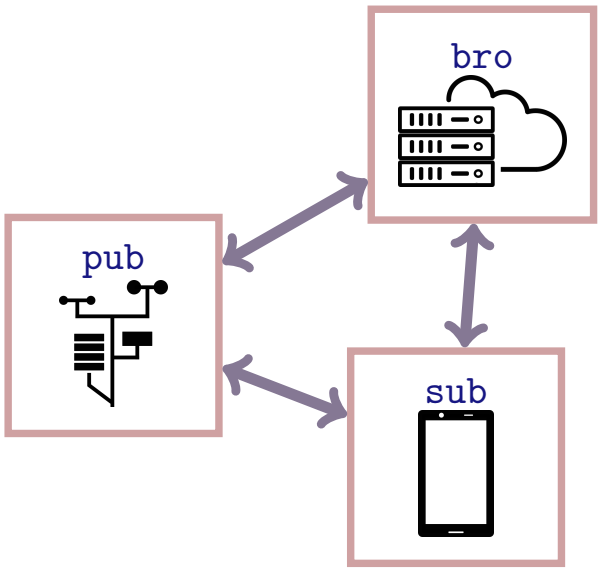
An operational semantics of interactions for verifying partially observed executions of distributed systems

Erwan Mahe

Université Paris-Saclay, CentraleSupélec

15th of July 2021

Distributed Systems

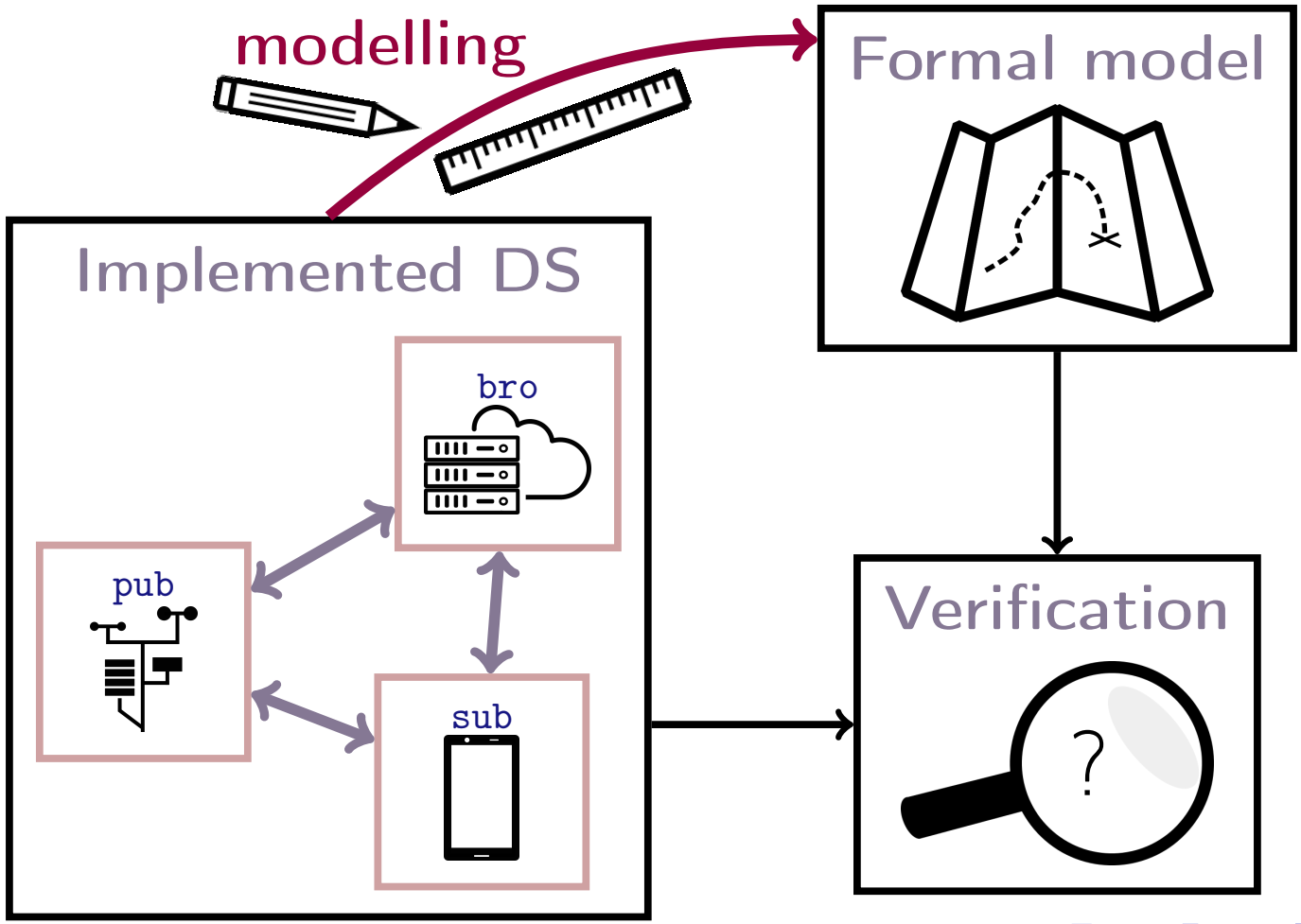


- distant machines (no global clock)
- asynchronous message passing

← MQTT example (publish/subscribe)

Formal analysis for increasing the quality of DSs

- allows verifying intractably (for humans) complex properties
- automatizable

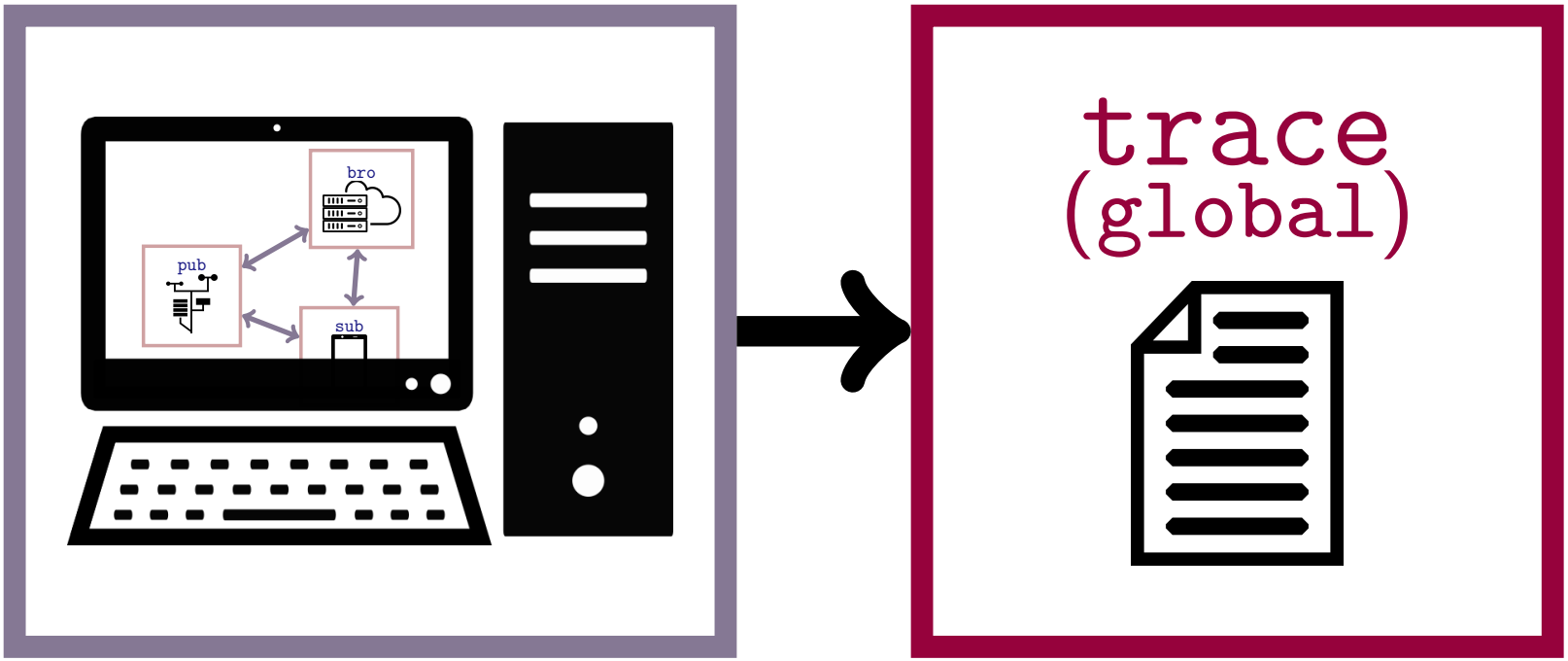


What can be verified ? **Executions**

- observing & logging events during execution
- different architectures
  (`Francalanza, Perez & Sanchez` 2018)

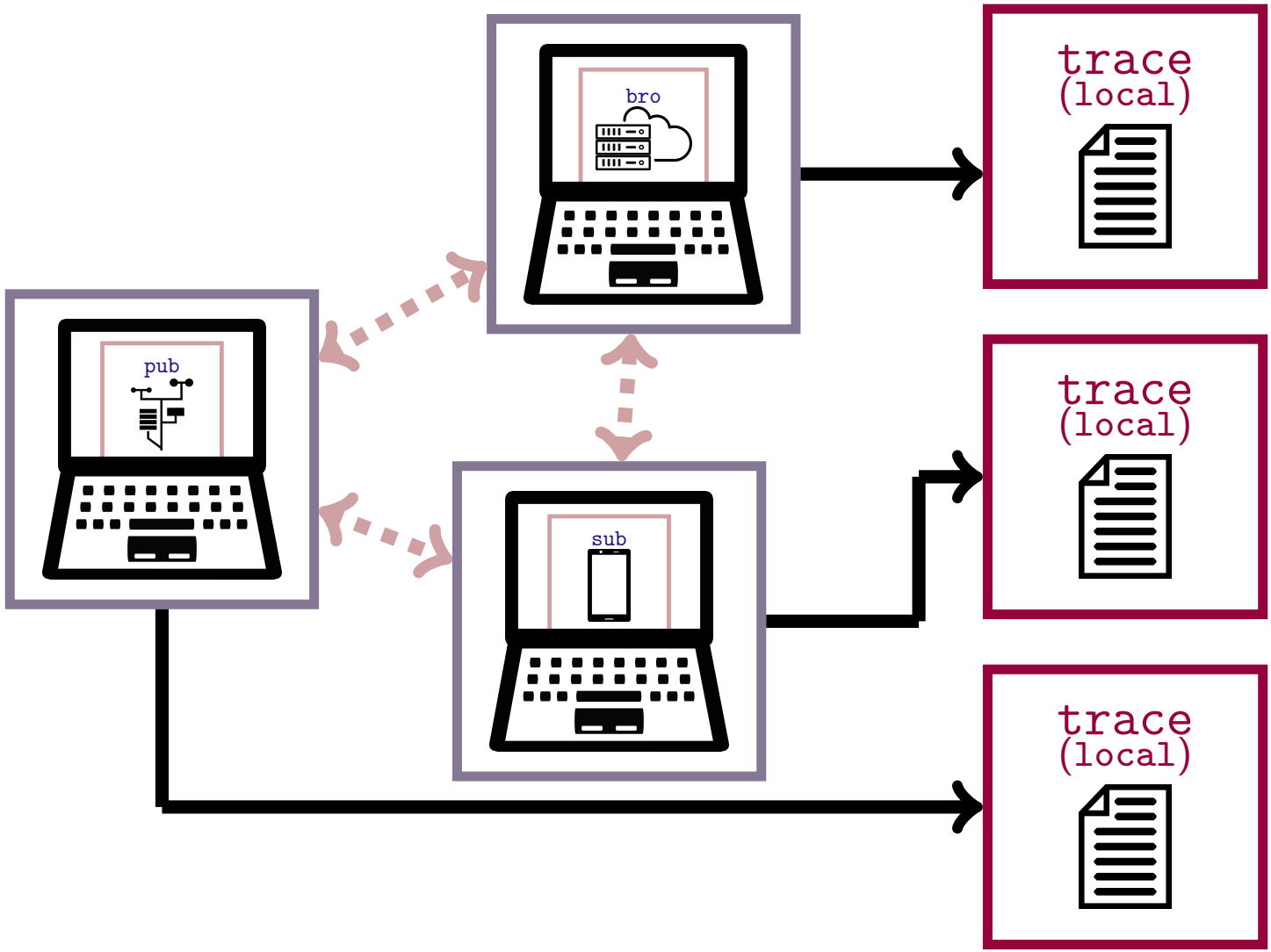
Different **architectures** for observation

- centralized case (single machine):

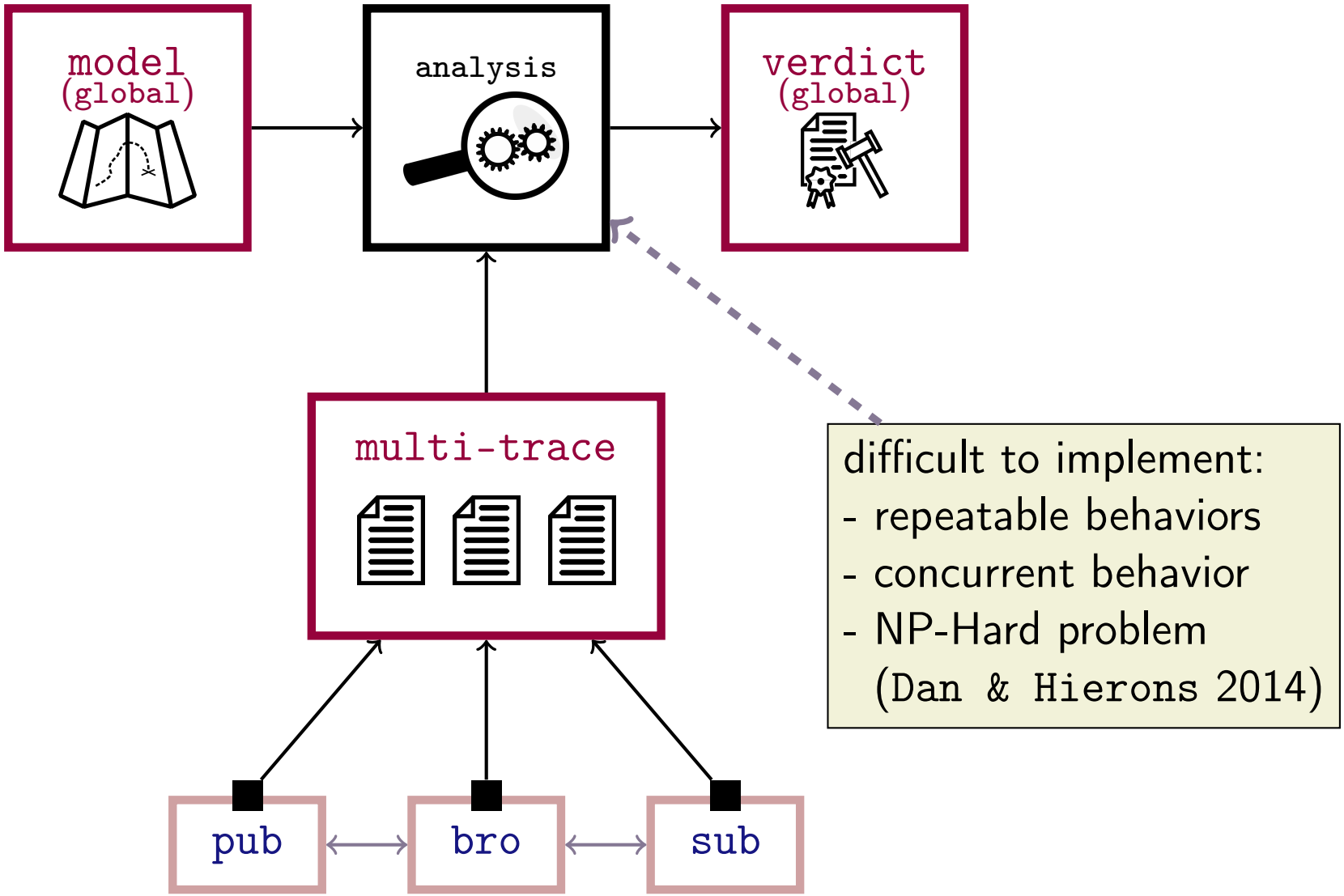


Different **architectures** for observation

- distributed case (distant machines):



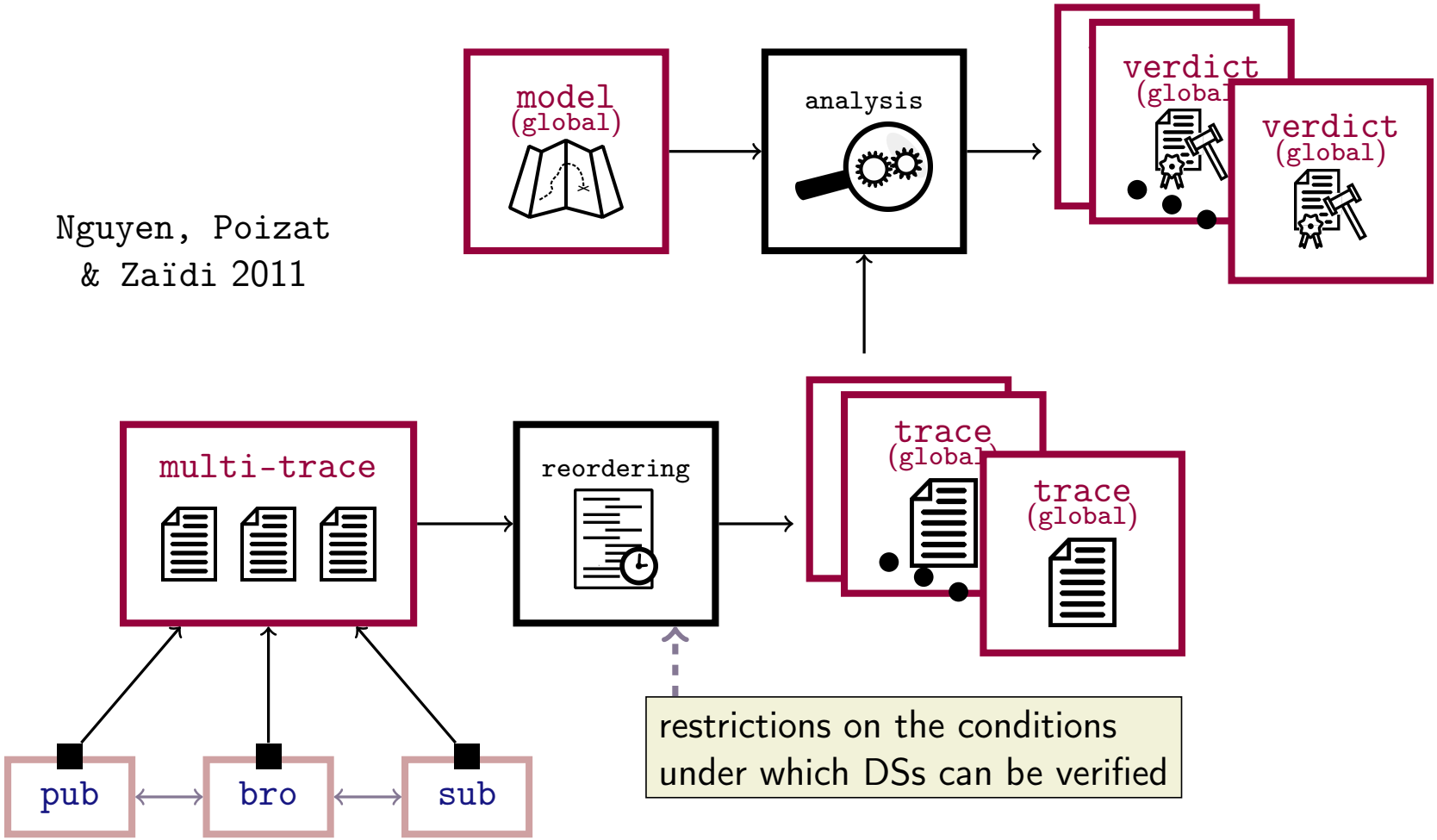
Offline analysis (Runtime Verification) from multi-traces



difficult to implement:
- repeatable behaviors
- concurrent behavior
- NP-Hard problem
  (Dan & Hierons 2014)

In practice: RV with reordered execution traces

Nguyen, Poizat & Zaïdi 2011

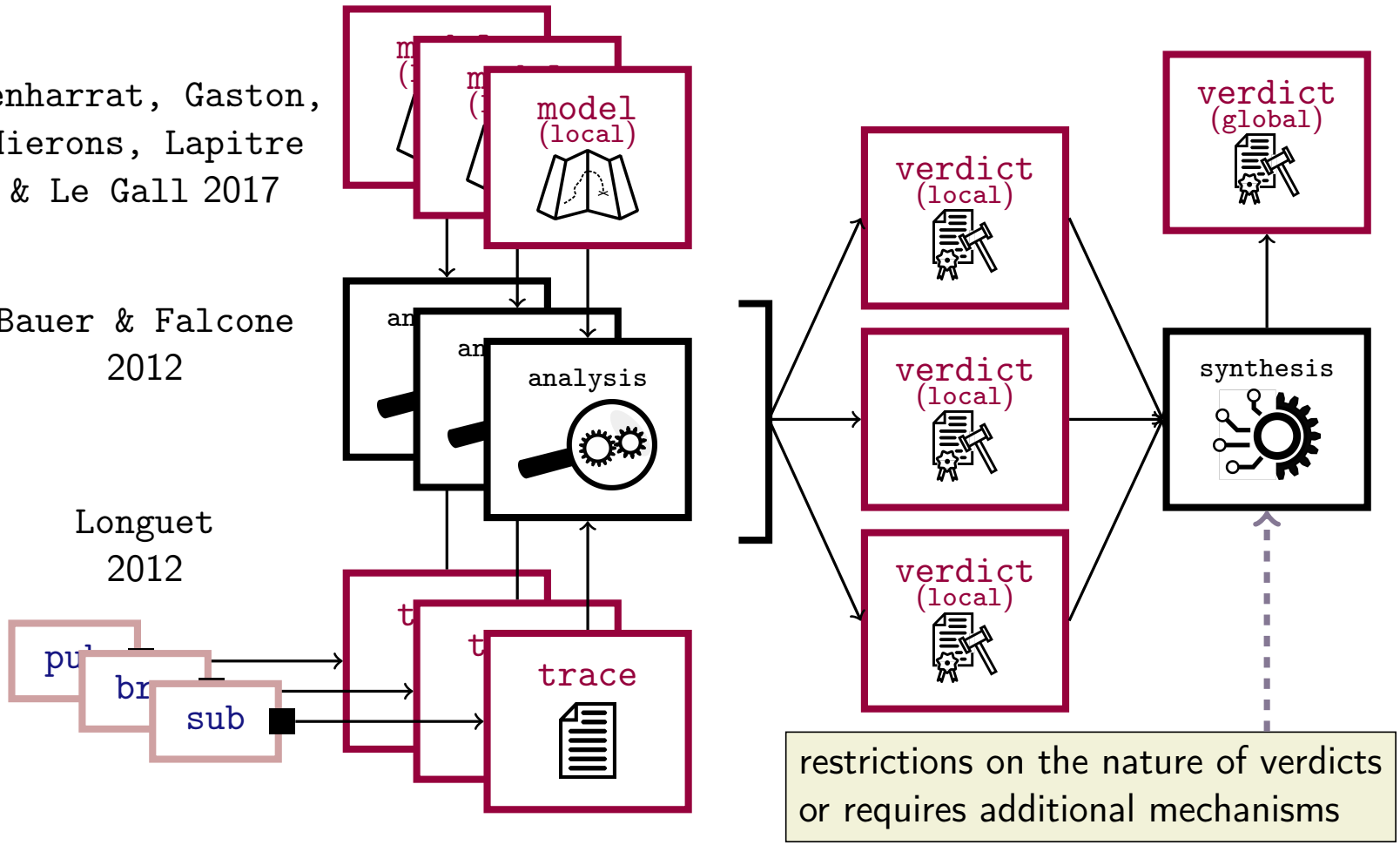


In practice: RV using local analyses

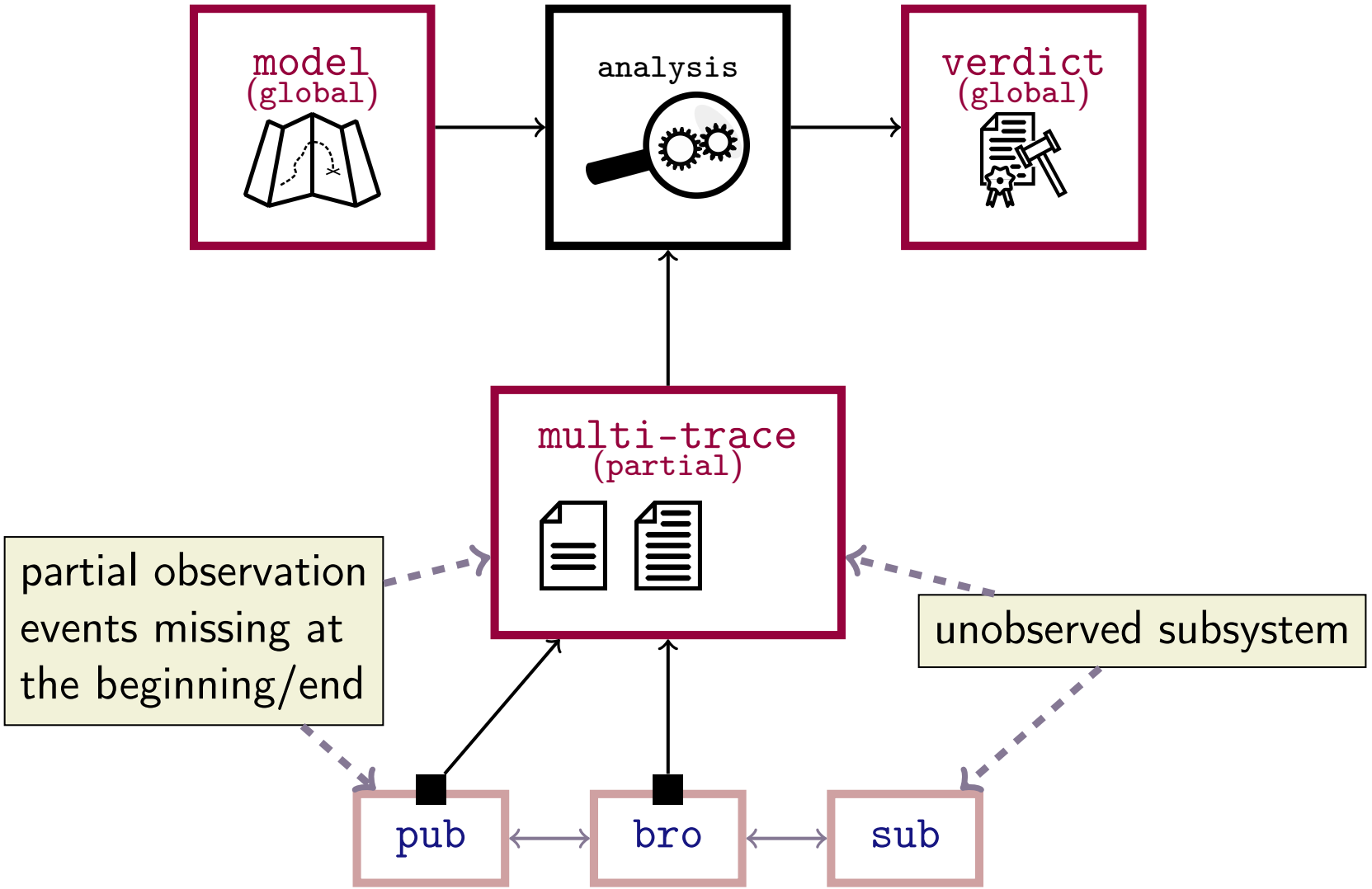
Benharrat, Gaston, Hierons, Lapitre & Le Gall 2017

