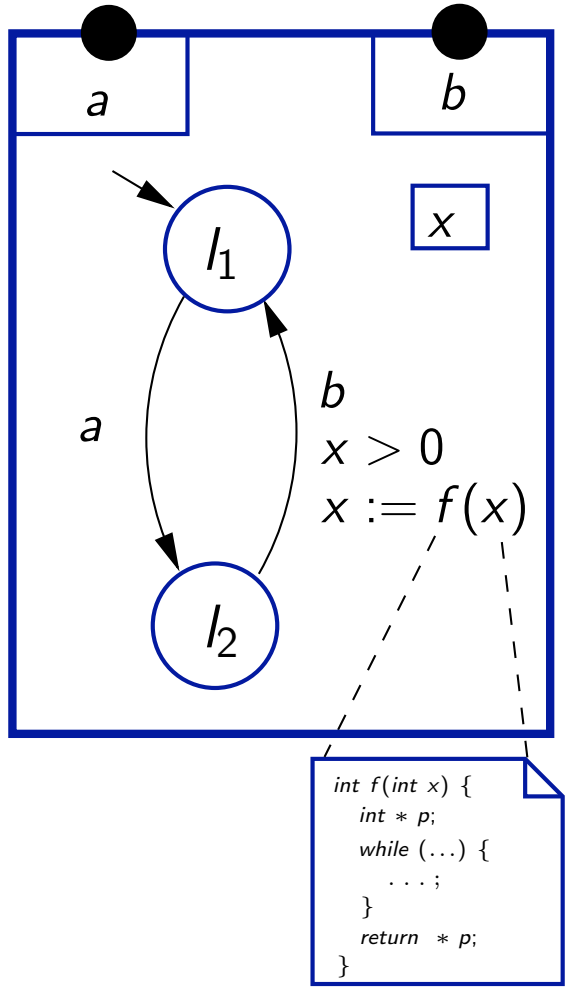
1. Specification of both atomic components and their coordination using well-defined operational semantics;
2. Efficient code generation.
3. Automatic verification and validation.

Behavior

Atomic Component

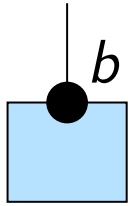
Labelled Transition System with data:

- *ports*, e.g, $\{a, b\}$
- *control locations*, e.g, $\{l_1, l_2\}$
- *variables*, e.g, $\{x\}$
- *transitions*
  - guards, e.g., $x > 0$
  - variable modifications (with external functions), e.g., $x := f(x)$



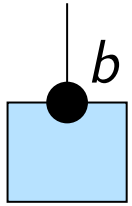
Interactions and Connectors: **Communication**

We consider the case of *synchron* (•) ports



Interactions and Connectors: **Communication**

We consider the case of *synchron* (•) ports

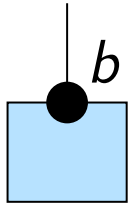


Interaction (= "a communication/collaboration")

An interaction is defined as a set of ports

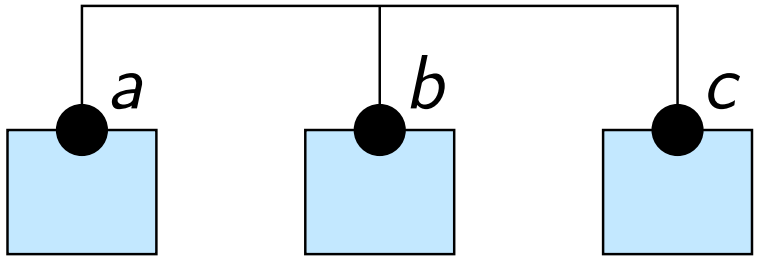
Interactions and Connectors: **Communication**

We consider the case of *synchron* (•) ports



Interaction (= "a communication/collaboration")

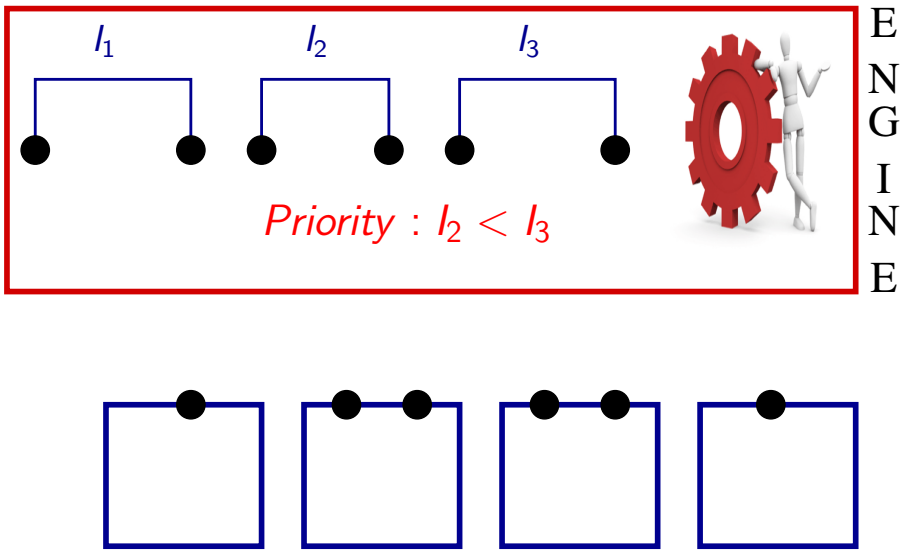
An interaction is defined as a set of ports



Composite component

Composite component

available atomic components + interactions + priority rules

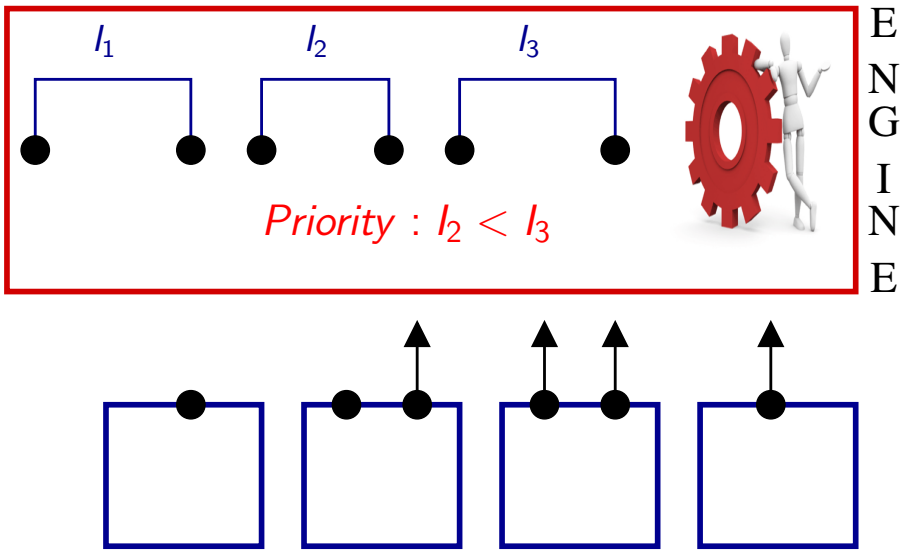


Engine Protocol

Composite component

Composite component

available atomic components + interactions + priority rules



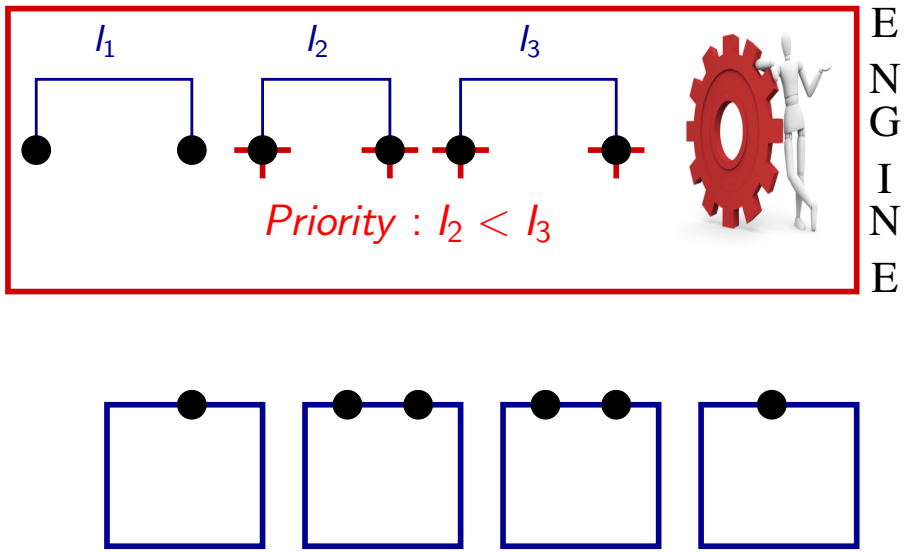
Engine Protocol

1. Atoms notify the engine of their enabled ports

Composite component

Composite component

available atomic components + interactions + priority rules



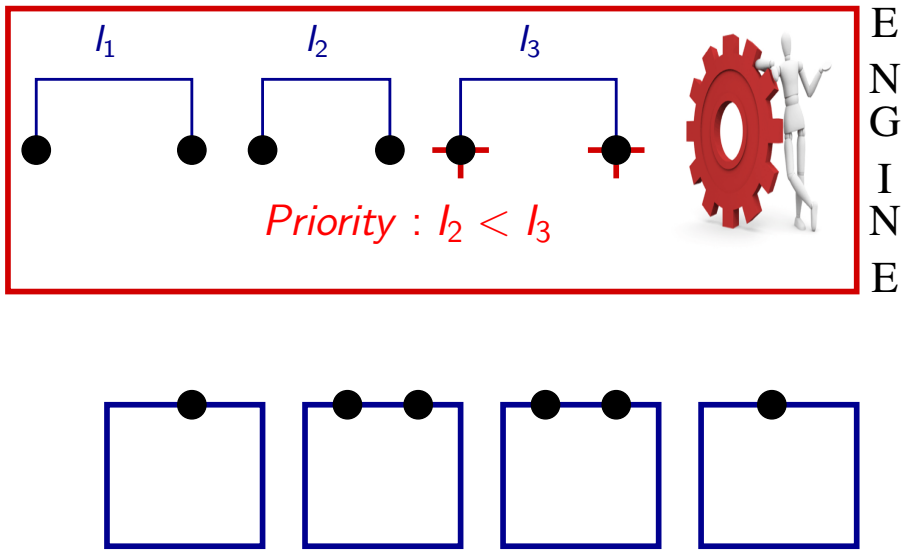
Engine Protocol

1. Atoms notify the engine of their enabled ports
2. The engine determines the enabled interactions

Composite component

Composite component

available atomic components + interactions + priority rules



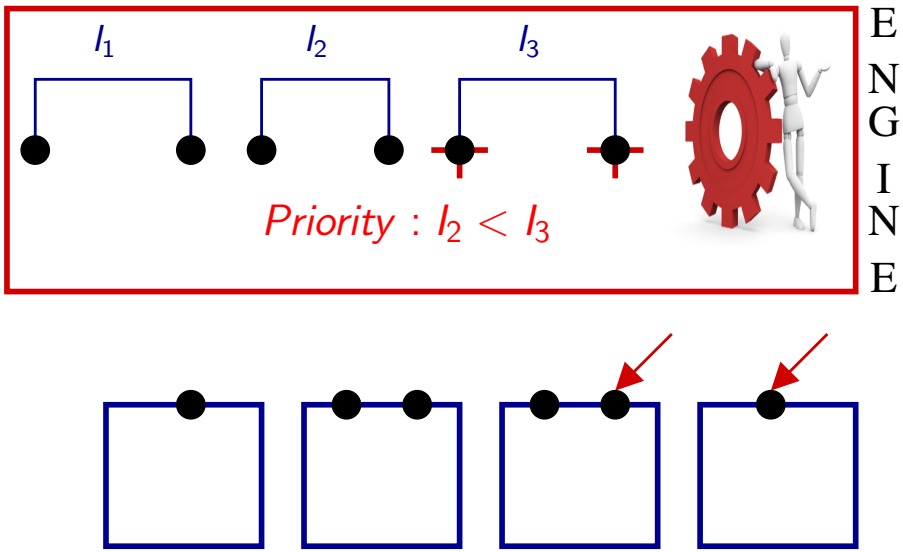
Engine Protocol

1. Atoms notify the engine of their enabled ports
2. The engine determines the enabled interactions
3. Chooses the highest priority interaction

Composite component

Composite component

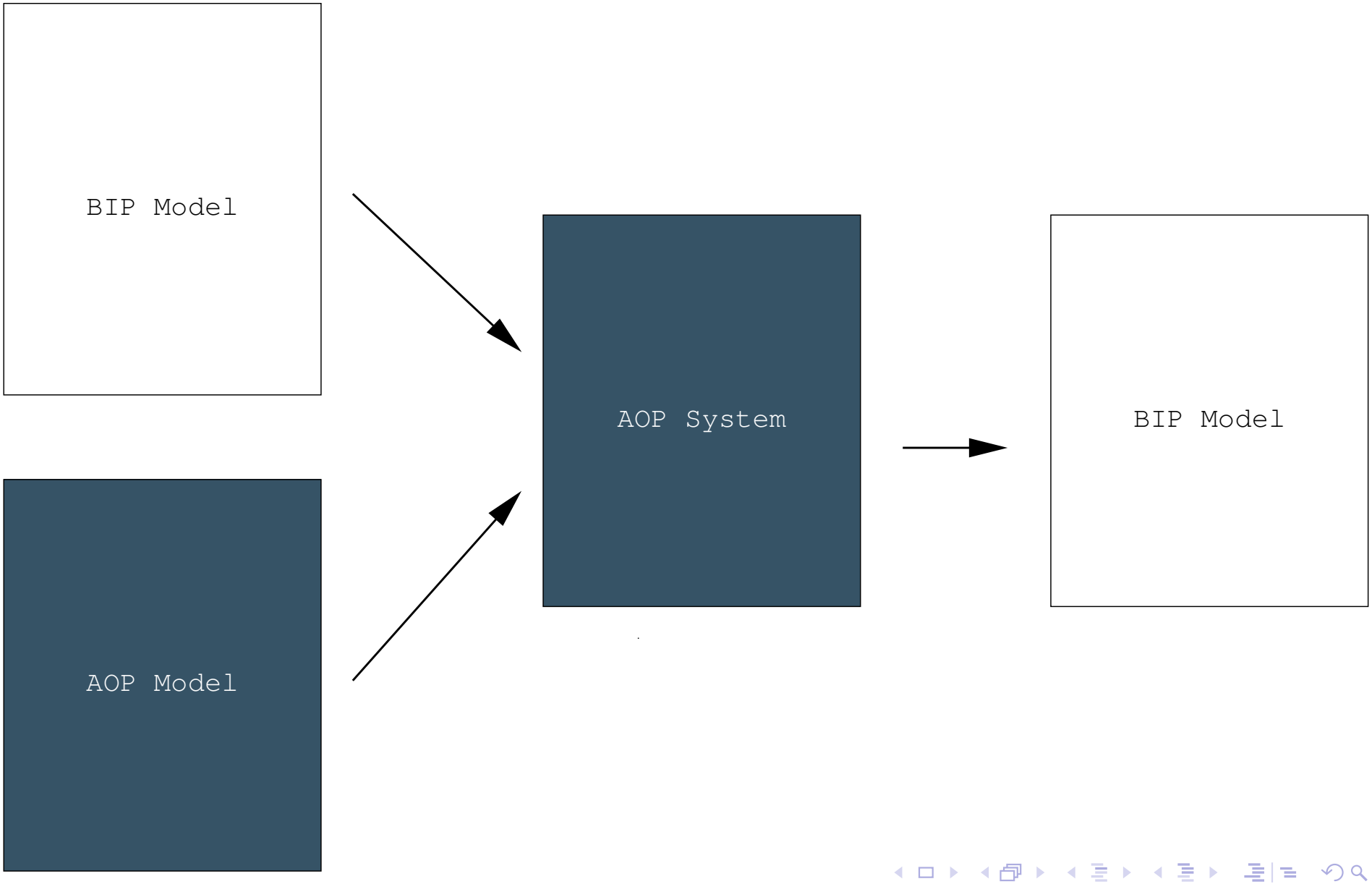
available atomic components + interactions + priority rules



Engine Protocol

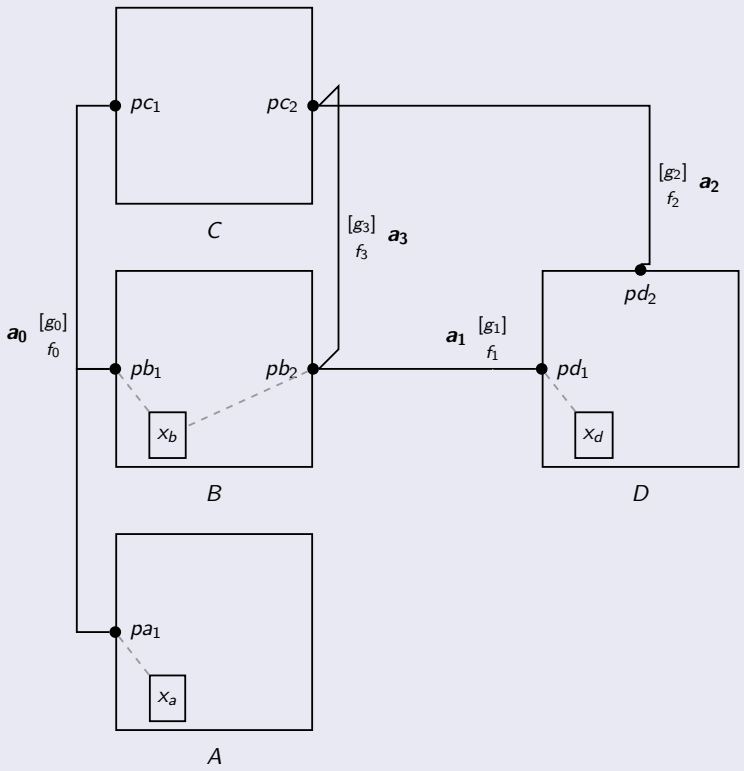
1. Atoms notify the engine of their enabled ports
2. The engine determines the enabled interactions
3. Chooses the highest priority interaction
4. Notifies the atoms

