On the Verification of Structured Parameterized Networks

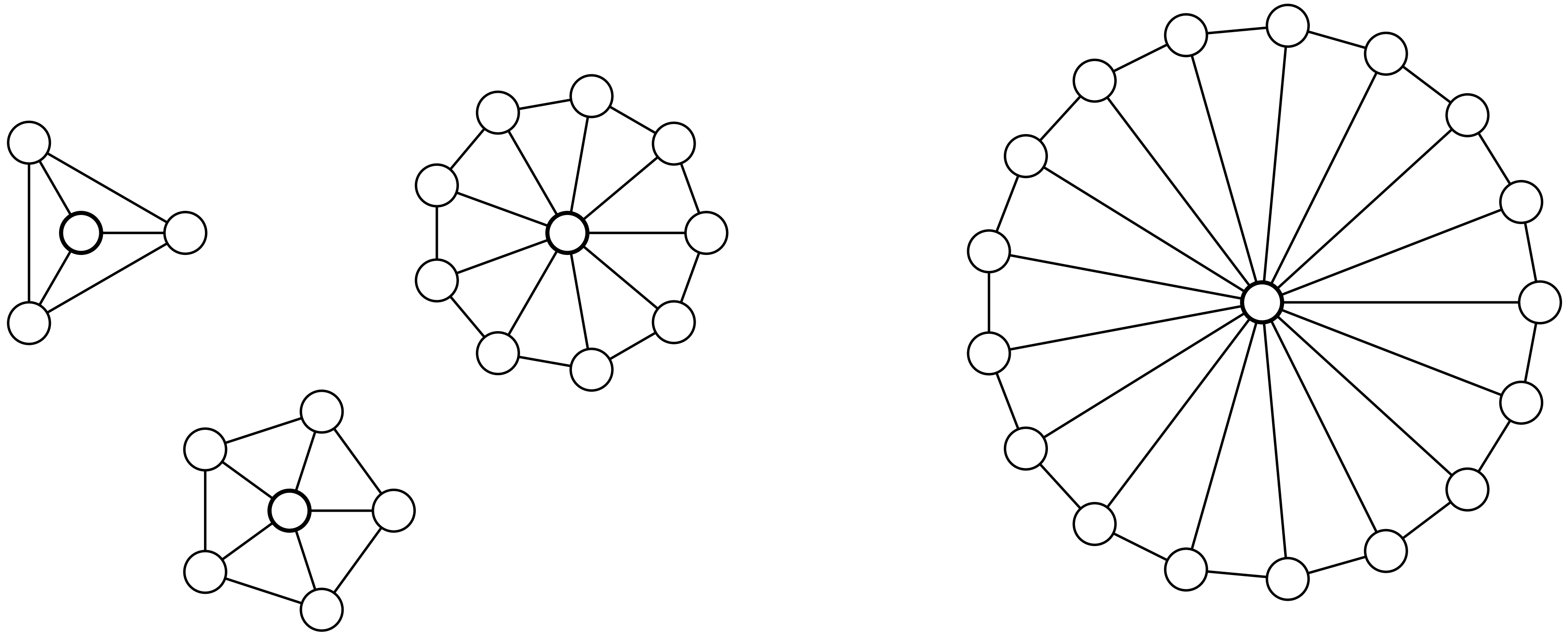
Neven Villani[1] , Marius Bozga[1], Radu Iosif[1], Arnaud Sangnier[2]

AVM'25 @ Timisoara; using material from NETYS'25, CAV'25, and work in progress

2025-09-24

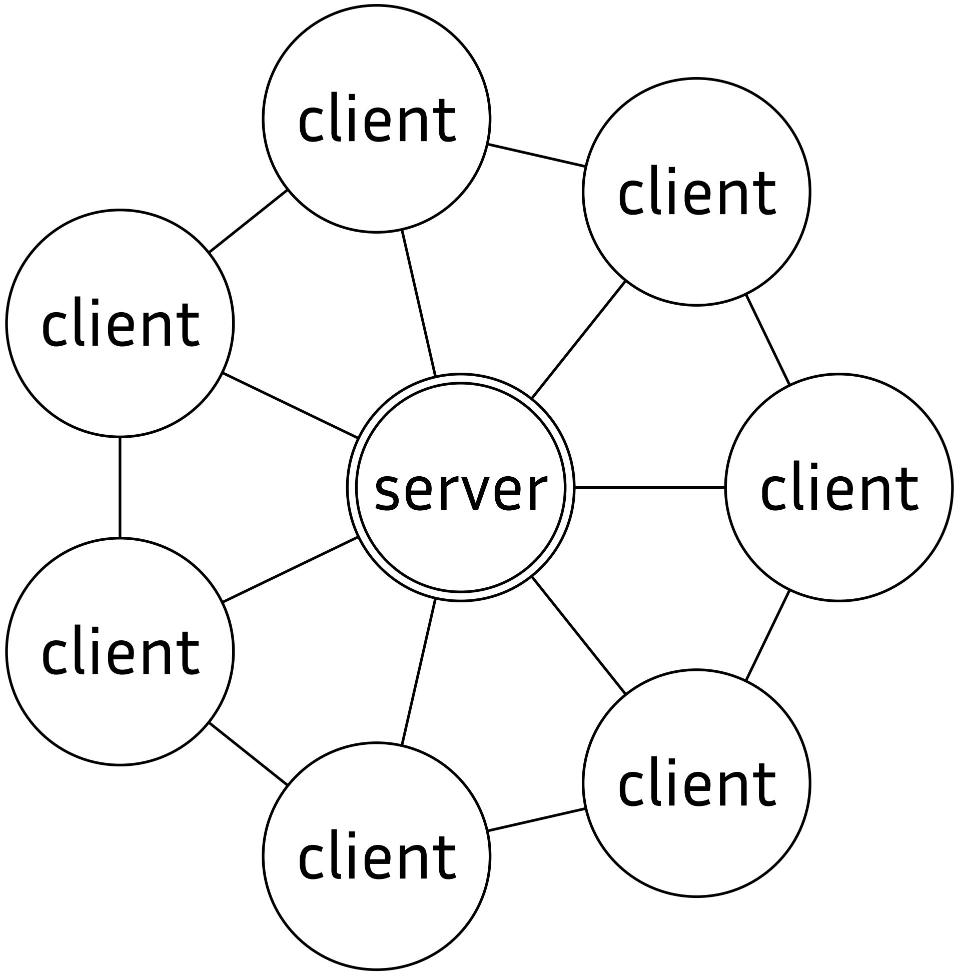
[1]VERIMAG, Univ. Grenoble Alpes, CNRS
[2]DIBRIS, Univ. di Genoa

- (automated) verification of networks
- challenge: size and architecture (communication topology)