Bauer & Falcone 2012

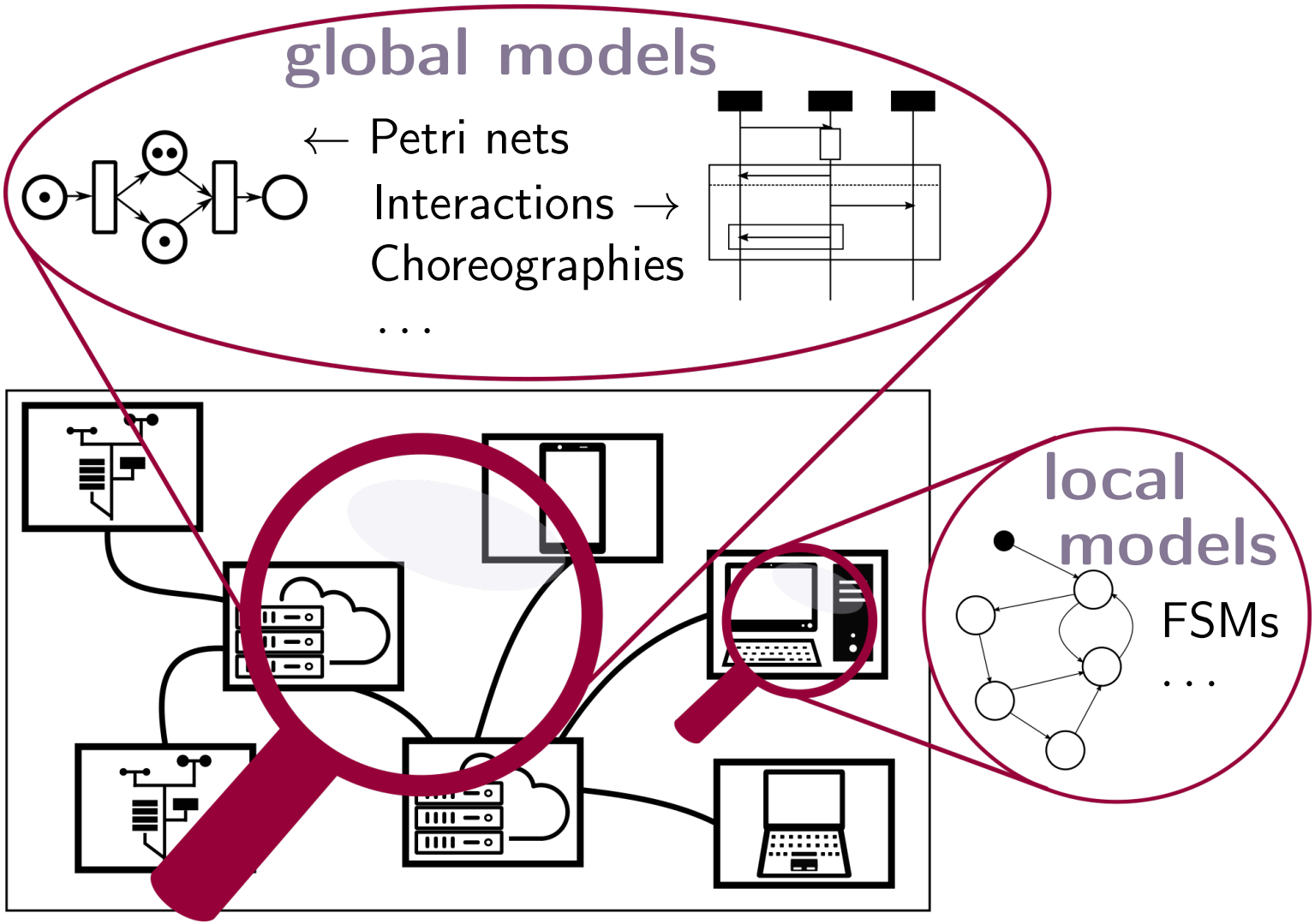
Longuet 2012



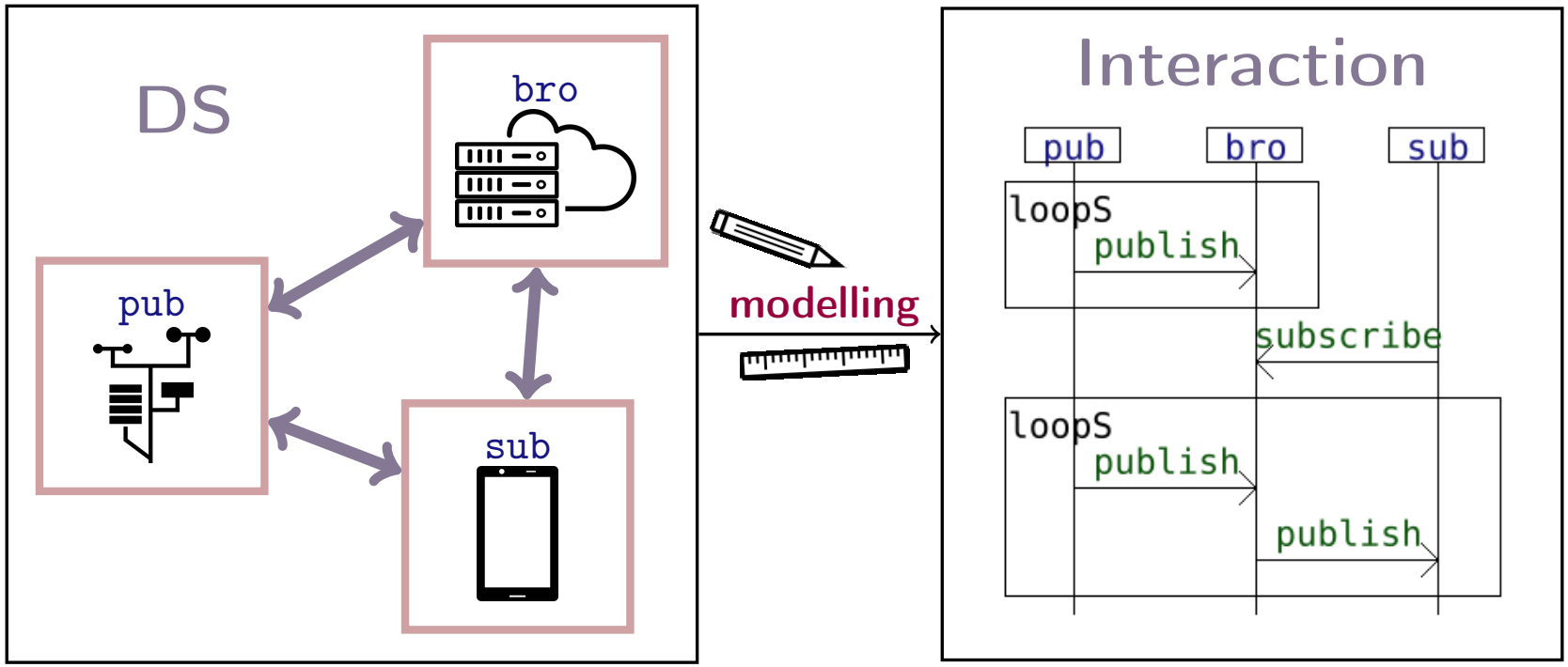
Our goal: **RV** from partial multi-traces without restrictions



How can **RV** be performed ? Various **formalisms**



Interaction models



- focuses on the flow of communications (global model)
- intuitive graphical representation
- nuanced specifications via dedicated operators

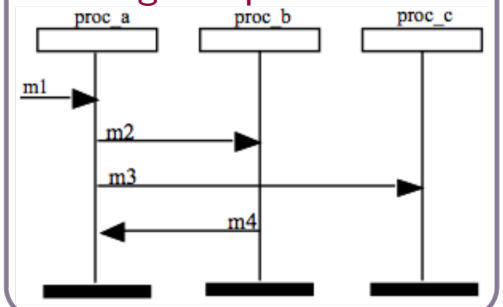
A short history of interaction languages

1993 MSC — ITU

1997 UML1.x — OMG OBJECT MANAGEMENT GROUP

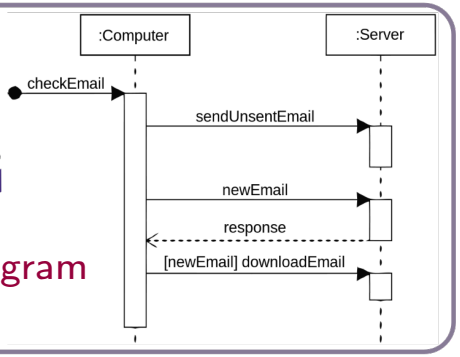
2007 UML2.x

Message Sequence Chart

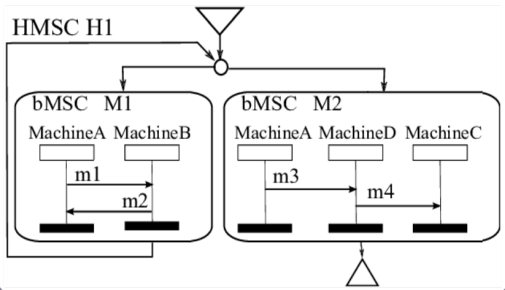


Sequence Diagram

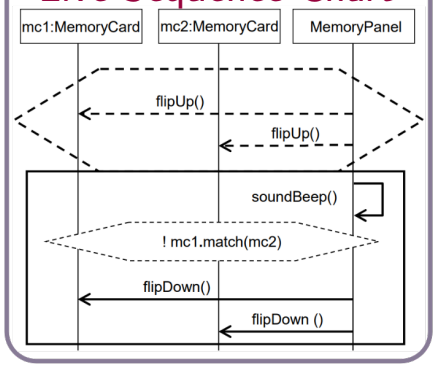
UNIFIED MODELING LANGUAGE™



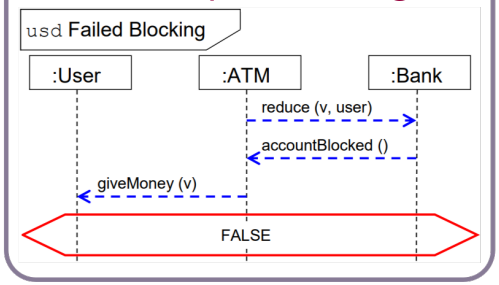
High-level MSC



Live Sequence Chart

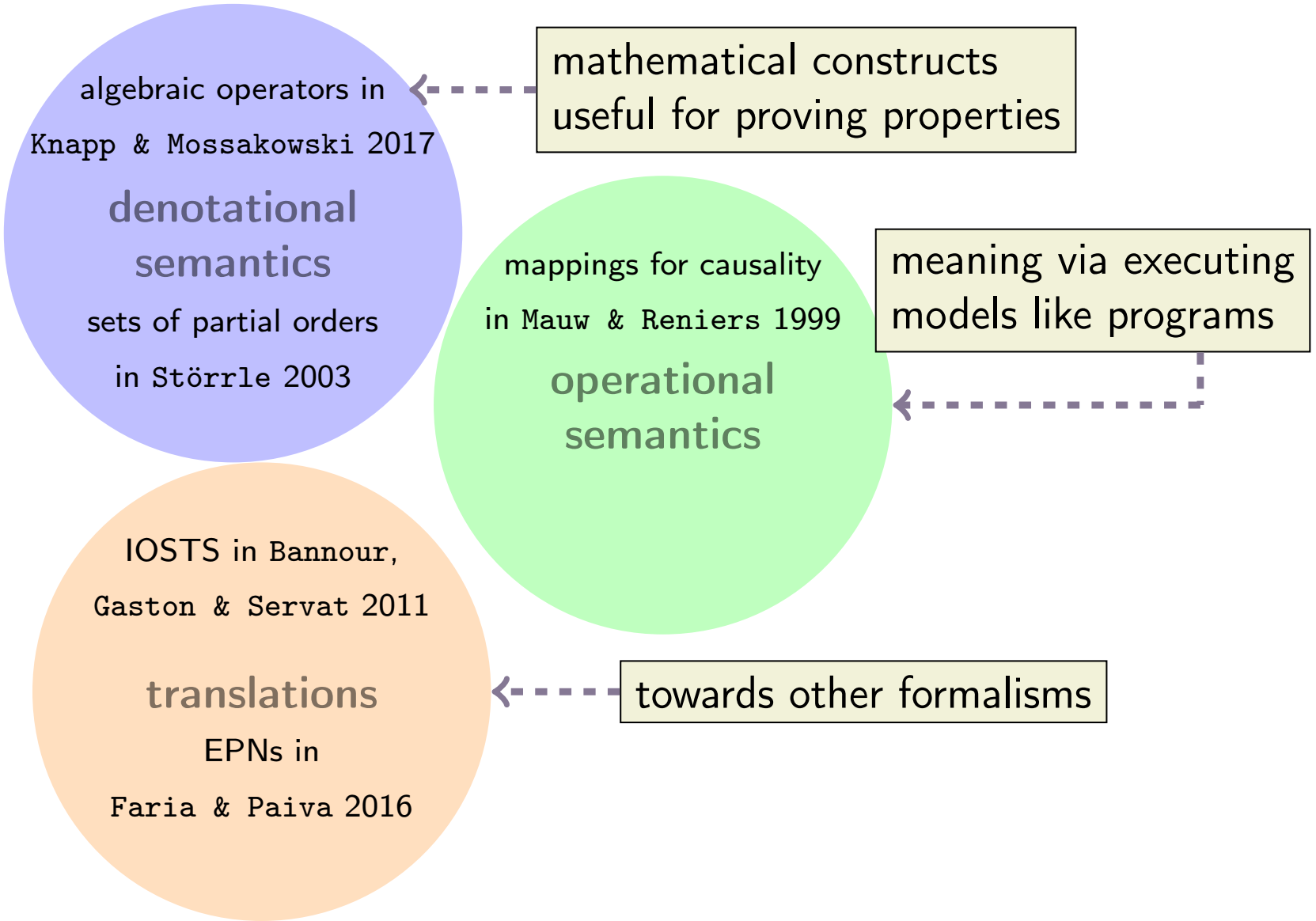


Modal Sequence Diagram



Mauw & Reniers 1997 — Damm & Harel 2001 — Harel & Maoz 2008

On the semantics of interactions

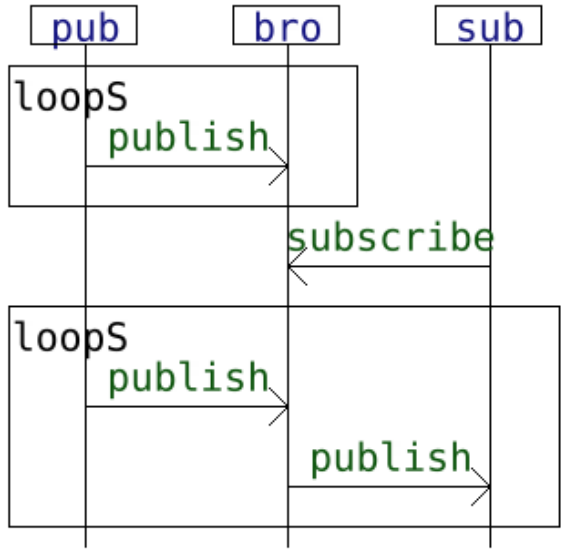


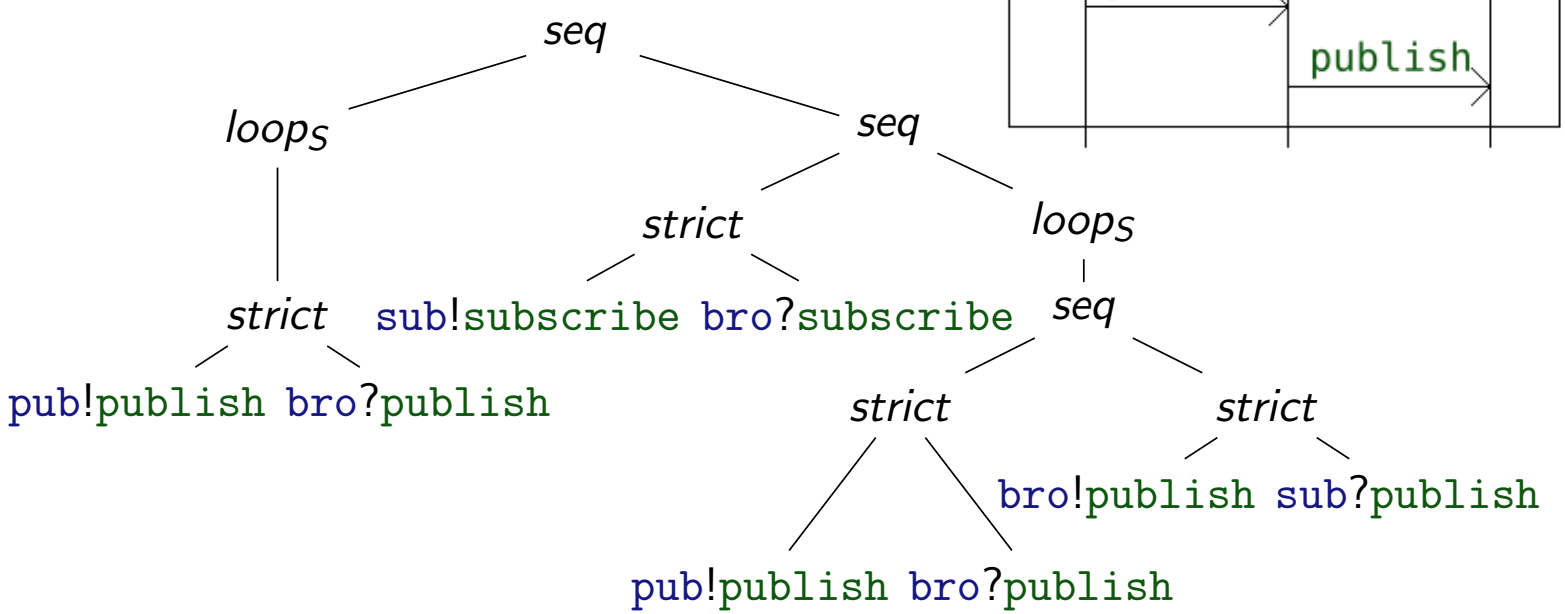
1. Interaction Language & denotational semantics
2. Operational semantics
3. Execution semantics
4. Multi-trace analysis algorithms
5. Implementation

Interactions as terms of a term algebra

$$\mathbb{A} = L \times \{!, ?\} \times M$$

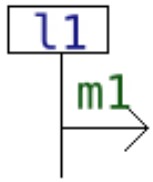
$$\left(\mathbb{I}, \left\{ \begin{array}{llll} \varnothing, & a \in \mathbb{A}, & & \\ alt, & strict, & seq, & par \\ loop_X, & loop_S, & loop_P & \end{array} \right\} \right)$$



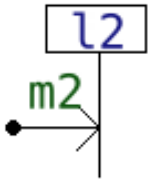


Empty interaction & Atomic emissions & receptions

$\varnothing$ $\{\epsilon\}$

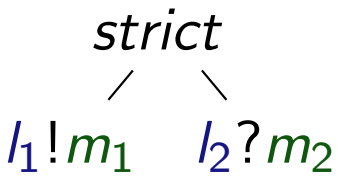
$l_1!m_1$ 

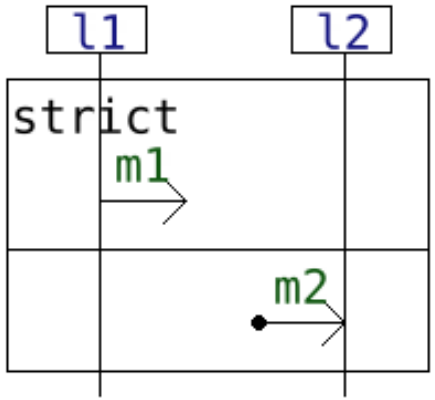
$\{l_1!m_1\}$

$l_2?m_2$ 

$\{l_2?m_2\}$

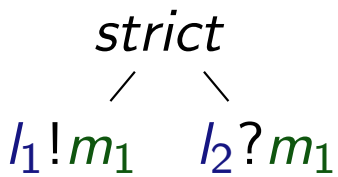
Strict Sequencing

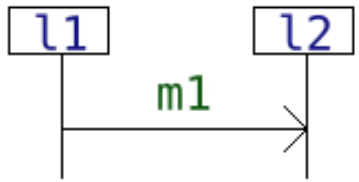




$$\{l_1!m_1\};\{l_2?m_2\} = \{l_1!m_1.l_2?m_2\}$$

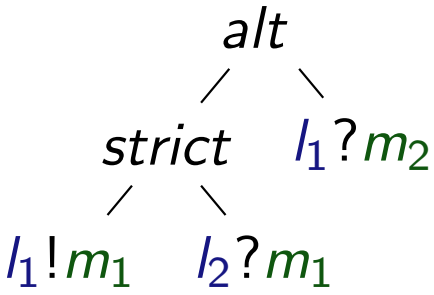
Strict Sequencing

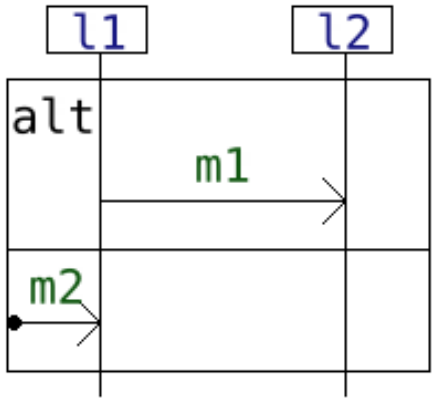




$$\{l_1!m_1\};\{l_2?m_1\} = \{l_1!m_1.l_2?m_1\}$$

Alternative

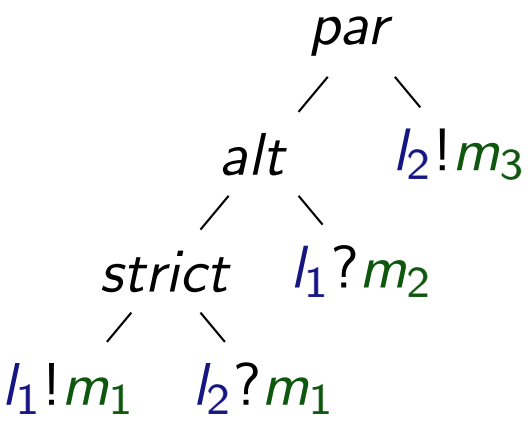




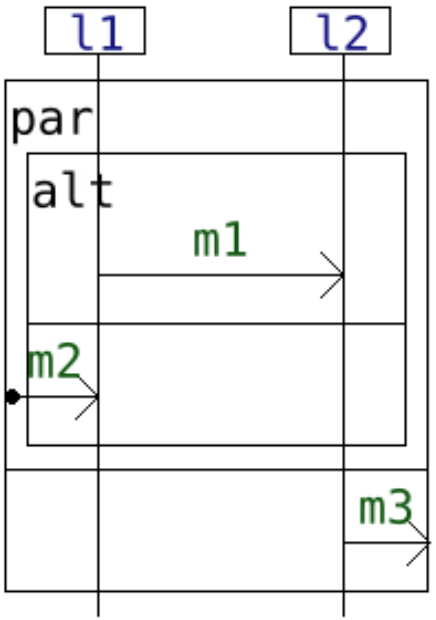
$$\{l_1!m_1.l_2?m_1\} \cup \{l_1?m_2\}$$

$$= \left\{ \begin{array}{l} l_1!m_1.l_2?m_1, \\ l_1?m_2 \end{array} \right\}$$

Interleaving

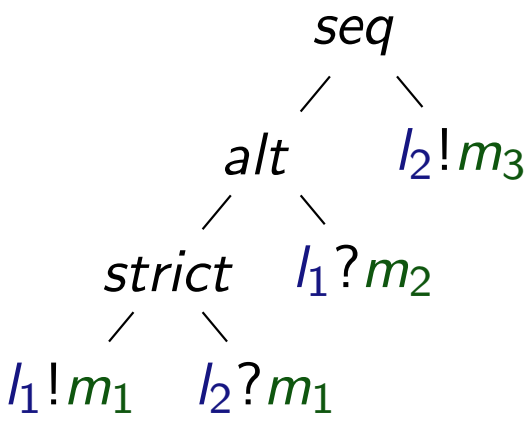


$$\left\{ \begin{array}{l} l_1!m_1.l_2?m_1, \\ l_1?m_2 \end{array} \right\} || \left\{ \; l_2!m_3 \; \right\}$$

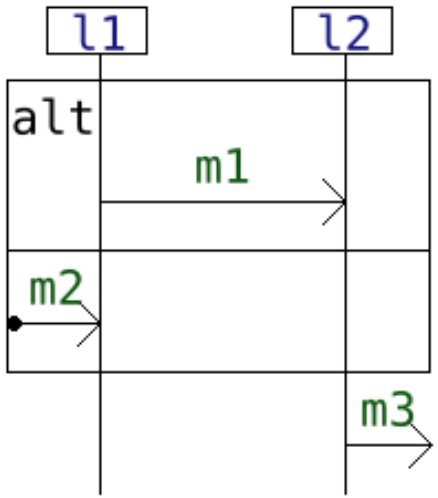


$$= \left\{ \begin{array}{l} l_2!m_3.l_1!m_1.l_2?m_1, \\ l_1!m_1.l_2!m_3.l_2?m_1, \\ l_1!m_1.l_2?m_1.l_2!m_3, \\ l_2!m_3.l_1?m_2, \\ l_1?m_2.l_2!m_3 \end{array} \right\}$$

Weak Sequencing



$$\left\{ \begin{array}{l} l_1!m_1.l_2?m_1, \\ l_1?m_2 \end{array} \right\} ;_{\times} \left\{ \; l_2!m_3 \; \right\}$$

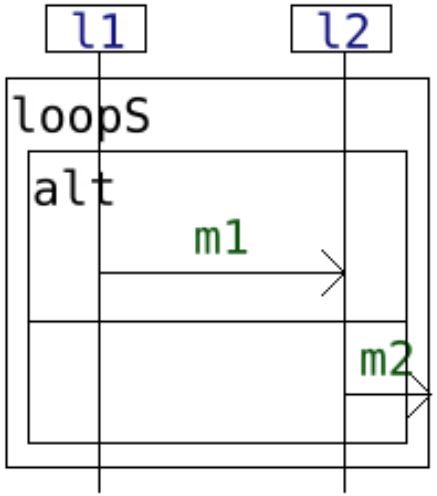


$$= \left\{ \begin{array}{l} l_1!m_1.l_2?m_1.l_2!m_3, \\ l_2!m_3.l_1?m_2, \\ l_1?m_2.l_2!m_3 \end{array} \right\}$$

Repetitions

loop operators (Kleene closures)

name	repetition with	operator
$loop_X$	*strict*	;*
$loop_S$	*seq*	;⁎*
$loop_P$	*par*	||*