High-Level View

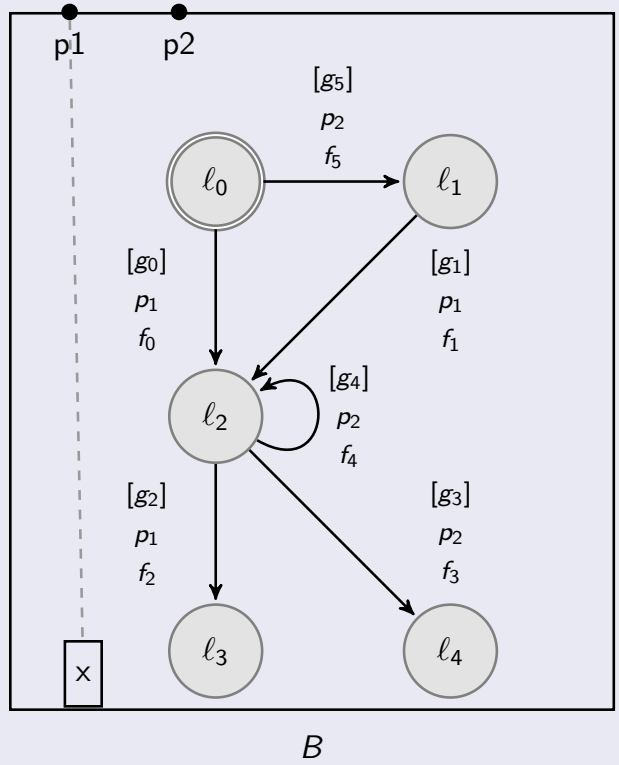


System Views

Global View



Local View



Execution Trace

Trace

- Alternating sequence of global state and interaction execution $(q_0, a_0, q_1, \ldots, q_{i-1}, a_{i-1}, q_i)$
- Global State: states of all atomic components
  - Location
  - Variable valuations

Execution Trace

Trace

- Alternating sequence of global state and interaction execution $(q_0, a_0, q_1, \ldots, q_{i-1}, a_{i-1}, q_i)$
- Global State: states of all atomic components
  - Location
  - Variable valuations

Global Event

- Starting global state $\rightarrow$ Interaction execution $\rightarrow$ Next global state
- $\langle q_k, a_k, q_{k+1} \rangle$ for $k \in [0, i-1]$
- $a_k = \langle p_k, f_k, g_k \rangle$

Execution Trace

Trace

- Alternating sequence of global state and interaction execution $(q_0, a_0, q_1, \ldots, q_{i-1}, a_{i-1}, q_i)$
- Global State: states of all atomic components
  - Location
  - Variable valuations

Global Event

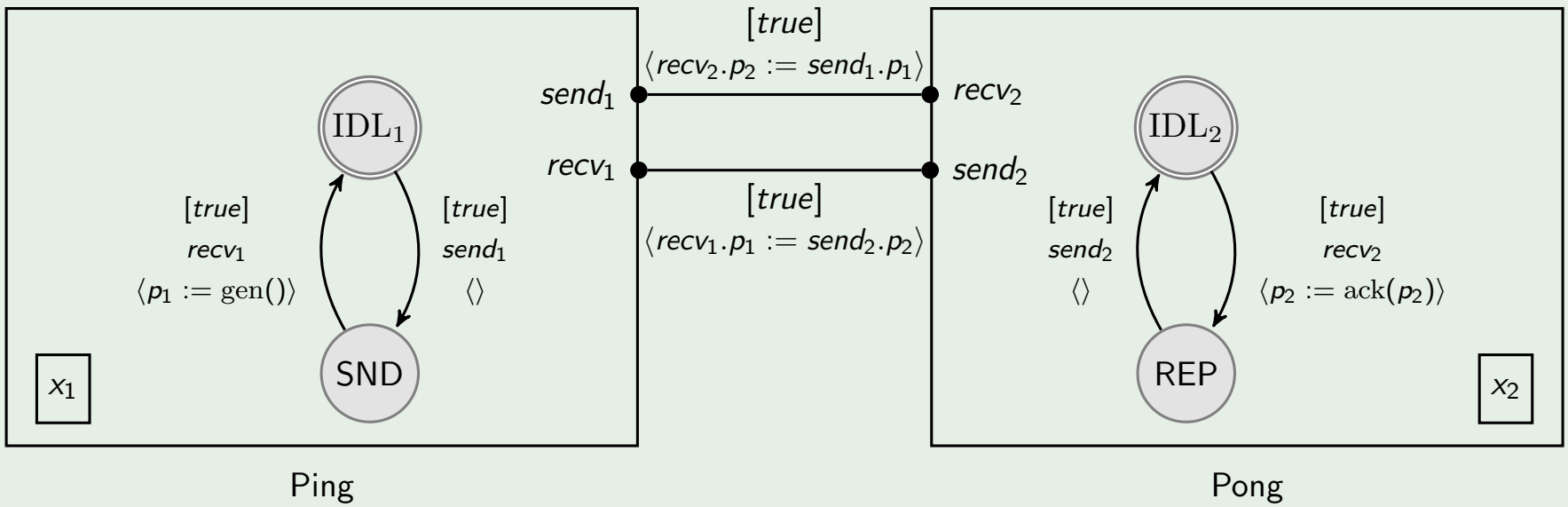
- Starting global state $\rightarrow$ Interaction execution $\rightarrow$ Next global state
- $\langle q_k, a_k, q_{k+1} \rangle$ for $k \in [0, i-1]$
- $a_k = \langle p_k, f_k, g_k \rangle$

Global Joinpoint

A global joinpoint is a global event.

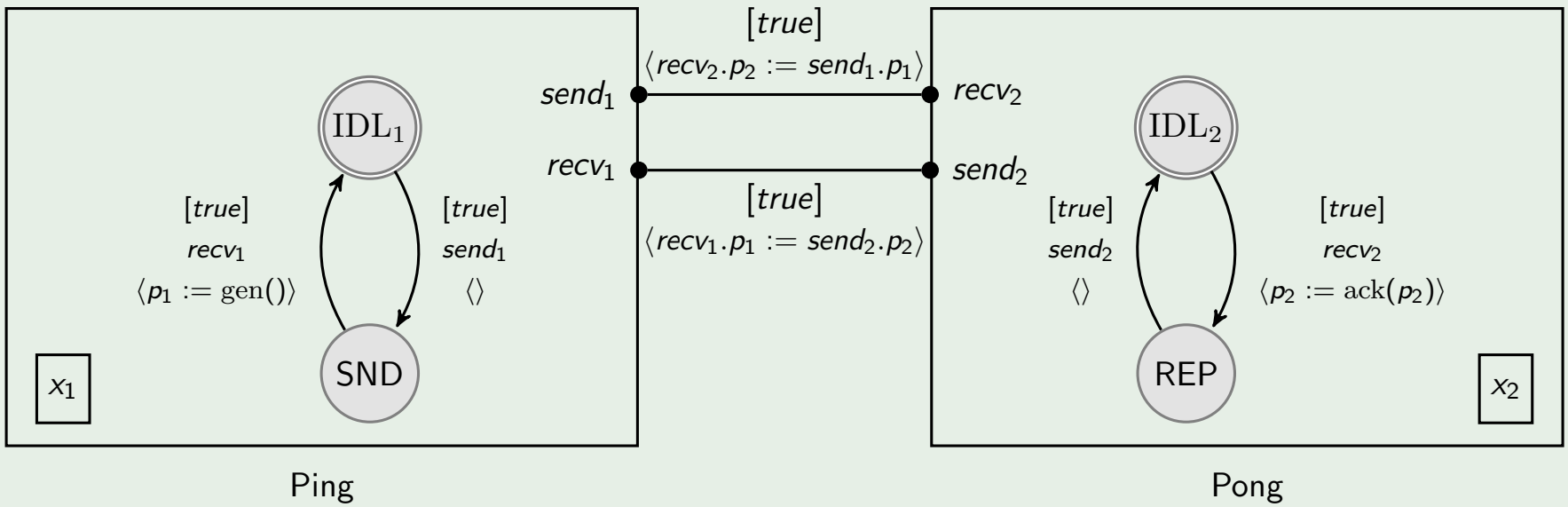
Example

Example (Global Event)



Example

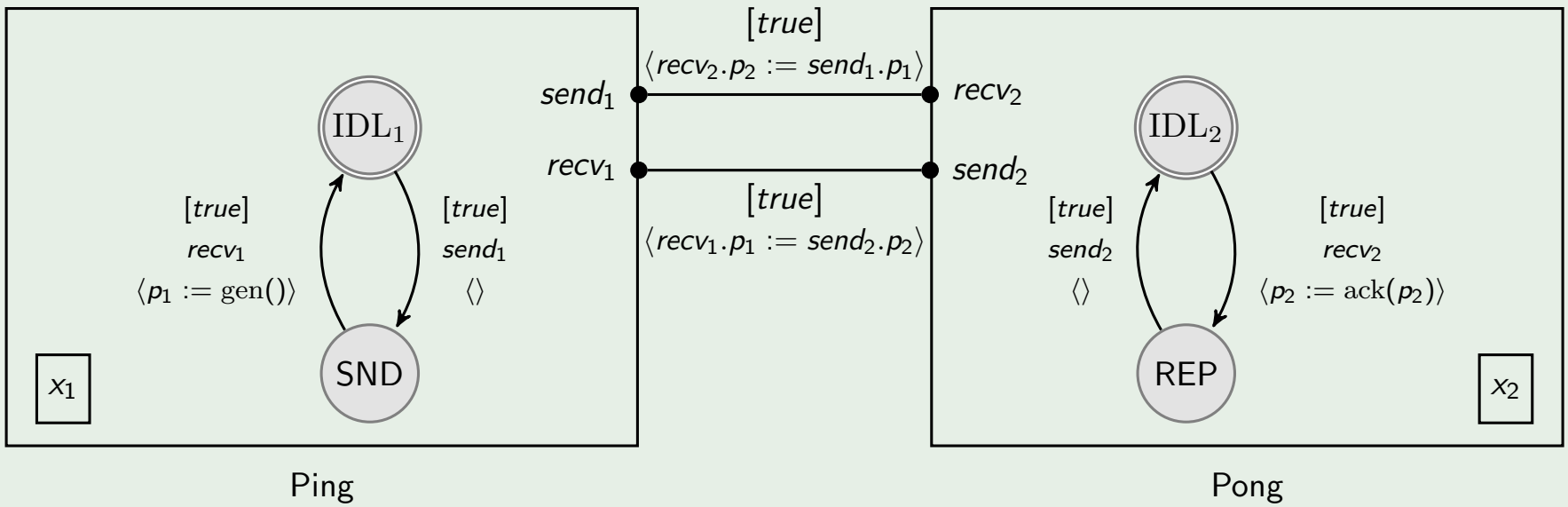
Example (Global Event)



$E_0 = \langle\langle\langle \mathtt{IDL}_1, \langle 1 \rangle\rangle, \langle \mathtt{IDL}_2, \langle 0 \rangle\rangle\rangle\rangle$,

Example

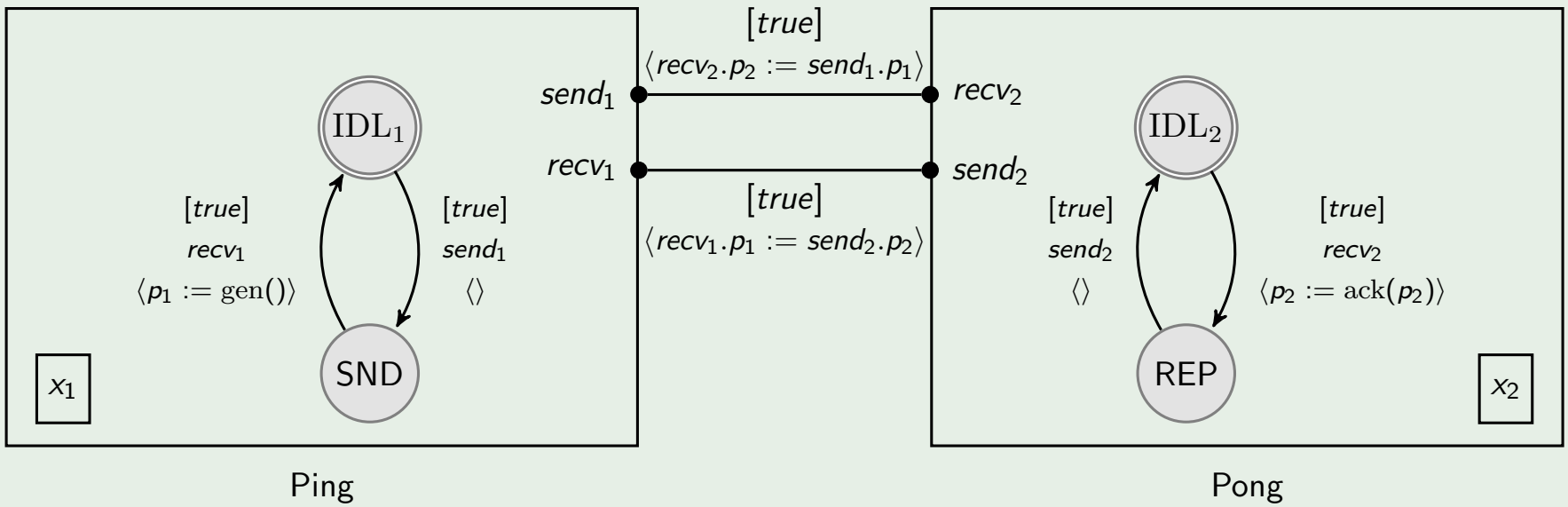
Example (Global Event)



$E_0 = \langle\langle\langle \mathtt{IDL}_1, \langle 1\rangle\rangle, \langle \mathtt{IDL}_2, \langle 0\rangle\rangle\rangle, \{\mathtt{send}_1, \mathtt{recv}_2\}\rangle$,

Example

Example (Global Event)



$E_0 = \langle\langle\langle \mathtt{IDL}_1, \langle 1\rangle\rangle, \langle \mathtt{IDL}_2, \langle 0\rangle\rangle\rangle, \{\mathtt{send}_1, \mathtt{recv}_2\}, \langle\langle \mathtt{SND}, \langle 1\rangle\rangle, \langle \mathtt{REP}, \langle 2\rangle\rangle\rangle\rangle$

Global Joinpoint

- We consider interaction execution when looking at coordination between atomic components.

Global Joinpoint

- We consider interaction execution when looking at coordination between atomic components.
- It can express the following:
  1. Synchronization between various atomic components;

Global Joinpoint

- We consider interaction execution when looking at coordination between atomic components.
- It can express the following:
  1. Synchronization between various atomic components;
  2. Data transfer between various atomic components.
     1. One ore more components are sending data.
     2. One or more components are receiving data.

Global Pointcut

Global Pointcut Expression

A global pointcut expression is defined as $gpc = \langle p, v_r, v_w \rangle$:

1. p : matches a subset of ports associated with the matching interaction.
2. v_r : variables read.
3. v_w: variables modified .

Global Pointcut

Global Pointcut Expression

A global pointcut expression is defined as $gpc = \langle p, v_{\mathrm{r}}, v_{\mathrm{w}} \rangle$:

1. p : matches a subset of ports associated with the matching interaction.
2. v_{r} : variables read.
3. v_{w}: variables modified .

Matching Global Joinpoints

$$\langle q, a, q' \rangle \vDash \langle p, v_r, v_w \rangle \text{ iff } p \subseteq a.ports \wedge v_r \subseteq \text{readvar}(a.func) \wedge v_w \subseteq \text{writevar}(a.func)$$

Global Advice

- The global advice consists of:
  1. A function f_b to execute before the interaction's computation.
  2. A function f_a to execute after the interaction's computation.

Global Advice

- The global advice consists of:
  1. A function f_b to execute before the interaction's computation.
  2. A function f_a to execute after the interaction's computation.
- The functions f_b, f_a are only defined over:
  1. Extra inter-type variables V.
  2. The variables of the context: the variables of the ports in the pointcut expression.

Global Weaving: Matching

- We use interactions as the base unit of instrumentation

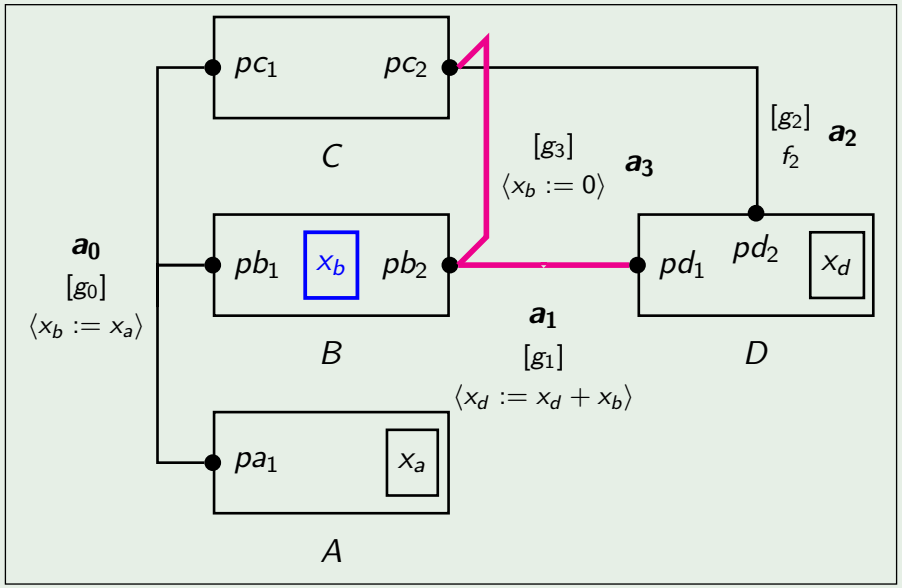
Global Weaving: Matching

- We use interactions as the base unit of instrumentation
- The syntax of the interactions is sufficient to capture the joinpoints

Global Weaving: Matching

- We use interactions as the base unit of instrumentation
- The syntax of the interactions is sufficient to capture the joinpoints

Example (Selecting Interactions)

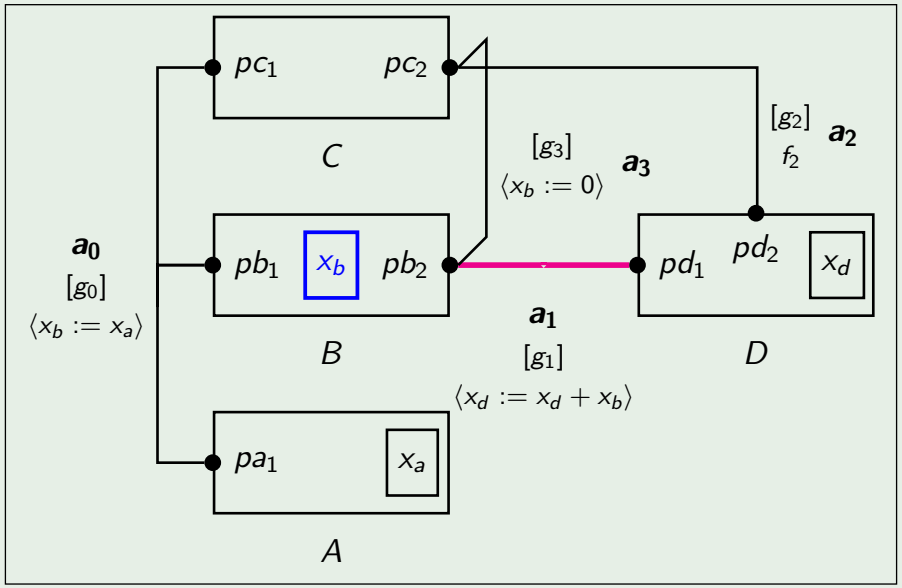


$$\langle \{pb_2\}, \emptyset, \emptyset \rangle$$

Global Weaving: Matching

- We use interactions as the base unit of instrumentation
- The syntax of the interactions is sufficient to capture the joinpoints

Example (Selecting Interactions)