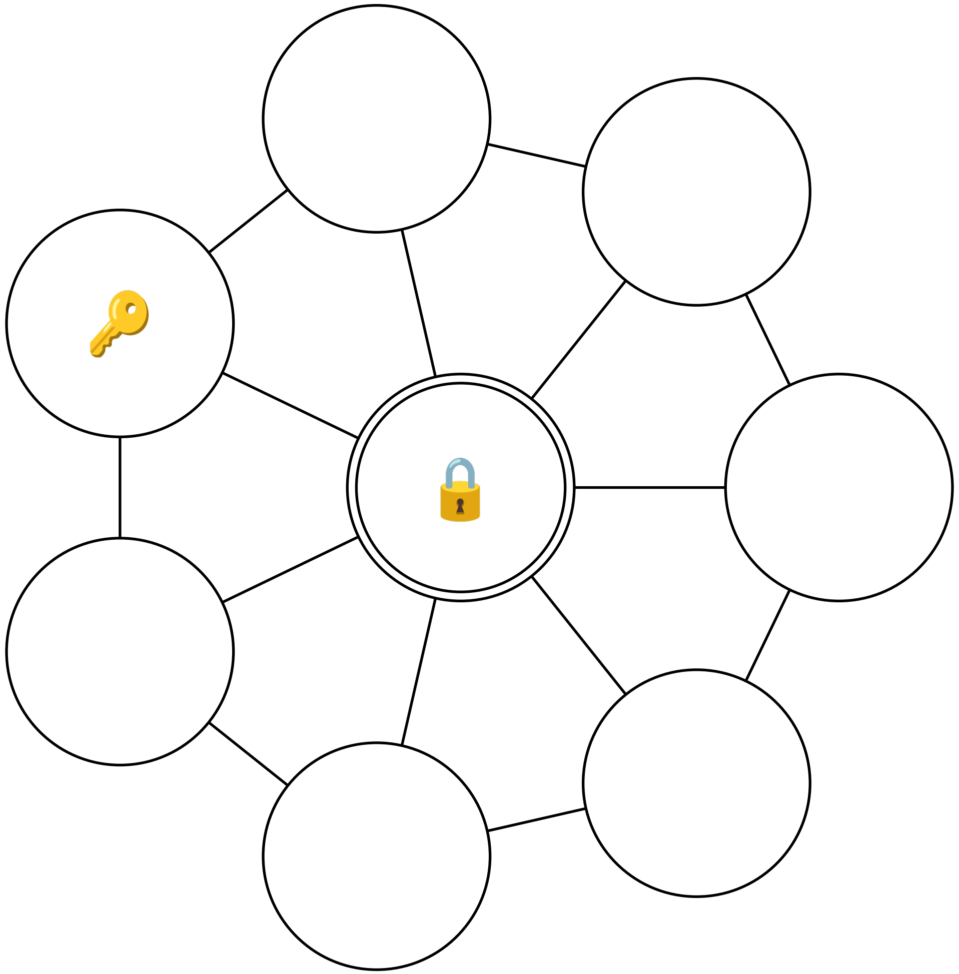


- undecidable $\rightsquigarrow$ abstraction

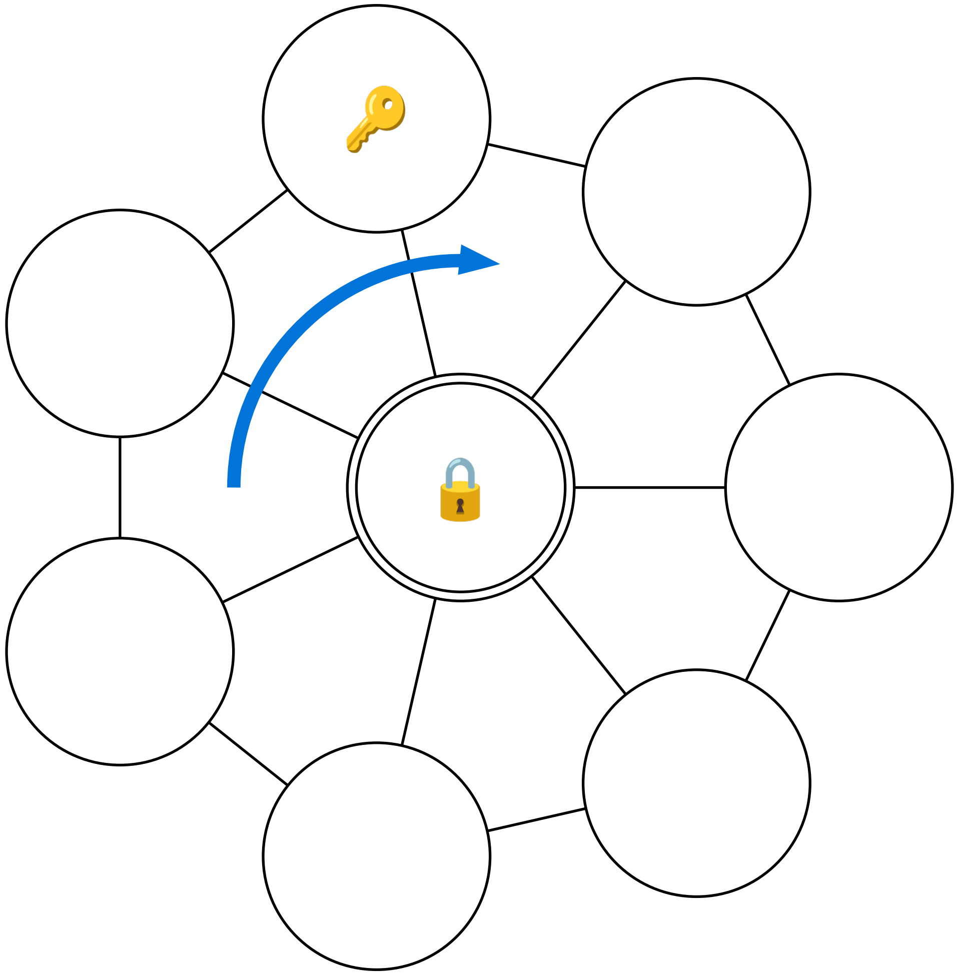
Token ring with resource



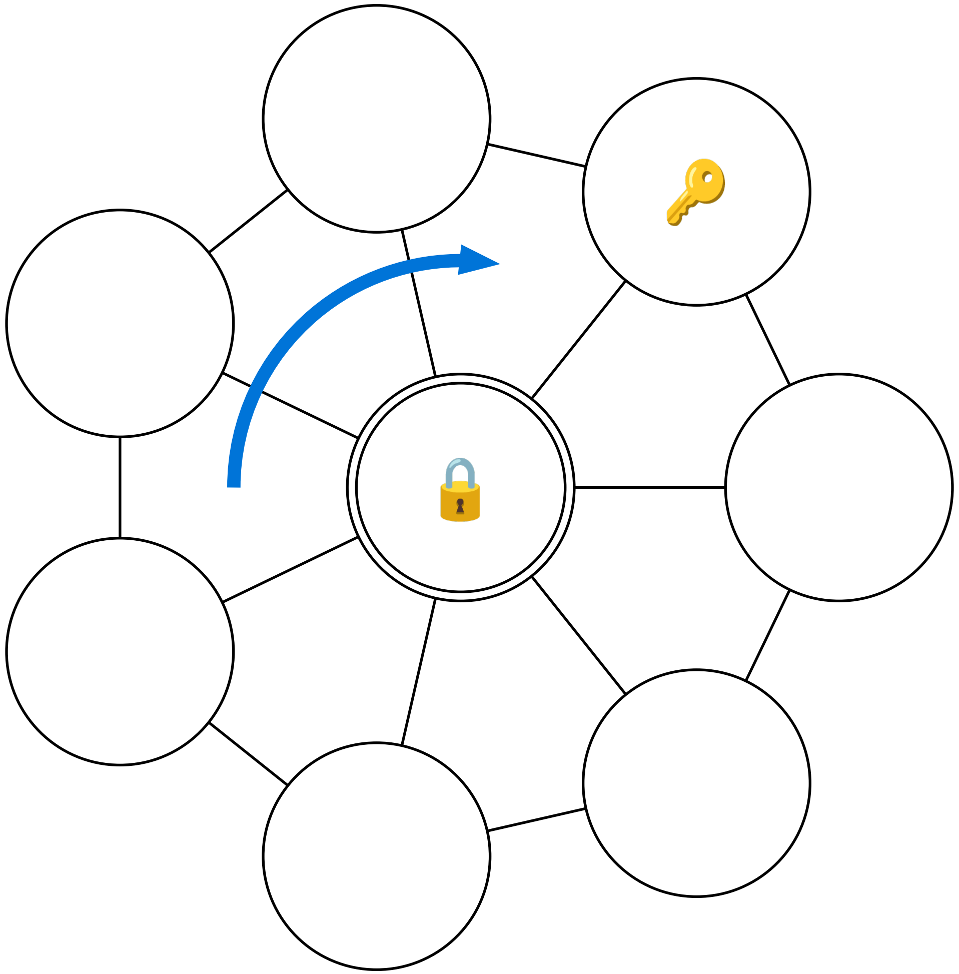
Token ring with resource



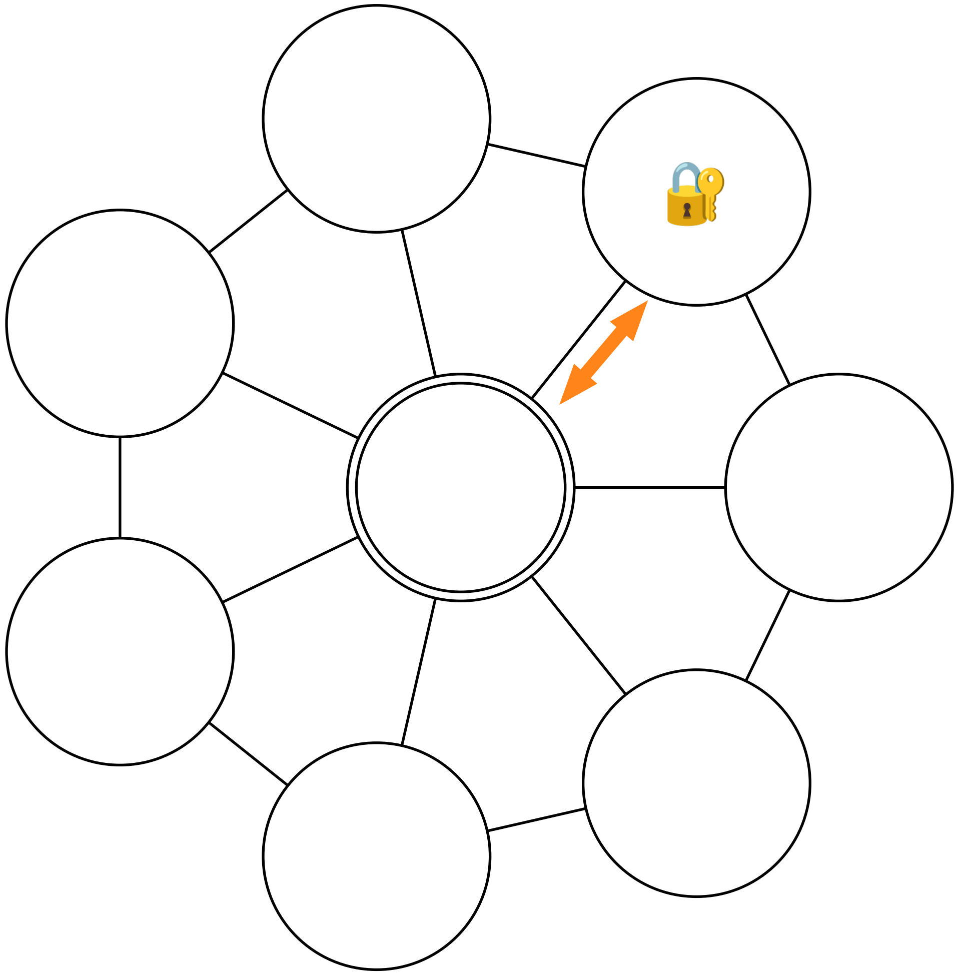
Token ring with resource



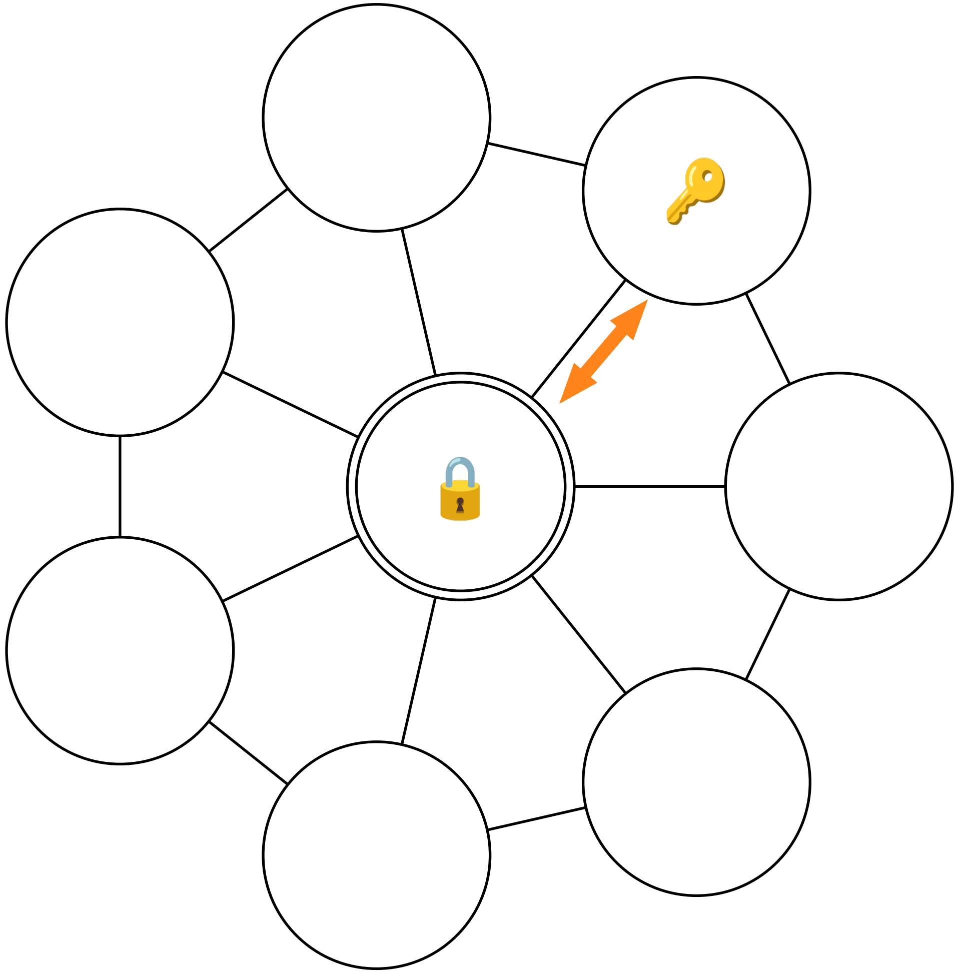
Token ring with resource



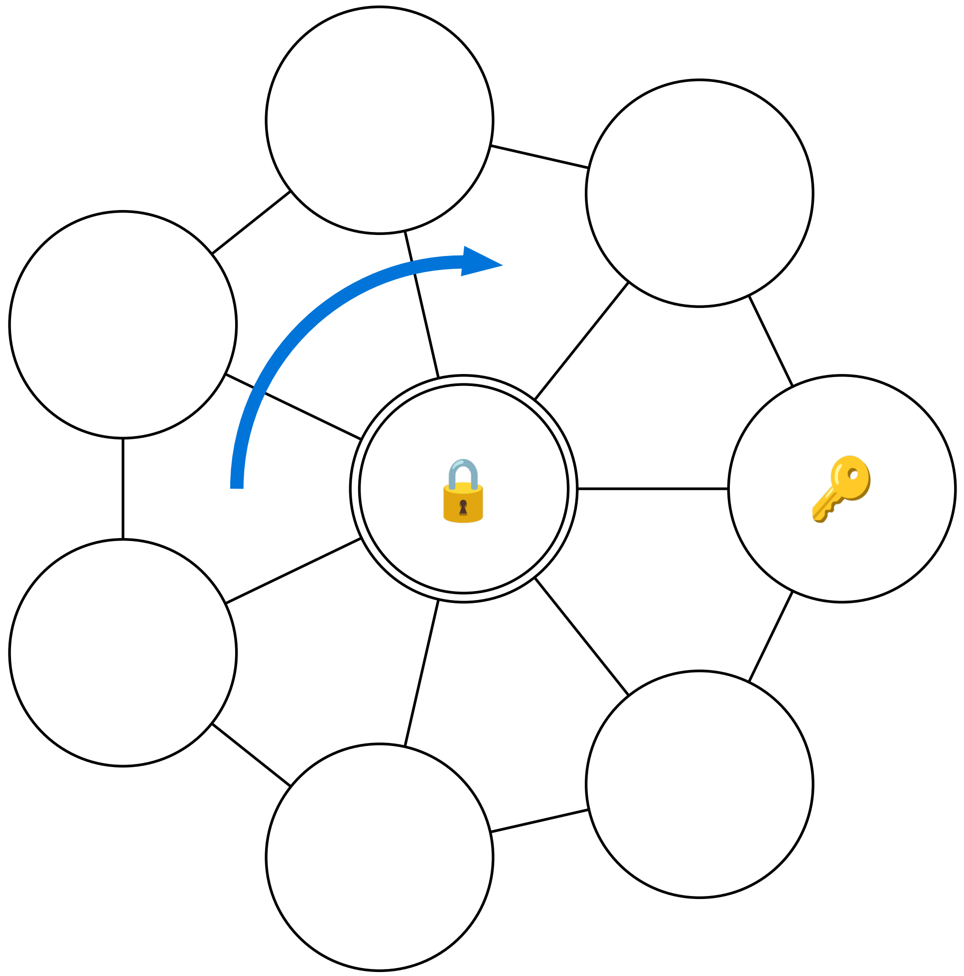
Token ring with resource



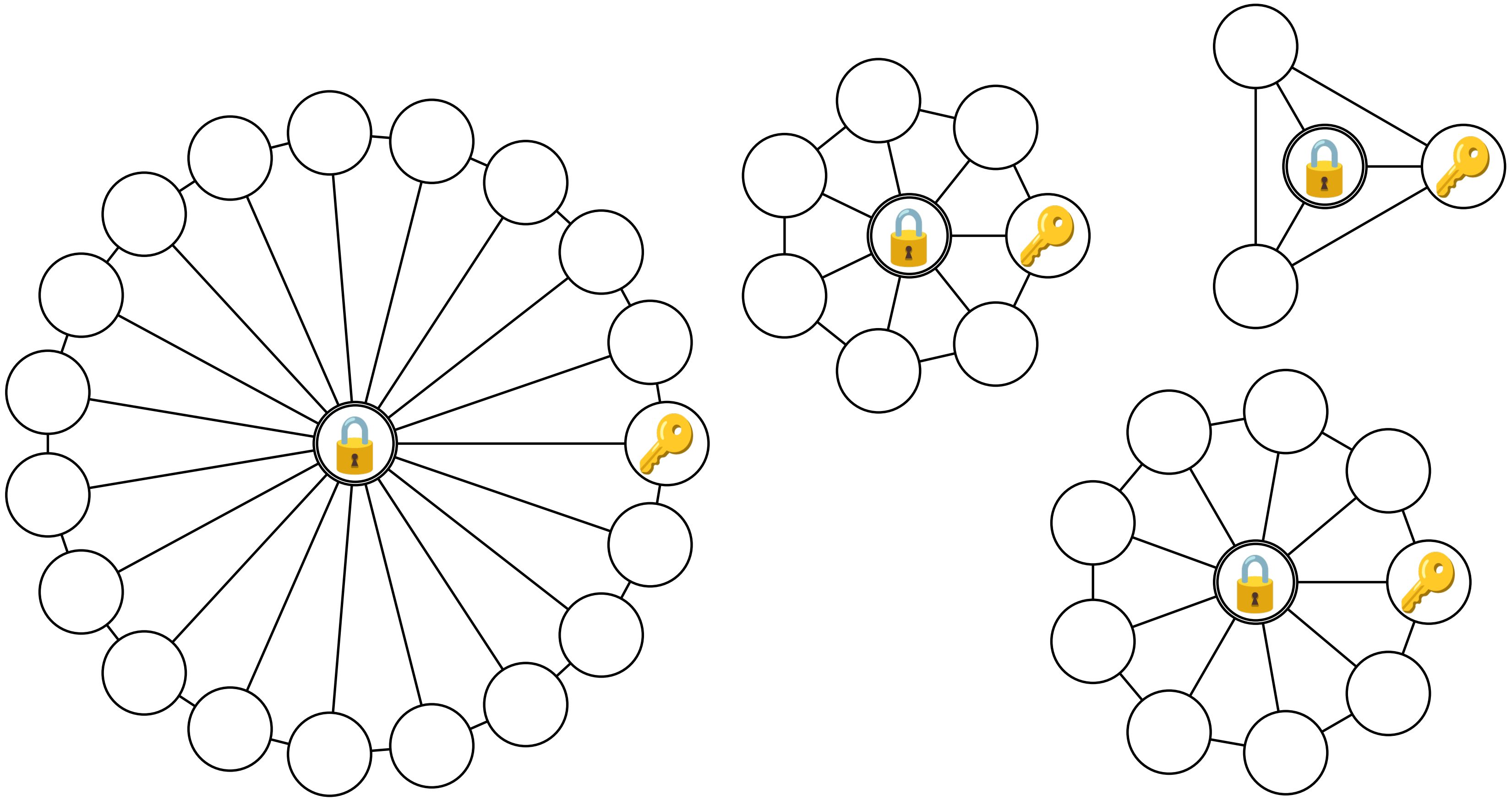
Token ring with resource



Token ring with resource



Token ring with resource



$$\forall n \geq 2$$

How would we automatically verify this ?

Techniques for non-finite-state systems...

Parameterized model checking of rendezvous systems

(B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith)