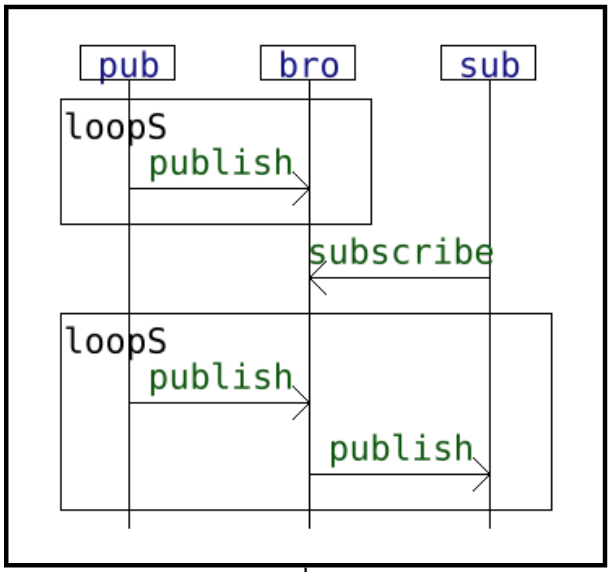


Syntax & denotational trace semantics

$$\left(\mathbb{I}, \left\{ \begin{array}{l} \varnothing,\ a \in \mathbb{A}, \\ alt,\ strict,\ seq,\ par \\ loop_X,\ loop_S,\ loop_P \end{array} \right\}\right)$$

homomorphism σ

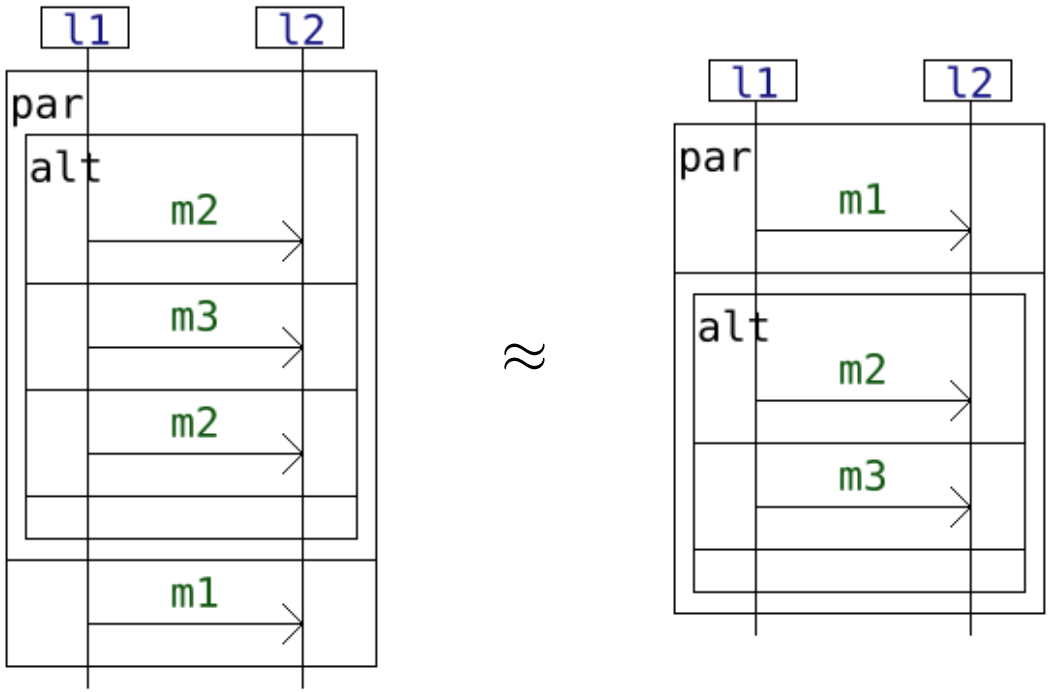
$$\left(\mathcal{P}(\mathbb{T}), \left\{ \begin{array}{l} \{\epsilon\},\ \{a\}, \\ \cup,\ ;,\ ;_{\Join},\ ||, \\ ;^*,\ ;_{\Join}{}^*,\ ||^* \end{array} \right\}\right)$$



$$\left\{ \begin{array}{l} \texttt{sub!subscribe.bro?subscribe}, \\ \cdots, \\ \texttt{pub!publish}.\cdots, \\ \cdots \end{array} \right\}$$

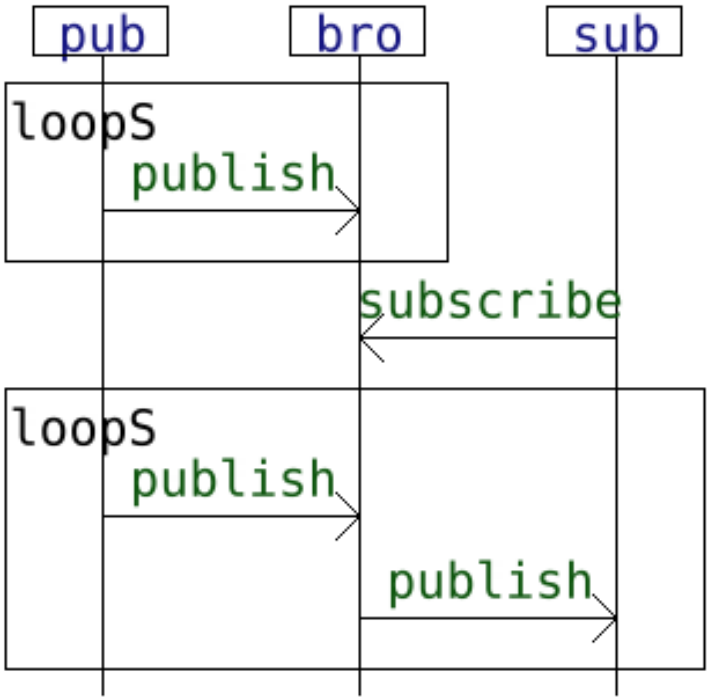
Equivalent terms & normal forms

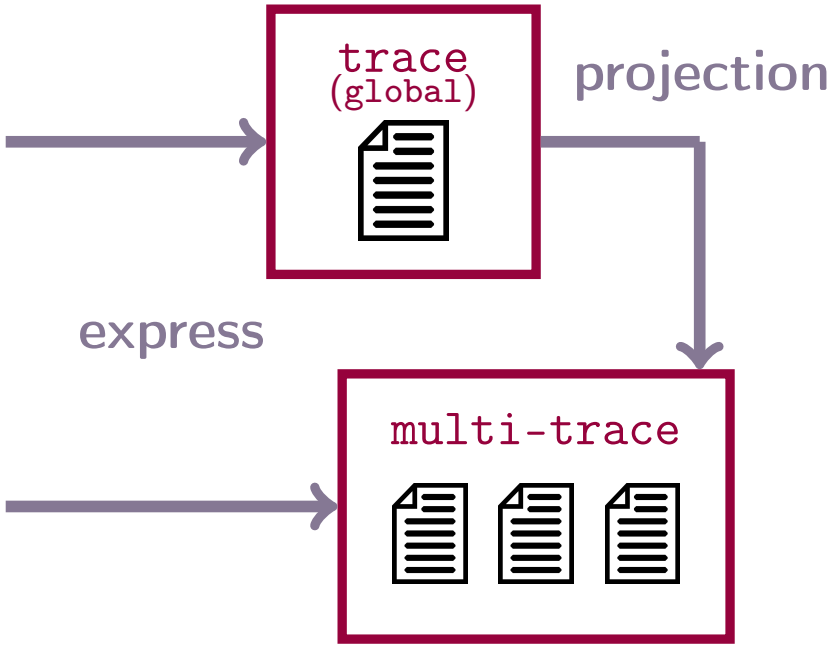
algebraic property of operator	$\rightarrow$	equation on interaction terms
$T_1 \| \| T_2 = T_2 \| \| T_1$	$\rightarrow$	$par(i_1, i_2) \approx par(i_2, i_1)$
$T ;_{\divideontimes} \{\epsilon\} = T$	$\rightarrow$	$seq(i, \varnothing) \approx i$



Computing **normal forms** via **term rewriting**

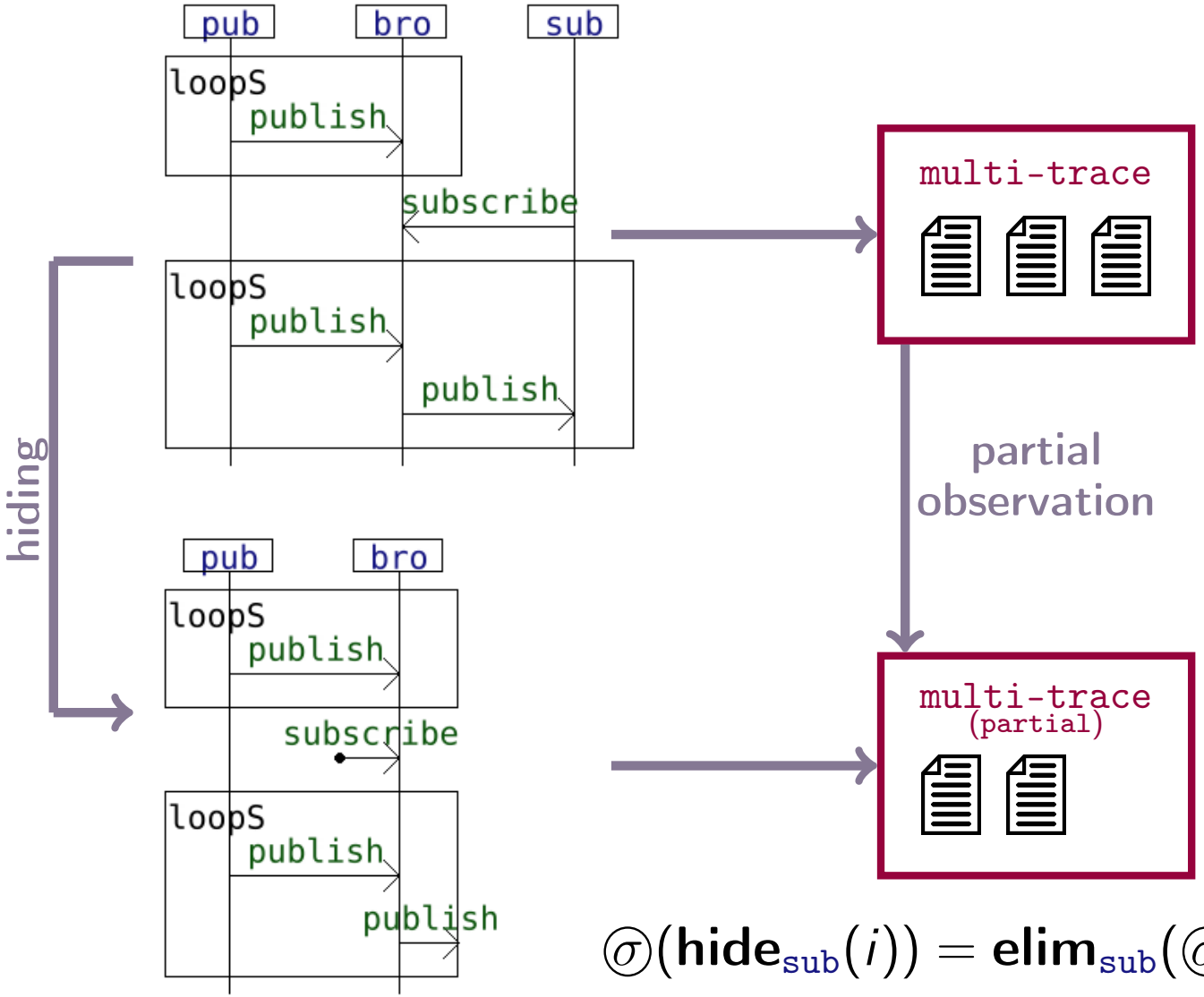
Multi-trace semantics





$$\circledcirc(i) = \mathbf{proj}(\sigma(i))$$

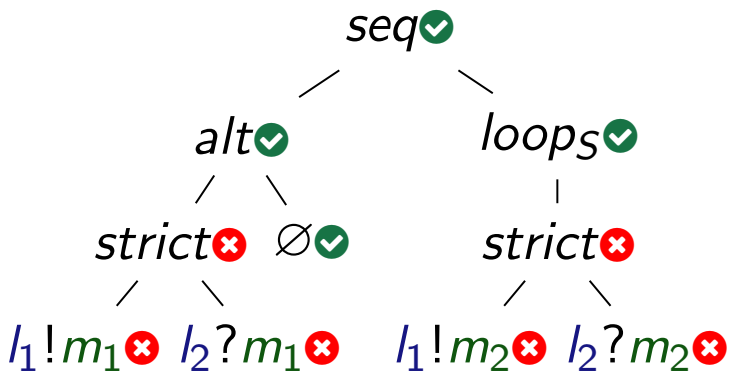
Hiding

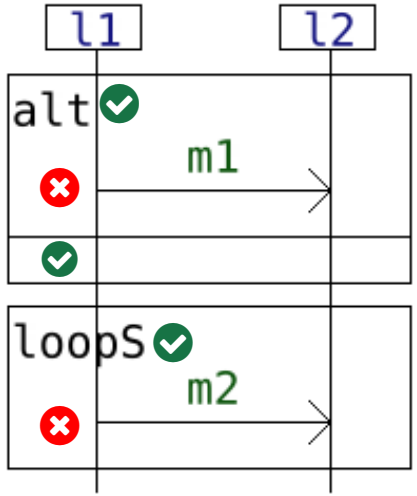


$$\circledcirc(\mathbf{hide}_{\mathtt{sub}}(i)) = \mathbf{elim}_{\mathtt{sub}}(\circledcirc(i))$$

Operational semantics

Termination predicate





Termination

$$\frac{}{\varnothing \downarrow} \qquad \frac{i_1 \downarrow}{alt(i_1, i_2) \downarrow} \qquad \frac{i_2 \downarrow}{alt(i_1, i_2) \downarrow}$$

$$\forall\ f \in \{strict, seq, par\} \qquad \frac{i_1 \downarrow \quad i_2 \downarrow}{f(i_1, i_2) \downarrow}$$

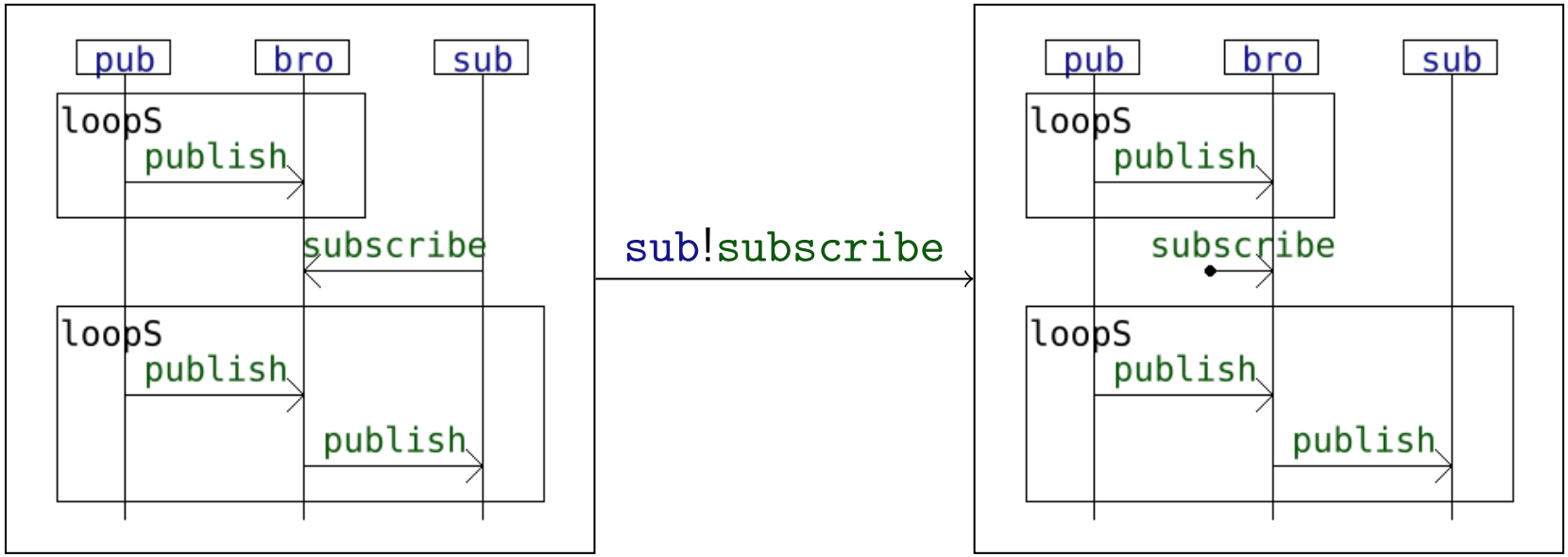
$$\forall\ k \in \{X, S, P\} \qquad \frac{}{loop_k(i_1) \downarrow}$$

Characterisation

For any $i \in \mathbb{I}$:

$$(i \downarrow) \Leftrightarrow (\epsilon \in \sigma(i))$$

Principle of the operational semantics



initial interaction

expression of an event

interaction specifying all continuations

rules & induction: process calculus, π-calculus, etc.

Rooda, van Beek & Baeten 2007

Despeyroux 2001

Rules of the operational semantics

$$\frac{}{a \xrightarrow{a} \varnothing} \qquad \frac{i_1 \xrightarrow{a} i_1'}{alt(i_1, i_2) \xrightarrow{a} i_1'} \qquad \frac{i_2 \xrightarrow{a} i_2'}{alt(i_1, i_2) \xrightarrow{a} i_2'}$$

$$\frac{i_1 \xrightarrow{a} i_1'}{par(i_1, i_2) \xrightarrow{a} par(i_1', i_2)} \qquad \frac{i_2 \xrightarrow{a} i_2'}{par(i_1, i_2) \xrightarrow{a} par(i_1, i_2')}$$

$$\frac{i_1 \xrightarrow{a} i_1'}{strict(i_1, i_2) \xrightarrow{a} strict(i_1', i_2)} \qquad \frac{i_2 \xrightarrow{a} i_2'}{strict(i_1, i_2) \xrightarrow{a} i_2'}\ i_1 \downarrow$$

$$\frac{i_1 \xrightarrow{a} i_1'}{seq(i_1, i_2) \xrightarrow{a} seq(i_1', i_2)} \qquad \frac{i_1 \overset{\theta(a)}{\nrightarrow} i_1' \qquad i_2 \xrightarrow{a} i_2'}{seq(i_1, i_2) \xrightarrow{a} seq(i_1', i_2')}$$

$$\frac{i_1 \xrightarrow{a} i_1'}{loop_X(i_1) \xrightarrow{a} strict(i_1', loop_X(i_1))} \qquad \frac{i_1 \xrightarrow{a} i_1'}{loop_P(i_1) \xrightarrow{a} par(i_1', loop_P(i_1))}$$

$$\frac{i_1 \xrightarrow{a} i_1' \qquad loop_S(i_1) \overset{\theta(a)}{\nrightarrow} i'}{loop_S(i_1) \xrightarrow{a} seq(i', seq(i_1', loop_S(i_1)))}$$

Rules of the operational semantics

Mauw & Reniers 1999, process algebra

$$\frac{}{a \xrightarrow{a} \varnothing} \qquad \frac{i_1 \xrightarrow{a} i_1'}{alt(i_1, i_2) \xrightarrow{a} i_1'} \qquad \frac{i_2 \xrightarrow{a} i_2'}{alt(i_1, i_2) \xrightarrow{a} i_2'}$$

$$\frac{i_1 \xrightarrow{a} i_1'}{par(i_1, i_2) \xrightarrow{a} par(i_1', i_2)} \qquad \frac{i_2 \xrightarrow{a} i_2'}{par(i_1, i_2) \xrightarrow{a} par(i_1, i_2')}$$

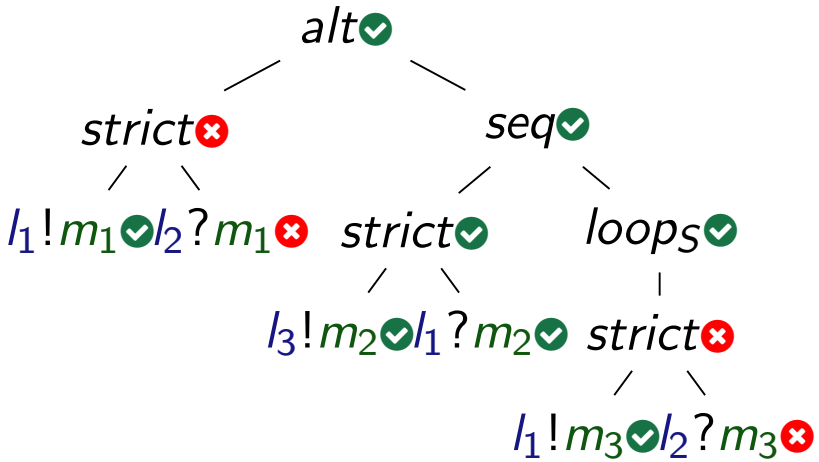
$$\frac{i_1 \xrightarrow{a} i_1'}{strict(i_1, i_2) \xrightarrow{a} strict(i_1', i_2)} \qquad \frac{i_2 \xrightarrow{a} i_2'}{strict(i_1, i_2) \xrightarrow{a} i_2'}\ i_1 \downarrow$$

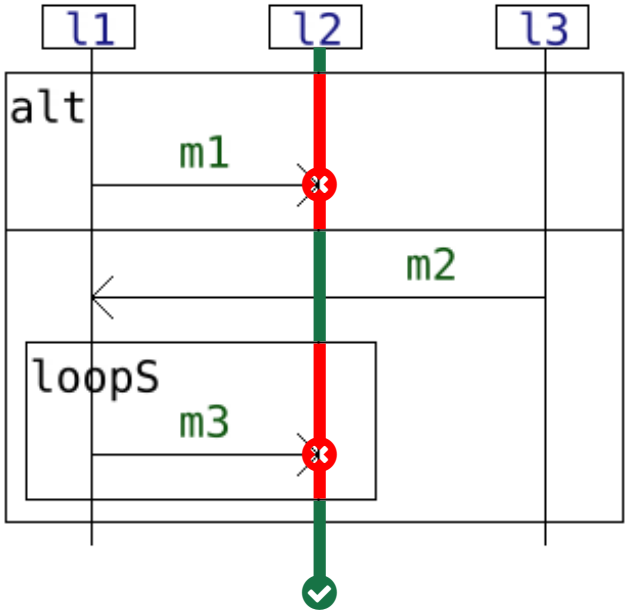
$$\frac{i_1 \xrightarrow{a} i_1'}{seq(i_1, i_2) \xrightarrow{a} seq(i_1', i_2)} \qquad \frac{i_1 \overset{\theta(a)}{\nrightarrow} i_1' \qquad i_2 \xrightarrow{a} i_2'}{seq(i_1, i_2) \xrightarrow{a} seq(i_1', i_2')}$$

$$\frac{i_1 \xrightarrow{a} i_1'}{loop_X(i_1) \xrightarrow{a} strict(i_1', loop_X(i_1))} \qquad \frac{i_1 \xrightarrow{a} i_1'}{loop_P(i_1) \xrightarrow{a} par(i_1', loop_P(i_1))}$$

$$\frac{i_1 \xrightarrow{a} i_1' \qquad loop_S(i_1) \overset{\theta(a)}{\nrightarrow} i'}{loop_S(i_1) \xrightarrow{a} seq(i', seq(i_1', loop_S(i_1)))}$$

Evasion predicate





Evasion

$$\frac{}{\varnothing \downarrow^{\divideontimes} l} \qquad \frac{\theta(a) \neq l}{a \downarrow^{\divideontimes} l}$$

$\cdots$

Characterisation

For any $i \in \mathbb{I}$ and $l \in L$:

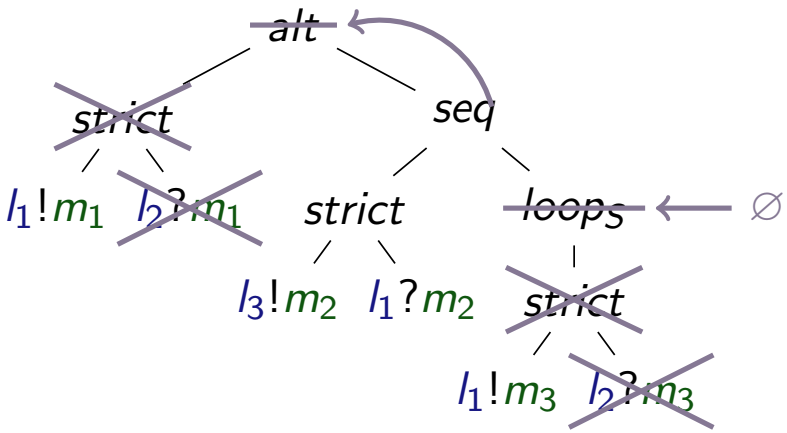
$$(i \downarrow^{\divideontimes} l) \Leftrightarrow (\exists\ t \in \sigma(i), \neg(t \divideontimes l))$$

Pruning relation

$$\frac{}{\varnothing \overset{l}{\nrightarrow} \varnothing} \qquad \frac{\theta(a) \neq l}{a \overset{l}{\nrightarrow} a} \qquad \frac{i_1 \overset{l}{\nrightarrow} i_1' \qquad i_2 \overset{l}{\nrightarrow} i_2'}{f(i_1, i_2) \overset{l}{\nrightarrow} f(i_1', i_2')}$$

$$\frac{i_1 \overset{l}{\nrightarrow} i_1' \qquad i_2 \overset{l}{\nrightarrow} i_2'}{alt(i_1, i_2) \overset{l}{\nrightarrow} alt(i_1', i_2')} \qquad \frac{i_1 \overset{l}{\nrightarrow} i_1'}{alt(i_1, i_2) \overset{l}{\nrightarrow} i_1'}\ i_2 \not\downarrow^{\nrightarrow} l \qquad \frac{i_2 \overset{l}{\nrightarrow} i_2'}{alt(i_1, i_2) \overset{l}{\nrightarrow} i_2'}\ i_1 \not\downarrow^{\nrightarrow} l$$

$$\frac{i_1 \overset{l}{\nrightarrow} i_1'}{loop_k(i_1) \overset{l}{\nrightarrow} loop_k(i_1')} \qquad \frac{}{loop_k(i_1) \overset{l}{\nrightarrow} \varnothing}\ i_1 \not\downarrow^{\nrightarrow} l$$