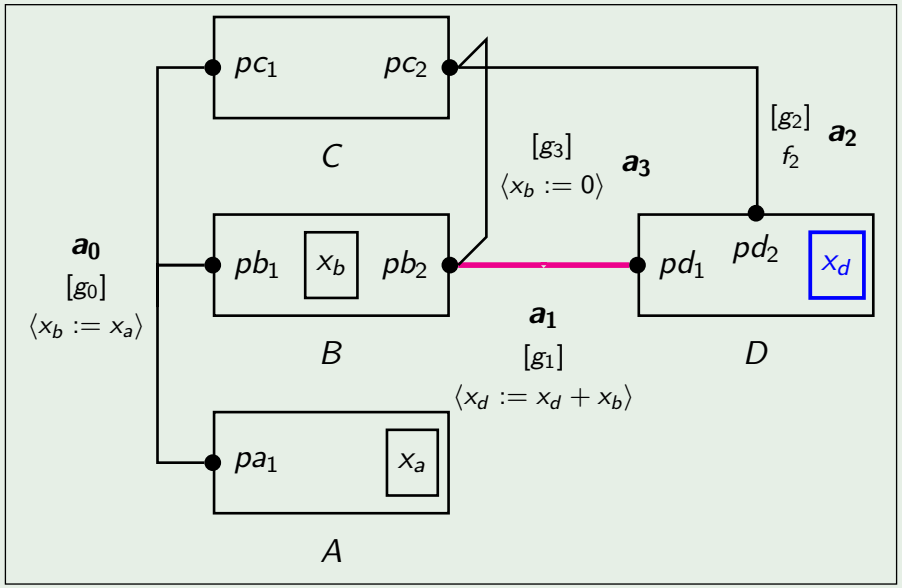


$$\langle \{pb_2\}, \{x_b\}, \emptyset \rangle$$

Global Weaving: Matching

- We use interactions as the base unit of instrumentation
- The syntax of the interactions is sufficient to capture the joinpoints

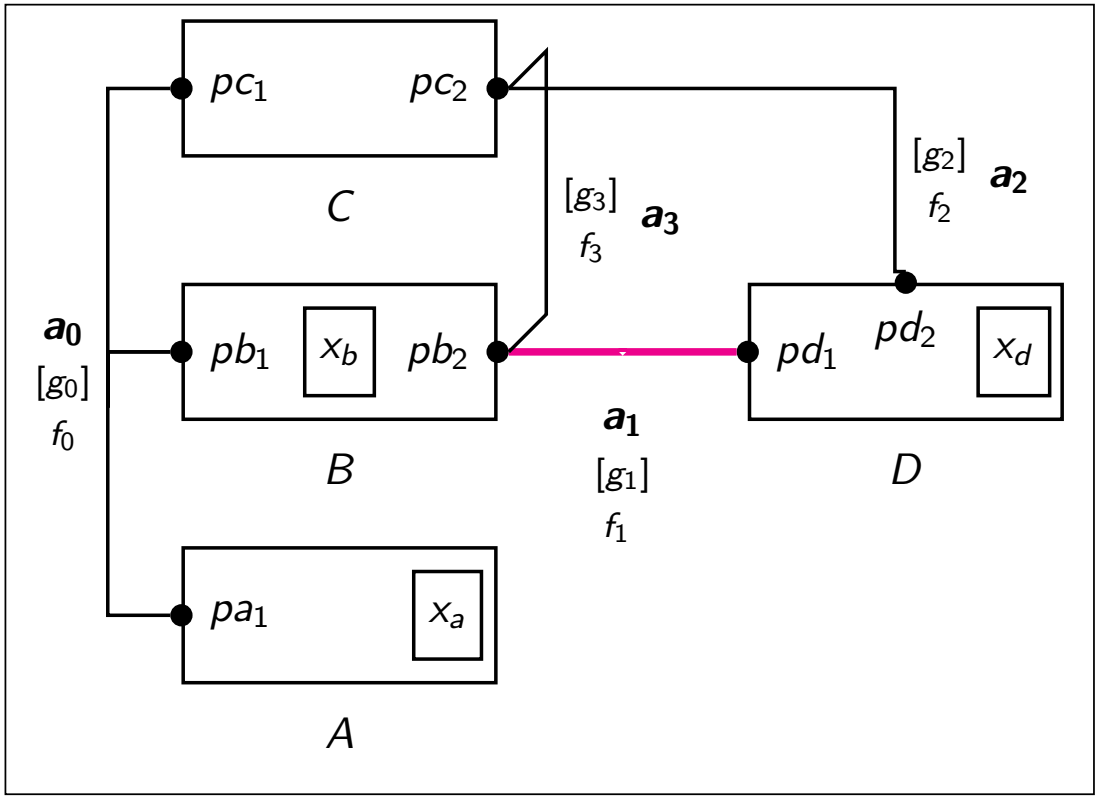
Example (Selecting Interactions)



$$\langle \{pd_1\}, \{x_d\}, \{x_d\} \rangle$$

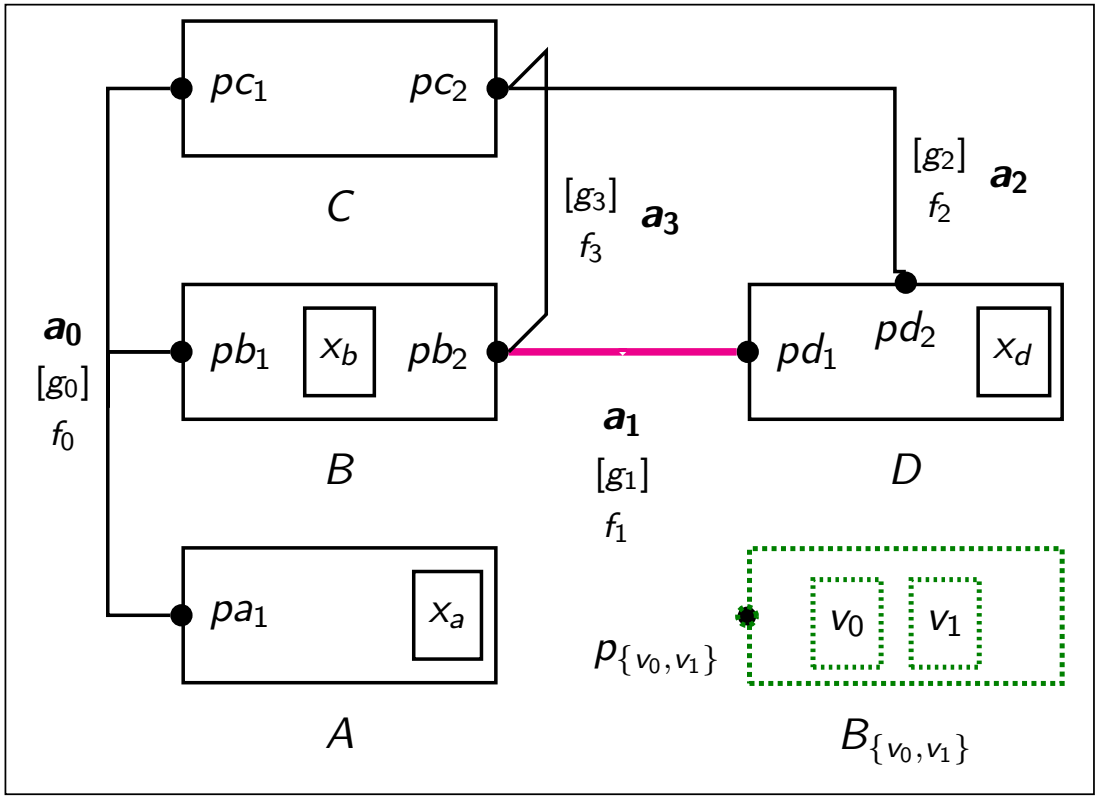
Global Weaving: Instrumentation

1. Select the match (set of interactions)



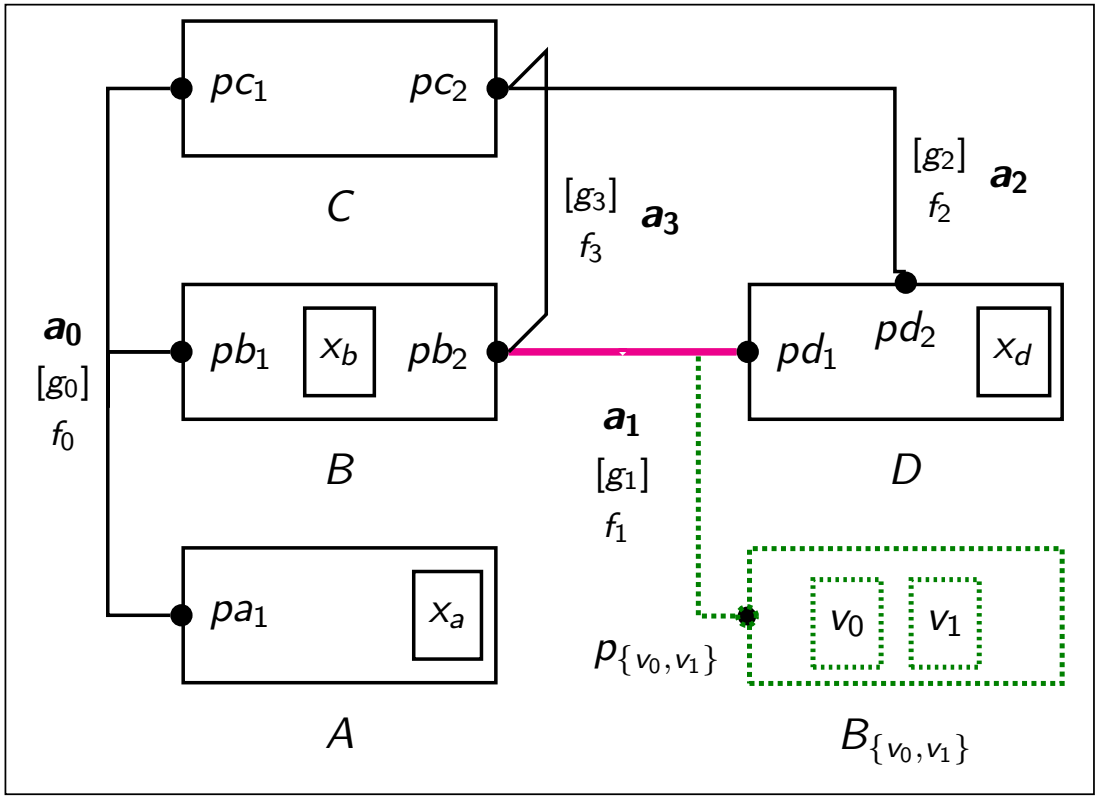
Global Weaving: Instrumentation

1. Select the match (set of interactions)
2. Add the inter-type component



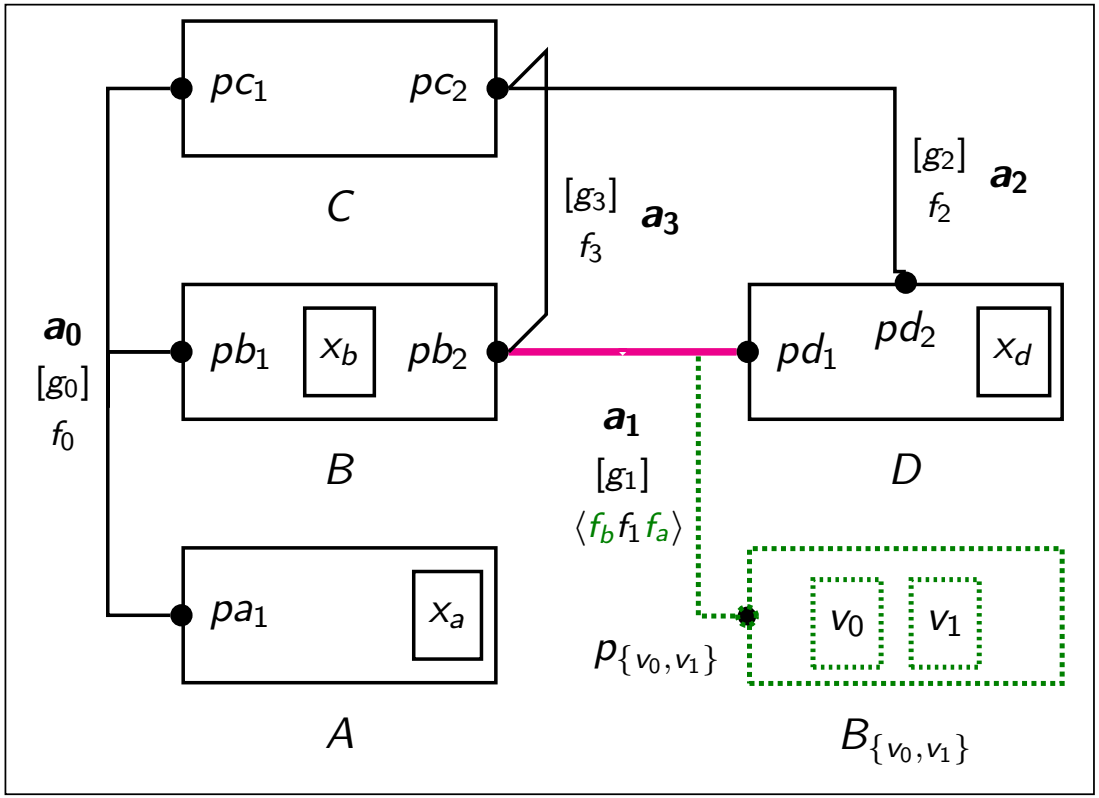
Global Weaving: Instrumentation

1. Select the match (set of interactions)
2. Add the inter-type component
3. Extend each interaction with the inter-type port



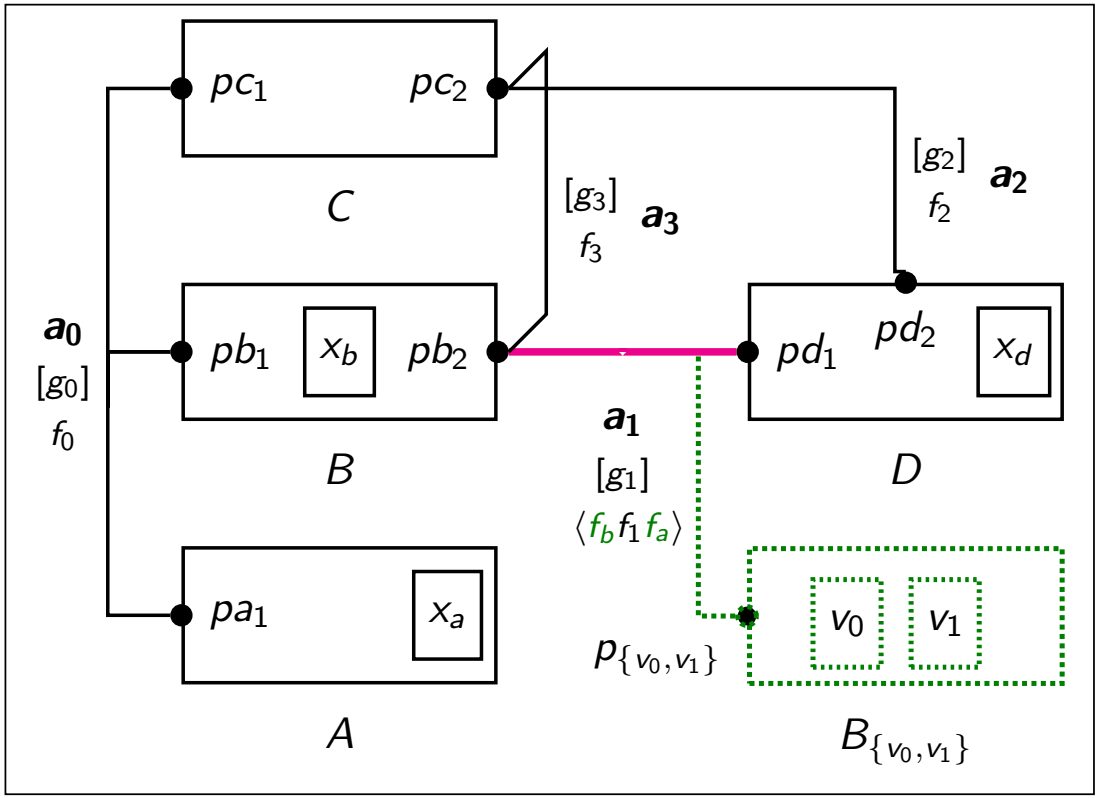
Global Weaving: Instrumentation

1. Select the match (set of interactions)
2. Add the inter-type component
3. Extend each interaction with the inter-type port
4. Inject the advice's update functions



Global Weaving: Instrumentation

1. Select the match (set of interactions)
2. Add the inter-type component
3. Extend each interaction with the inter-type port
4. Inject the advice's update functions
5. Priority is unchanged



Global Aspect and Correctness

Global Aspect

- A global aspect is defined on a composite component it contains:
  1. Inter-type variables
  2. A global pointcut expression
  3. A global advice

Global Aspect and Correctness

Global Aspect

- A global aspect is defined on a composite component it contains:
  1. Inter-type variables
  2. A global pointcut expression
  3. A global advice
- Acts as a constraint between:
  1. The ports in the pointcut
  2. The variables in the advice

Global Aspect and Correctness

Global Aspect

- A global aspect is defined on a composite component it contains:
  1. Inter-type variables
  2. A global pointcut expression
  3. A global advice
- Acts as a constraint between:
  1. The ports in the pointcut
  2. The variables in the advice

Proposition: Correctness of a Single Weave

- Consider all events in the resulting system
- Define rem_g that removes the advice from the global event (if it exists)
- Verify that the advice is applied to an event e iff $\mathrm{rem}_g(e) \vDash gpc$

Local Execution Trace

Projection

Extract the local state from the global state: $q = \langle q_0, \ldots, q_n \rangle$, $q \upharpoonright B_k = q_k$

Local Execution Trace

Projection

Extract the local state from the global state: $q = \langle q_0, \ldots, q_n \rangle$, $q \upharpoonright B_k = q_k$

Local Event

- Global Event $\rightarrow$ Local Event

Local Execution Trace

Projection

Extract the local state from the global state: $q = \langle q_0, \ldots, q_n \rangle$, $q \upharpoonright B_k = q_k$

Local Event

- Global Event $\rightarrow$ Local Event
- Participation of B_k in an interaction a: $\exists p \in B_k.ports : p \in a.ports$,

Local Execution Trace

Projection

Extract the local state from the global state: $q = \langle q_0, \ldots, q_n \rangle$, $q \upharpoonright B_k = q_k$

Local Event

- Global Event $\rightarrow$ Local Event
- Participation of B_k in an interaction a: $\exists p \in B_k.ports : p \in a.ports$,
- Local component participates: $\langle \langle l, v \rangle, \tau, \langle l', v' \rangle \rangle$
- Local component does not participate: ϵ

Local Execution Trace

Projection

Extract the local state from the global state: $q = \langle q_0, \ldots, q_n \rangle$, $q \upharpoonright B_k = q_k$