❌ not homogeneous (2 kinds of processes)

How would we automatically verify this ?

Techniques for non-finite-state systems...

Parameterized model checking of rendezvous systems

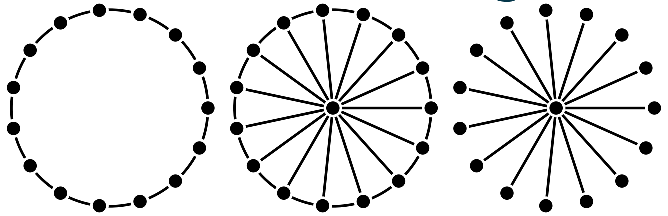
(B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith)

❌ not homogeneous (2 kinds of processes)

Parameterized Verification of Algorithms for Oblivious Robots on a Ring

(A. Sangnier, N. Sznajder, M. Potop-Butucaru, S. Tixeuil)

❌ not a standard architecture (clique, ring, star)



How would we automatically verify this ?

Techniques for non-finite-state systems...

Parameterized model checking of rendezvous systems

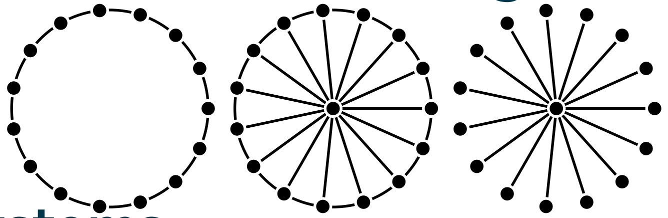
(B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith)