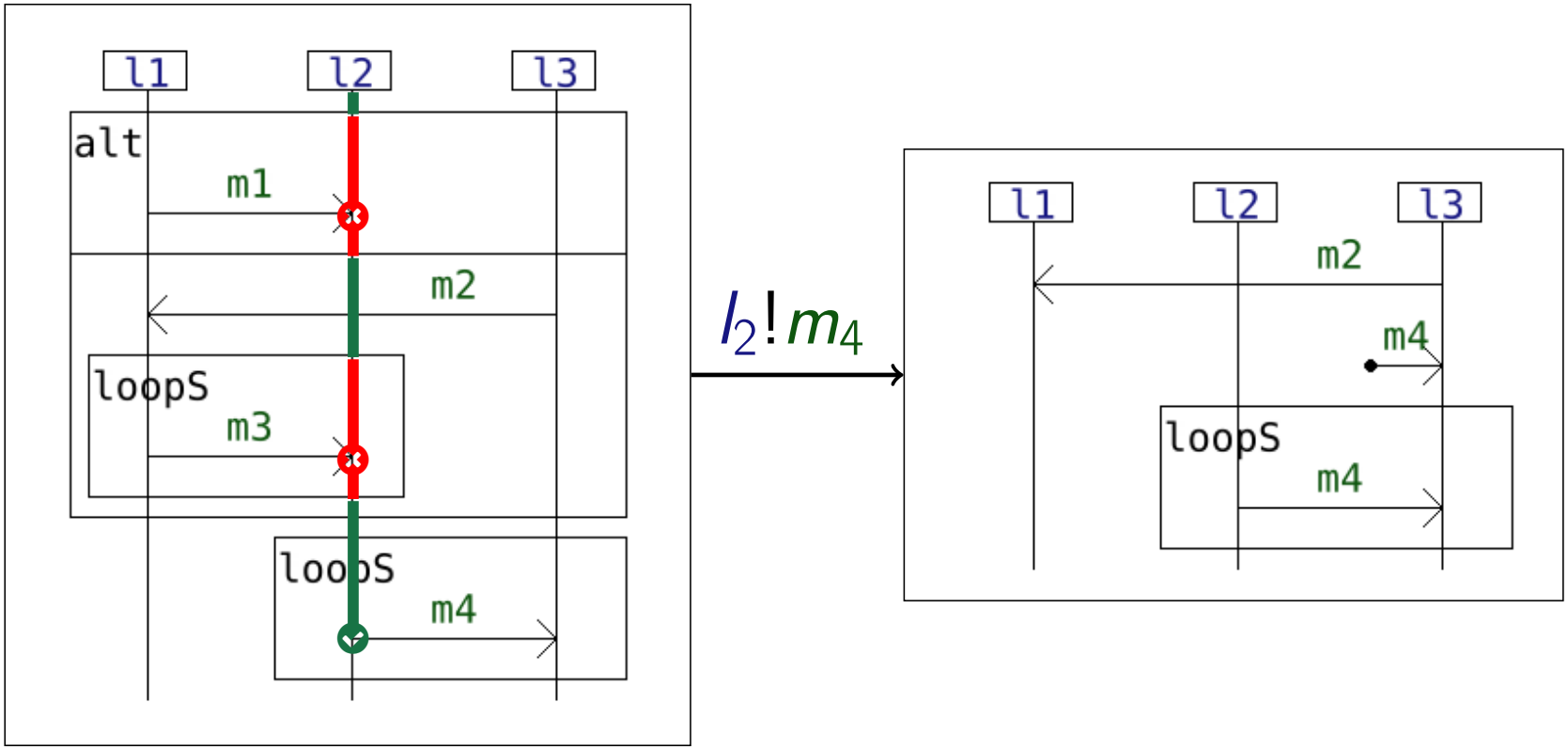


Characterisation

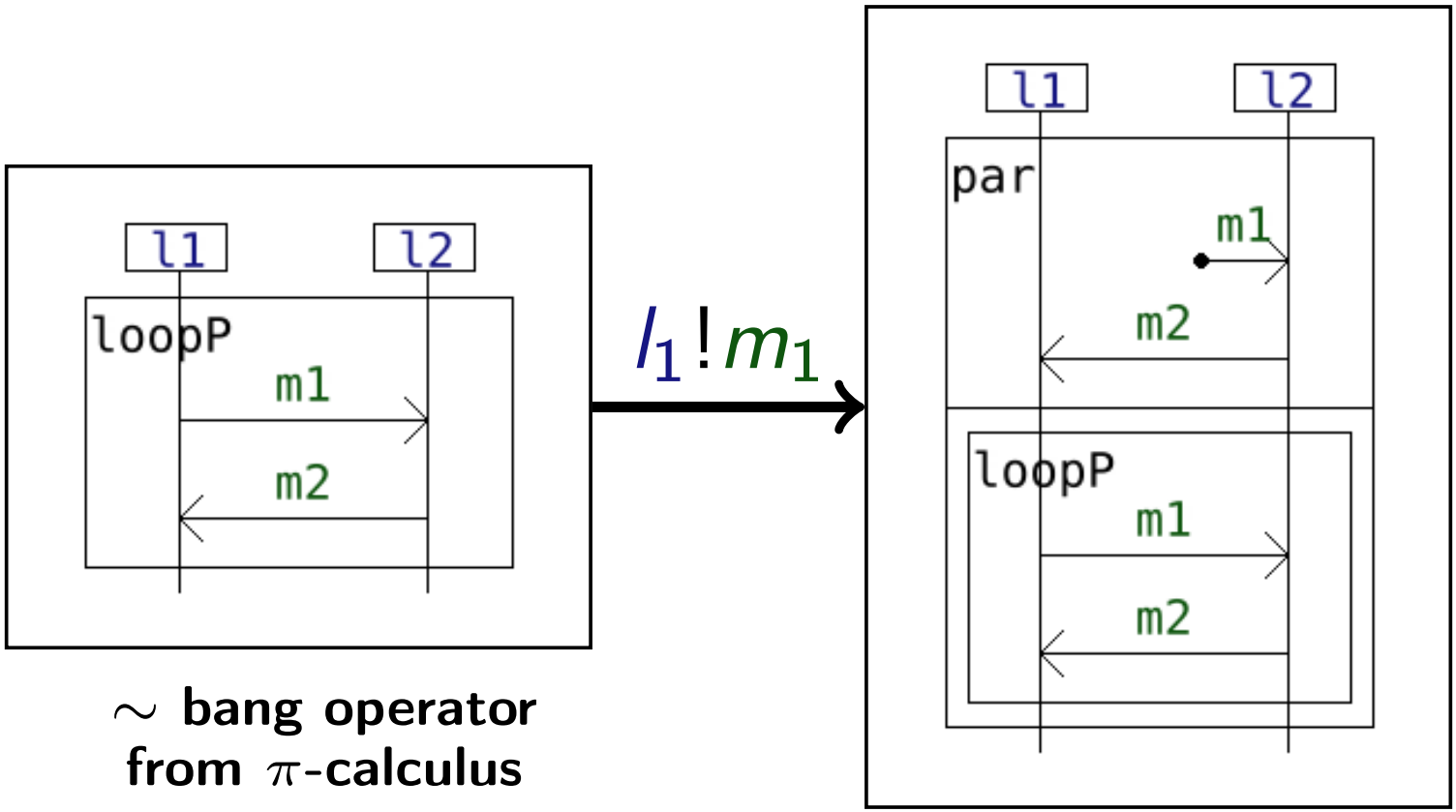
For any $i \in \mathbb{I}$ and $l \in L$:

$$(i \overset{l}{\nrightarrow} i') \Rightarrow \begin{pmatrix} \sigma(i') = \\ \{t \in \sigma(i) \mid \neg(t \nrightarrow l)\} \end{pmatrix}$$

Executing actions on the right of *seq*



Executing actions underneath a $loop_P$



$\sim$ **bang operator from π-calculus**

Operational semantics & equivalence

Operational semantics σ_o

We define $\sigma_o : \mathbb{I}_\Omega \to \mathcal{P}(\mathbb{T}_\Omega)$ by:

$$\frac{i\downarrow}{\epsilon \in \sigma_o(i)} \qquad \frac{t \in \sigma_o(i') \qquad i \xrightarrow{a} i'}{a.t \in \sigma_o(i)}$$

Equivalence of the σ_o and σ semantics

For any interaction $i \in \mathbb{I}_\Omega$:

$$\sigma_o(i) = \sigma(i)$$

Need for an algorithmization

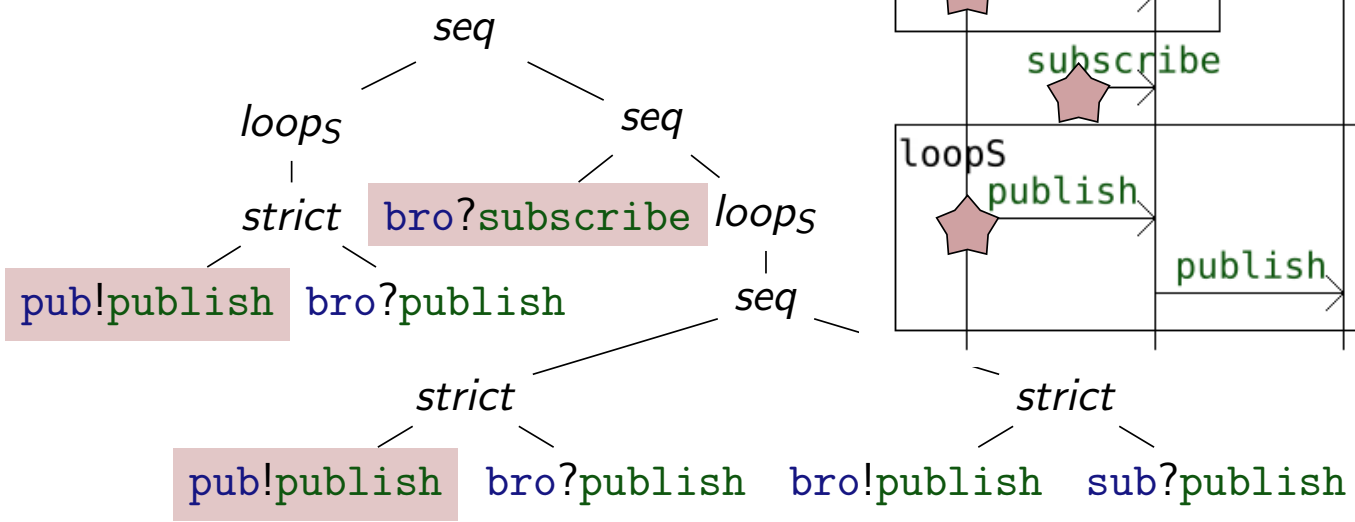
- problems:
  - prove that $i \xrightarrow{a} i'$
  - determine if i can express a and compute i'
- naive implementation of operational semantics:
  - computes all (a, i') .s.t. $i \xrightarrow{a} i'$
  - no term simplification $\rightarrow$ degradation of term at each step
- non-trivial implementation: "execution semantics"

Execution semantics

Frontier of execution

- $\texttt{frt}(i) =$ **match** i **with**

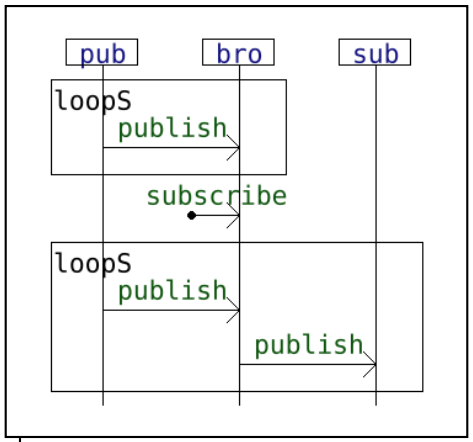
$\mid \varnothing$	$\rightarrow$	$\emptyset$
$\mid a \in \mathbb{A}_\Omega$	$\rightarrow$	$\{\epsilon\}$
$\mid \mathit{strict}(i_1, i_2)$	$\rightarrow$	$\begin{cases} 1.\texttt{frt}(i_1) \cup 2.\texttt{frt}(i_2) & \text{if } i_1 \downarrow \\ 1.\texttt{frt}(i_1) & \text{else} \end{cases}$
$\mid \mathit{seq}(i_1, i_2)$	$\rightarrow$	$1.\texttt{frt}(i_1) \cup \{p \mid p \in 2.\texttt{frt}(i_2)\ ,\ i_1 \downarrow^{\times} \theta(i_{\mid p})\}$
$\mid f(i_1, i_2)$	$\rightarrow$	$1.\texttt{frt}(i_1) \cup 2.\texttt{frt}(i_2)$ **for** $f \in \{\mathit{alt}, \mathit{par}\}$
$\mid \mathit{loop}_k(i_1)$	$\rightarrow$	$1.\texttt{frt}(i_1)$ **for** $k \in \{X, S, P\}$