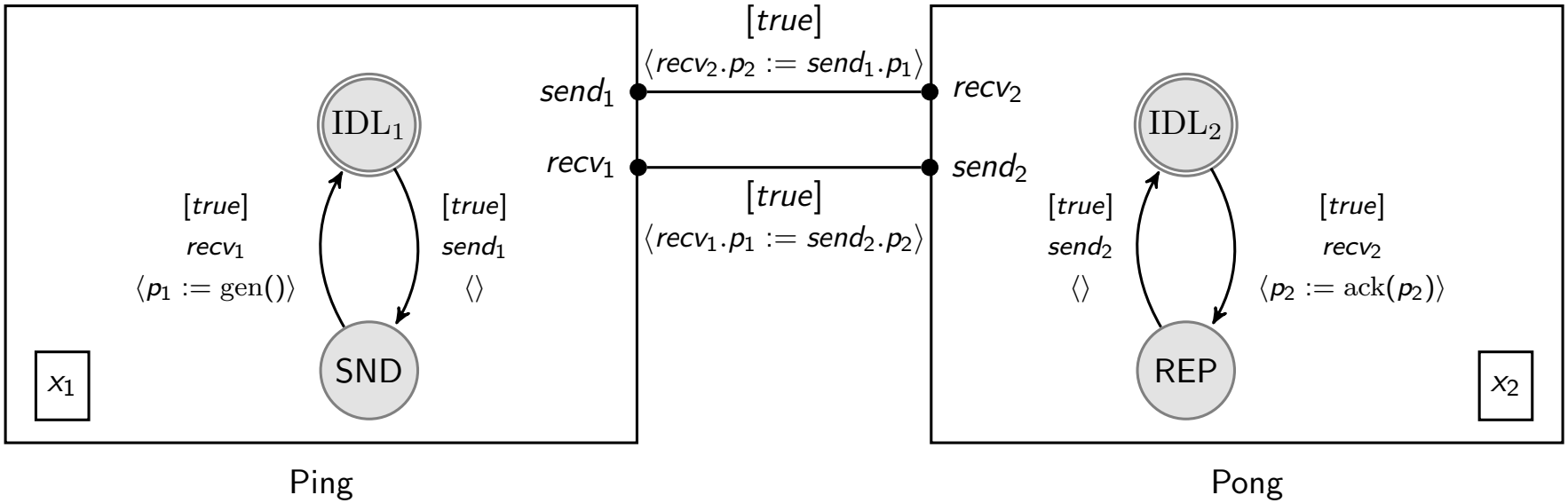
Local Event

- Global Event $\rightarrow$ Local Event
- Participation of B_k in an interaction a: $\exists p \in B_k.ports : p \in a.ports$,
- Local component participates: $\langle \langle l, v \rangle, \tau, \langle l', v' \rangle \rangle$
- Local component does not participate: ϵ

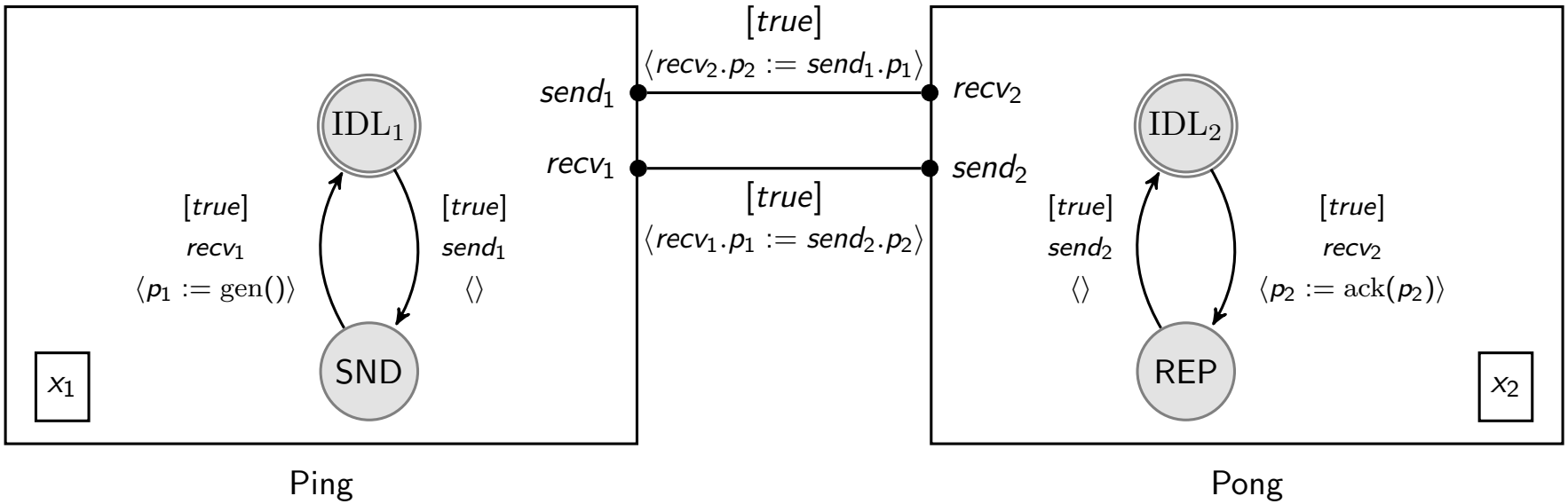
Local Joinpoint

A local joinpoint is a local event.

Example

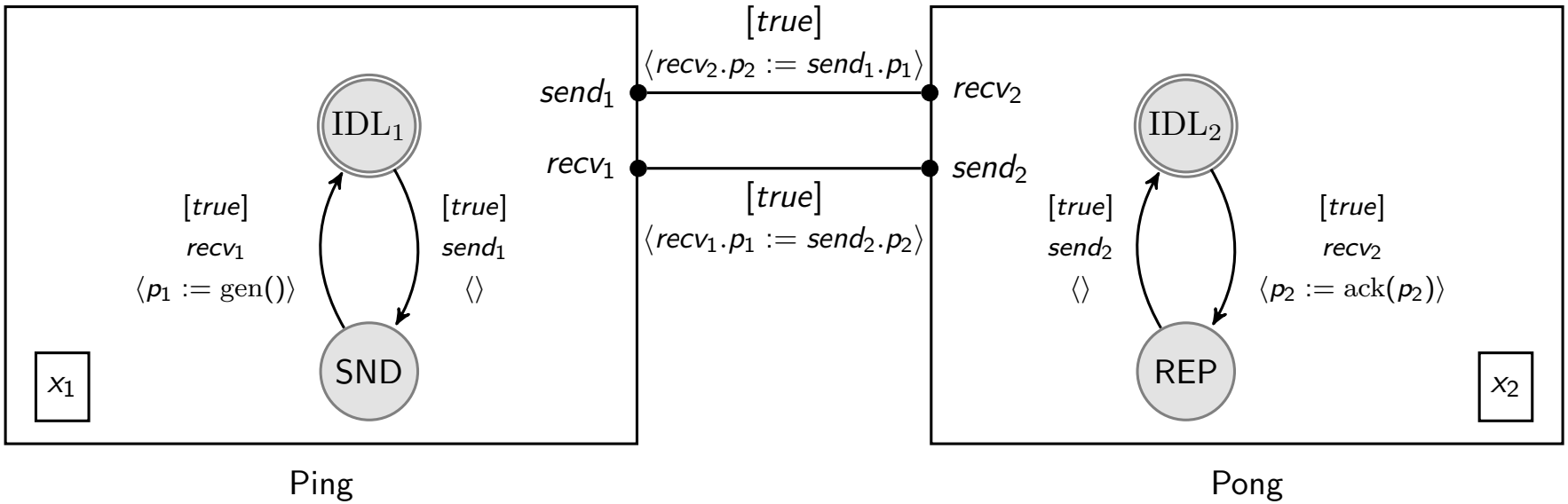

[true]
⟨recv2.p2 := send1.p1⟩
send1
recv2
recv1
send2
[true]
⟨recv1.p1 := send2.p2⟩
IDL1
[true]
recv1
⟨p1 := gen()⟩
[true]
send1
⟨⟩
SND
x1
Ping
IDL2
[true]
send2
⟨⟩
[true]
recv2
⟨p2 := ack(p2)⟩
REP
x2
Pong

Example



- $E_0 = \langle\langle\langle \mathtt{IDL}_1, \langle 1\rangle\rangle, \langle \mathtt{IDL}_2, \langle 0\rangle\rangle\rangle, \{\mathtt{send}_1, \mathtt{recv}_2\}, \langle\langle \mathtt{SND}, \langle 1\rangle\rangle, \langle \mathtt{REP}, \langle 2\rangle\rangle\rangle\rangle$

Example



- $E_0 = \langle\langle\langle \mathtt{IDL}_1, \langle 1\rangle\rangle, \langle \mathtt{IDL}_2, \langle 0\rangle\rangle\rangle, \{\mathtt{send}_1, \mathtt{recv}_2\}, \langle\langle \mathtt{SND}, \langle 1\rangle\rangle, \langle \mathtt{REP}, \langle 2\rangle\rangle\rangle\rangle$
- $\mathrm{map}(E_0, B_0) = \langle\langle \mathtt{IDL}_1, \langle 1\rangle\rangle, \mathtt{send}_1, \langle \mathtt{SND}, \langle 1\rangle\rangle\rangle$
- $\mathrm{map}(E_0, B_1) = \langle\langle \mathtt{IDL}_2, \langle 0\rangle\rangle, \mathtt{recv}_2, \langle \mathtt{REP}, \langle 2\rangle\rangle\rangle$

Local Pointcuts

Location

$e = \langle\langle l, v\rangle, \tau, \langle l', v'\rangle\rangle \vDash lpc$ iff match lpc with:

$$\mid \text{atLocation}(\ell) \qquad \rightarrow (l = \ell)$$

Local Pointcuts

Variables

$e = \langle \langle l, v \rangle, \tau, \langle l', v' \rangle \rangle \vDash lpc$ iff match lpc with:

$$
\begin{array}{ll}
\mid \mathrm{atLocation}(\ell) & \rightarrow (l = \ell) \\
\mid \mathrm{readVarGuard}(x) & \rightarrow (\exists t \in B_k.trans : t.src = l \wedge x \in \mathsf{readguard}(t)) \\
\mid \mathrm{readVarFunc}(x) & \rightarrow (x \in \mathsf{readvar}(\tau.func)) \\
\mid \mathrm{write}(x) & \rightarrow (x \in \mathsf{writevar}(\tau.func))
\end{array}
$$

Local Pointcuts

Ports

$e = \langle \langle l, v \rangle, \tau, \langle l', v' \rangle \rangle \vDash lpc$ iff match lpc with:

$$
\begin{array}{ll}
\mid \mathrm{atLocation}(\ell) & \rightarrow (l = \ell) \\
\mid \mathrm{readVarGuard}(x) & \rightarrow (\exists t \in B_k.trans : t.src = l \wedge x \in \mathsf{readguard}(t)) \\
\mid \mathrm{readVarFunc}(x) & \rightarrow (x \in \mathsf{readvar}(\tau.func)) \\
\mid \mathrm{write}(x) & \rightarrow (x \in \mathsf{writevar}(\tau.func)) \\
\mid \mathrm{portEnabled}(p) & \rightarrow (\exists q' : \langle \langle l, v \rangle, p, q' \rangle \in \rightarrow_{B_k})
\end{array}
$$

Local Pointcuts

Ports

$e = \langle \langle l, v \rangle, \tau, \langle l', v' \rangle \rangle \vDash lpc$ iff match lpc with:

$$
\begin{array}{ll}
\mid \mathrm{atLocation}(\ell) & \rightarrow (l = \ell) \\
\mid \mathrm{readVarGuard}(x) & \rightarrow (\exists t \in B_k.trans : t.src = l \wedge x \in \mathsf{readguard}(t)) \\
\mid \mathrm{readVarFunc}(x) & \rightarrow (x \in \mathsf{readvar}(\tau.func)) \\
\mid \mathrm{write}(x) & \rightarrow (x \in \mathsf{writevar}(\tau.func)) \\
\mid \mathrm{portEnabled}(p) & \rightarrow (\exists q' : \langle \langle l, v \rangle, p, q' \rangle \in \rightarrow_{B_k}) \\
\mid \mathrm{portExecute}(p) & \rightarrow (\tau.port = p)
\end{array}
$$

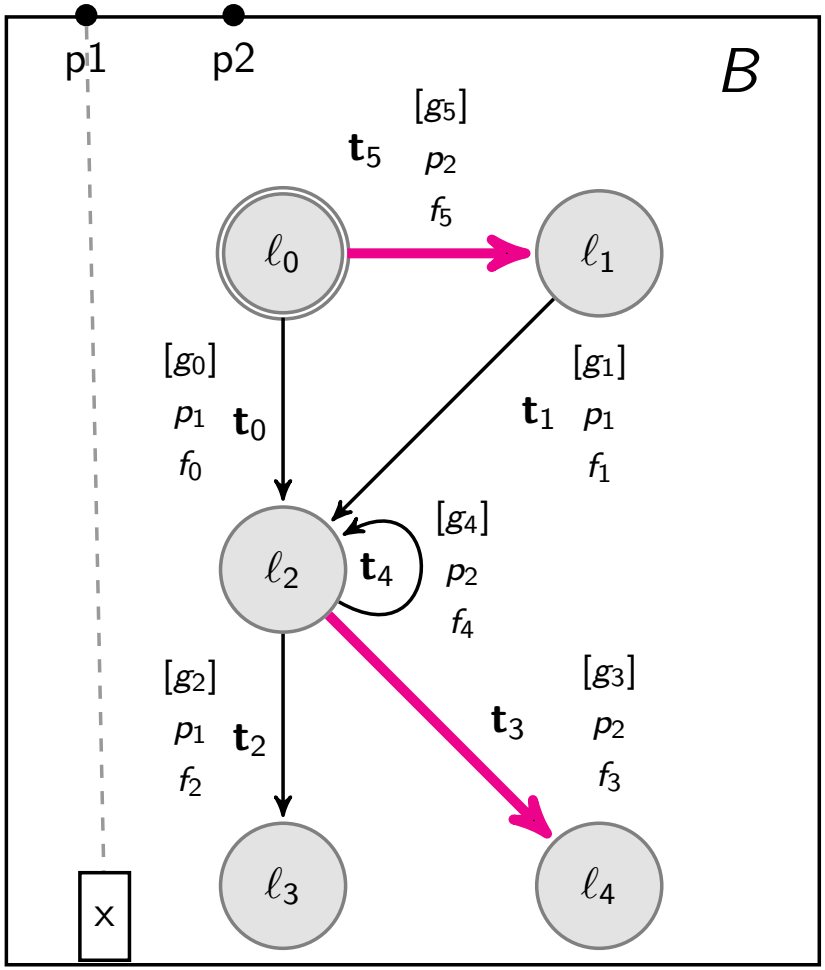
Local Pointcuts

Combination

$e = \langle \langle l, v \rangle, \tau, \langle l', v' \rangle \rangle \vDash lpc$ iff match lpc with:

$$
\begin{array}{ll}
\mid \mathrm{atLocation}(\ell) & \rightarrow (l = \ell) \\
\mid \mathrm{readVarGuard}(x) & \rightarrow (\exists t \in B_k.trans : t.src = l \wedge x \in \mathsf{readguard}(t)) \\
\mid \mathrm{readVarFunc}(x) & \rightarrow (x \in \mathsf{readvar}(\tau.func)) \\
\mid \mathrm{write}(x) & \rightarrow (x \in \mathsf{writevar}(\tau.func)) \\
\mid \mathrm{portEnabled}(p) & \rightarrow (\exists q' : \langle \langle l, v \rangle, p, q' \rangle \in \rightarrow_{B_k}) \\
\mid \mathrm{portExecute}(p) & \rightarrow (\tau.port = p) \\
\mid \phi \text{ and } \phi' & \rightarrow e \vDash \phi \wedge e \vDash \phi'
\end{array}
$$

Syntactic Representation



$$M = \{t_3, t_5\}$$

ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$\mathbf{t_5} : \langle \mathbf{g_5}, \mathbf{p_2}, \mathbf{f_5} \rangle$	ℓ_1

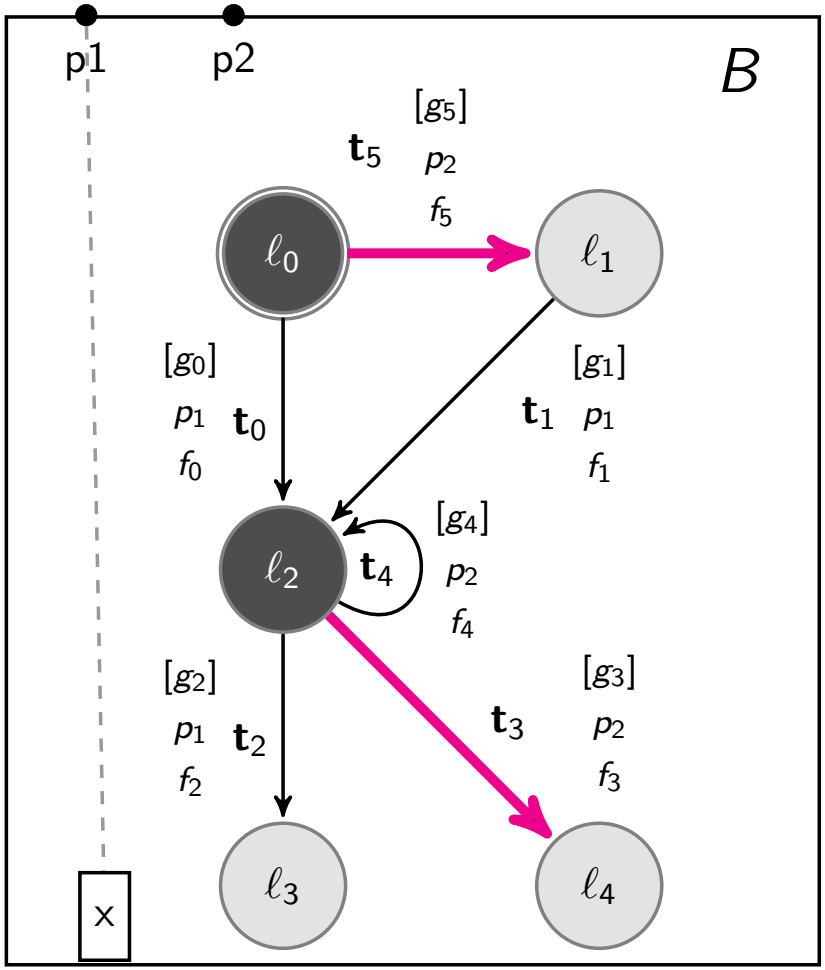
ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$\mathbf{t_3} : \langle \mathbf{g_3}, \mathbf{p_2}, \mathbf{f_3} \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3

ℓ_4

Syntactic Representation



origin(M)