❌ not homogeneous (2 kinds of processes)

Parameterized Verification of Algorithms for Oblivious Robots on a Ring

(A. Sangnier, N. Sznajder, M. Potop-Butucaru, S. Tixeuil)

❌ not a standard architecture (clique, ring, star)



Parameterized Model Checking of Token-Passing Systems

(B. Aminof, S. Jacobs, A. Khalimov, S. Rubin)

❌ not a token-passing system (key and lock don't behave like tokens)

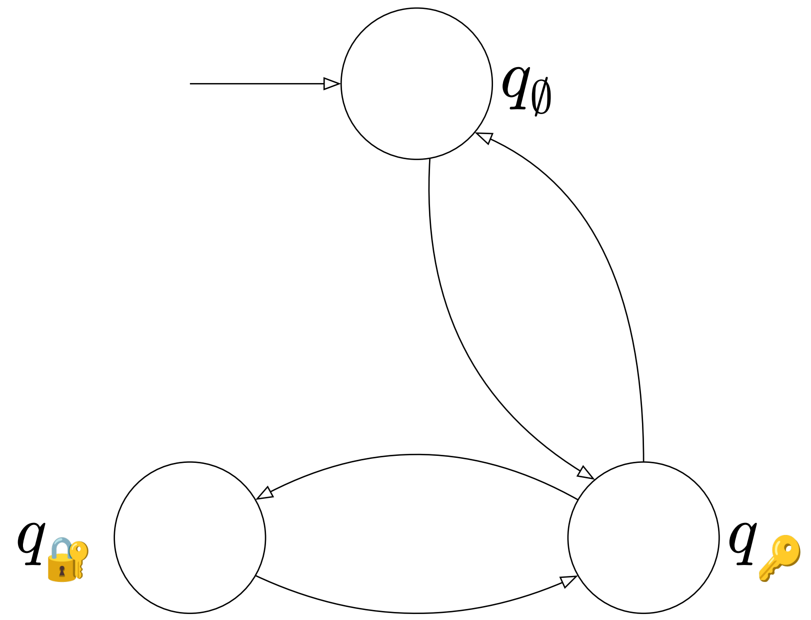
Framework requirements

We must be able to express

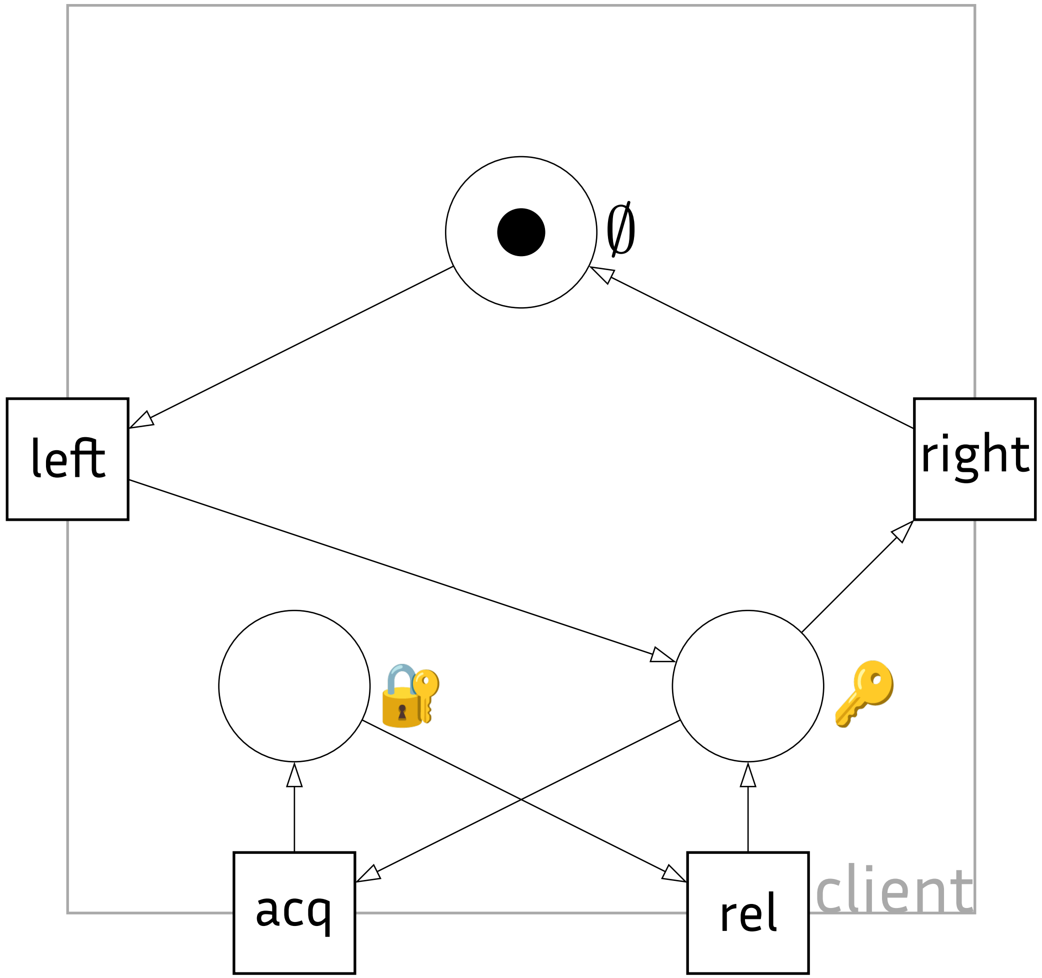
- an encoding of the **local behavior** of processes
- a description of the **interactions** and **architectures** of arbitrary size
- a **specification language** for safety properties