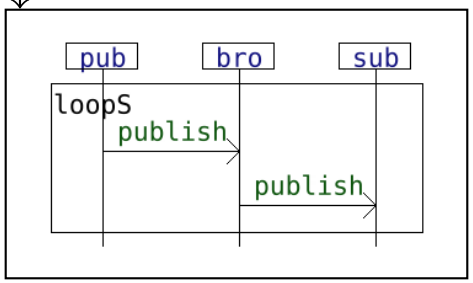
frontier

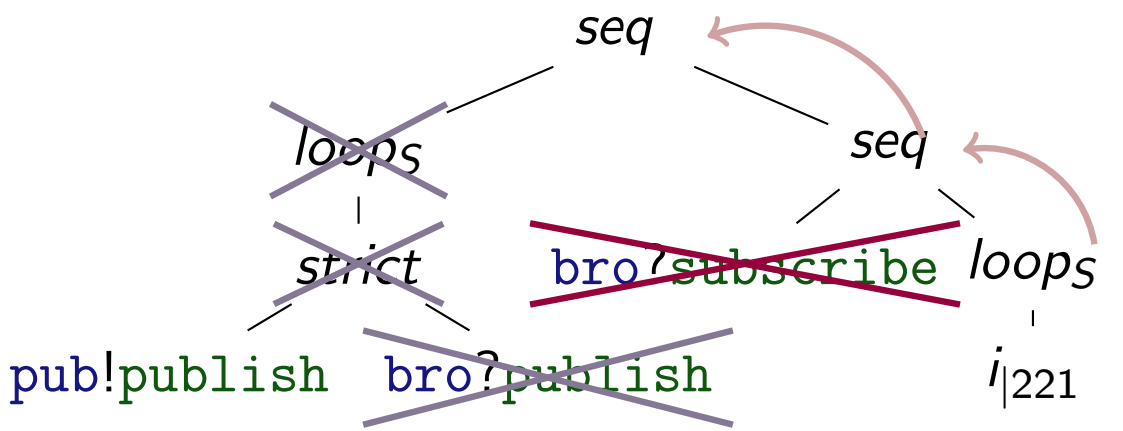
$\texttt{frt}(i) = \{111, 21, 22111\}$

Execution



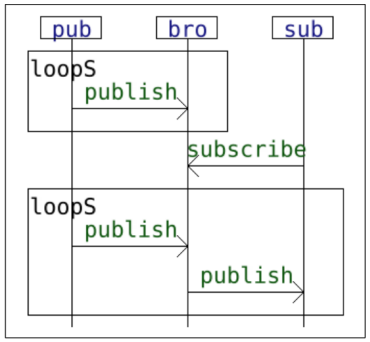
bro?subscribe

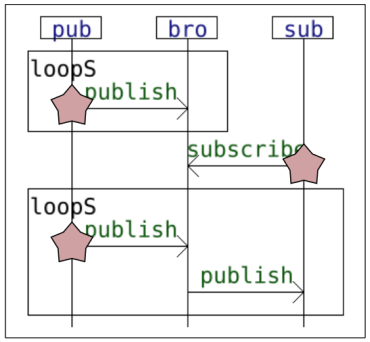


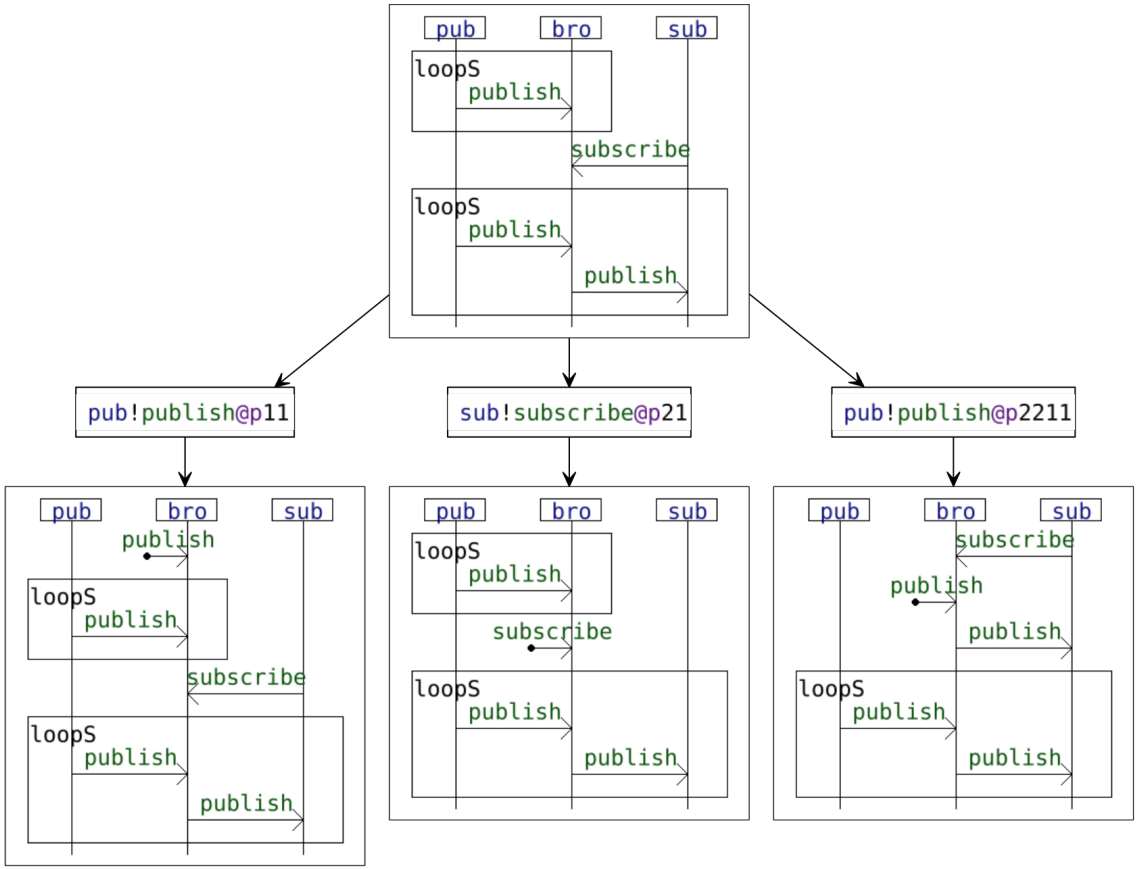


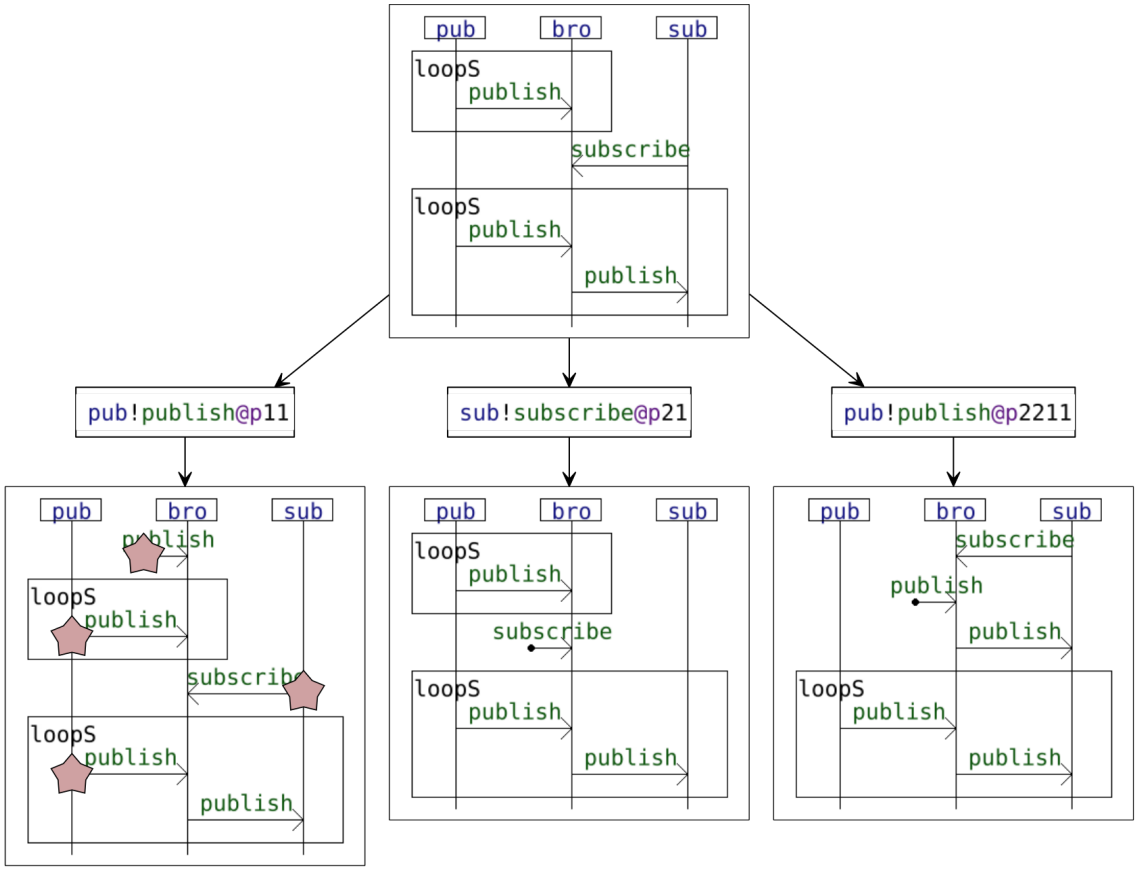
pruning | eχecution | simplification

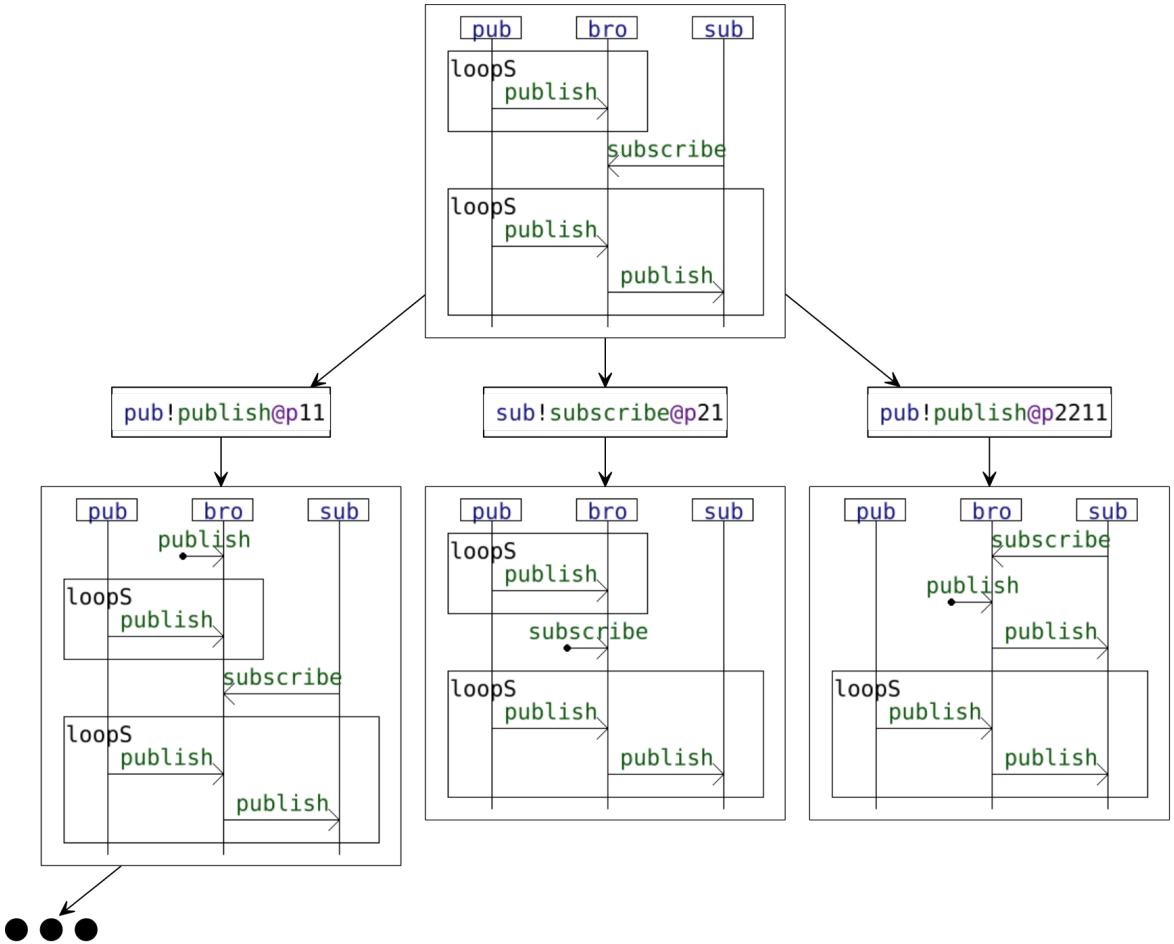
Muroor Nadumane 2020

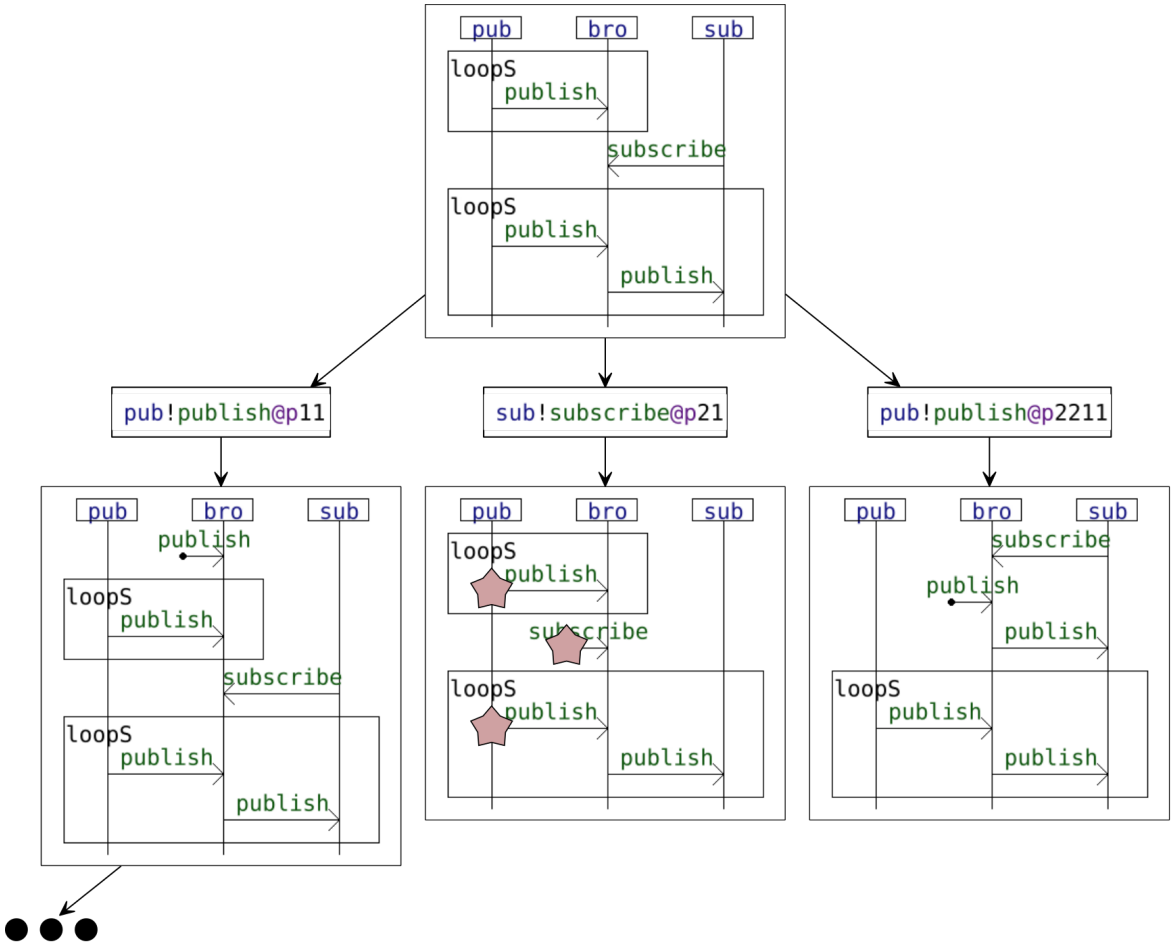

pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish

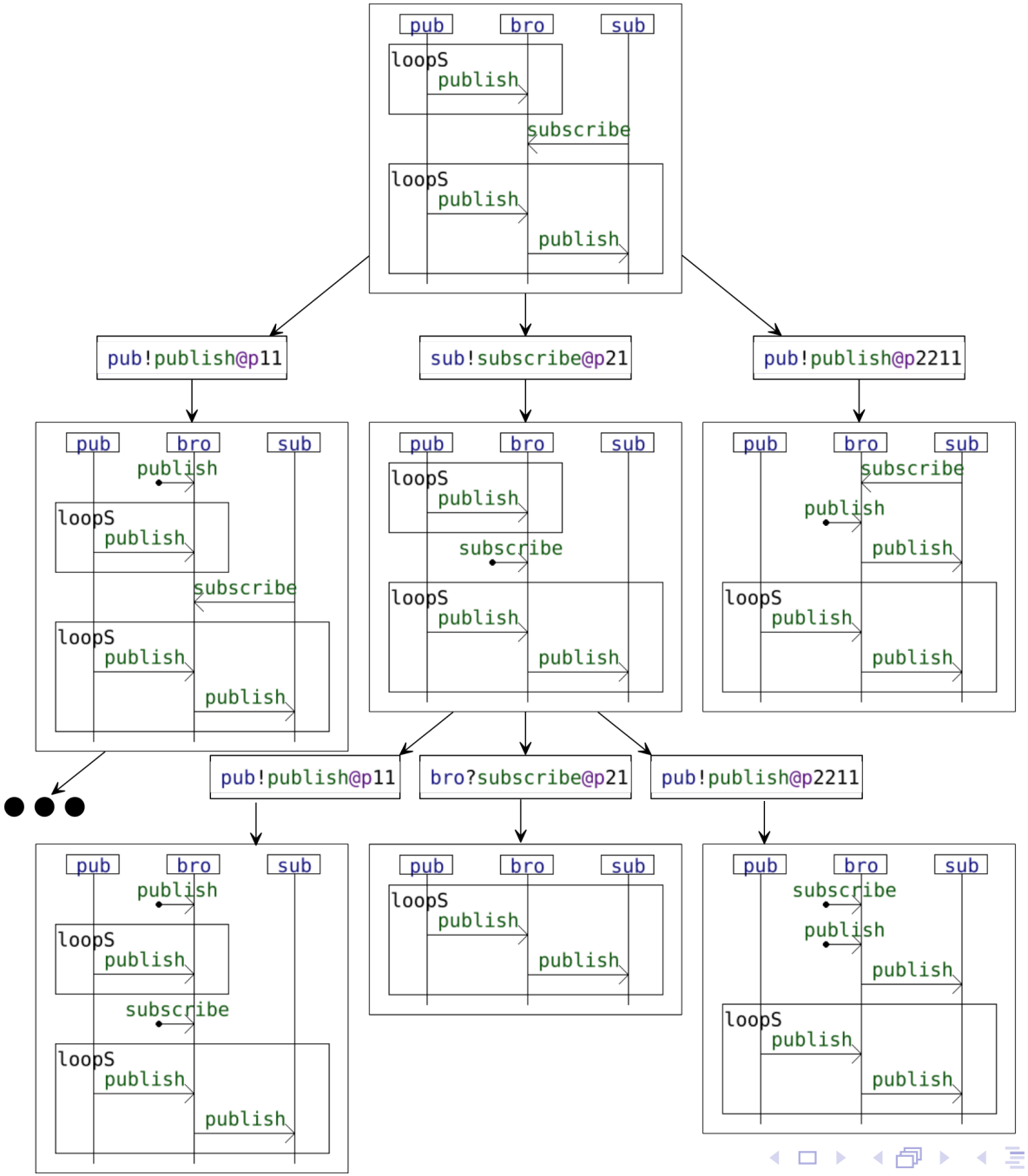

pub
bro
sub
loopS
publish
subscrib
loopS
publish
publish

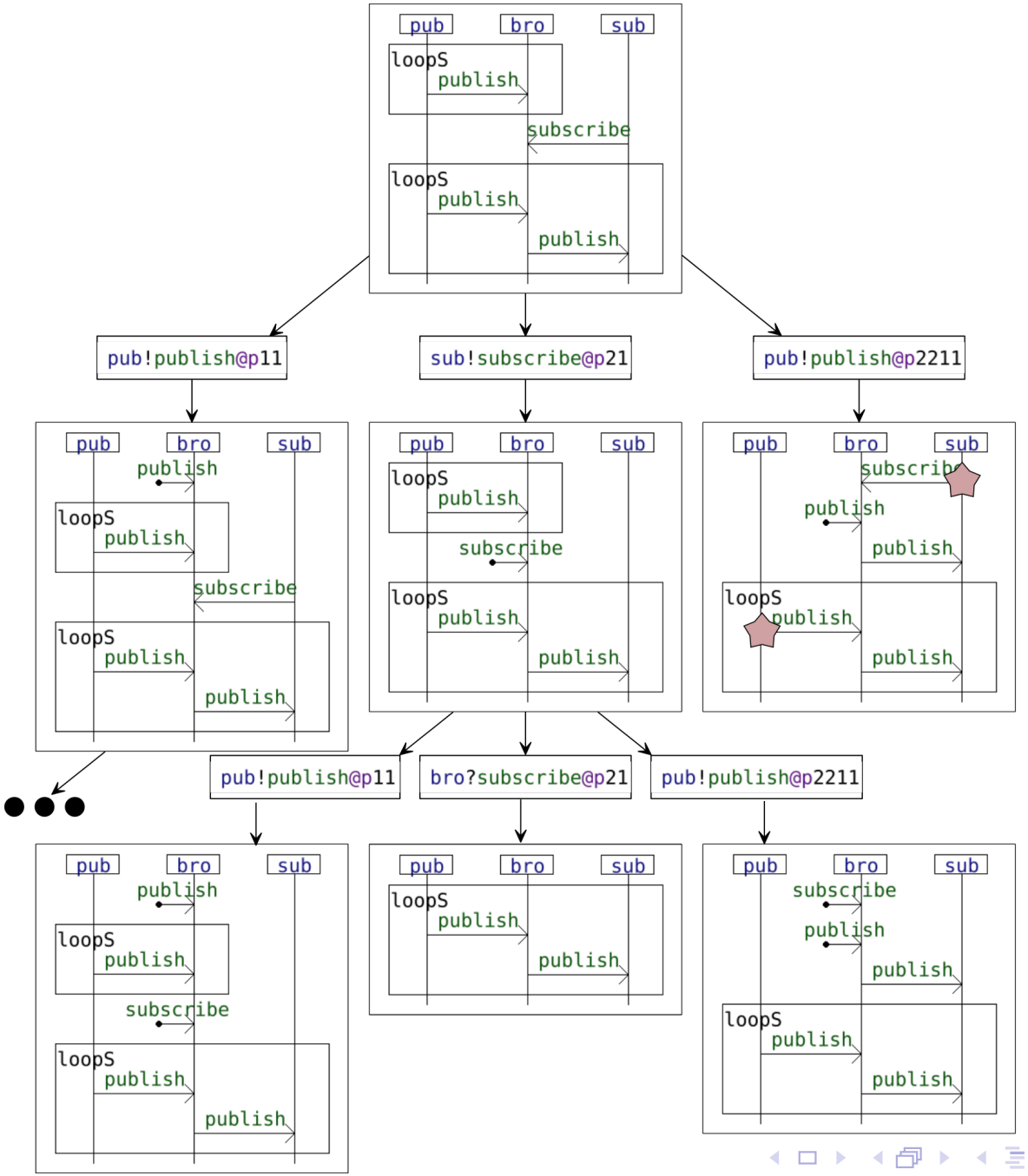

pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211
pub
bro
sub
publish
loopS
publish
subscribe
loopS
publish
publish
pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish
pub
bro
sub
subscribe
publish
publish
loopS
publish
publish

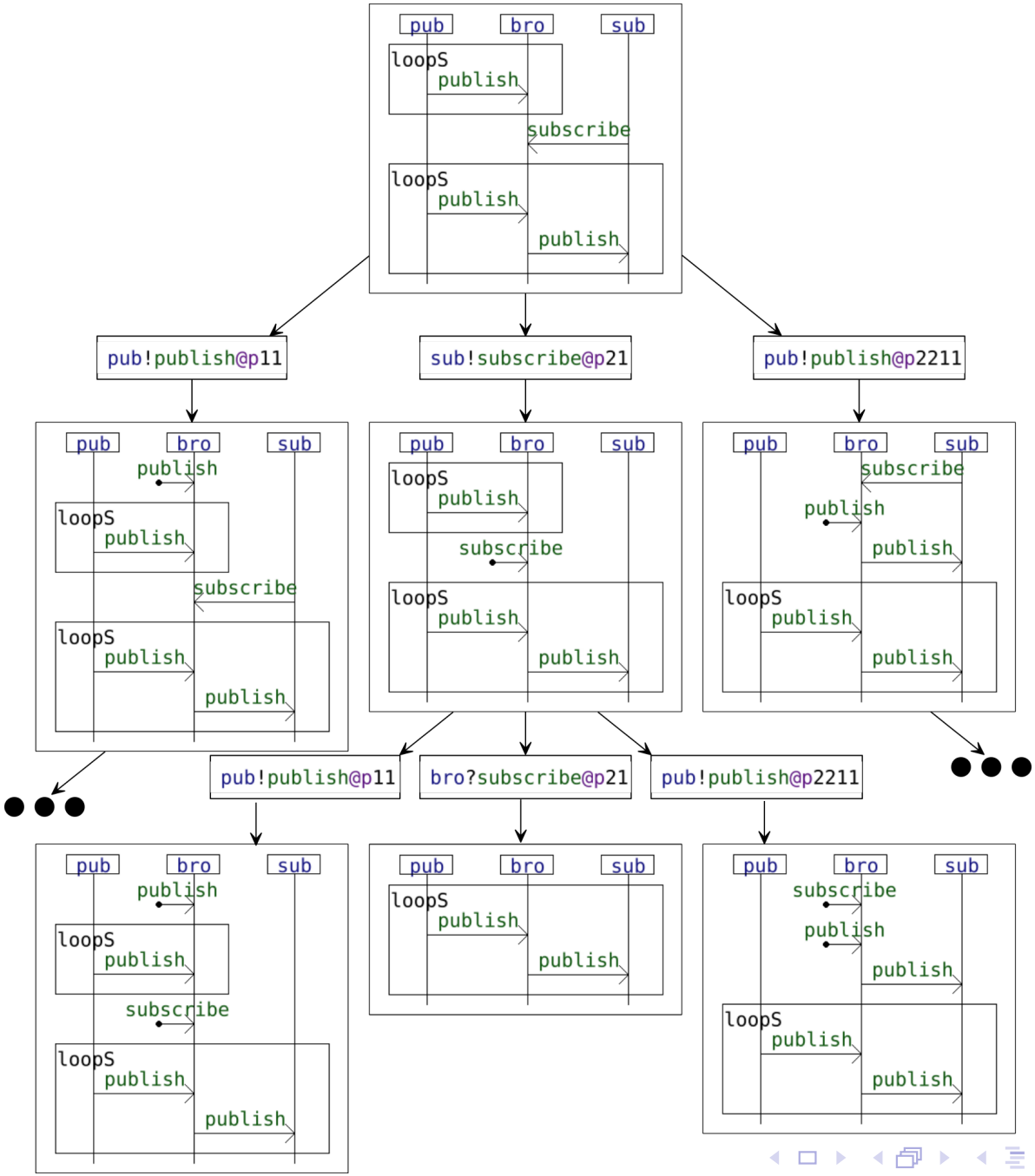

pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211
pub
bro
sub
loopS
publish
publish
subscribe
loopS
publish
publish
pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish
pub
bro
sub
subscribe
publish
publish
loopS
publish
publish


pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211
pub
bro
sub
publish
loopS
publish
subscribe
loopS
publish
publish
pub
bro
sub
loopS
publish
subscribe
loopS
publish
publish
pub
bro
sub
subscribe
publish
publish
loopS
publish
publish


pub
bro
sub
loopS
publish
subscribe
publish
publish
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211


pub
bro
sub
loopS
publish
subscribe
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211
bro?subscribe@p21


pub
bro
sub
loopS
publish
subscribe
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211
bro?subscribe@p21


pub
bro
sub
loopS
publish
subscribe
pub!publish@p11
sub!subscribe@p21
pub!publish@p2211
bro?subscribe@p21

Execution semantics & equivalence

Execution semantics σ_e

We define $\sigma_e : \mathbb{I}_\Omega \to \mathcal{P}(\mathbb{T}_\Omega)$ as:

$$\sigma_e(i) = empty(i) \cup \bigcup_{p \in front(i)} i_{|p}.\sigma_e(\chi(i,p))$$

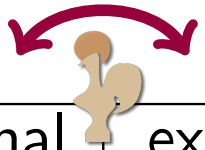
with $$empty(i) = \begin{cases} \{\epsilon\} & \text{if } i \downarrow \\ \emptyset & \text{if } i \not\downarrow \end{cases}$$

Equivalence of the semantics

For any interaction $i \in \mathbb{I}_\Omega$:

$$\sigma_e(i) = \sigma_o(i)$$

Comparison of semantics

	denotational	operational	execution
intuition of meaning	✓✓	✓	✗
related to	Knapp & Mossakowski 2017	Mauw & Reniers 1999	Muroor Nadumane 2020
use for proofs	✓✓ (hiding)		✗
easy implem. & use for RV	✗		✓✓
also	✓ (normal forms)	✓ (execution tree)	

Multi-trace analysis algorithms

Full observation (accepted behavior)

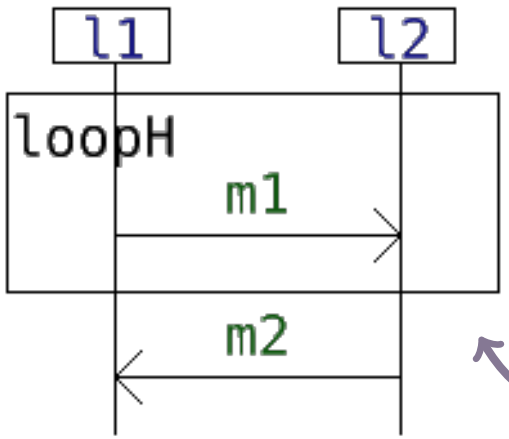
executed behavior as a global trace

$l_1!m_1$

$l_2?m_1$

$l_2!m_2$

$l_1?m_2$



[l1] ←l1!m1.l1?m2

[l2] ←l2?m1.l2!m2

fully observed multi-trace

interaction model

Projection of a global prefix (accepted behavior)

executed behavior as a global trace

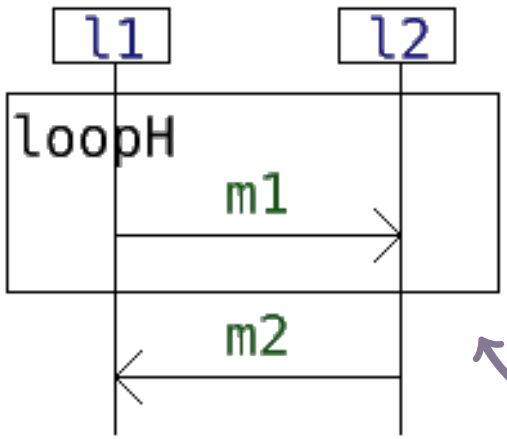
$l_1!m_1$

$l_2?m_1$

$l_2!m_2$

~~$l_1?m_2$~~

unobserved



[l1] ←l1!m1

[l2] ←l2?m1.l2!m2

partial observation

interaction model

Prefix in the sense of multi-traces (accepted behavior)

executed behavior as a global trace

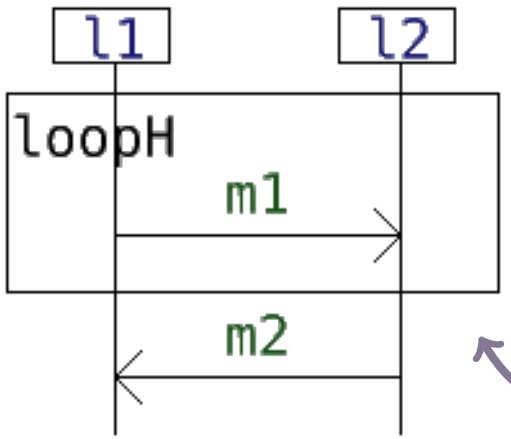
$l_1!m_1$

$l_2?m_1$

~~$l_2!m_2$~~

$l_1?m_2$

unobserved



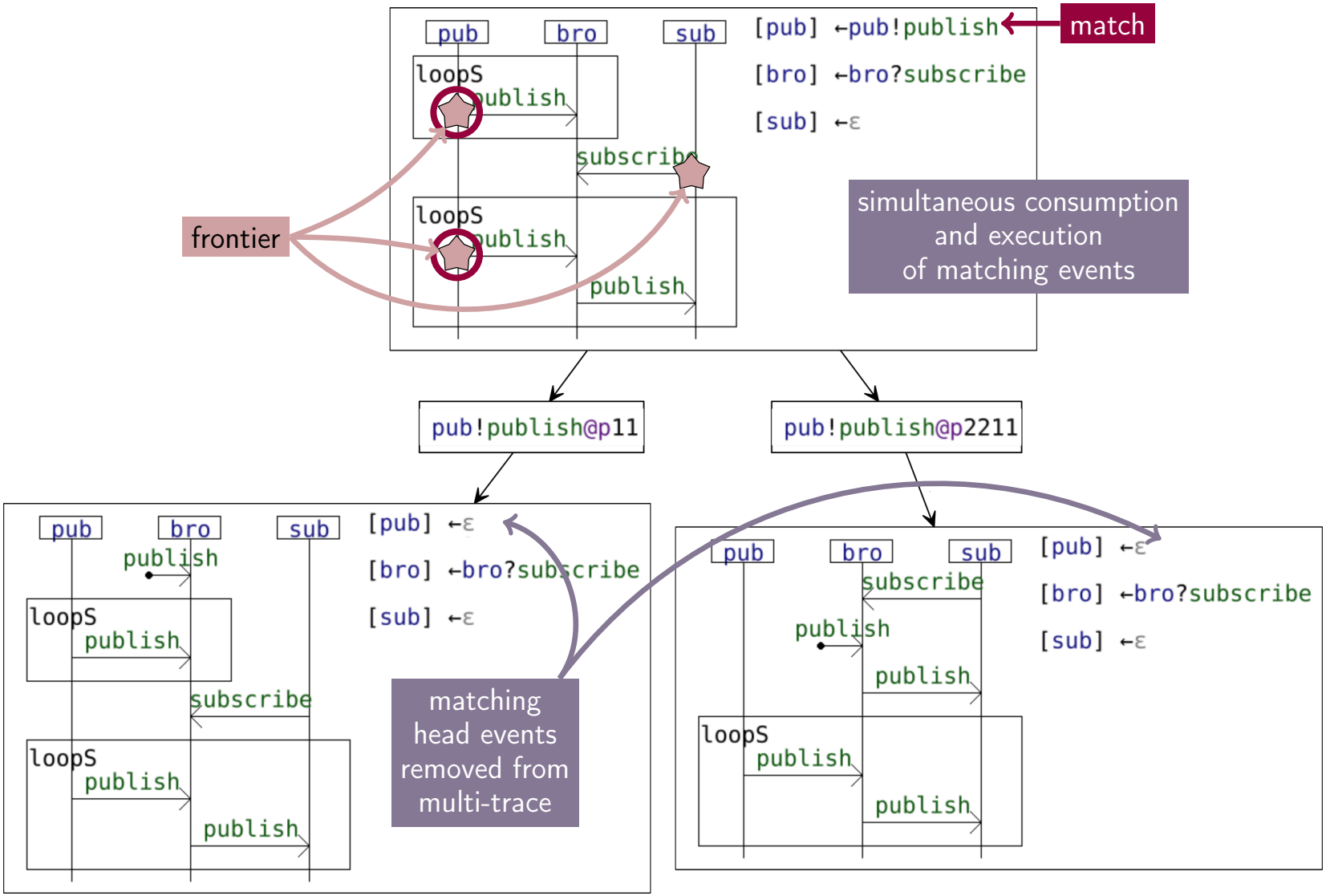
[l1] ←l1!m1.l1?m2

[l2] ←l2?m1

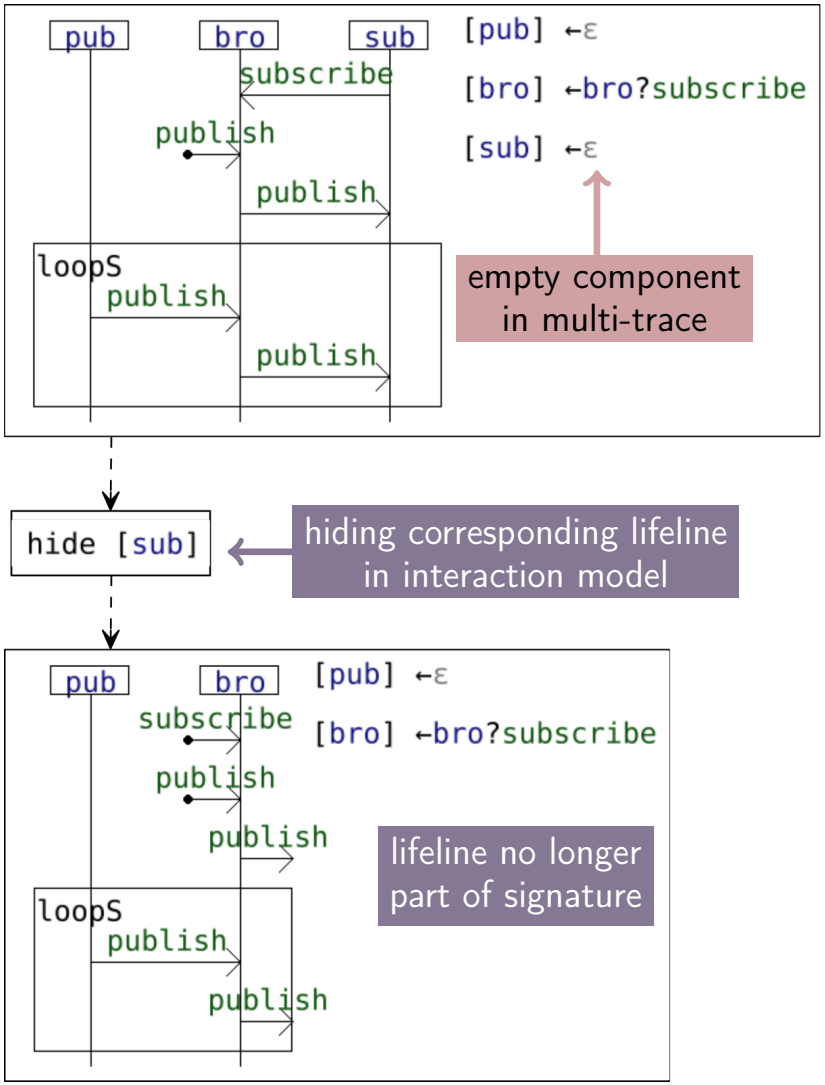
partial observation

interaction model

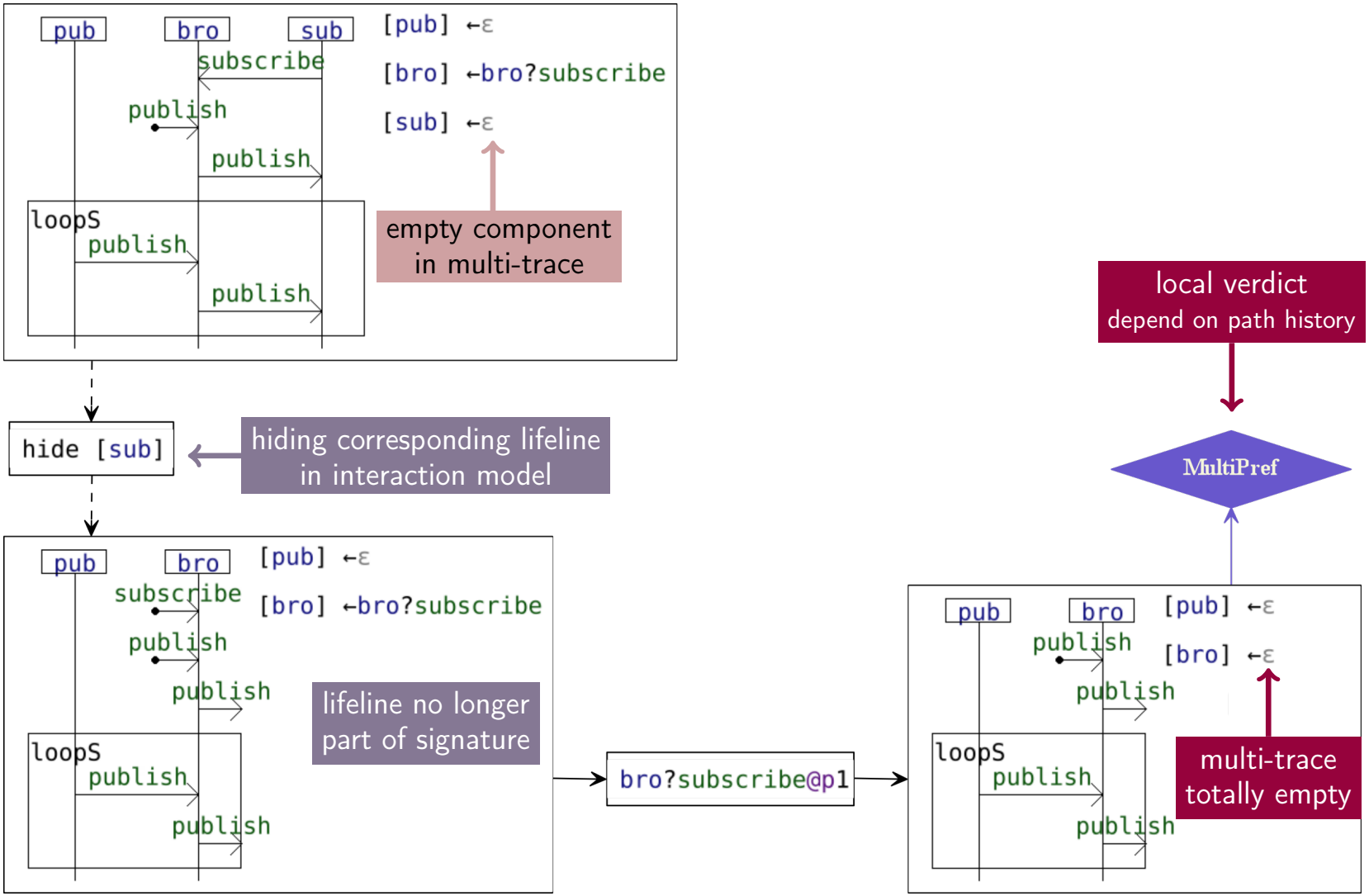
Consuming events



Hiding


pub
bro
sub
subscribe
publish
publish
loopS
publish
publish
[pub] ←ε
[bro] ←bro?subscribe
[sub] ←ε
empty component in multi-trace
hide [sub]
hiding corresponding lifeline in interaction model
pub
bro
subscribe
publish
publish
loopS
publish
publish
[pub] ←ε
[bro] ←bro?subscribe
lifeline no longer part of signature