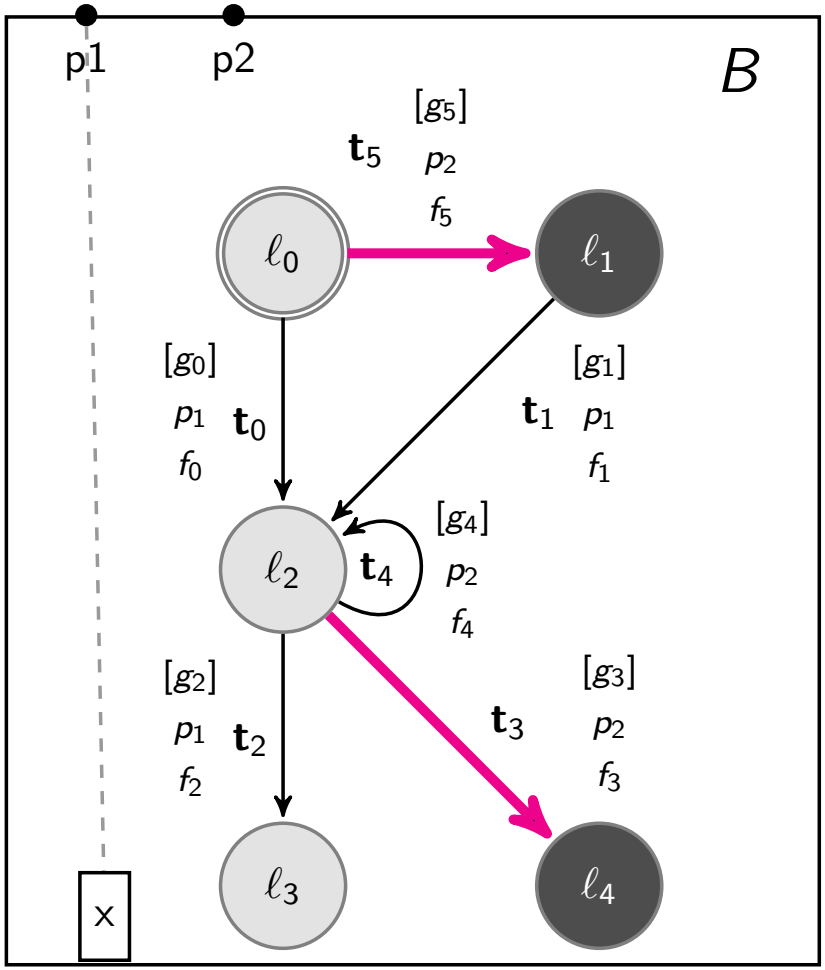
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$\mathbf{t_5 : \langle g_5, p_2, f_5 \rangle}$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$\mathbf{t_3 : \langle g_3, p_2, f_3 \rangle}$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Syntactic Representation



$\mathrm{dest}(M)$

ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$\mathbf{t_5 : \langle g_5, p_2, f_5 \rangle}$	ℓ_1

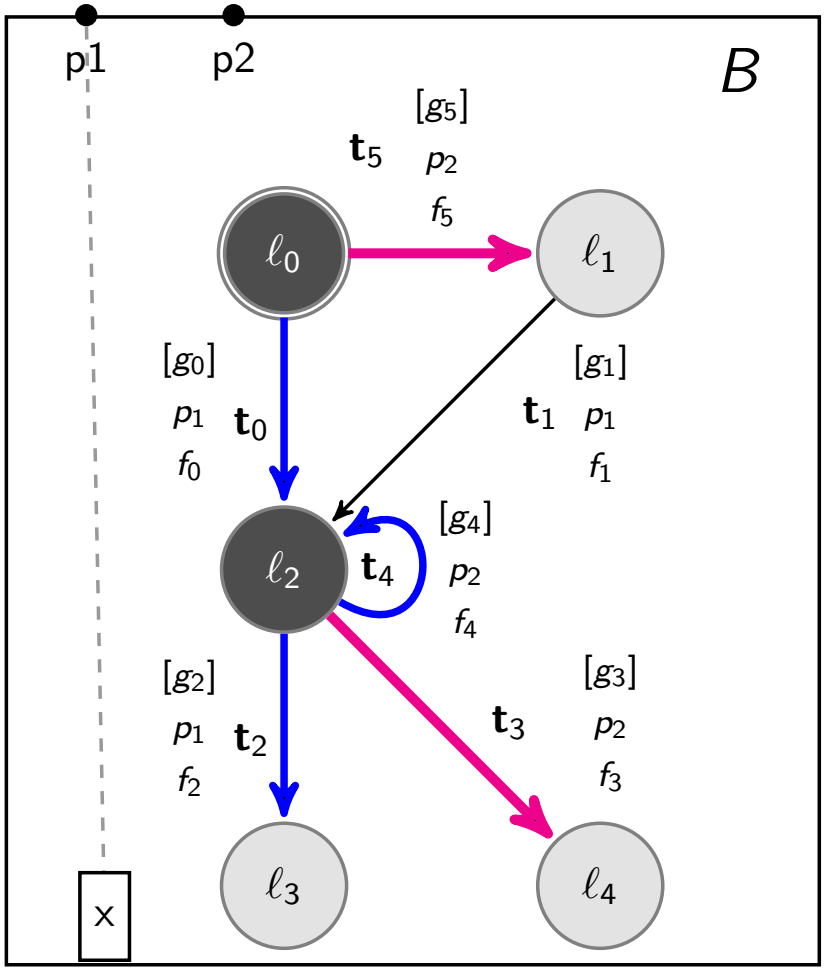
ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$\mathbf{t_3 : \langle g_3, p_2, f_3 \rangle}$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3

ℓ_4

Syntactic Representation



siblings(M)

ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$\mathbf{t_5} : \langle \mathbf{g_5}, \mathbf{p_2}, \mathbf{f_5} \rangle$	ℓ_1

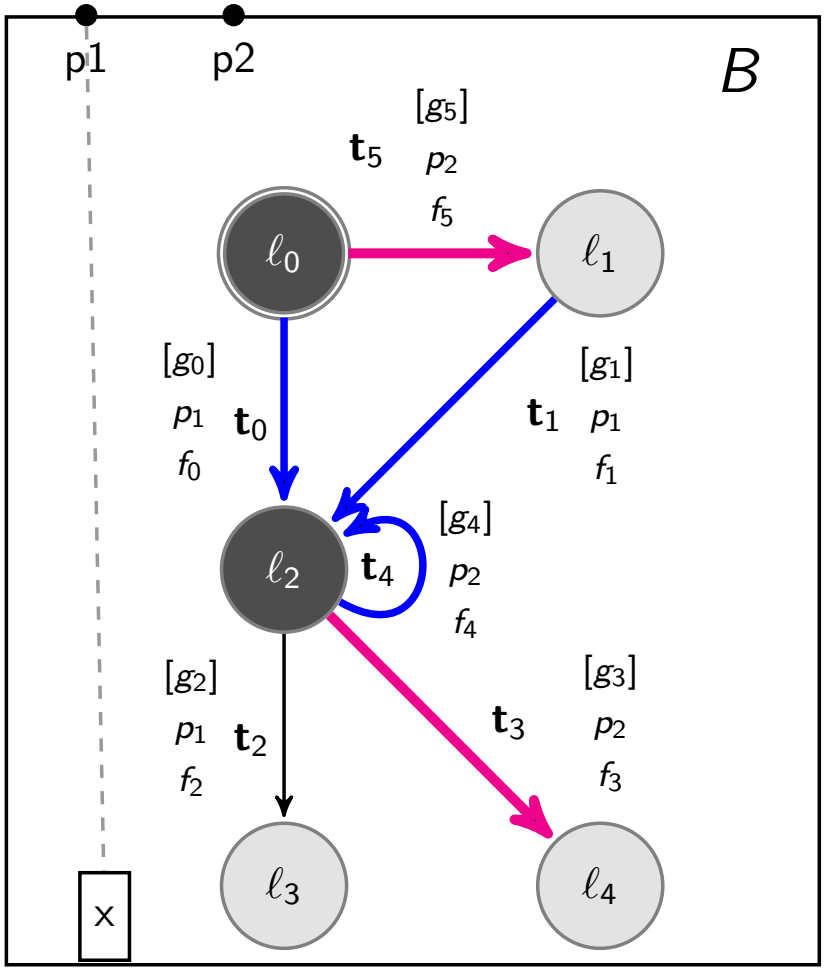
ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$\mathbf{t_3} : \langle \mathbf{g_3}, \mathbf{p_2}, \mathbf{f_3} \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3

ℓ_4

Syntactic Representation



previous(M)

ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$\mathbf{t_5} : \langle \mathbf{g_5}, \mathbf{p_2}, \mathbf{f_5} \rangle$	ℓ_1

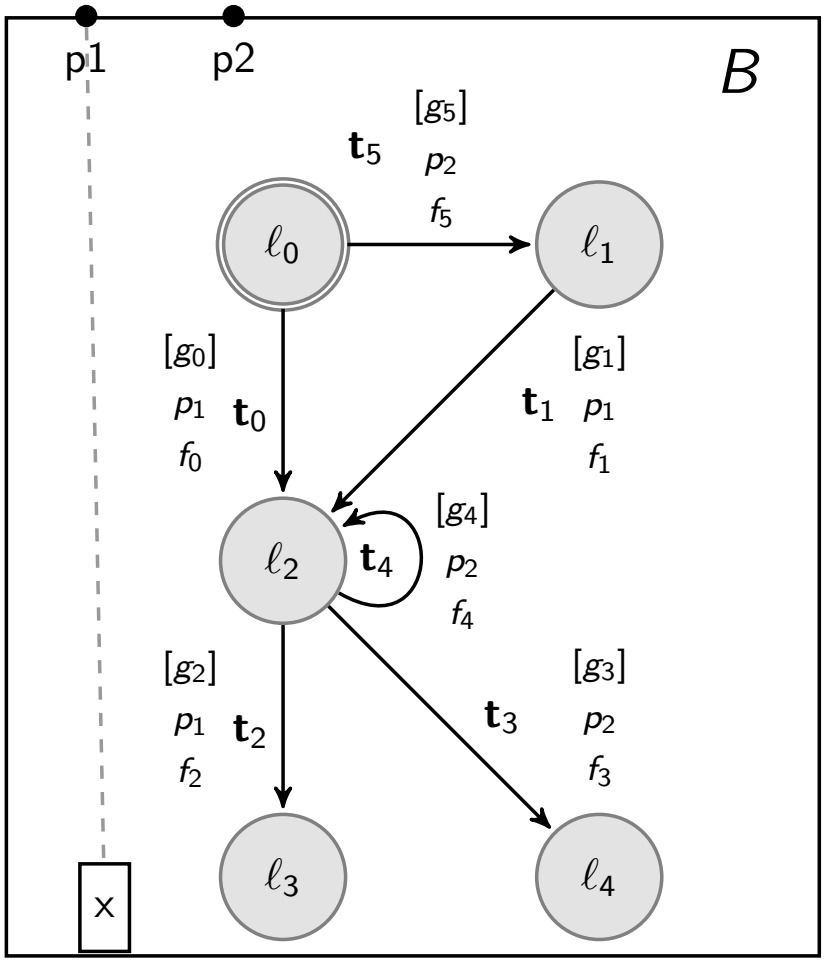
ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$\mathbf{t_3} : \langle \mathbf{g_3}, \mathbf{p_2}, \mathbf{f_3} \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3

ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

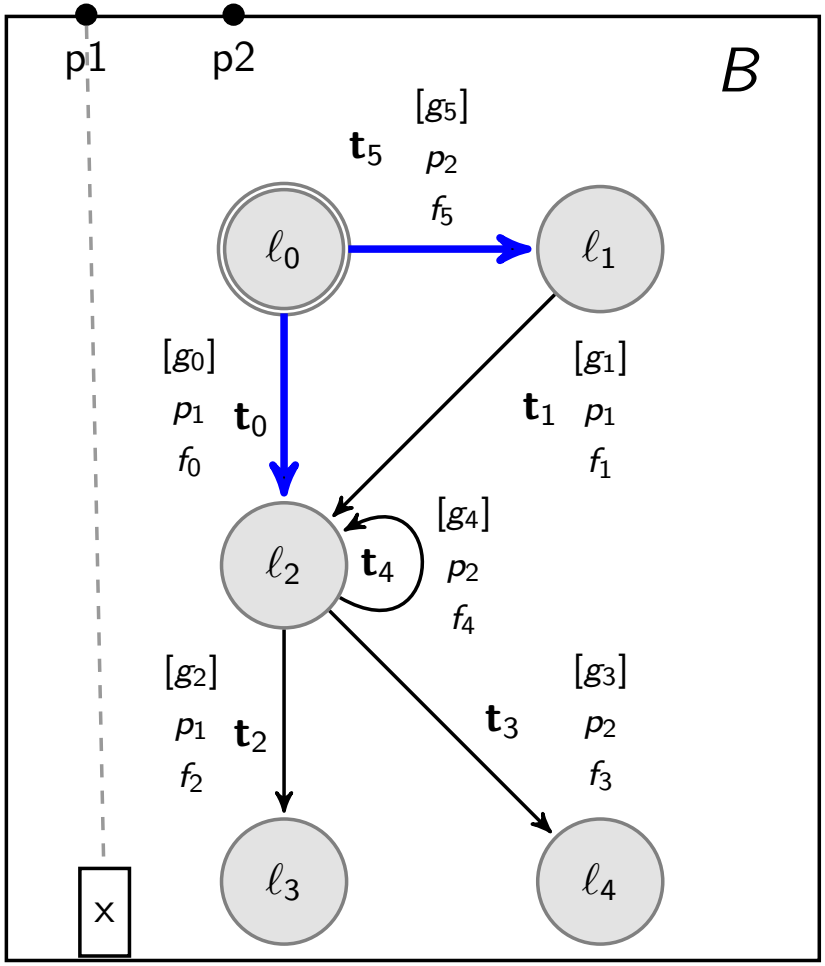
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

$\text{atLocation}(\ell_0)$

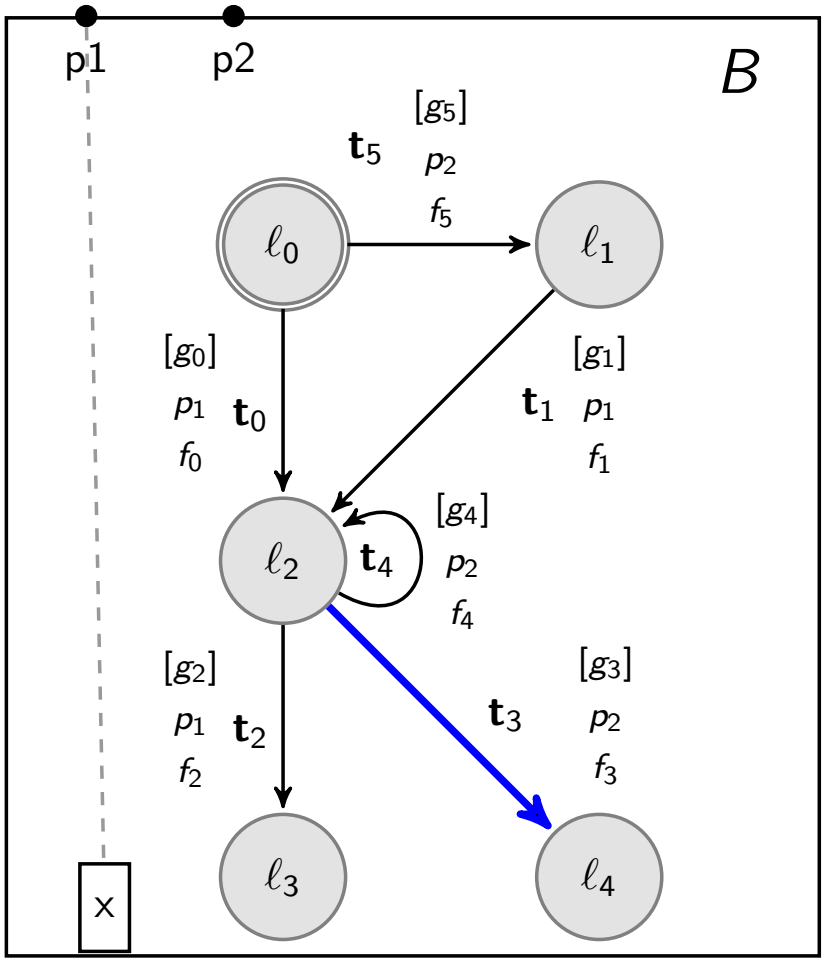
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

$\mathrm{readVarGuard}(x)$

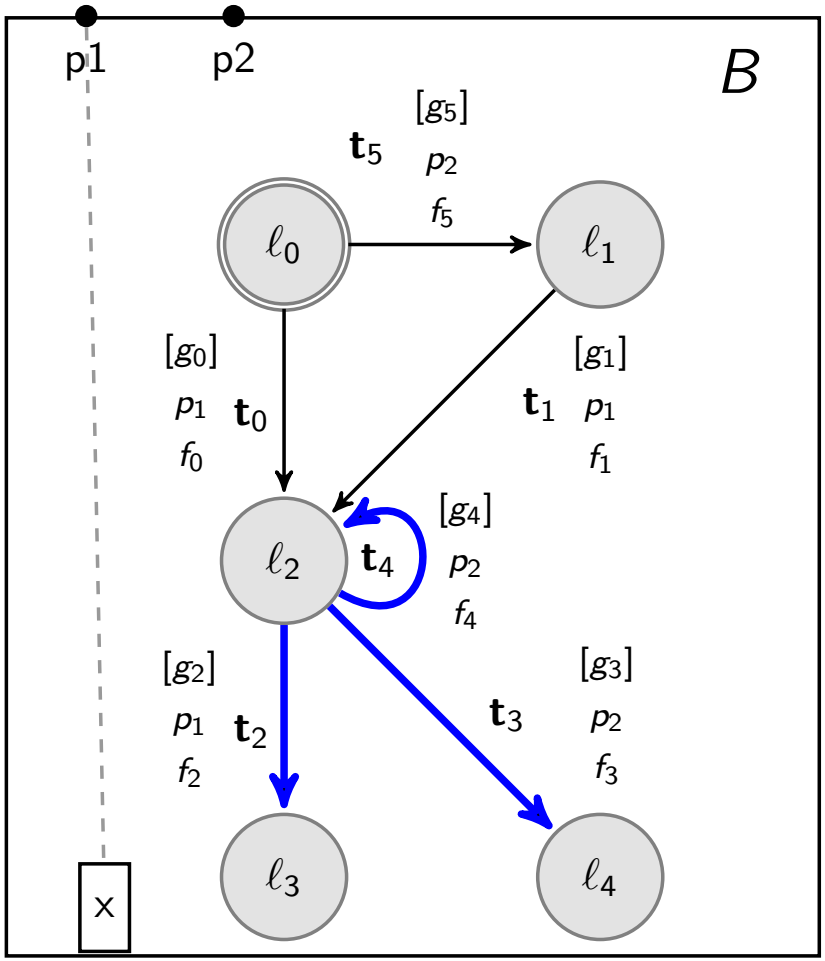
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

readVarGuard(x) [Siblings]