Framework

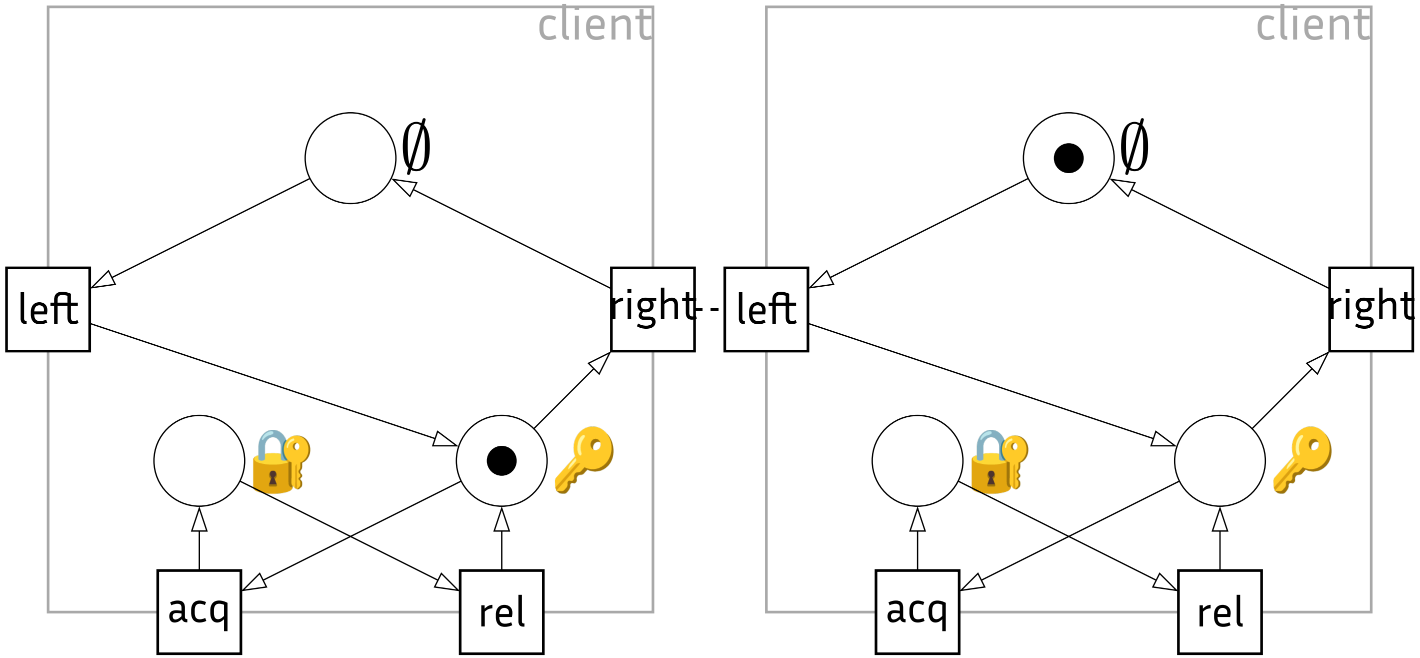
Client



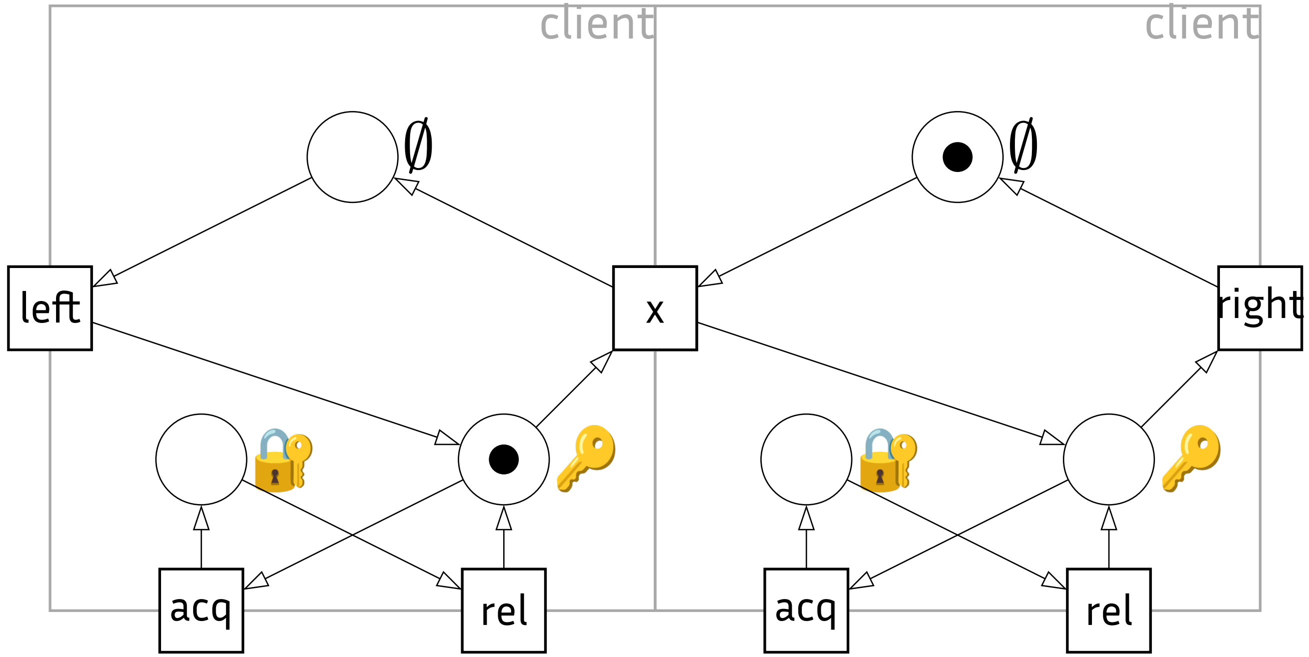
Client



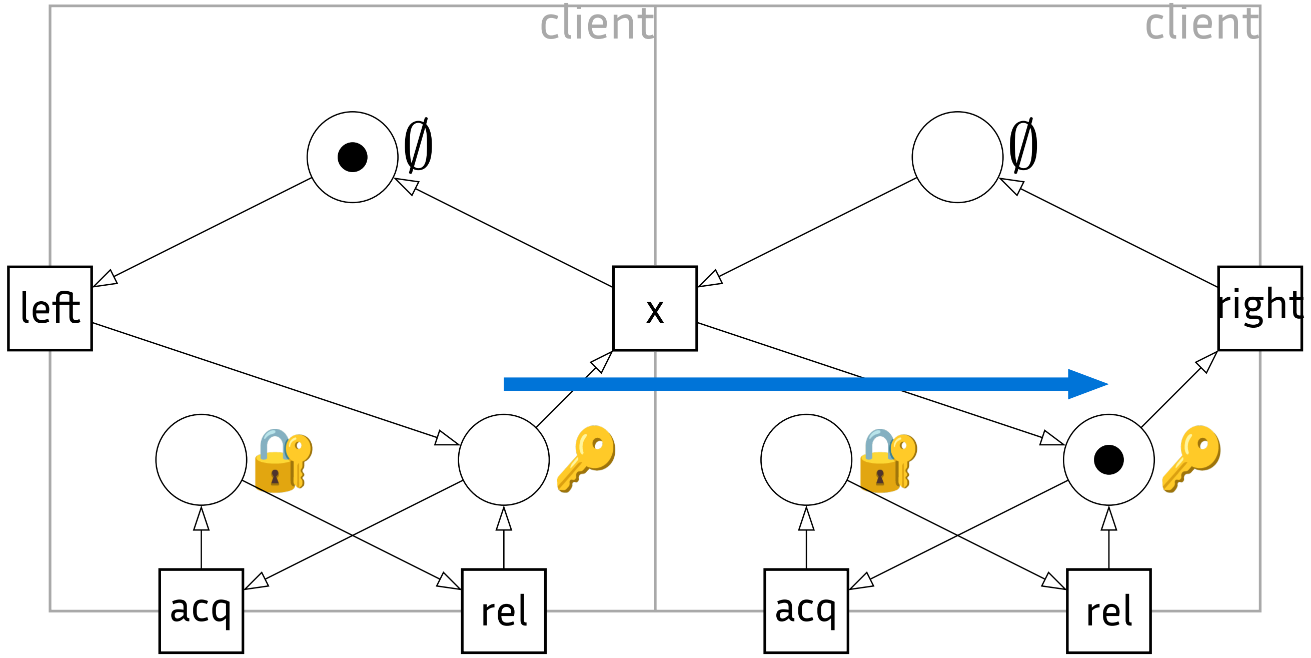
Interactions through composition



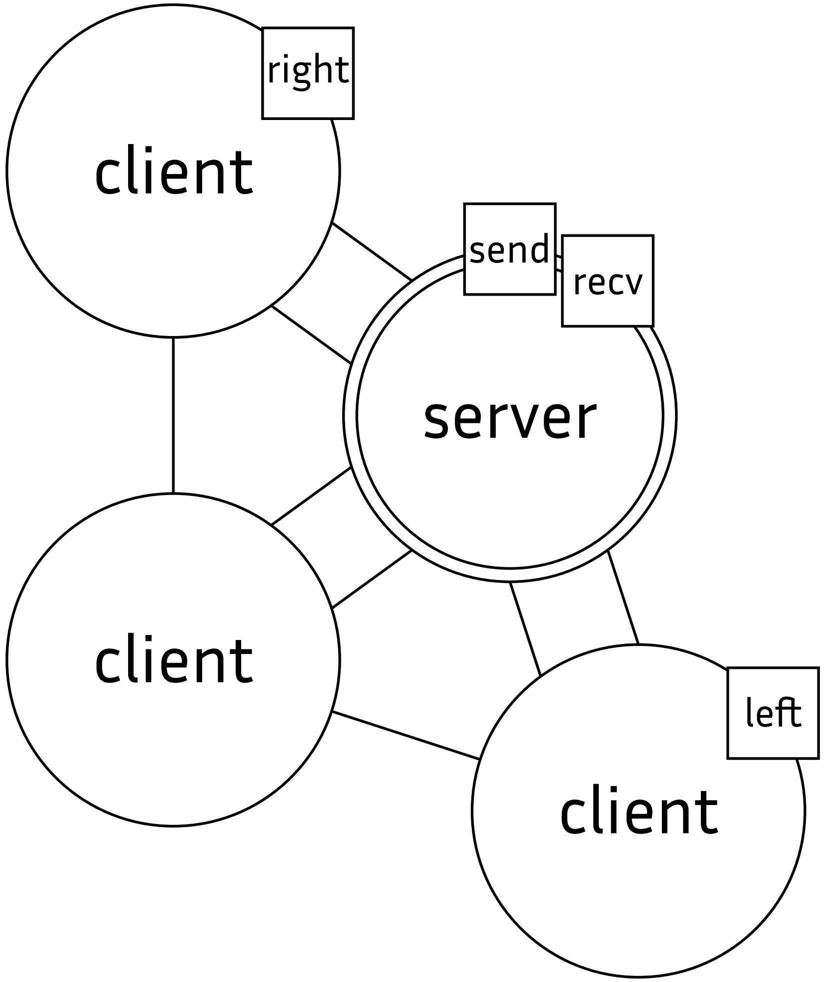
Interactions through composition



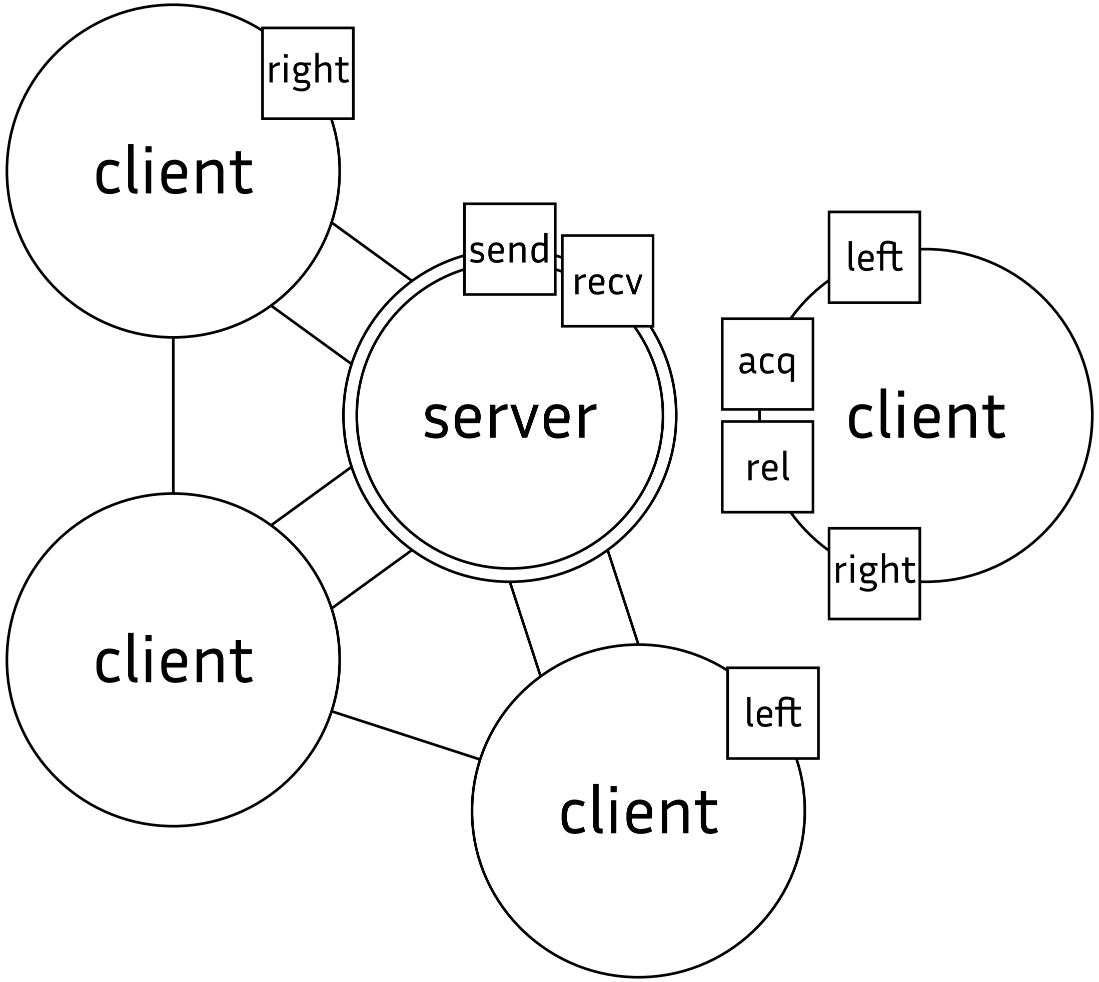
Interactions through composition



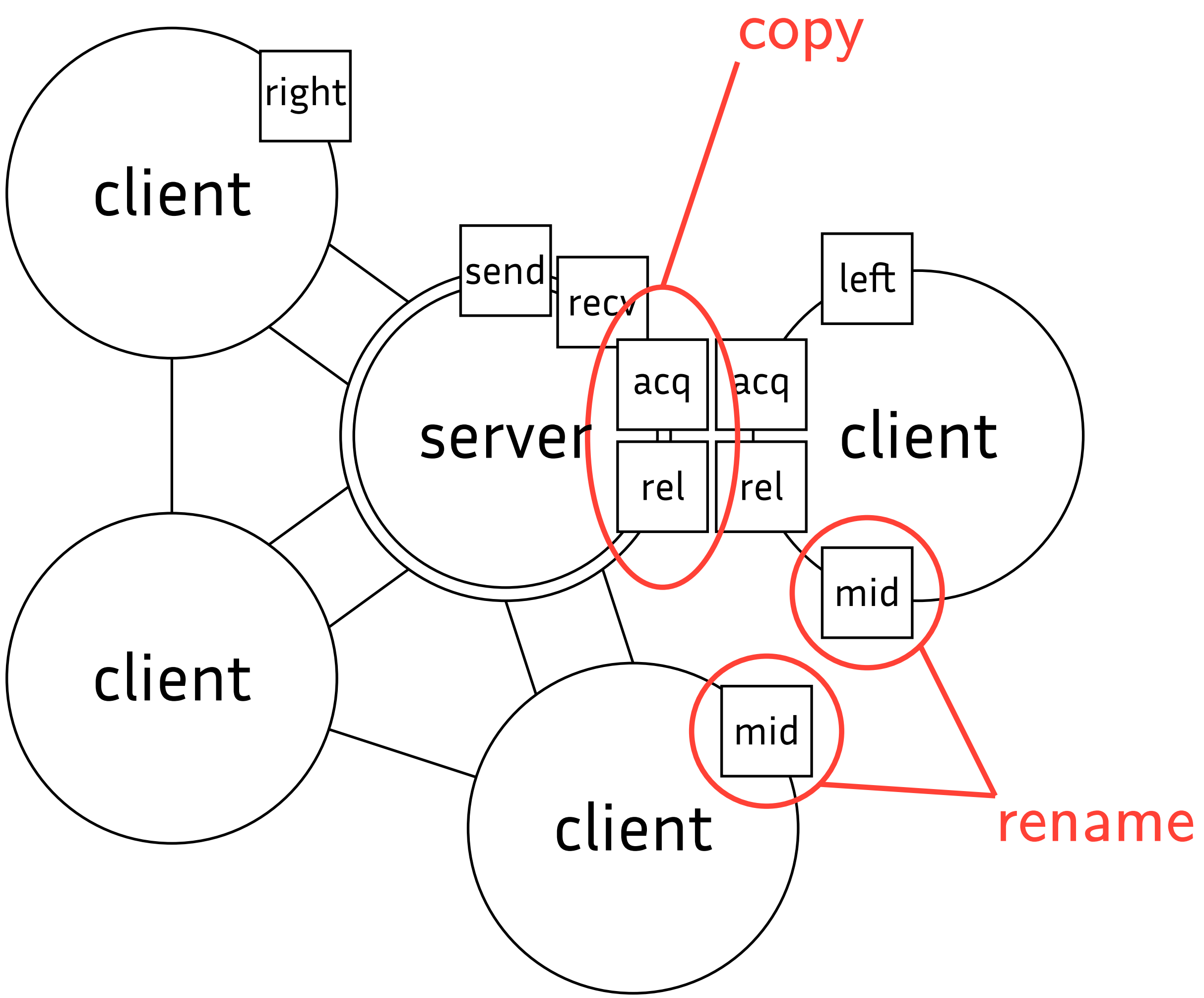
Structure: inductive step



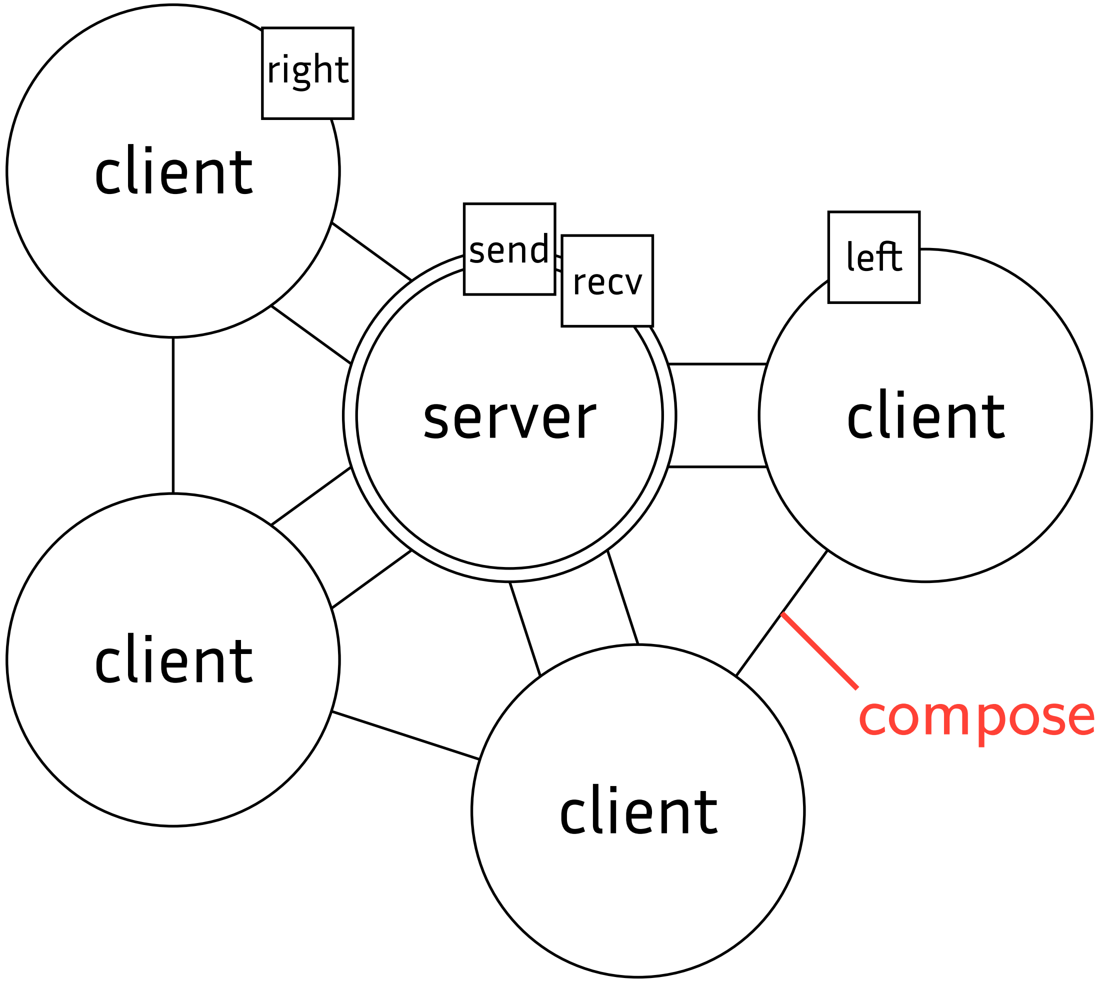
Structure: inductive step



Structure: inductive step



Structure: inductive step

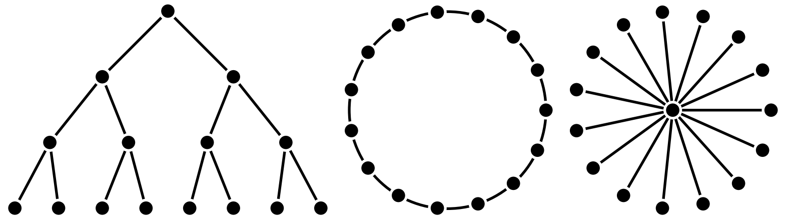


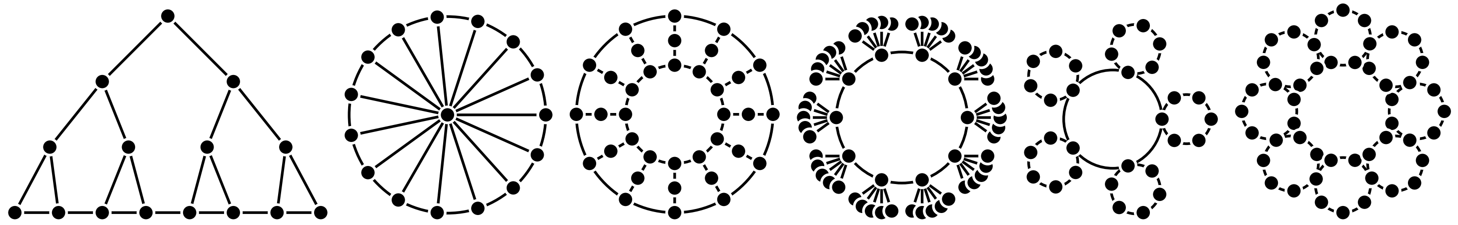
Structure represented by a grammar

$$
\begin{aligned}
\mathrm{X} \longrightarrow \mathsf{compose}\Big(& \\