Hiding & verdicts



An analysis algorithm with hiding steps

Analysis relation $\rightsquigarrow$ on nodes of $(\mathbb{I} \times \mathbb{M}) \cup \{Obs, Out\}$

The analysis relation $\rightsquigarrow$ is s.t.:

$$(R_h)\ \frac{i \quad \mu}{\mathbf{hide}_I(i) \quad \mathbf{elim}_I(\mu)}\ \mu_{|I} = \epsilon \qquad (R_e)\ \frac{i \quad a \vec{\odot} \mu}{i' \quad \mu}\ i \xrightarrow{a} i'$$

$$(R_p)\ \frac{i \quad \epsilon_L}{Obs} \qquad (R_f)\ \frac{i \quad \mu}{Out}\ \begin{cases} (\nexists\ I \in L \text{ s.t. } \mu_{|I} = \epsilon) \\ \wedge (\nexists\ i \xrightarrow{a} i' \text{ s.t. } \mu = a \vec{\odot} \mu') \end{cases}$$

Algorithm

$\omega : \mathbb{I} \times \mathbb{M} \to \{Pass, Fail\}$ is s.t.:

- $\omega(i, \mu) = Pass$ iff $\exists$ a path $(i, \mu) \overset{*}{\rightsquigarrow} Obs$
- $\omega(i, \mu) = Fail$ otherwise

Correctness

For any $i \in \mathbb{I}$ and $\mu \in \mathbb{M}$:

$(\omega(i, \mu) = Pass) \Leftrightarrow (\mu \in \overline{\circledcirc(i)})$

Implementation

Coq proofs

- proof of equivalence between 3 semantics:

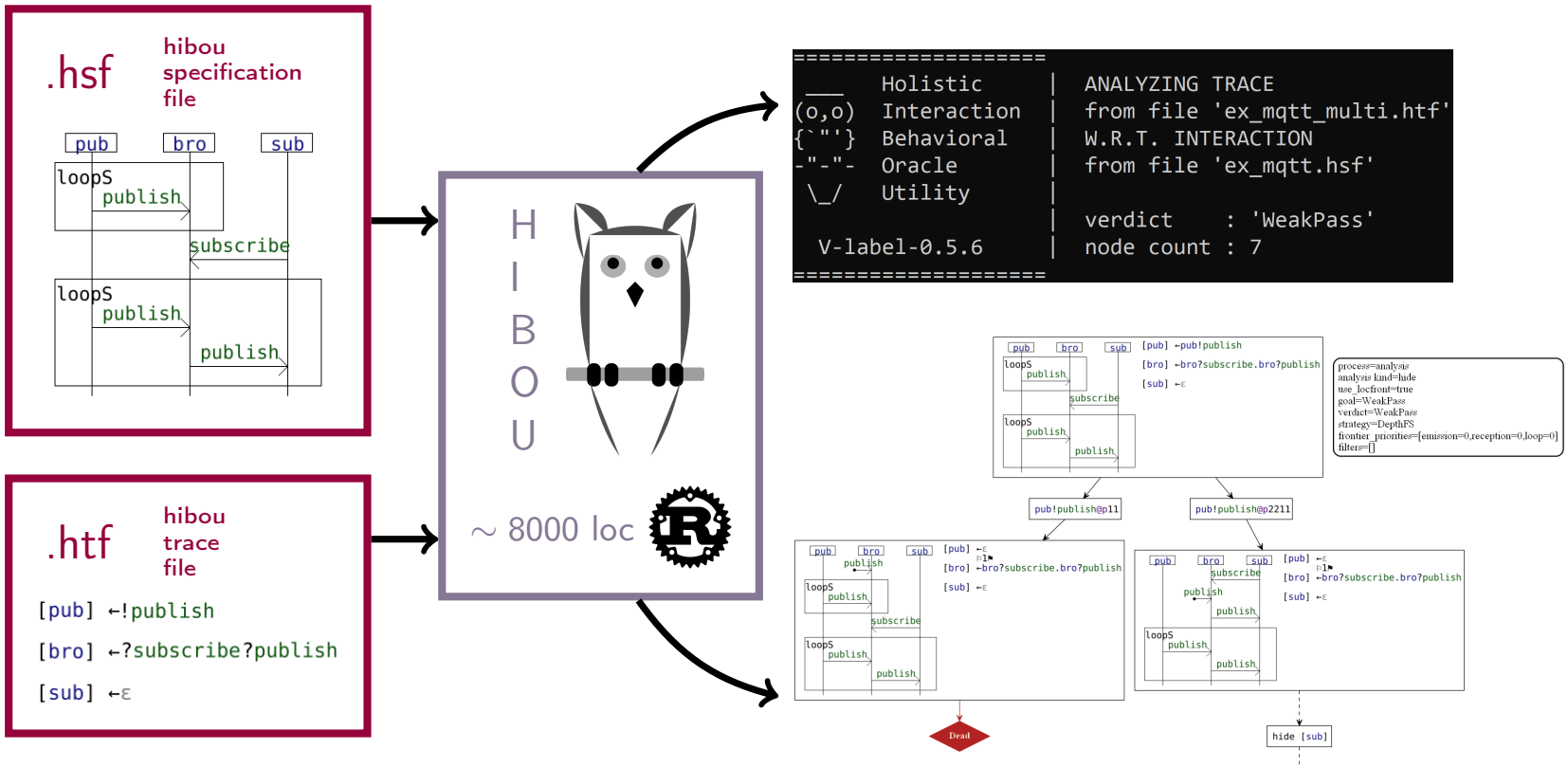
  `github.com/erwanM974/coq_hibou_label_semantics_equivalence`

  $\sim$ 3100 loc
- proof of correctness for analysis algorithm:

  `github.com/erwanM974/coq_hibou_label_multi_trace_analysis`

  $\sim$ 1300 loc

HIBOU tool



github.com/erwanM974/hibou_label

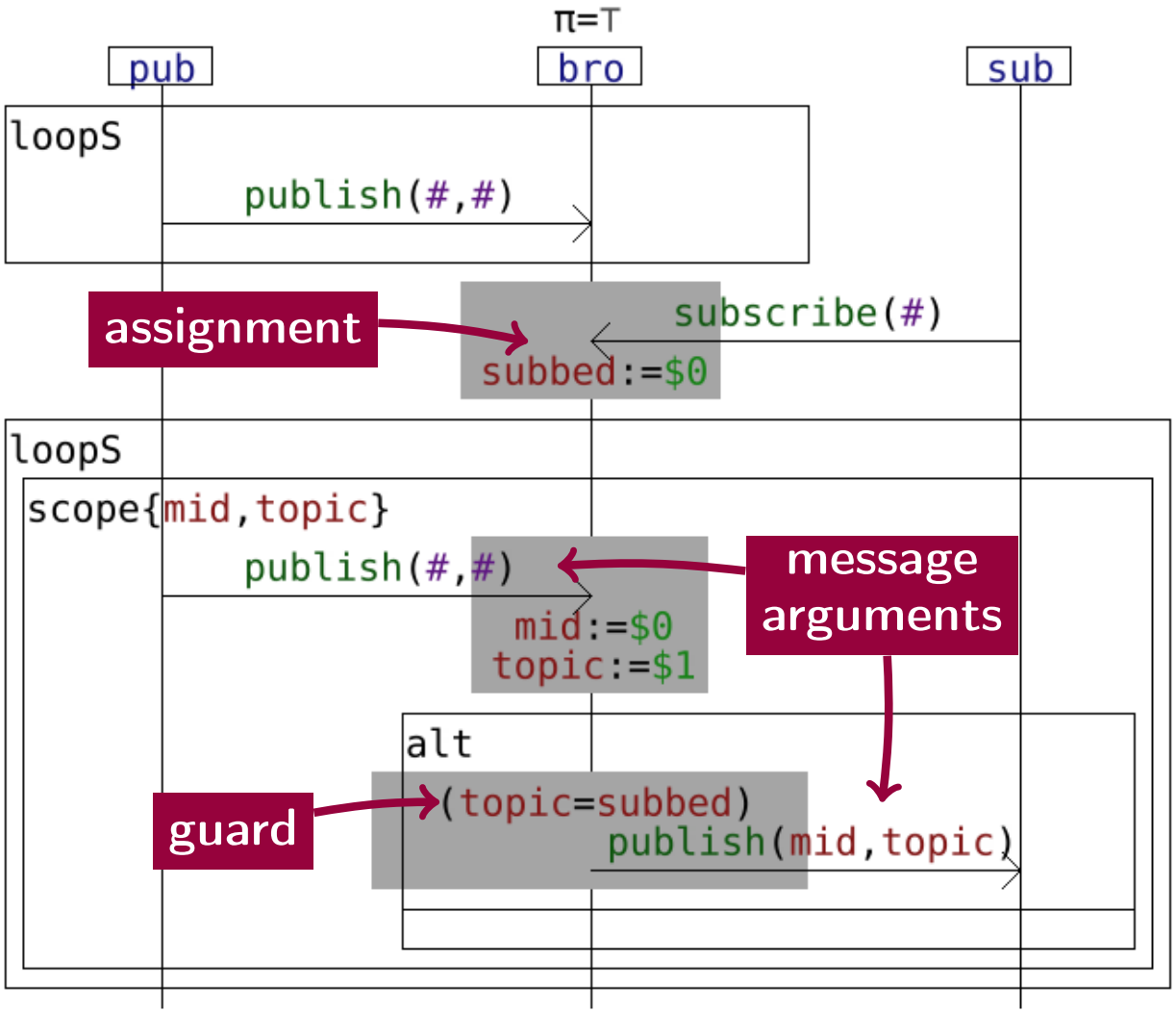
- multi-trace analysis (hide, simulation, local frontiers, goals, heuristics, filters, etc.)
- exploration of execution trees
- computation of normal forms
- ease of use features (input languages, configuration, graphical outputs, etc.)

The need to handle data

concrete message arguments

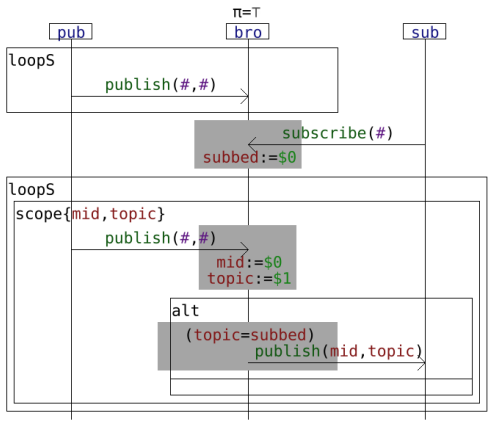
```
[pub] ← !publish(429,"a")

[bro] ← ?subscribe("b")...+1

[sub] ← ε
```

Enriching interactions with data



HIBOUX tool

.hxsf — hiboux specification file



.hxtf — hiboux trace file

```
[pub] ← !publish(429,"a")
[bro] ← ?subscribe("b")...+1
[sub] ← ε
```

github.com/erwanM974/hibou_efm

HIBOUX

~ 13000 loc

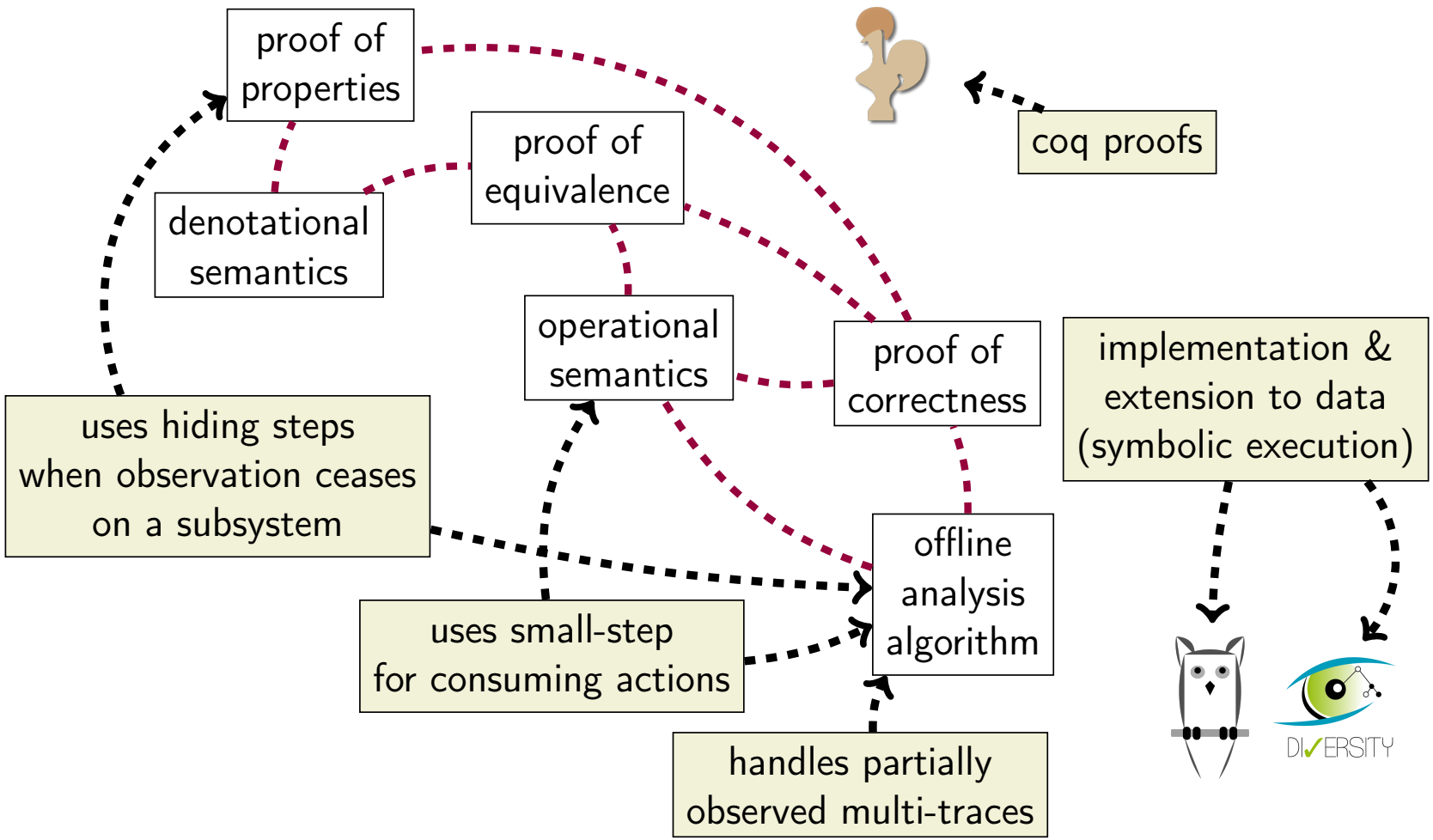
outputs

pilots the firing of symbolic actions (flow of control)

DIVERSITY

performs symbolic execution (handles data)

Conclusion


proof of properties
proof of equivalence
coq proofs
denotational semantics
operational semantics
proof of correctness
implementation & extension to data (symbolic execution)
uses hiding steps when observation ceases on a subsystem
offline analysis algorithm
uses small-step for consuming actions
handles partially observed multi-traces
DIVERSITY

Perspectives

- further validation & use of the tools on case studies
- formalizing data & time in interaction models
- horizontal composition of interactions
  (reverse of hiding $\rightarrow$ from local to global)
- online analysis

Bibliography

Francalanza, Perez & Sanchez 2018
- Runtime Verification for Decentralized and Distributed Systems

Dan & Hierons 2014
- The Oracle Problem When Testing from MSCs

Nguyen, Poizat & Zaïdi 2011
- Passive conformance testing of service choreographies

Benharrat, Gaston, Hierons, Lapitre & Le Gall
- Constraint-Based Oracles for Timed Distributed Systems

Bauer & Falcone 2012
- Decentralized LTL Monitoring

Longuet 2012
-Global and local testing from Message Sequence Charts

Mauw & Reniers 1997
- High-level Message Sequence Charts

Damm & Harel 2001
- LSCs: Breathing Life into Message Sequence Charts

Harel & Maoz 2008
- Assert and negate revisited: Modal semantics for UML Sequence diagrams

Faria & Paiva 2016
- A toolset for conformance testing against UML sequence diagrams based on event-driven colored Petri nets

Knapp & Mossakowski 2017
- UML Interactions Meet State Machines - An Institutional Approach

Störrle 2003
- Semantics of Interactions in UML 2.0

Mauw & Reniers 1999
- Operational Semantics for MSC'96

Rooda, van Beek & Baeten 2007
- Process Algebra (in Handbook of Dynamic System Modeling)

Despeyroux 2001
- A Higher-Order Specification of the π-Calculus

Muroor Nadumane 2020
- Models and Verification for Composition and Reconfiguration of Web of Things Applications

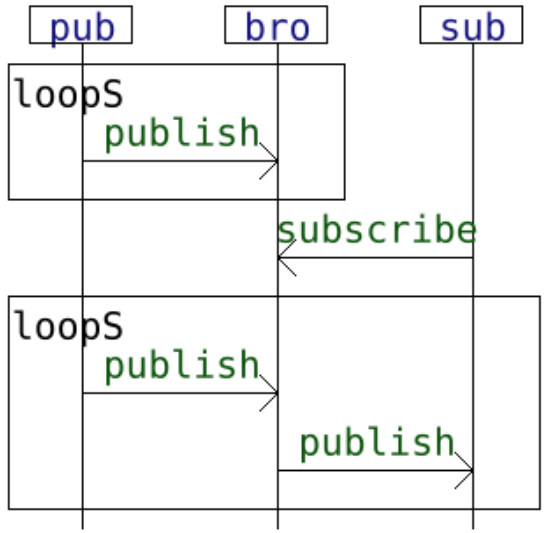
Appendix

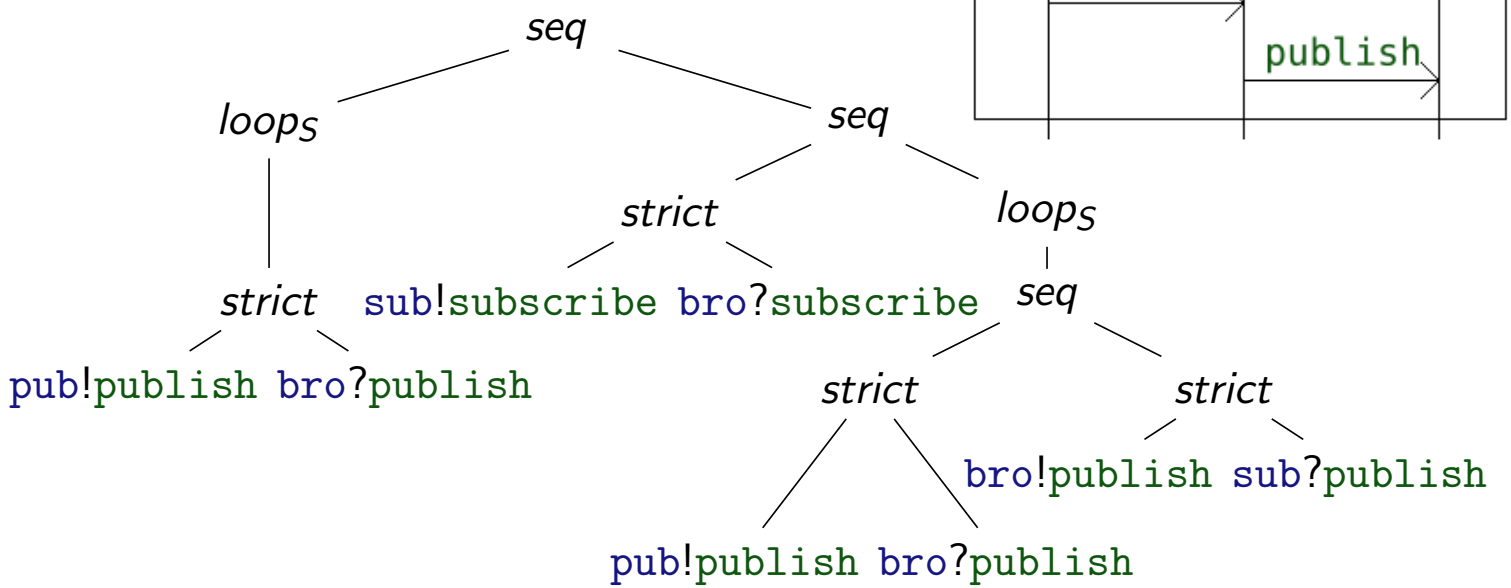
Details on the syntax

$$\Omega = (L, M)$$

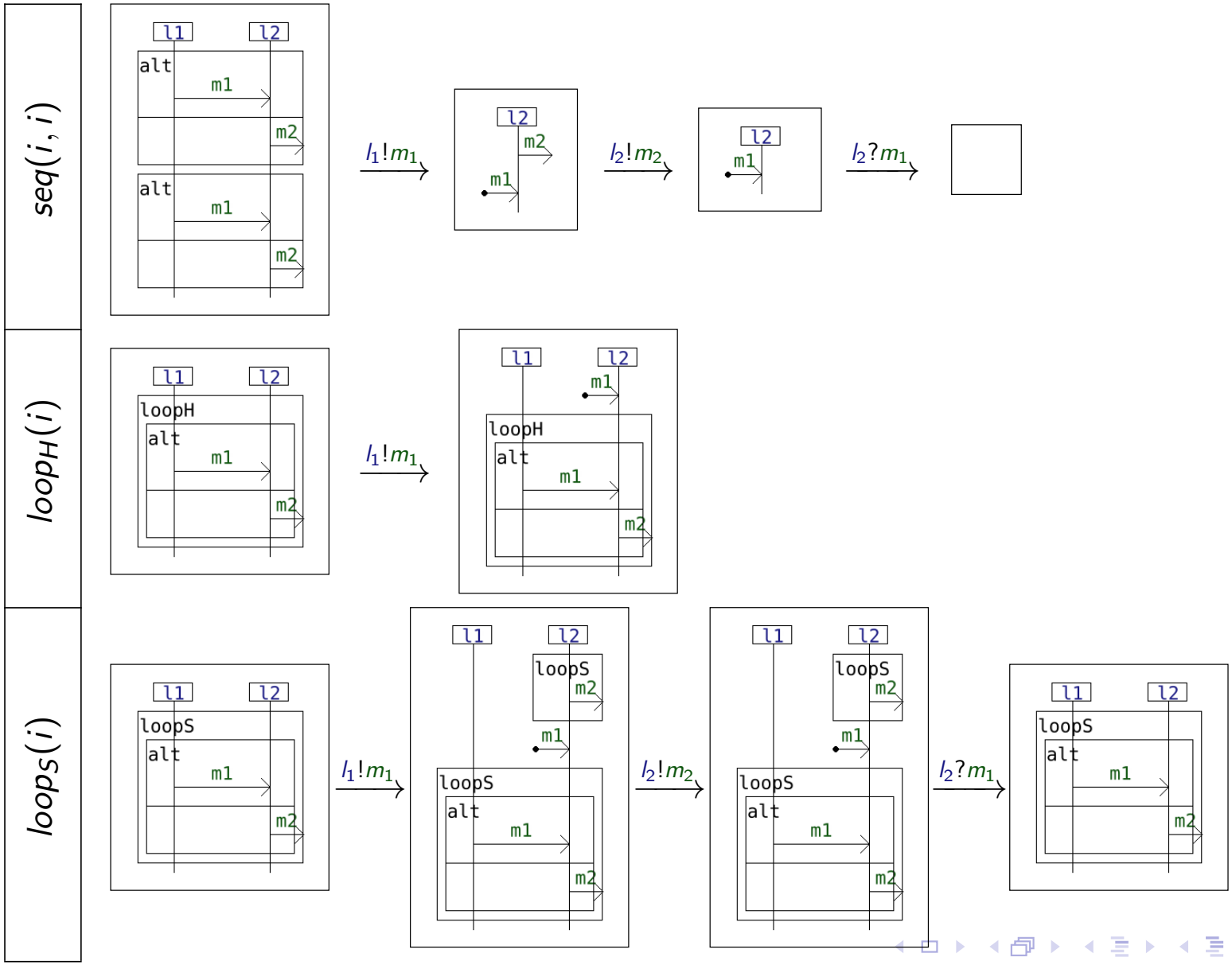
$$\mathbb{A}_\Omega = L \times \{!, ?\} \times M$$

$$\left(\mathbb{I}_\Omega, \left\{ \begin{array}{llll} \varnothing, & a \in \mathbb{A}_\Omega, & & \\ alt, & strict, & seq, & par \\ loop_X, & loop_H, & loop_S, & loop_P \end{array} \right\} \right)$$

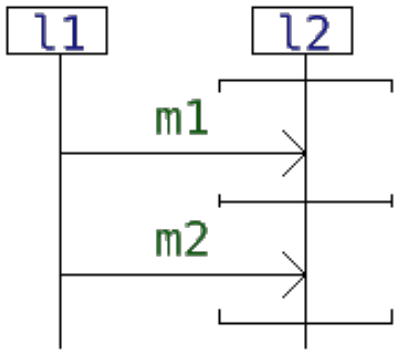
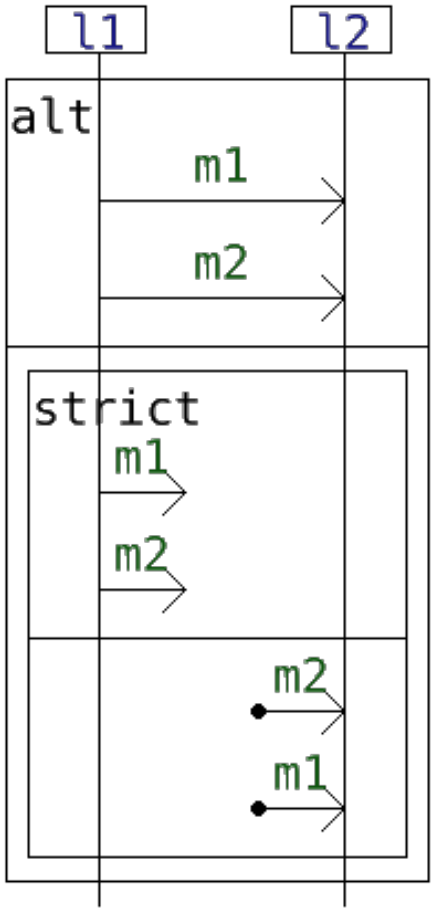