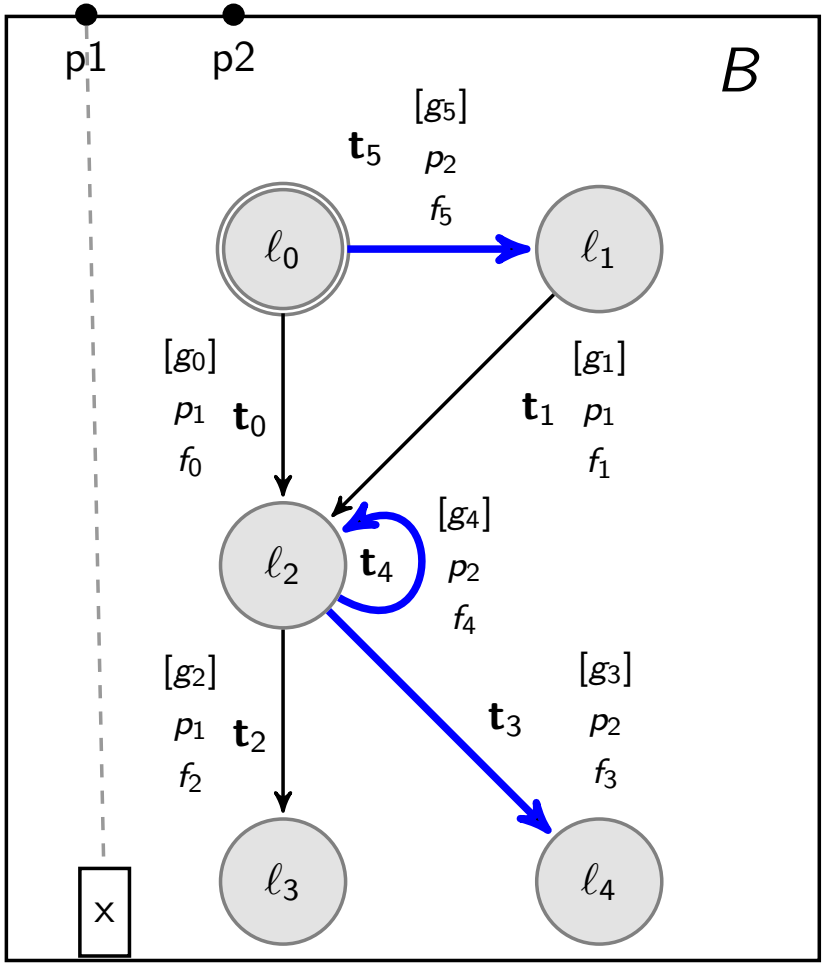
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

portExecute(p_2)

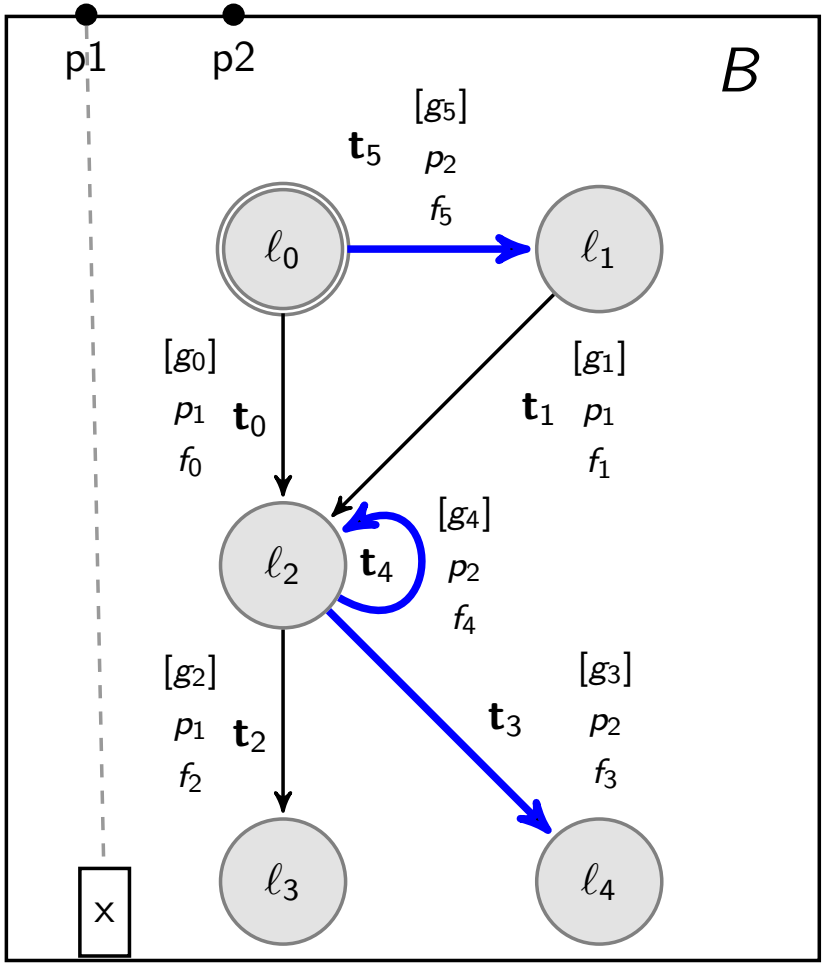
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

$\text{portEnabled}(p_2)$

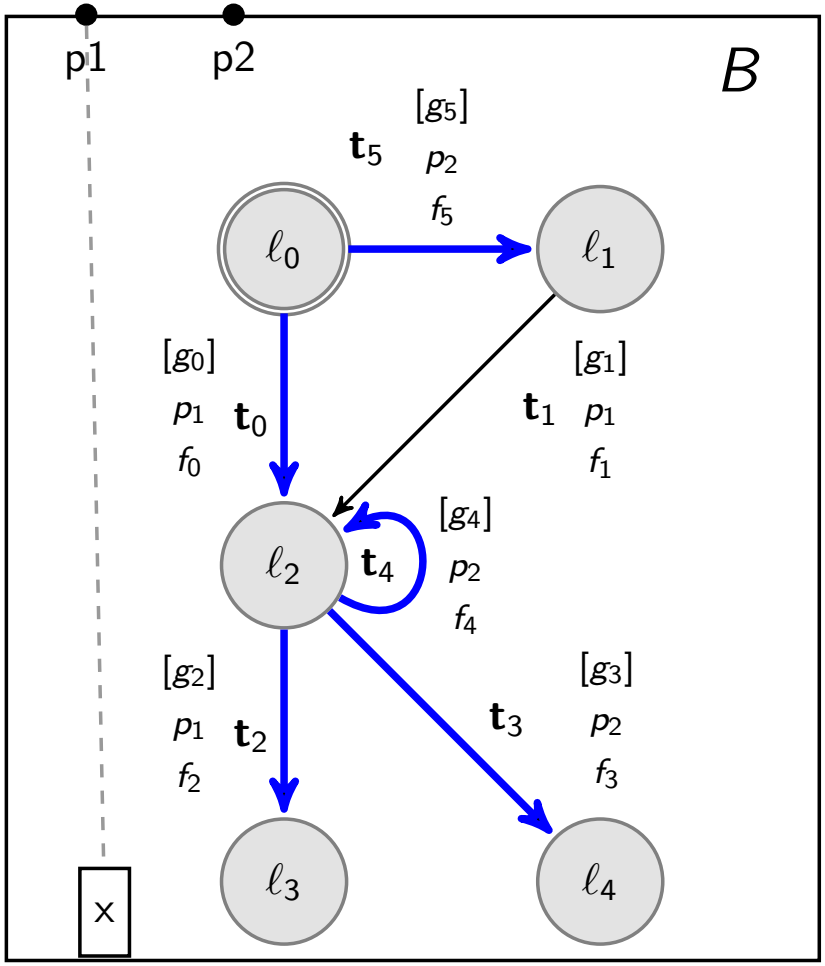
ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Matching Pointcuts



$$g_3 = x > 3$$

$\mathrm{portEnabled}(p_2)[\mathrm{Siblings}]$

ℓ_0	
$t_0 : \langle g_0, p_1, f_0 \rangle$	ℓ_2
$t_5 : \langle g_5, p_2, f_5 \rangle$	ℓ_1

ℓ_2	
$t_2 : \langle g_2, p_1, f_2 \rangle$	ℓ_3
$t_3 : \langle g_3, p_2, f_3 \rangle$	ℓ_4
$t_4 : \langle g_4, p_2, f_4 \rangle$	ℓ_2

ℓ_1	
$t_1 : \langle g_1, p_1, f_1 \rangle$	ℓ_2

ℓ_3
ℓ_4

Local Advice

1. A function f_b to execute before the update function.
2. A function f_a to execute after the update function.

Local Advice

1. A function f_b to execute before the update function.
2. A function f_a to execute after the update function.
3. A set of "reset location" pairs $\langle l, g \rangle$: after joinpoint move to l if g holds.

Local Weaving

- Add inter-type variables to the component directly

Local Weaving

- Add inter-type variables to the component directly
- Determine what is before and after the element selection

Local Weaving

- Add inter-type variables to the component directly
- Determine what is before and after the element selection
- Dynamic behavior: portEnabled, reset location

Local Weaving

- Add inter-type variables to the component directly
- Determine what is before and after the element selection
- Dynamic behavior: portEnabled, reset location
  - Use an extra boolean variable to denote entry/exit of a pointcut area

Local Weaving

- Add inter-type variables to the component directly
- Determine what is before and after the element selection
- Dynamic behavior: portEnabled, reset location
  - Use an extra boolean variable to denote entry/exit of a pointcut area
  - Pre-evaluate guards in an earlier temporary location

Local Weaving

- Add inter-type variables to the component directly
- Determine what is before and after the element selection
- Dynamic behavior: portEnabled, reset location
  - Use an extra boolean variable to denote entry/exit of a pointcut area
  - Pre-evaluate guards in an earlier temporary location
  - Simulate if/else construct by duplicating transitions

Composition (Overview)

- Conditions for interference
  - Multiple concerns can overlap one region
  - Not all concerns are independant
  - Possibility to have non-determinism or unwanted side-effects

Composition (Overview)

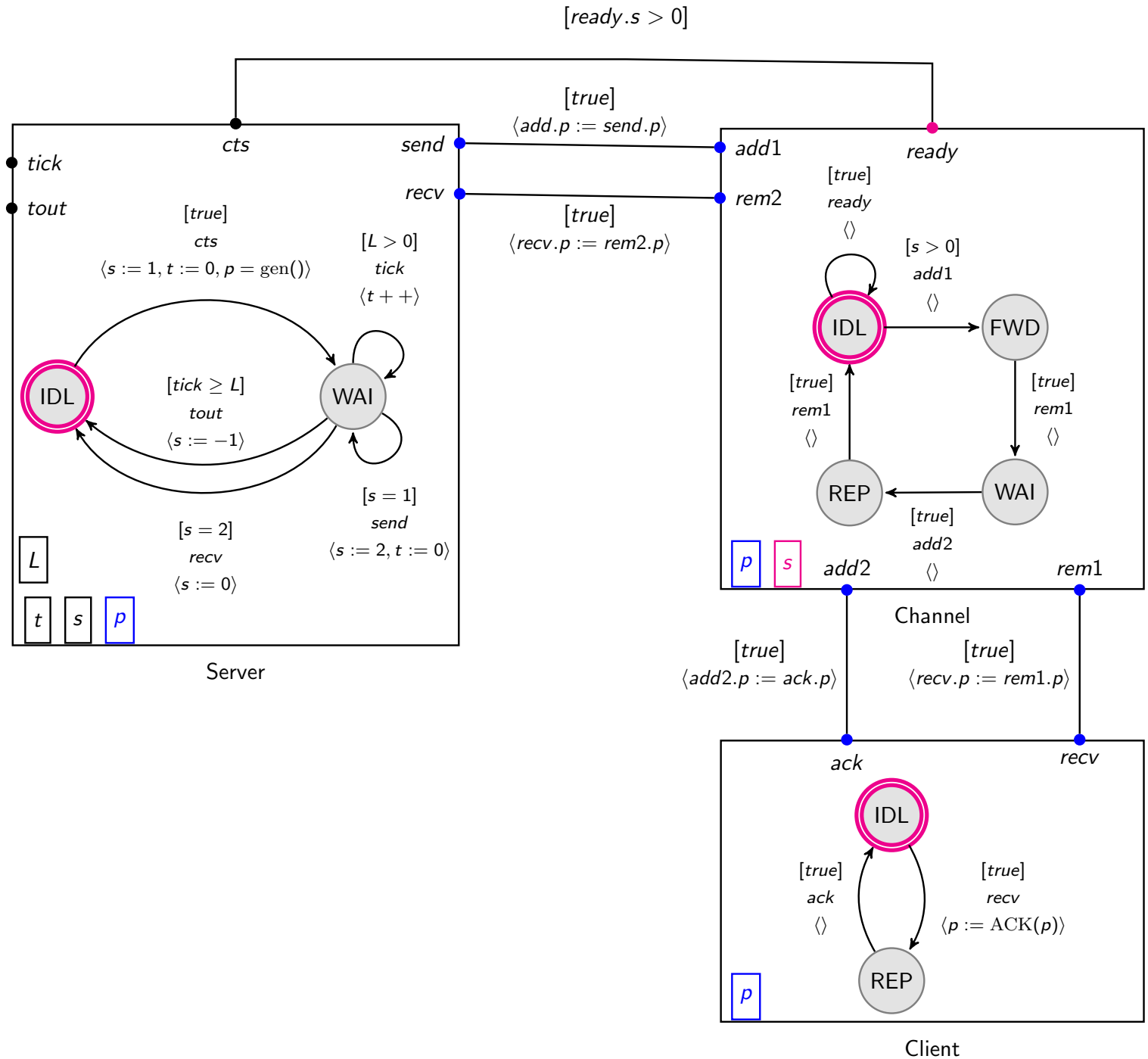
- Conditions for interference
  - Multiple concerns can overlap one region
  - Not all concerns are independant
  - Possibility to have non-determinism or unwanted side-effects
- Dealing with interference
  - Aspect containers: extra constraints on the aspects (ordering and sharing)

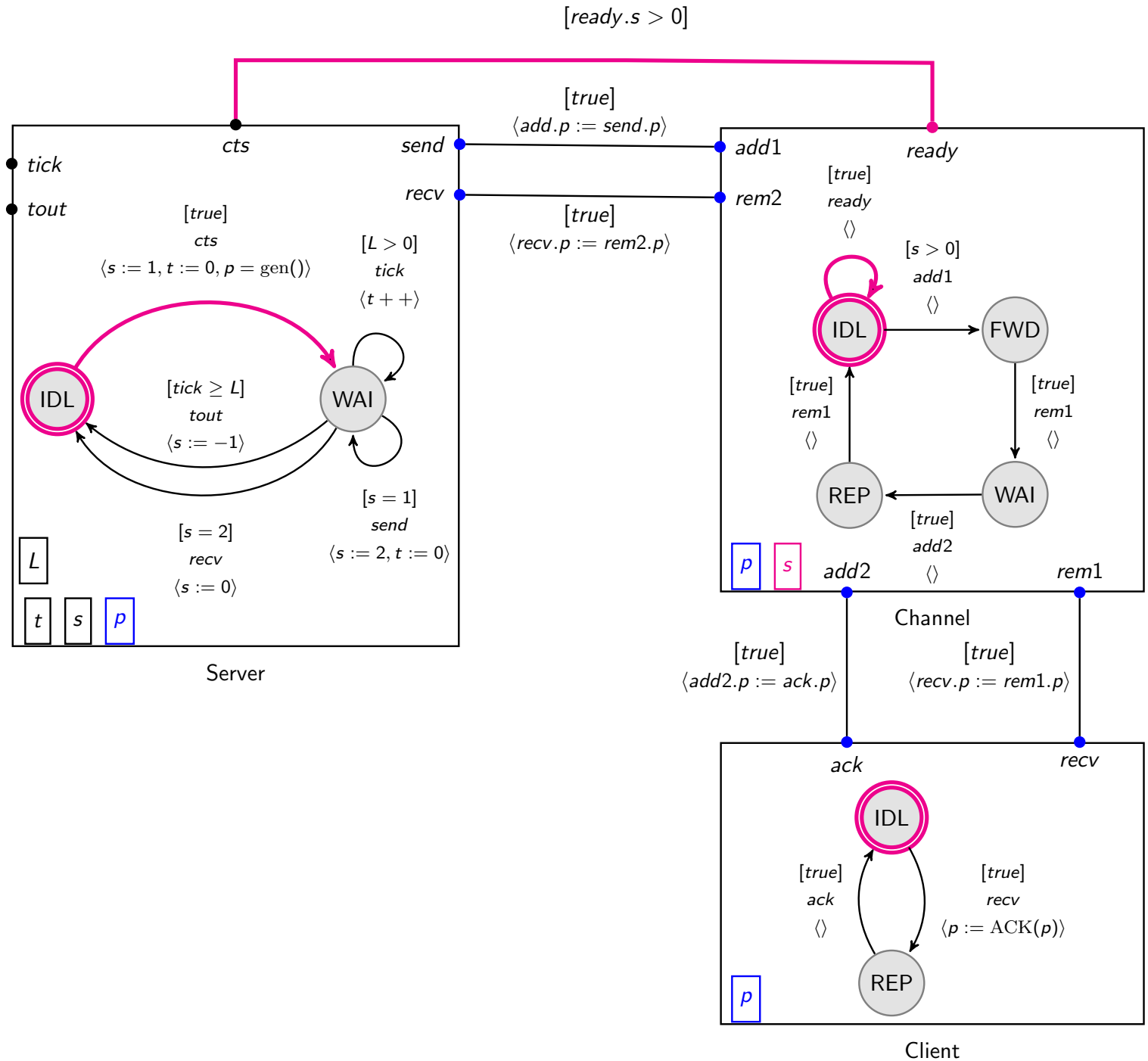
Composition (Overview)