&\mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)\big), \\
&\mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client}) \\
\Big)&
\end{aligned}
$$

Representable architectures

Encoded as a CFG for graphs[1] $\Longrightarrow$ families of bounded TW are representable (missing: ~~square grids~~, cliques)





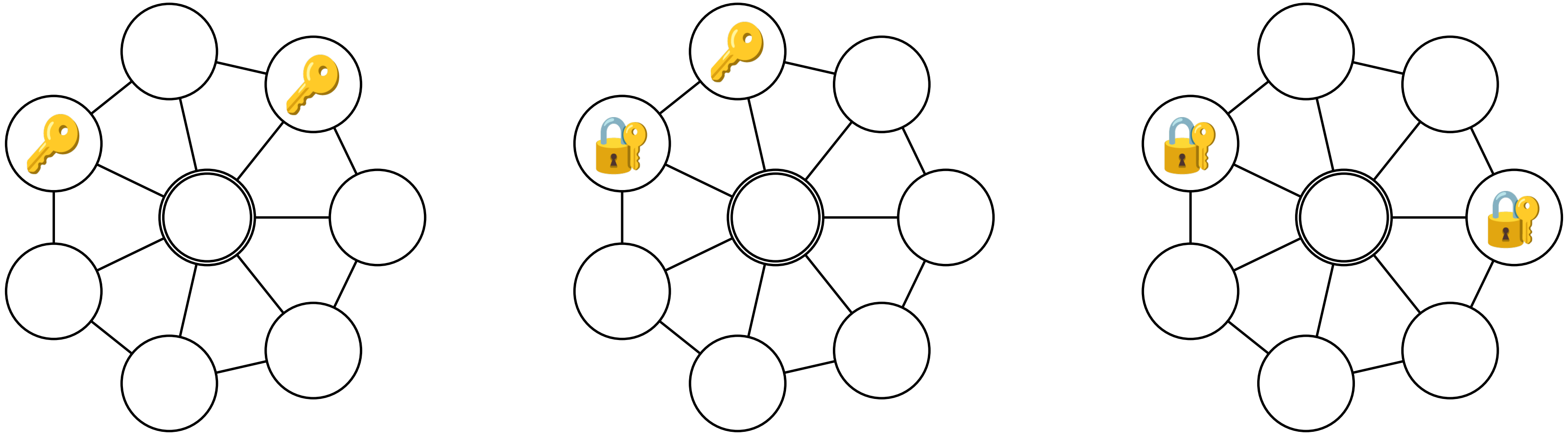
[1]Graph Structure and Monadic Second Order Logic; by B. Courcelle, J. Engelfriet

Safety specification

Safety properties

#(🔑): number of tokens on 🔑

$\sim$ number of clients who claim to own the key



$$\#(🔑) + \#(🔐) > 1$$

Proving safety $\approx$ reachability problem in an infinite family of PNs

Expressible Properties

- **mutual exclusion**
  "at most k processes can enter a critical section simultaneously"
- **uniqueness**
  "the entire system contains at most k instances of a resource"
- **uncoverability**
  "no process can reach a bad state"

Examples: leader election, semaphores, dining philosophers, …

Missing: ~~liveness~~, ~~deadlock freedom~~

An Abstraction Technique

Verification pipeline

$$\mathrm{Sys} \longrightarrow \mathsf{compose}\big(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')\big)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\big(\ \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)\big),\ \mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client})\ \big)$$