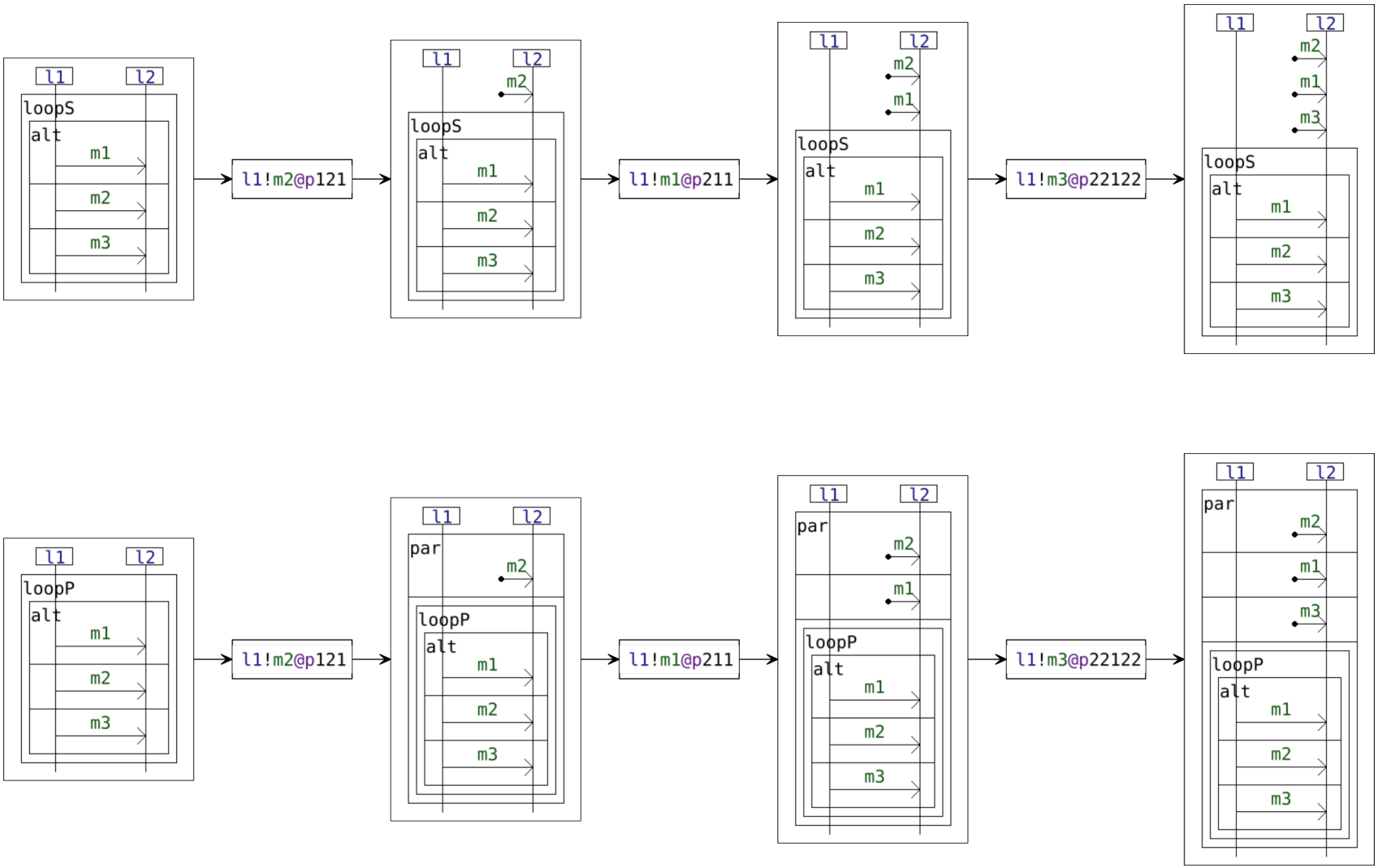
$loop_H$ & $loop_S$


seq(i, i)
loopH(i)
loopS(i)
l1
l2
alt
m1
m2
loopH
loopS
$l_1!m_1$
$l_2!m_2$
$l_2?m_1$

Co-regions

diagram	syntax	equivalent
	$coreg(\{l_2\})$ $strict$ $strict$ $l_1!m_1\ l_2?m_1\ l_1!m_2\ l_2?m_2$	

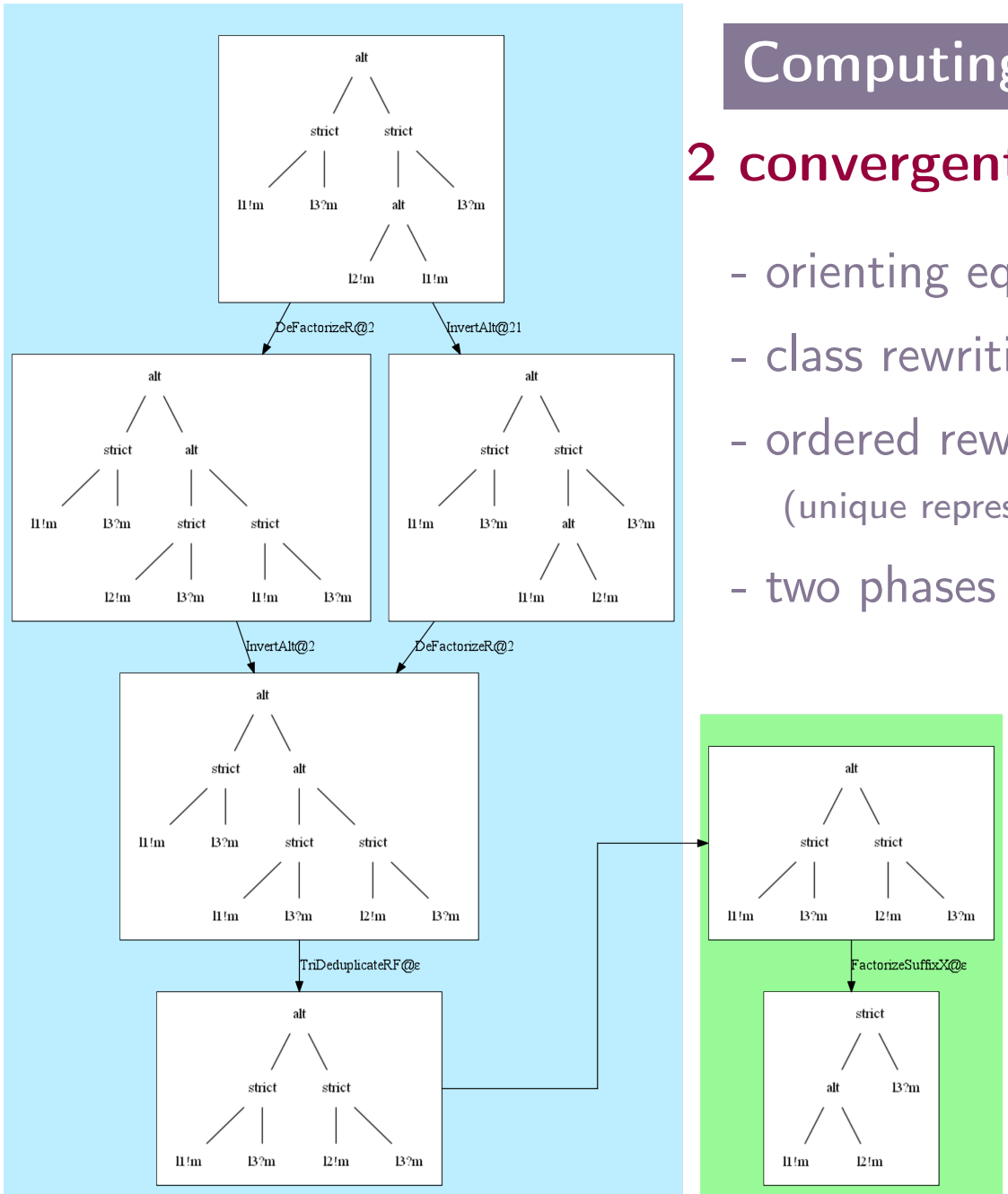
Communication medium within the language itself (buffers, bags, etc.)



Equations on interactions

$$
\begin{array}{ll}
\forall\ f \in \{strict, seq, par\}, & f(\varnothing, y) \approx y \\
\forall\ f \in \{strict, seq, par\}, & f(x, \varnothing) \approx x \\
\forall\ f \in \{strict, seq, par, alt\}, & f(f(x, y), z) \approx f(x, f(f, z)) \\
\forall\ f \in \{par, alt\}, & f(x, y) \approx f(y, x) \\
 & alt(x, x) \approx x \\
\forall\ f \in \{strict, seq, par\}, & f(x, alt(y, z)) \approx alt(f(x, y), f(x, z)) \\
\forall\ f \in \{strict, seq, par\}, & f(alt(x, y), z) \approx alt(f(x, z), f(y, z)) \\
\forall\ k \in \{X, H, S, P\}, & loop_k(\varnothing) \approx \varnothing \\
\forall\ (k_1, k_2) \in \{X, H, S, P\}^2, & loop_{k_1}(loop_{k_2}(x)) \approx loop_{min(k_1, k_2)}(x)
\end{array}
$$

Match algebraic properties of underlying operators
(neutral element, associativity, commutativity, etc.)

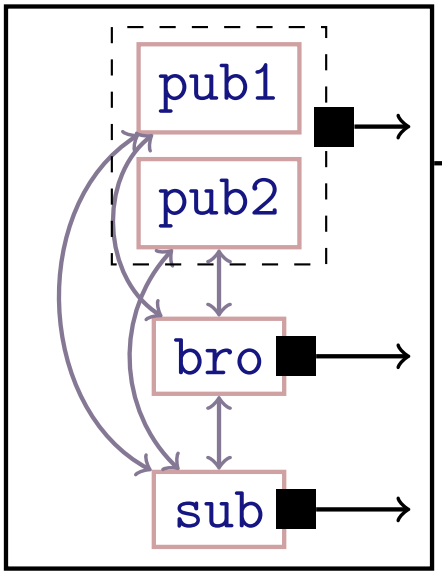


Computing normal forms

2 convergent rewrite systems

- orienting equations (termination)
- class rewriting (AC)
- ordered rewriting
  (unique representants of AC-classes)
- two phases (distributivity)

Having co-localized lifelines



execution
& logging

given the partition of L

$$C = \begin{pmatrix} \{\texttt{pub1}, \texttt{pub2}\}, \\ \{\texttt{bro}\}, \\ \{\texttt{sub}\} \end{pmatrix}$$

multitrace μ

$$\begin{pmatrix} \mu_{|\{\texttt{pub1},\texttt{pub2}\}} & = & \texttt{pub1!publish}.\cdots, \\ \mu_{|\{\texttt{bro}\}} & = & \texttt{bro?publish}.\cdots, \\ \mu_{|\{\texttt{sub}\}} & = & \texttt{sub!subscribe}.\cdots \end{pmatrix}$$

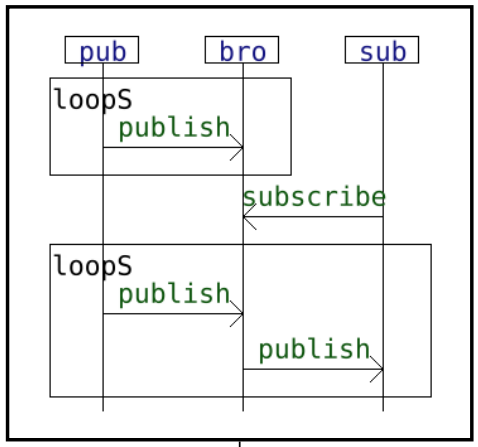
Denotational multi-trace semantics (general case)

Given a partition C of L defining the components of multi-traces:

$$\left(\mathbb{I}_\Omega, \left\{ \begin{array}{l} \varnothing,\ a \in \mathbb{A}_\Omega, \\ \mathit{alt},\ \mathit{strict},\ \mathit{seq},\ \mathit{par} \\ \mathit{loop}_X,\ \mathit{loop}_H,\ \mathit{loop}_S,\ \mathit{loop}_P \end{array} \right\}\right)$$

homomorphism $@_C$

$$\left(\mathcal{P}(\mathbb{T}_{\Omega|C}), \left\{ \begin{array}{l} \{\epsilon_C\},\ \{a \vec{\odot} \epsilon_C\}, \\ \cup,\ \textcircled{;},\ \textcircled{\otimes},\ \textcircled{\Vert}, \\ \textcircled{;}*,\ \textcircled{\otimes}^{\curvearrowleft}*,\ \textcircled{\otimes}*,\ \textcircled{\Vert}* \end{array} \right\}\right)$$



$$\left\{ \begin{array}{l} (!\texttt{subscribe},\ ?\texttt{subscribe},\ \epsilon), \\ \cdots, \\ (!\texttt{publish}.\cdots,\ ?\texttt{publish},\ \epsilon), \\ \cdots \end{array} \right\}$$

Multi-trace semantics up to the discrete partition

In the case where $C = (\{l\})_{l \in L} = \check{L}$, we have $\circledast = \circledcirc$ and hence:

$$\left(\mathbb{I}_\Omega, \left\{ \begin{array}{l} \varnothing,\ a \in \mathbb{A}_\Omega, \\ alt,\ strict,\ seq,\ par \\ loop_X,\ loop_H,\ loop_S,\ loop_P \end{array} \right\}\right)$$

σ ↙ ; $\sigma_{|\check{L}} = \mathbf{proj}_{\check{L}} \circ \sigma$; $\circledast_{\check{L}}$ ↘

$$\left(\mathcal{P}(\mathbb{T}_\Omega), \left\{ \begin{array}{l} \{\epsilon\},\ \{a\}, \\ \cup,\ ;,\ ;_\circledast,\ \|, \\ ;^*,\quad ;_\circledast^{\curvearrowleft *},\quad ;_\circledast^*,\quad \|^* \end{array} \right\}\right) \xrightarrow{\mathbf{proj}_{\check{L}}} \left(\mathcal{P}(\mathbb{T}_{\Omega|\check{L}}), \left\{ \begin{array}{l} \{\epsilon_{\check{L}}\},\ \{a \vec{\odot} \epsilon_{\check{L}}\}, \\ \cup,\ \circledcirc,\ \circledast,\ \circledparallel, \\ \circledcirc^*,\quad \circledast^{\curvearrowleft *},\quad \circledast^*,\quad \circledparallel^* \end{array} \right\}\right)$$

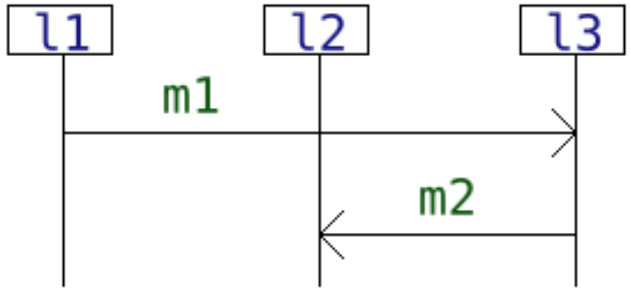
Projection preserves some algebraic structures

For any sets of traces T, T_1 and T_2:

$$\mathbf{proj}_C(T_1 \cup T_2) = \mathbf{proj}_C(T_1) \cup \mathbf{proj}_C(T_2)$$
$$\mathbf{proj}_C(T_1; T_2) = \mathbf{proj}_C(T_1) \fatsemi \mathbf{proj}_C(T_2)$$
$$\mathbf{proj}_C(T_1 || T_2) = \mathbf{proj}_C(T_1) \fatbslash\!\!\fatbslash \mathbf{proj}_C(T_2)$$
$$\mathbf{proj}_C(T^{;*}) = \mathbf{proj}_C(T)^{\fatsemi *}$$
$$\mathbf{proj}_C(T^{||*}) = \mathbf{proj}_C(T)^{\fatbslash\!\!\fatbslash *}$$

Counter example

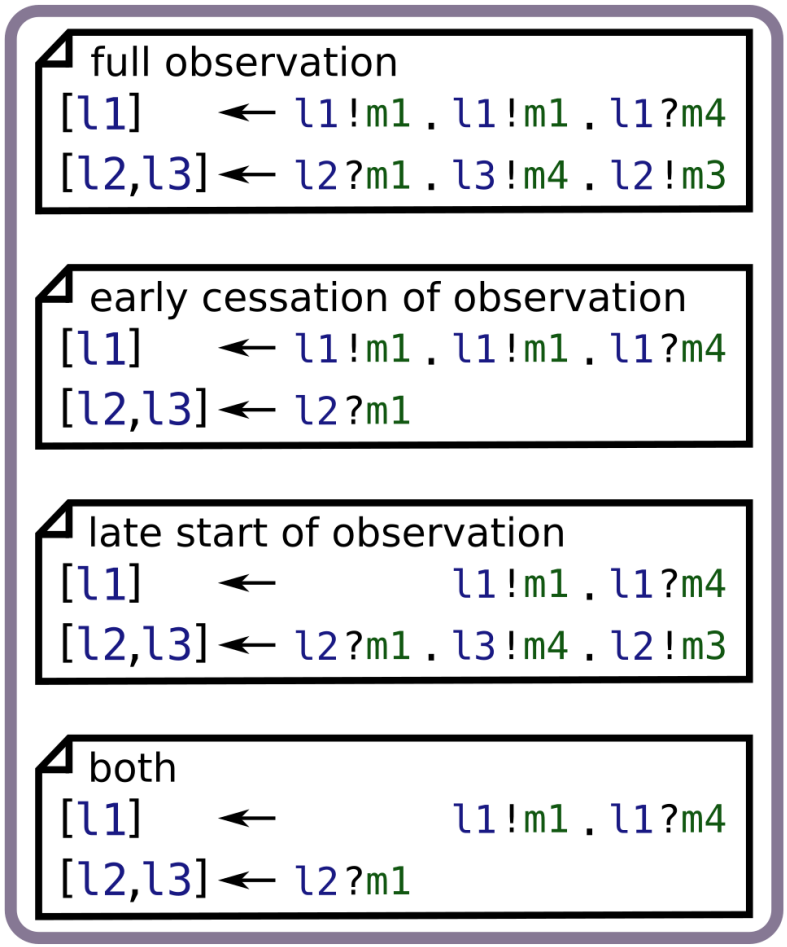
Given $L = \{l_1, l_2, l_3\}$ and the partition $C = (\{l_1, l_2\},\ \{l_3\})$



$$\mathbf{proj}_C \left(\begin{array}{l} \{l_1!m_1.l_3?m_1\} \\ ;\!\!\times \{l_3!m_2.l_2?m_2\} \end{array} \right) = \mathbf{proj}_C(\{l_1!m_1.l_3?m_1.l_3!m_2.l_2?m_2\}) = \left\{ \left(\begin{array}{l} l_1!m_1.l_2?m_2, \\ l_3?m_1.l_3!m_2 \end{array} \right) \right\}$$

$$\left(\begin{array}{l} \mathbf{proj}_C(\{l_1!m_1.l_3?m_1\}) \\ \circledast_C \mathbf{proj}_C(\{l_3!m_2.l_2?m_2\}) \end{array} \right) = \left\{ \left(\begin{array}{l} l_1!m_1, \\ l_3?m_1 \end{array} \right) \right\} \circledast_C \left\{ \left(\begin{array}{l} l_2?m_2, \\ l_3!m_2 \end{array} \right) \right\} = \left\{ \left(\begin{array}{l} l_1!m_1.l_2?m_2, \\ l_3?m_1.l_3!m_2 \end{array} \right), \left(\begin{array}{l} l_2?m_2.l_1!m_1, \\ l_3?m_1.l_3!m_2 \end{array} \right) \right\}$$

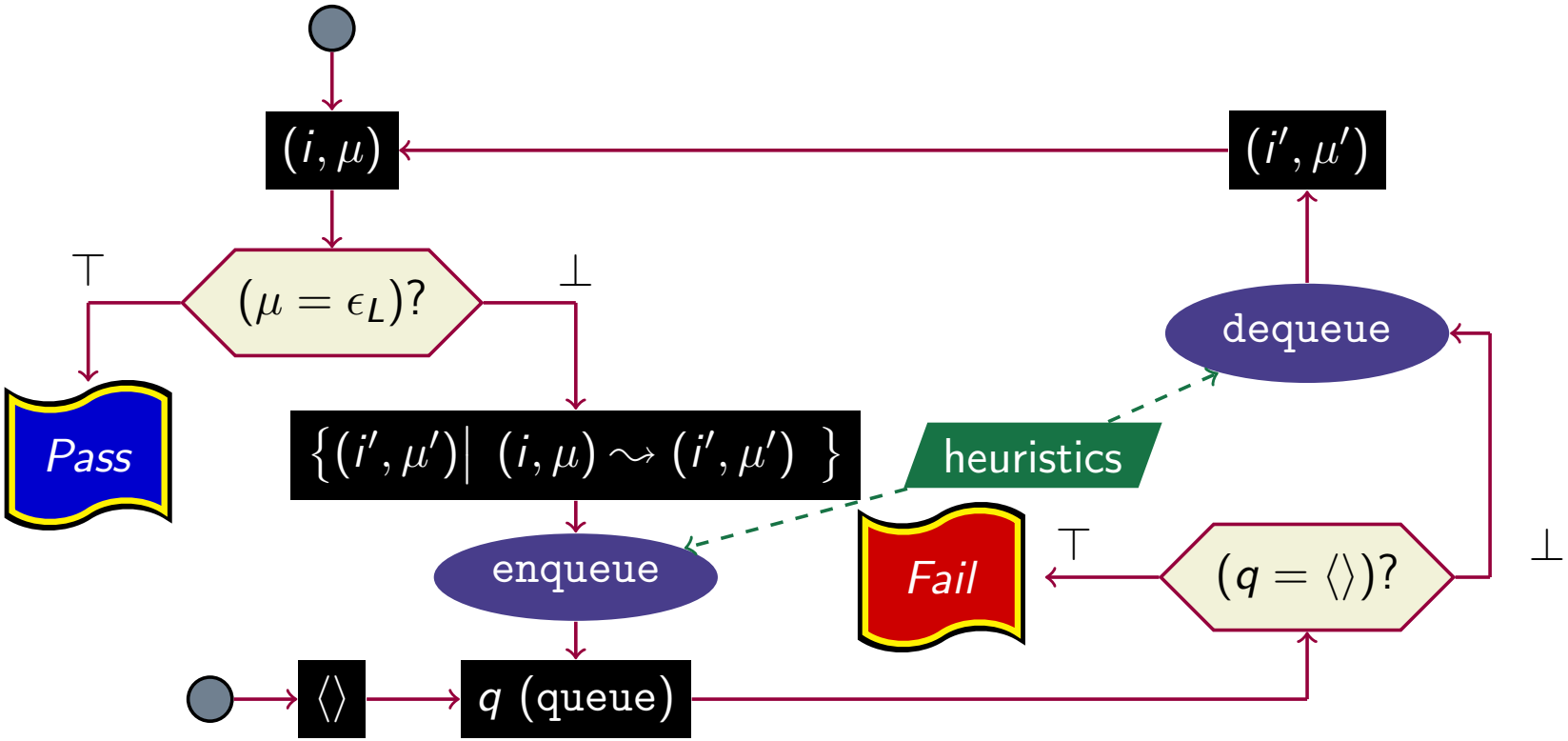
Semantics of prefixes and slices



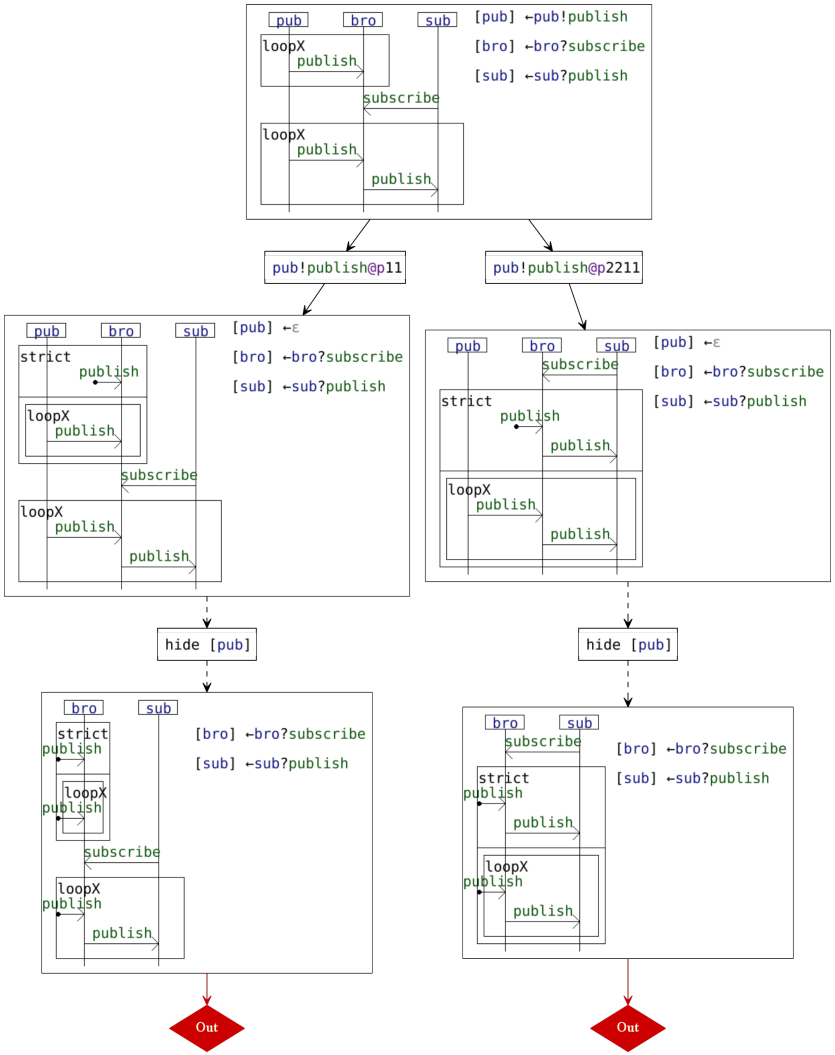
For any interaction $i \in \mathbb{I}_\Omega$:

- $\sigma^{\dagger}_{|C}(i) = \mathbf{proj}_C(\overline{\sigma(i)})$
- $\overline{\sigma_{|C}}(i) = \overline{\sigma_{|C}(i)}$
- $\overline{\circledcirc_C}(i) = \overline{\circledcirc_C(i)}$
- $\underline{\sigma_{|C}}(i) = \underline{\sigma_{|C}(i)}$

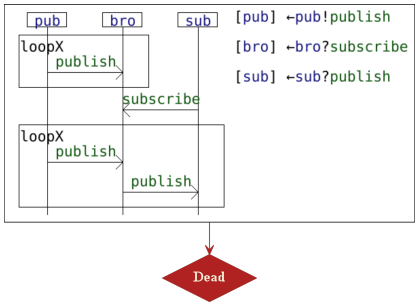
Using heuristics, etc.