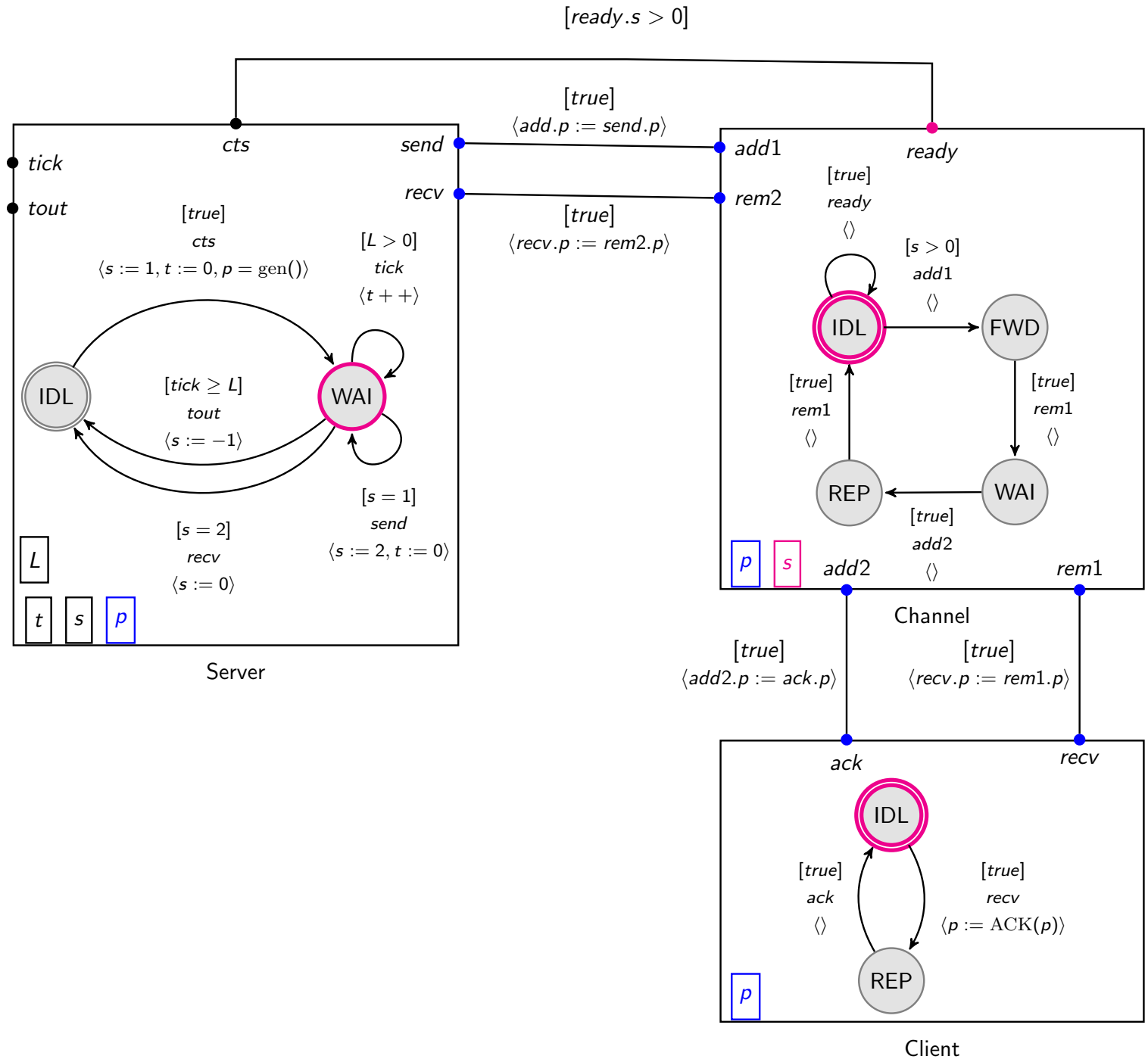
- Conditions for interference
  - Multiple concerns can overlap one region
  - Not all concerns are independant
  - Possibility to have non-determinism or unwanted side-effects
- Dealing with interference
  - Aspect containers: extra constraints on the aspects (ordering and sharing)
  - Weaving procedures: match/weave strategy

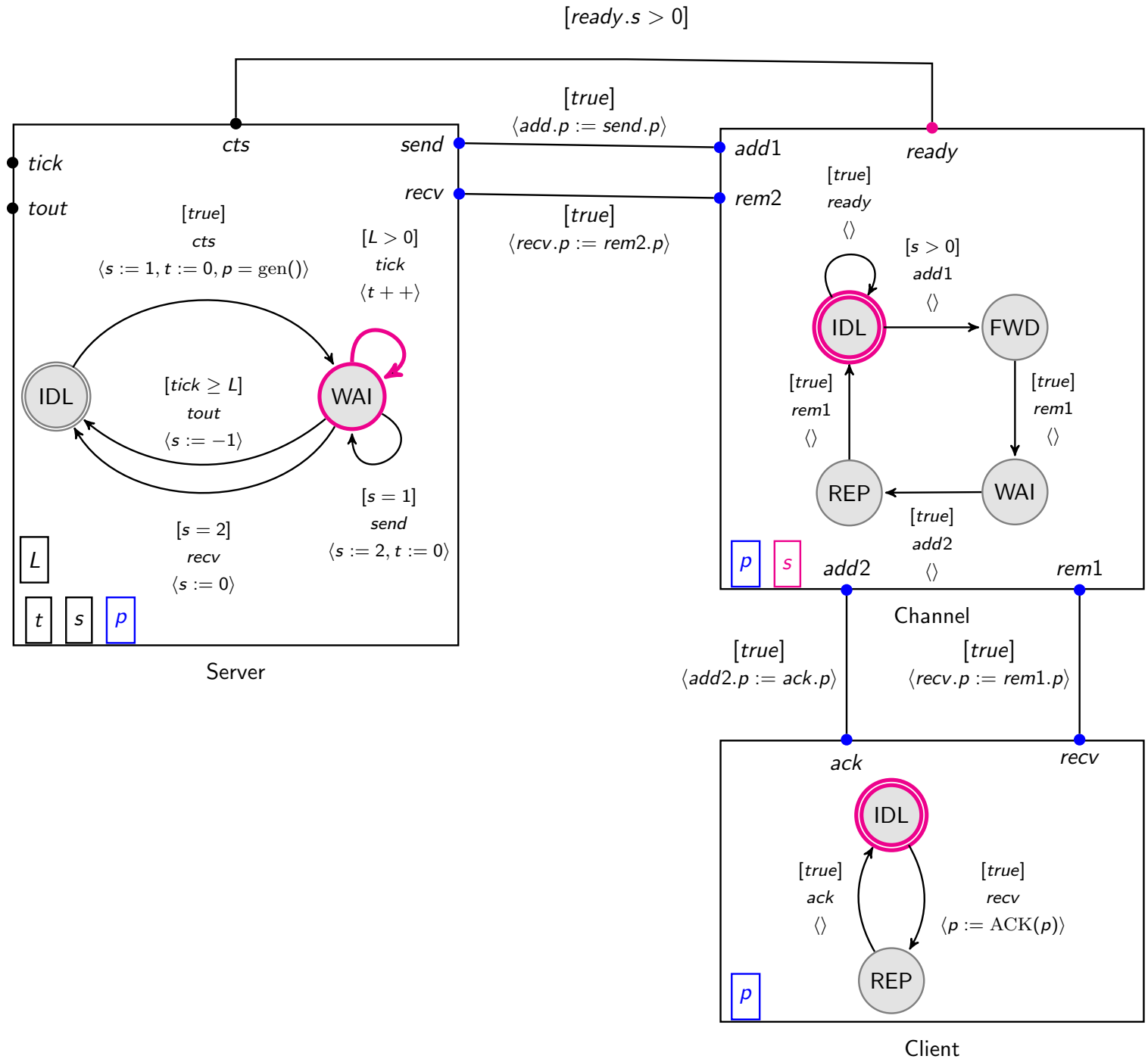
Using AOP-BIP
Network Example [1]

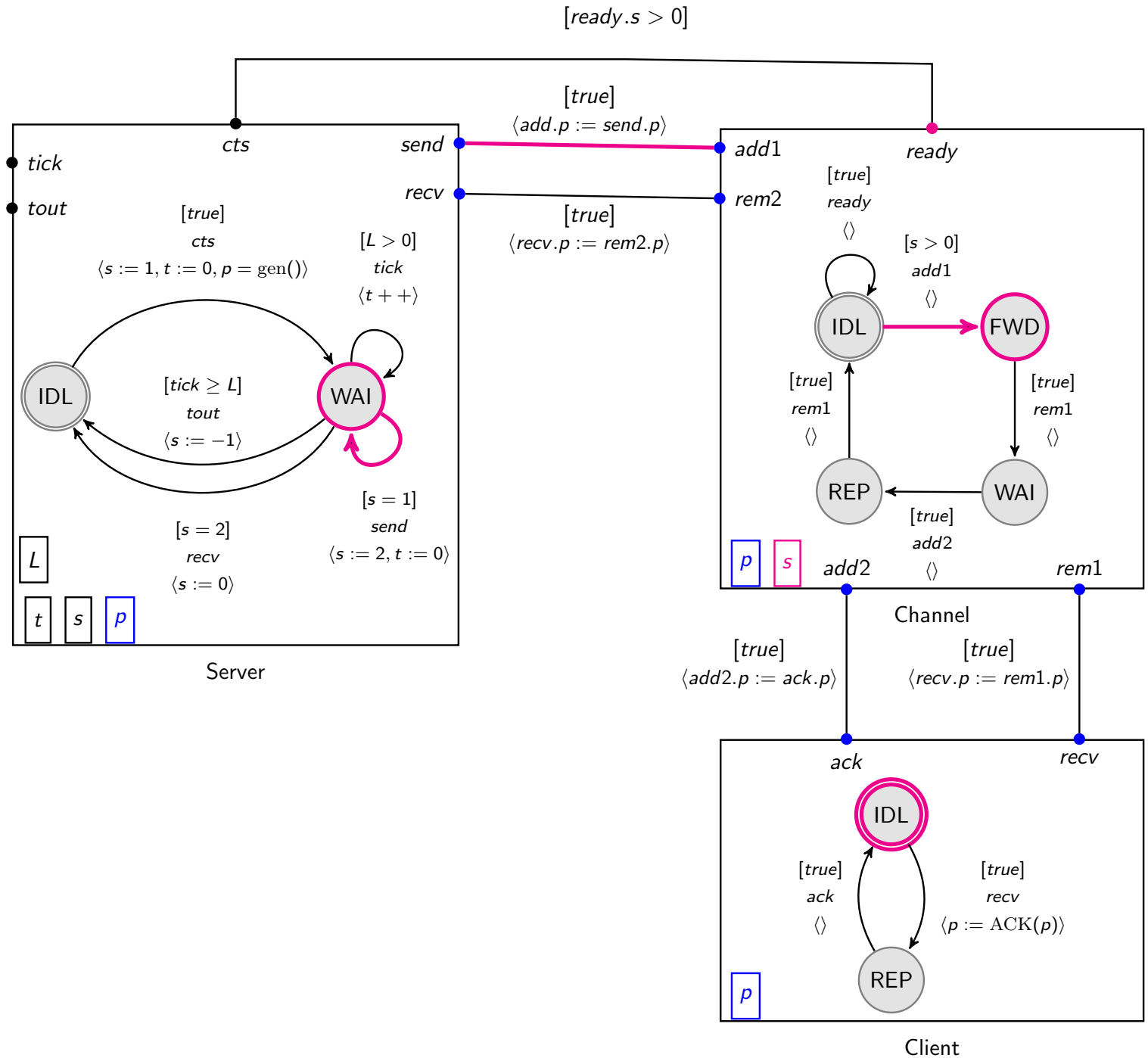
[1] A modified version of the network in [Bonakdarpour et al., 2012]

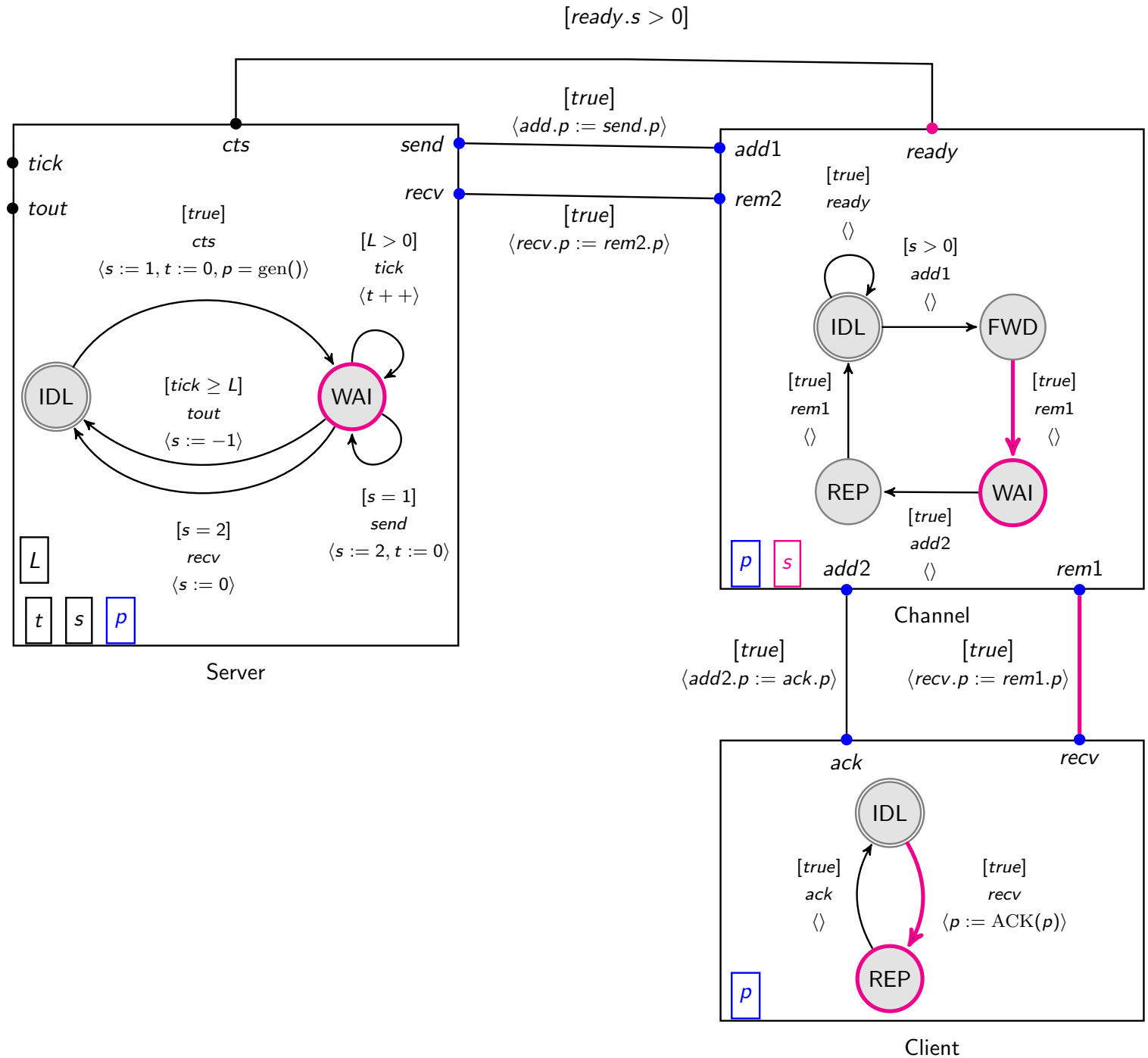

[ready.s > 0]
[true]
⟨add.p := send.p⟩
[true]
⟨recv.p := rem2.p⟩
cts
send
recv
tick
tout
add1
rem2
ready
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
IDL
WAI
[tick ≥ L]
tout
⟨s := −1⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
Server
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client

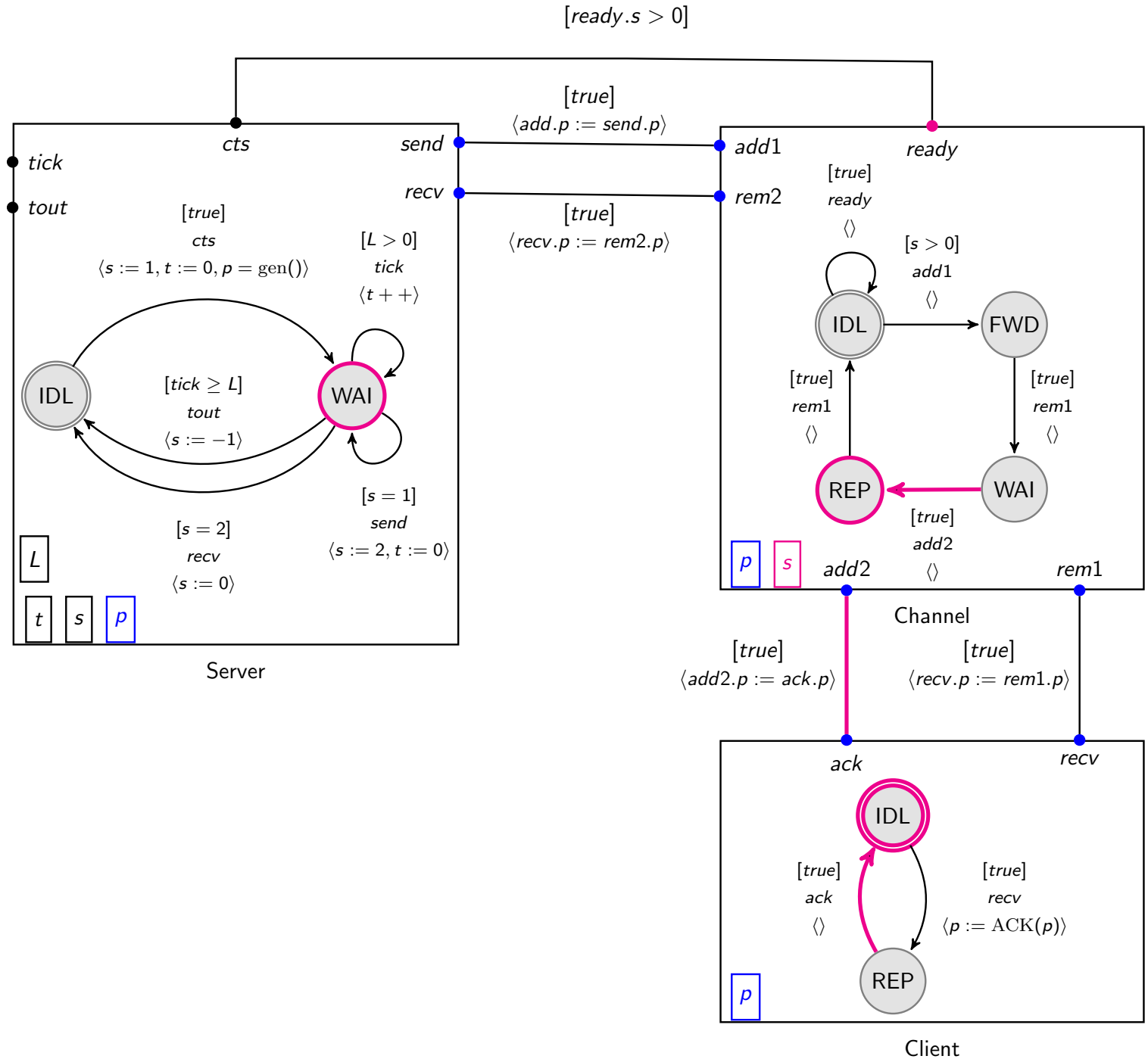

[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
tout
recv
rem2
[true]
⟨recv.p := rem2.p⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[tick ≥ L]
tout
⟨s := −1⟩
WAI
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
p
s
add2
rem1
Server
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client