Reducing the search space with local frontiers

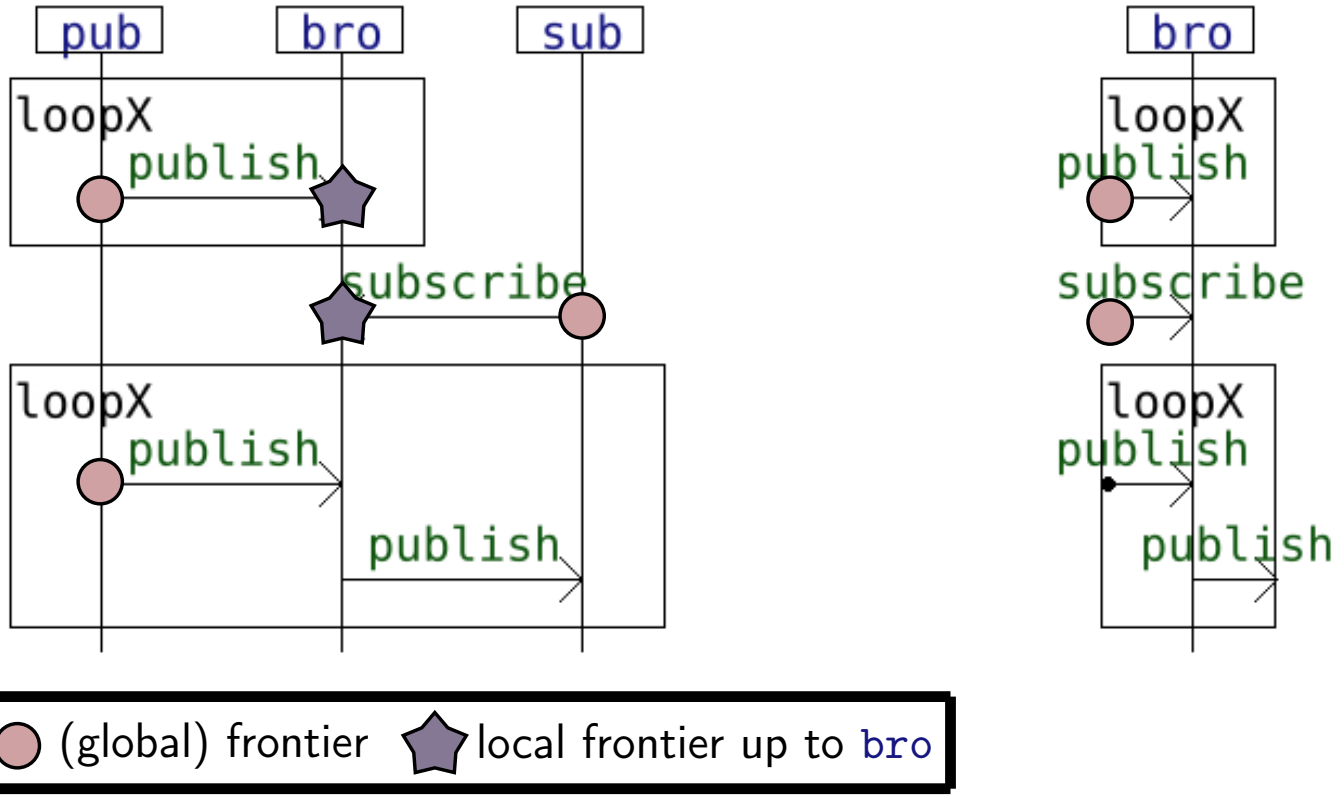


pointlessly explored paths

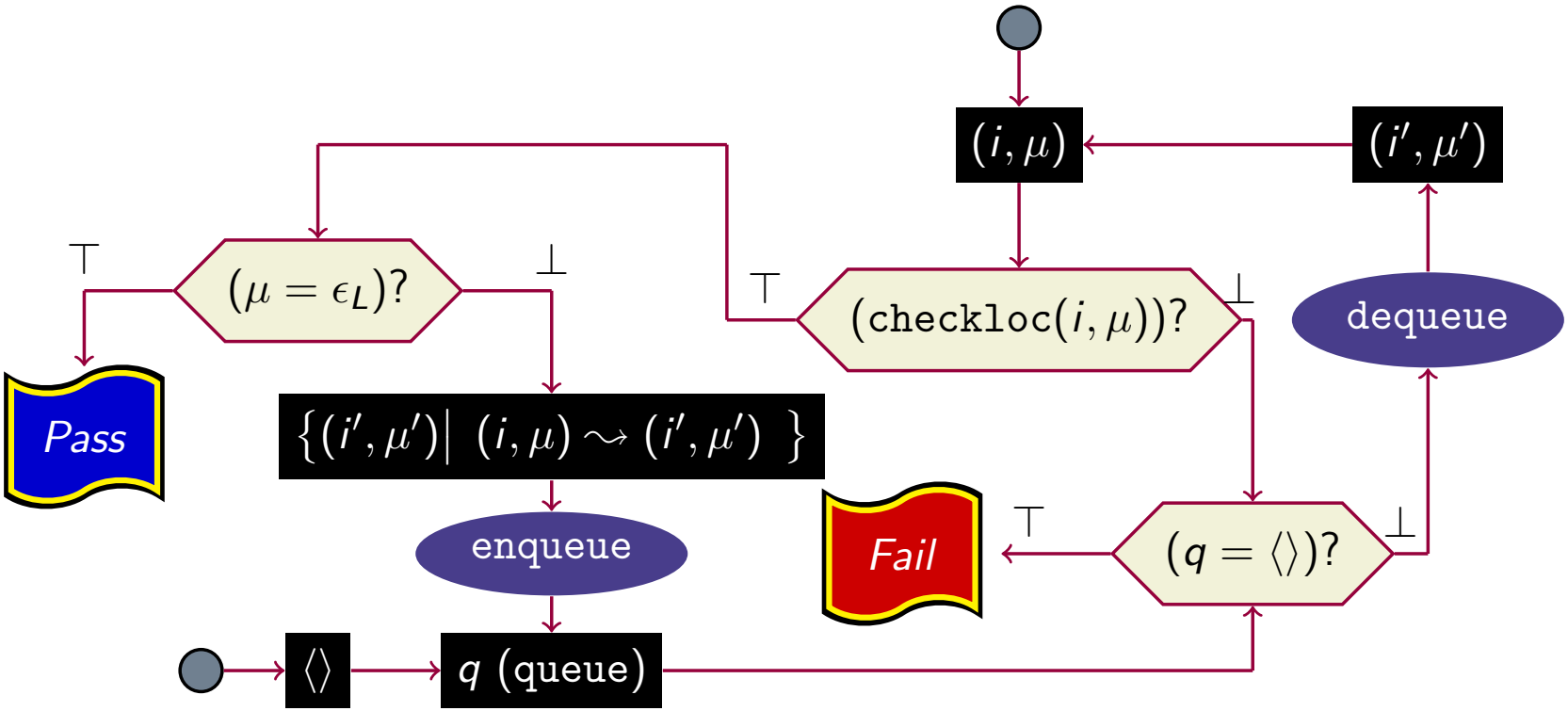


direct verdict

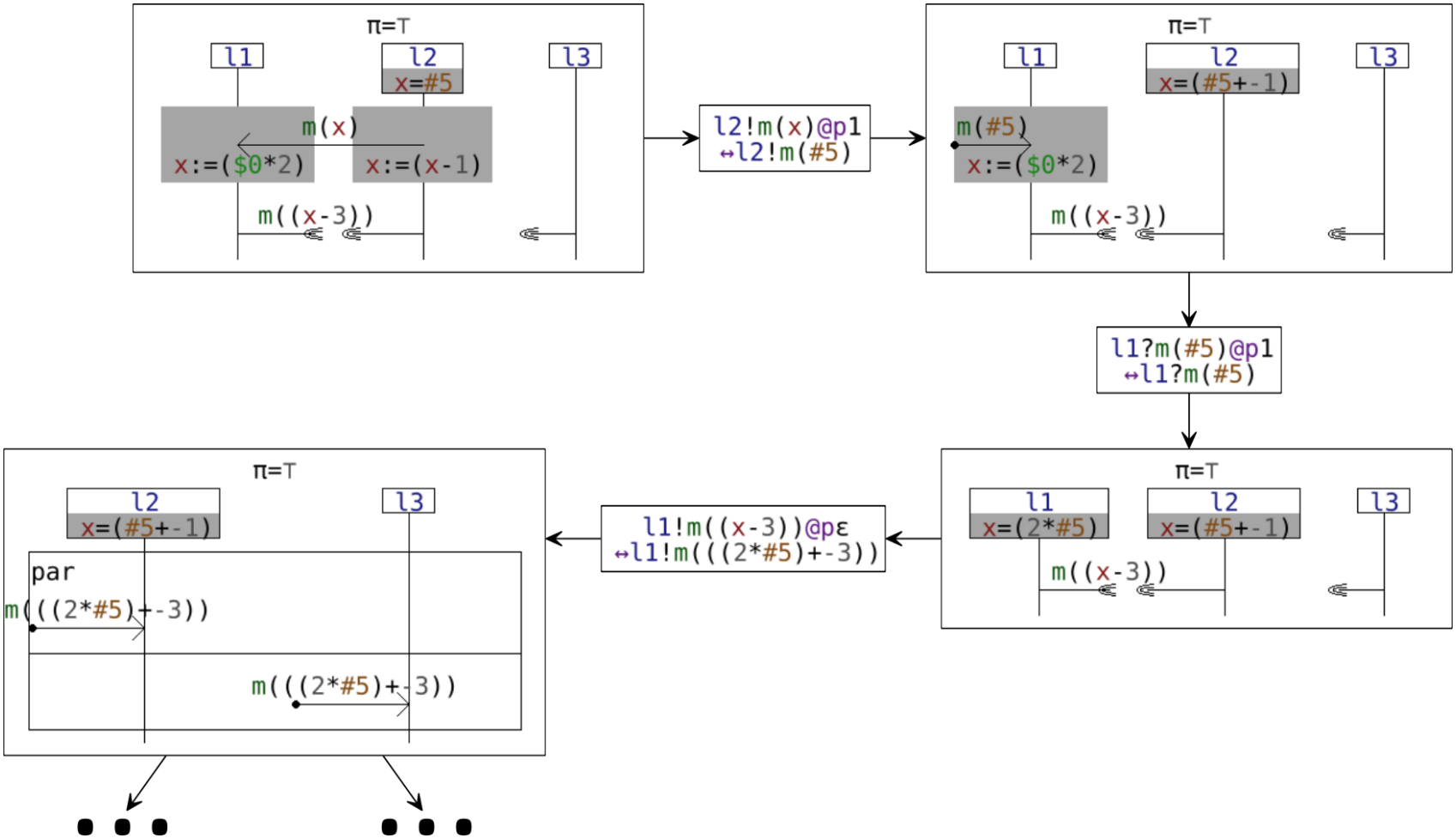
Local frontiers

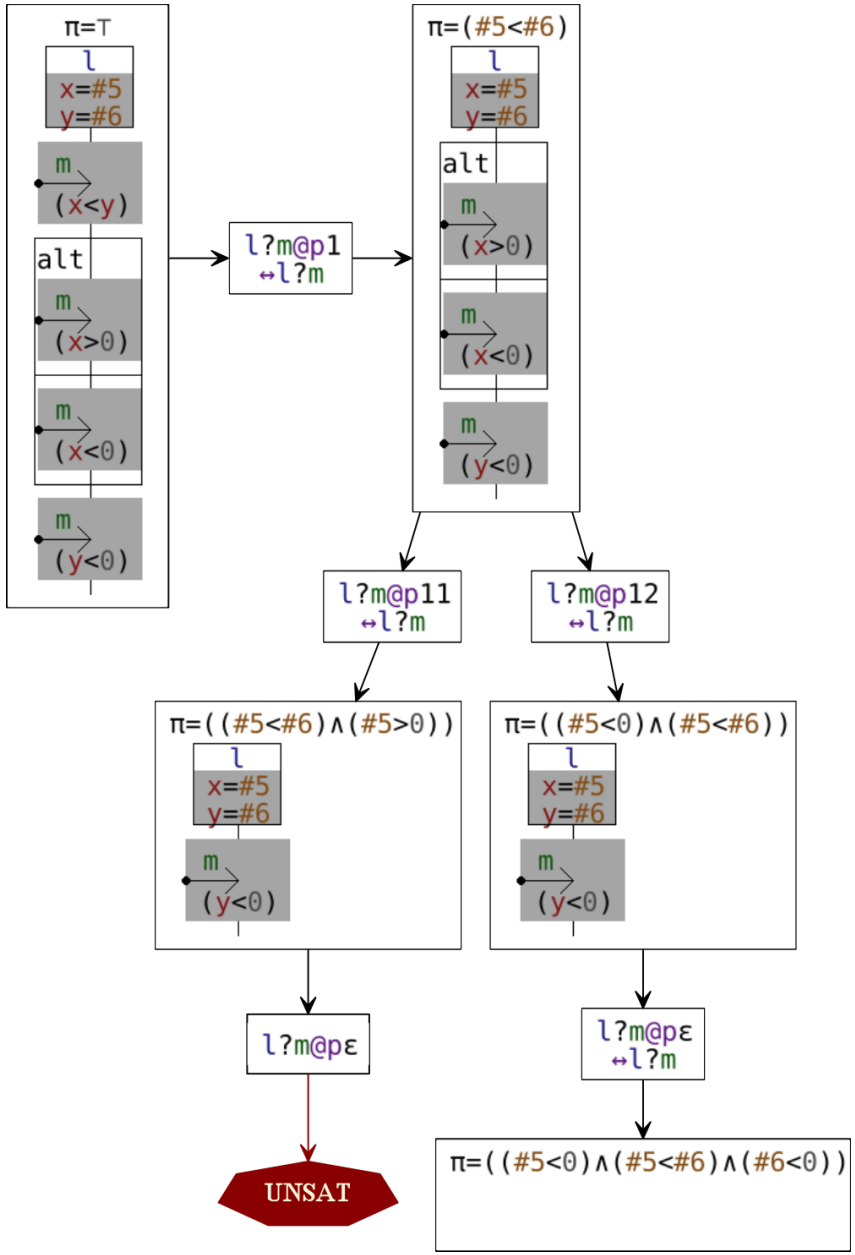


Algorithm with the checking of local frontiers

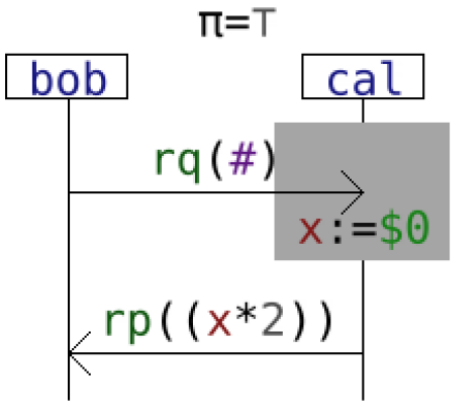


Value passing & symbolic execution




π=⊤
l
x=#5
y=#6
m
(x<y)
alt
m
(x>0)
m
(x<0)
m
(y<0)
l?m@p1
↔l?m
π=(#5<#6)
l
x=#5
y=#6
alt
m
(x>0)
m
(x<0)
m
(y<0)
l?m@p11
↔l?m
l?m@p12
↔l?m
π=((#5<#6)∧(#5>0))
l
x=#5
y=#6
m
(y<0)
π=((#5<0)∧(#5<#6))
l
x=#5
y=#6
m
(y<0)
l?m@pε
l?m@pε
↔l?m
UNSAT
π=((#5<0)∧(#5<#6)∧(#6<0))

Analyses with data



Accepted behavior

```
[bob] bob!rq(4);
[cal] cal?rq(4).cal!rp(8)
```

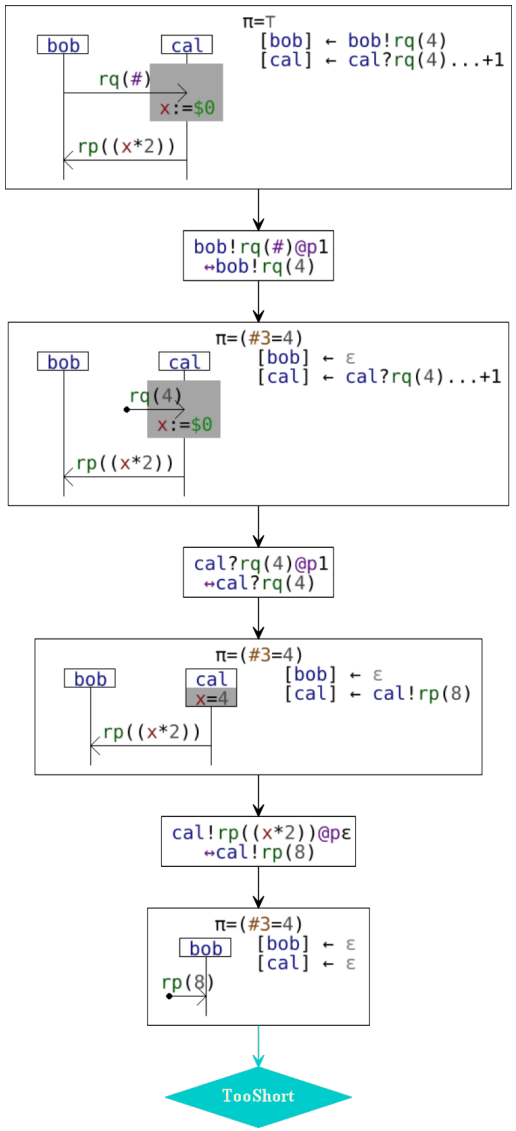
Wrong computation

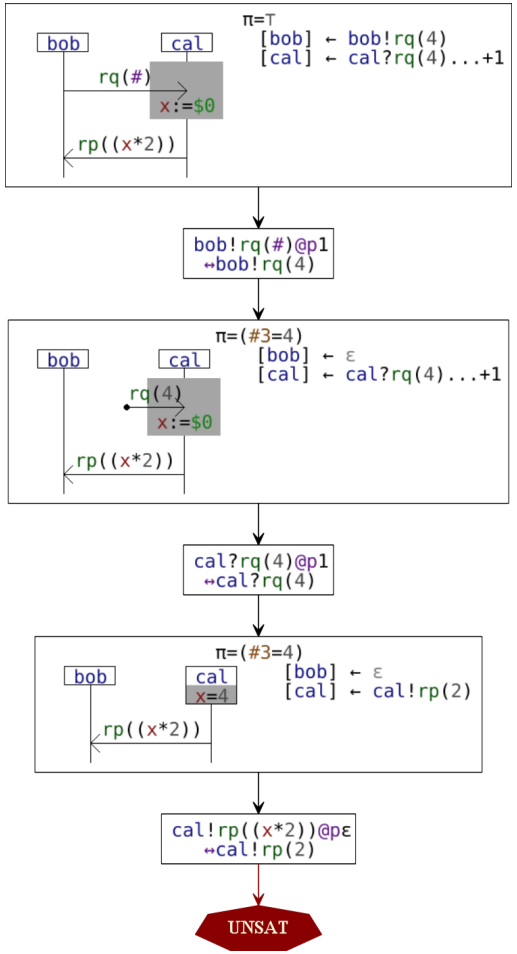
```
[bob] bob!rq(4);
[cal] cal?rq(4).cal!rp(2)
```

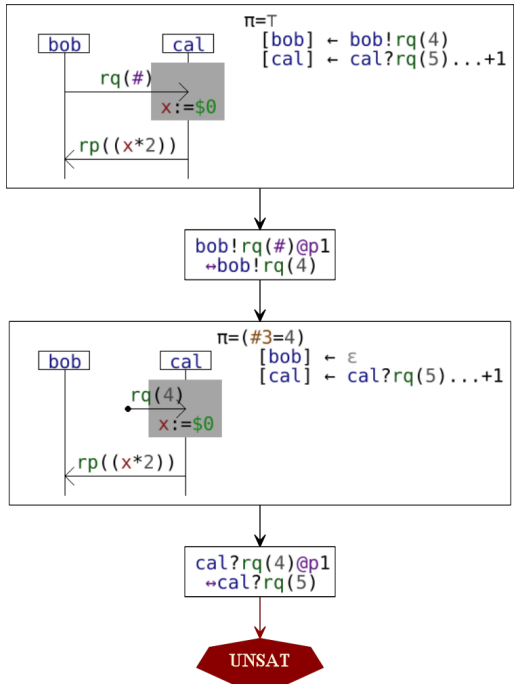
Man-in-the-middle

```
[bob] bob!rq(4);
[cal] cal?rq(5).cal!rp(10)
```

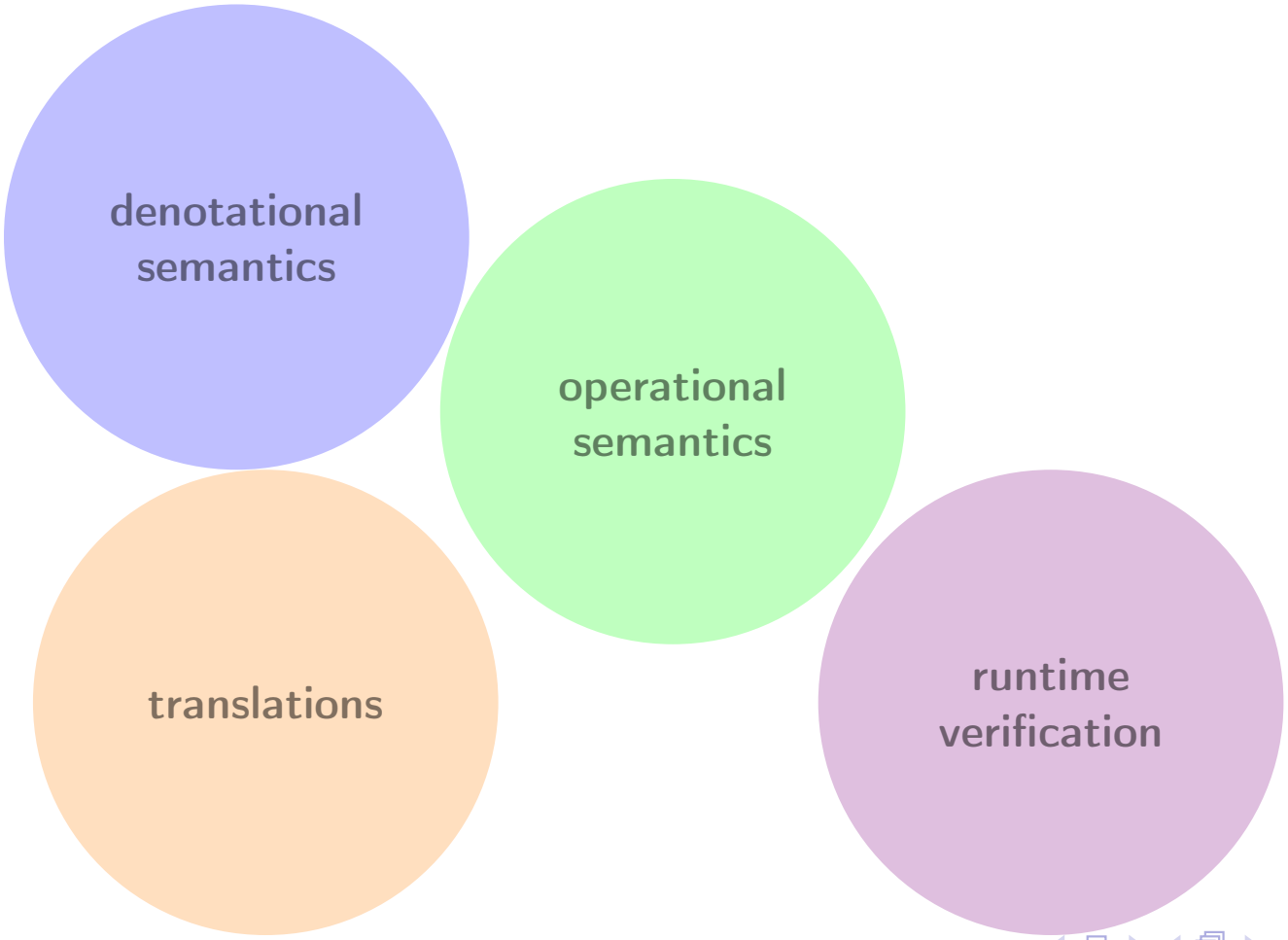
Analyses with data



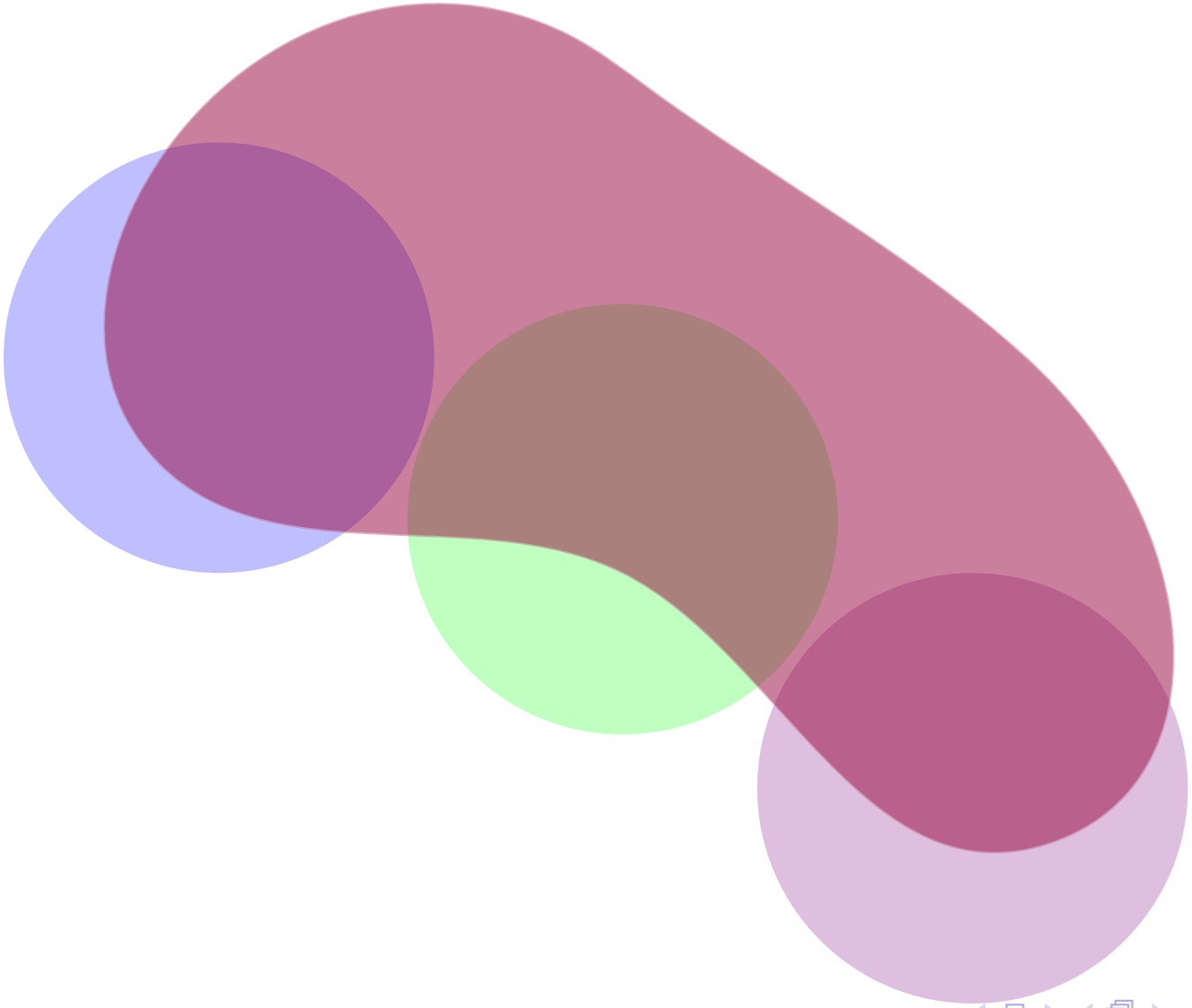




Position and main results of the thesis



Position and main results of the thesis



Position and main results of the thesis