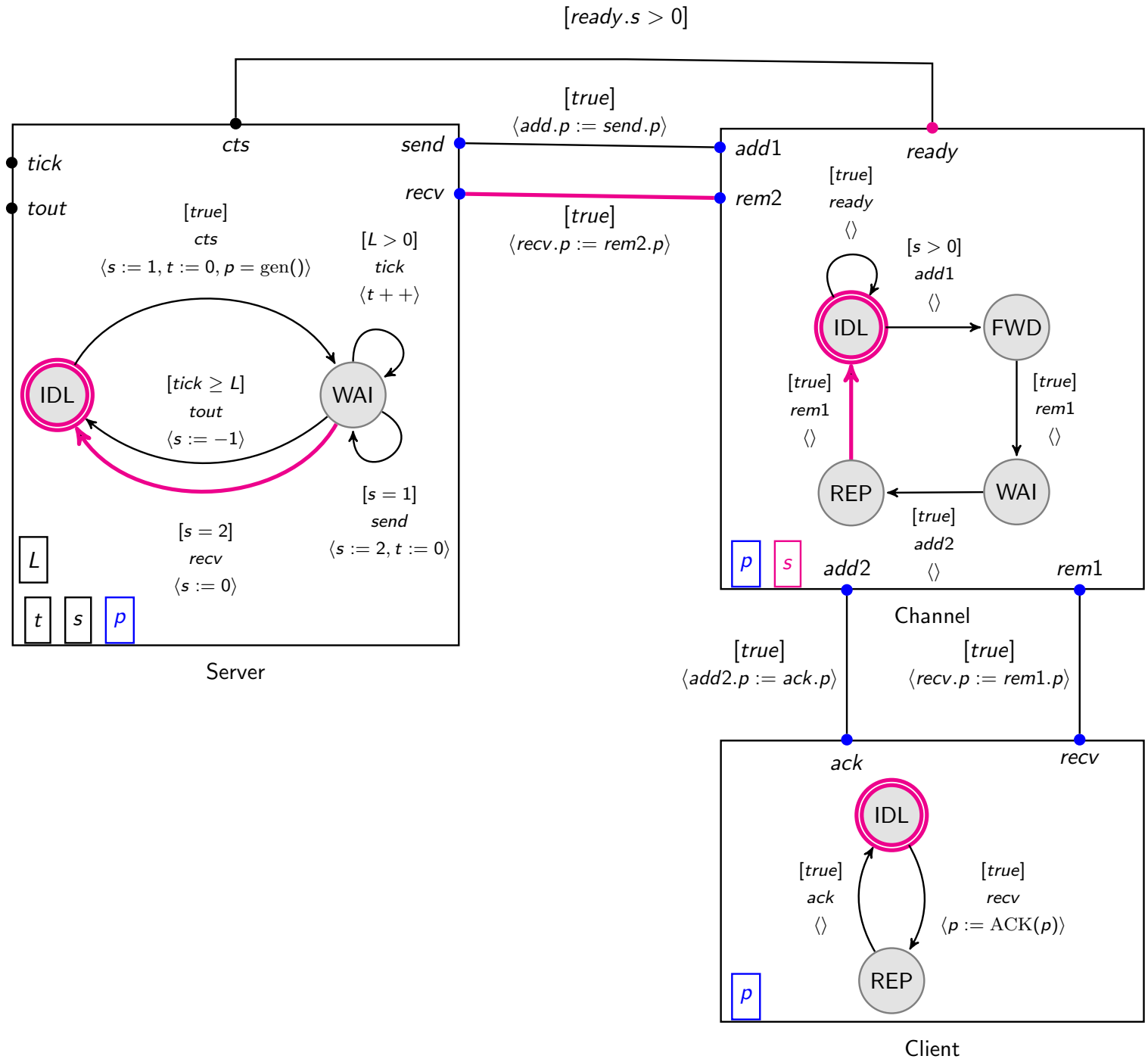

[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
tout
recv
rem2
[true]
⟨recv.p := rem2.p⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
IDL
WAI
[tick ≥ L]
tout
⟨s := −1⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
Server
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client

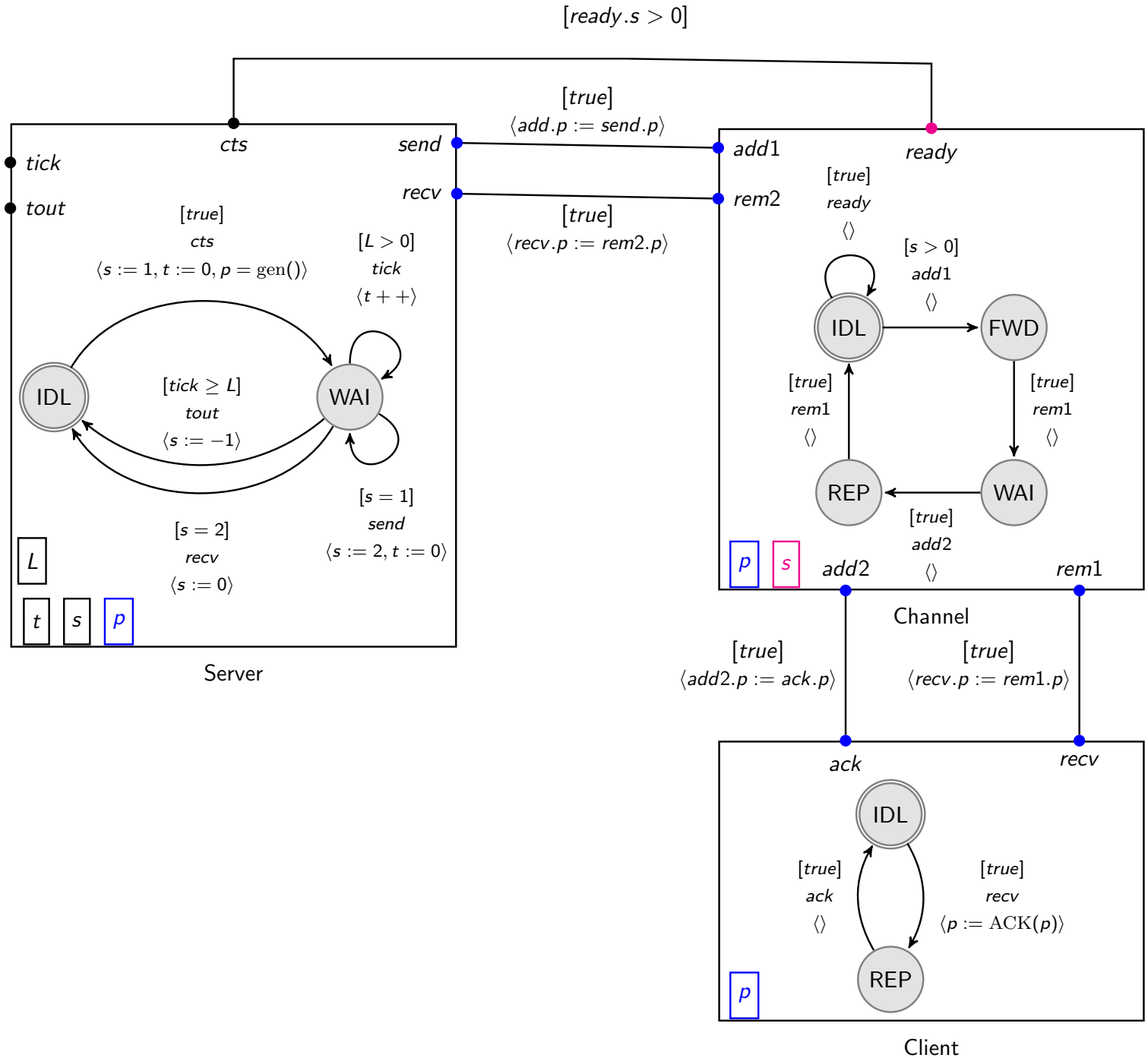

[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
recv
rem2
tout
[true]
⟨recv.p := rem2.p⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[tick ≥ L]
tout
⟨s := −1⟩
WAI
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
Server
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client


[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
tout
recv
rem2
[true]
⟨recv.p := rem2.p⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t + +⟩
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[tick ≥ L]
tout
⟨s := −1⟩
WAI
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
Server
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client


[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
recv
rem2
tout
[true]
⟨recv.p := rem2.p⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[tick ≥ L]
tout
⟨s := −1⟩
WAI
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
Server
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client


[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
tout
recv
rem2
[true]
⟨recv.p := rem2.p⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[tick ≥ L]
tout
⟨s := −1⟩
WAI
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
Server
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client


[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
add1
ready
tick
recv
rem2
tout
[true]
⟨recv.p := rem2.p⟩
[true]
ready
⟨⟩
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[tick ≥ L]
tout
⟨s := −1⟩
WAI
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[s = 1]
send
⟨s := 2, t := 0⟩
[true]
add2
⟨⟩
[s = 2]
recv
⟨s := 0⟩
L
p
s
add2
rem1
t
s
p
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Server
Client


[ready.s > 0]
[true]
⟨add.p := send.p⟩
cts
send
tick
tout
recv
[true]
cts
⟨s := 1, t := 0, p = gen()⟩
[L > 0]
tick
⟨t++⟩
IDL
WAI
[tick ≥ L]
tout
⟨s := −1⟩
[s = 1]
send
⟨s := 2, t := 0⟩
[s = 2]
recv
⟨s := 0⟩
L
t
s
p
Server
[true]
⟨recv.p := rem2.p⟩
add1
rem2
ready
[true]
ready
⟨⟩
[s > 0]
add1
⟨⟩
IDL
FWD
[true]
rem1
⟨⟩
[true]
rem1
⟨⟩
REP
WAI
[true]
add2
⟨⟩
p
s
add2
rem1
Channel
[true]
⟨add2.p := ack.p⟩
[true]
⟨recv.p := rem1.p⟩
ack
recv
IDL
[true]
ack
⟨⟩
[true]
recv
⟨p := ACK(p)⟩
REP
p
Client

Authentication Concern

```
Aspect AddHash (server)  {
  portExecute(cts)
  do {} {p = wrap(p); }
}
Aspect VerifyHash (channel) {
  data int clear = 0
  portExecute(add1)
  do {} {clear = check(p); p = unwrap(p);}
  {(IDL, clear == 0)}
}
Aspect Carol {
  ports(a:server.send b:channel.add1)
  readPortVars(a.r)
  do {} {b.r = pfake(a.r);}
}
```

Authentication Concern

```
Aspect AddHash (server)  {
  portExecute(cts)
  do {} {p = wrap(p); }
}
Aspect VerifyHash (channel) {
  data int clear = 0
  portExecute(add1)
  do {} {clear = check(p); p = unwrap(p);}
  {(IDL, clear == 0)}
}
Aspect Carol {
  ports(a:server.send b:channel.add1)
  readPortVars(a.r)
  do {} {b.r = pfake(a.r);}
}
```

```
[Server.cts  ] Clear to Send
[Server.send ] -> 886|6 (Time: 0)
[Channel.add1] <- 386|6
[Channel.rem1] -> 386
[Client.recv ] <- ACK
[Client.ack  ] -> ACK
[Channel.add2] <- ACK
[Channel.rem2] -> ACK
[Server.recv ] <- ACK (Time: 3)

[Server.cts  ] Clear to Send
[Server.send ] -> 763|3 (Time: 0)
[Channel.add1] <- 736|3
[Server.tout ] Timeout
```

Concern Coverage

- Complexity to implement a crosscutting concern
- Scattering and Tangling

Concern Coverage

- Complexity to implement a crosscutting concern
- Scattering and Tangling

		Scattering		Tangling		
#	**Concern**	**Transitions**	**Interactions**	**OT**	**OI**	**OC**
1	Logging	10	0	5	0	2, 3
2	Authentication	2	1	2	1	1, 3, 4
3	Congestion	5	0	4	0	1, 2
4	Fault Tolerance	0	3	0	1	2
	`Network`	12	5			

OT: Overlapping Transitions | OI: Overlapping Interactions | OC: Overlapping Concerns

Applications to RV

- Runtime Verification is a crosscutting concern $\rightarrow$ use `AOP-BIP` for RV

Applications to RV

- Runtime Verification is a crosscutting concern $\rightarrow$ use `AOP-BIP` for RV
  1. Use pointcuts to describe joinpoints that are of interest (generate events)
  2. Use advice functions to process events and also output trace (for offline)
  3. Use inter-type to store monitor state

Applications to RV

- Runtime Verification is a crosscutting concern $\rightarrow$ use `AOP-BIP` for RV
  1. Use pointcuts to describe joinpoints that are of interest (generate events)
  2. Use advice functions to process events and also output trace (for offline)
  3. Use inter-type to store monitor state
- Considerations:
  - Simpler to describe transformations
  - Not as expressive as RV-BIP
    (improves with extending pointcuts, advices and composition)

RV Example

- Data Robot: modules consisting of services
- Examples of existing tools `RV – BIP` [Falcone et al., 2015]
- Simplified simulation:
  1. `clock` : components providing the global clock (`getTime`)
  2. `poster`: interface to exchange data with other services
  3. `reader`: reads data of a poster (`read`)
  4. `writer`: writes data to a poster (`writev` → `finishWrite`)
- Data exchanged is: `time` and `taskid`

RV Example - Monitors

```
Aspect MonMutex (poster) {
  data int c = 0;

  portExecute(writev)
  do {checkConcurrent(c); c++;} {}

  portExecute(finishWrite)
  do {}{c--;}
}

Aspect MonFresh {
  ports(r:reader.read p:poster.read c:clck.getTime)
  do  { checkFresh(c.x, p.x); } {}
}

Aspect MonOrder {
  data int lastTask = 0

  ports(p:poster.writev) writePortVars(p.x)
  do {} { checkOrder(p.x, lastTask); lastTask = p.x; }
}
```

```
void checkConcurrent(int c) {
  if(c > 0)
    //Print violation
}
void checkFresh(int clock, int poster) {
  if(clock - poster > 2)
    //Print violation
}
void checkOrder(int task, int lastTask) {
  if (task != lastTask + 1)
    //Print violation
}
```

Conclusion

Summary

- Proposed a framework for crosscutting concerns in the BIP framework
- Focused on concerns at the level of the coordination between components
- Briefly overviewed local aspects and aspect composition
- Presented `AOP-BIP` and used it on a network example
  (More details in the technical report and ujf-aub.bitbucket.org/aop-bip)
- Showed how it can be used for runtime verification of CBSs

Conclusion

Summary

- Proposed a framework for crosscutting concerns in the BIP framework
- Focused on concerns at the level of the coordination between components
- Briefly overviewed local aspects and aspect composition
- Presented `AOP-BIP` and used it on a network example
  (More details in the technical report and ujf-aub.bitbucket.org/aop-bip)
- Showed how it can be used for runtime verification of CBSs

Future Work

1. Expand the expressiveness of both pointcuts and advices
2. Study in more detail aspect composition (strategies, designs)
3. Implement Model-to-Model transformation using Domain Specific Languages (DSLs) inspired form ATL and compare expressiveness.

Related Work

- AOP for CBS (not formal)
  1. [Pessemier et al., 2008] Defining extra *advice* interfaces and using separate components for aspects.
  2. [Duclos et al., 2002] Applying AOP to CBS by defining two languages
  3. [Lieberherr et al., 2003] Investigating aspects in modules (Java Packages)
  4. [David and Ledoux, 2006] Using AOP in Fractal to define adaptation policies

Related Work

- AOP for CBS (not formal)
  1. [Pessemier et al., 2008] Defining extra *advice* interfaces and using separate components for aspects.
  2. [Duclos et al., 2002] Applying AOP to CBS by defining two languages
  3. [Lieberherr et al., 2003] Investigating aspects in modules (Java Packages)
  4. [David and Ledoux, 2006] Using AOP in Fractal to define adaptation policies
- Formal Approaches (not as simple/expressive)
  1. [Katz, 2006, Djoko et al., 2012] Categorizing aspects and studying which properties are preserved for which classes of aspects
  2. [Altisen et al., 2006] AOP for reactive systems presented as Mealy automata
  3. [Dihego and Sampaio, 2015] AOP for BRIC component model presented as Communicating Sequential Processes (CSP)

Language

```
aspect    : pointcuts 'do' advice;
pointcuts : (pointcut)+ ;
pointcut  : pctype '(' IDENTIFIER ')';

pctype    : 'atLocation'
          | 'readVarGuard'
          | 'readVarFunc'
          | 'write'
          | 'portEnabled'
          | 'portExecute'
          ;

advice    : (before) (after) (resetlocs)? ;
before    : '{' actions '}';
after     : '{' actions '}';
resetlocs : '{' (rlocpair (',' rlocpair)*)? };
rlocpair  : '(' IDENTIFIER ',' expression  );

gaspect   : gpoint 'do' before after
          ;
gpoint    : 'ports' '(' (portspec)+ ')' (gread)? (gwrite)?
          | 'ports' '(' (portspec)+ ')' (gwrite)? (gread)?
          ;

gwrite    : 'writePortVars' '(' port_var+ ')';
gread     : 'readPortVars'  '(' port_var+ ')';

portspec  : IDENTIFIER ':' port_name ;
port_name : IDENTIFIER '.' IDENTIFIER;
port_var  : IDENTIFIER '.' IDENTIFIER;
```

Language

```
file      : (CODE)? (container)+  ;

container : 'Aspect' IDENTIFIER '{' intertype (gaspect)+  '}'
          | 'Aspect' IDENTIFIER '(' IDENTIFIER ')'
            '{' intertype (aspect)+ '}'
          ;
CODE      : '{#' .*? '#}';
intertype : (intertypedef)*
              ;
intertypedef
          : 'data' IDENTIFIER IDENTIFIER
          | 'data' IDENTIFIER IDENTIFIER '=' literal_expression
          ;
```

Altisen, K., Maraninchi, F., and Stauch, D. (2006).
Aspect-oriented programming for reactive systems: Larissa, a proposal in the synchronous framework.
Sci. Comput. Program., 63(3):297–320.

Bonakdarpour, B., Bozga, M., and Gößler, G. (2012).
A theory of fault recovery for component-based models.
In Richa, A. W. and Scheideler, C., editors, *Stabilization, Safety, and Security of Distributed Systems - 14th Int. Symposium, SSS 2012, Toronto, Canada, October 1-4, 2012. Proc.*, volume 7596 of *LNCS*, pages 314–328. Springer.

Czarnecki, K., Eisenecker, U. W., and Steyaert, P. (1997).
Beyond objects: Generative programming.
In *The 23rd Int. Conf. On Software Engineering*, pages 5–14.

David, P. and Ledoux, T. (2006).
An aspect-oriented approach for developing self-adaptive Fractal components.

In *Proc. of 5th Int. Symposium on Software Composition*, volume 4089 of *LNCS*, pages 82–97.


Dihego, J. and Sampaio, A. (2015).
Aspect-oriented development of trustworthy component-based systems.
In *Proc. of Theoretical Aspects of Computing*, volume 9399 of *LNCS*, pages 425–444.


Djoko, S. D., Douence, R., and Fradet, P. (2012).
Aspects preserving properties.
Sci. Comput. Program., 77(3):393–422.


Duclos, F., Estublier, J., and Morat, P. (2002).
Describing and using non functional aspects in component based applications.

In *AOSD*, pages 65–75.


Falcone, Y., Jaber, M., Nguyen, T., Bozga, M., and Bensalem, S. (2015).
Runtime verification of component-based systems in the BIP framework with formally-proved sound and complete instrumentation.
Software and System Modeling, 14(1):173–199.


Katz, S. (2006).
Aspect categories and classes of temporal properties.
In *Transactions on aspect-oriented software development I*, pages 106–134. Springer.


Lieberherr, K. J., Lorenz, D. H., and Ovlinger, J. (2003).
Aspectual collaborations: Combining modules and aspects.
Comput. J., 46(5):542–565.


Pessemier, N., Seinturier, L., Duchien, L., and Coupaye, T. (2008).
A component-based and aspect-oriented model for software evolution.
IJCAT, 31(1/2):94–105.