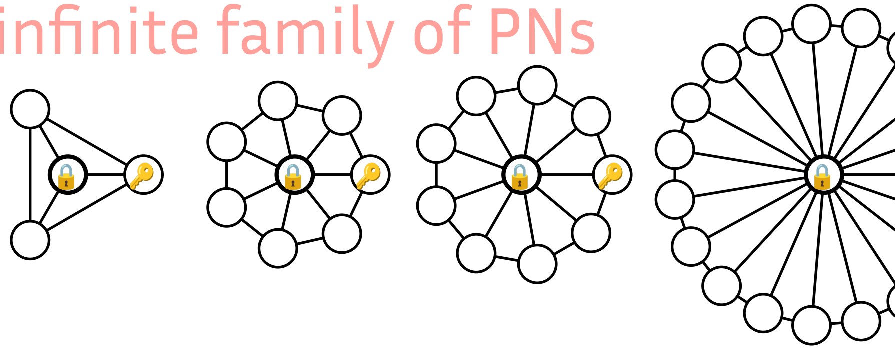
$$\mathrm{X} \longrightarrow \mathsf{compose}\big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(\mathrm{server}), \mathrm{client}\big)$$

Verification pipeline

$$
\begin{aligned}
\mathrm{Sys} &\longrightarrow \textsf{compose}\big(\mathrm{X}, \textsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')\big)\\
\mathrm{X} &\longrightarrow \textsf{compose}\big(\\
&\qquad \textsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\big(\textsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)\big),\\
&\qquad \textsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client})\\
&\ \big)\\
\mathrm{X} &\longrightarrow \textsf{compose}\big(\textsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(\mathrm{server}), \mathrm{client}\big)
\end{aligned}
$$

$$\xrightarrow{\text{language}}$$

infinite family of PNs



Verification pipeline

$$\mathrm{Sys} \longrightarrow \mathsf{compose}\big(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')\big)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\big($$

$$\quad \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)\big),$$

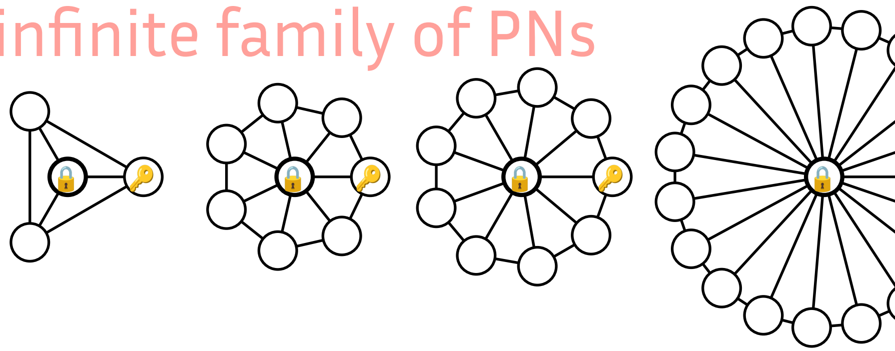
$$\quad \mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client})$$

$$\big)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(\mathrm{server}), \mathrm{client}\big)$$

$\xrightarrow{\text{language}}$

infinite family of PNs



$\Big\downarrow$ safety

$$\boxed{\#(🔑) + \#(🔐) > 1}$$

Verification pipeline

$\text{Sys} \longrightarrow \mathsf{compose}(\text{X}, \mathsf{rename}_{\text{left}\mapsto\text{right},\ \text{right}\mapsto\text{left}}(\text{client}'))$

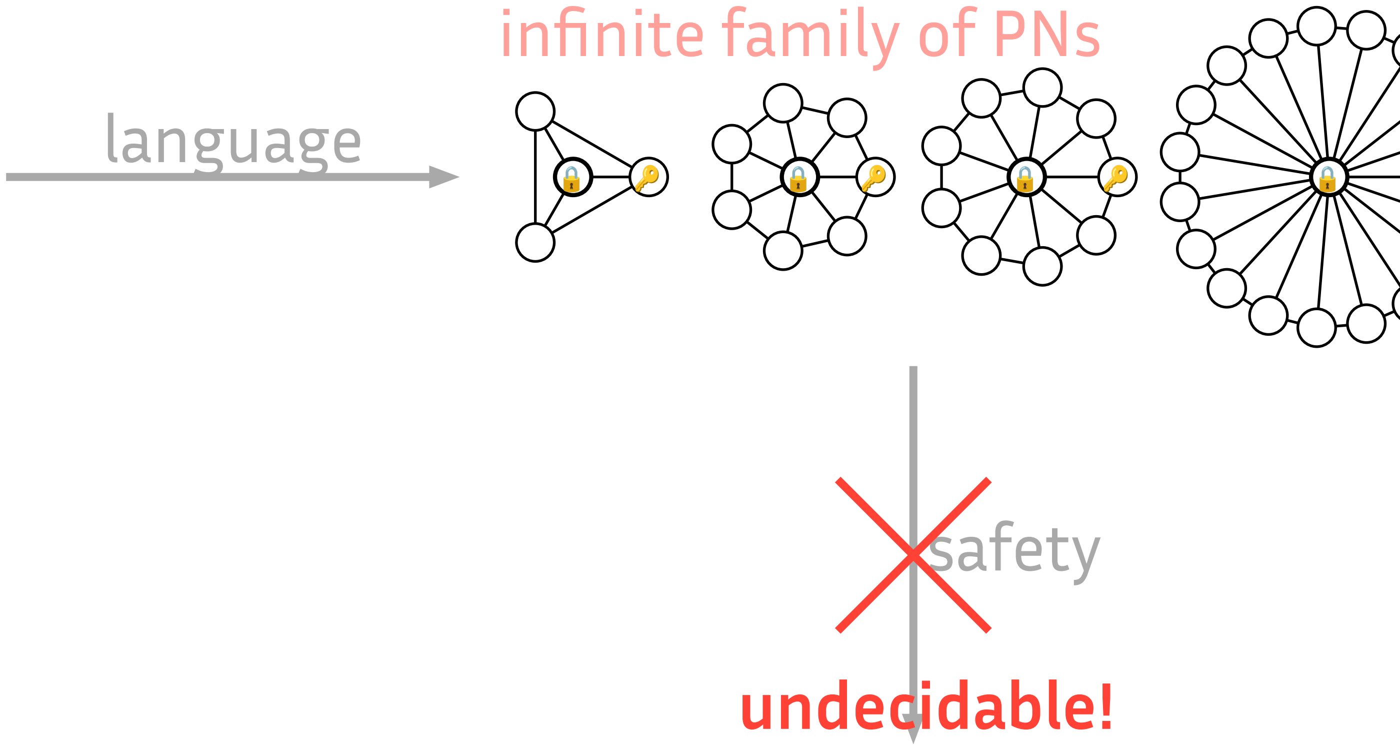
$\text{X} \longrightarrow \mathsf{compose}($

$\quad \mathsf{rename}_{\text{left}\mapsto\text{mid}}(\mathsf{copy}_{\text{send}\rightsquigarrow\text{acq},\text{recv}\rightsquigarrow\text{rel}}(X)),$

$\quad \mathsf{rename}_{\text{right}\mapsto\text{mid}}(\text{client})$

$)$

$\text{X} \longrightarrow \mathsf{compose}(\mathsf{copy}_{\text{send}\rightsquigarrow\text{acq},\text{recv}\rightsquigarrow\text{rel}}(\text{server}), \text{client})$



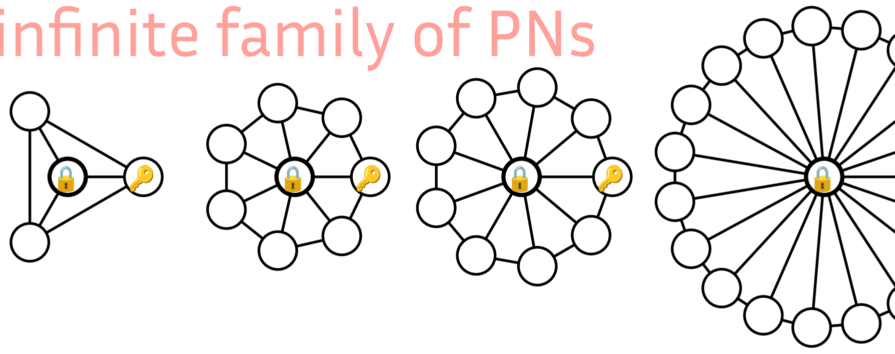
$\boxed{\#(🔑) + \#(🔐) > 1}$

Verification pipeline

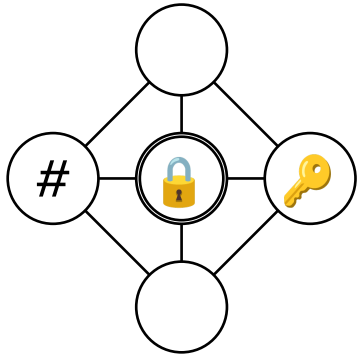
$$
\begin{aligned}
\mathrm{Sys} &\longrightarrow \mathsf{compose}(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}'))\\
\mathrm{X} &\longrightarrow \mathsf{compose}(\\
&\qquad \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)),\\
&\qquad \mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client})\\
&\ )\\
\mathrm{X} &\longrightarrow \mathsf{compose}(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(\mathrm{server}), \mathrm{client})
\end{aligned}
$$

language

infinite family of PNs



abstraction



$\supseteq$

(soundness)

safety

undecidable!

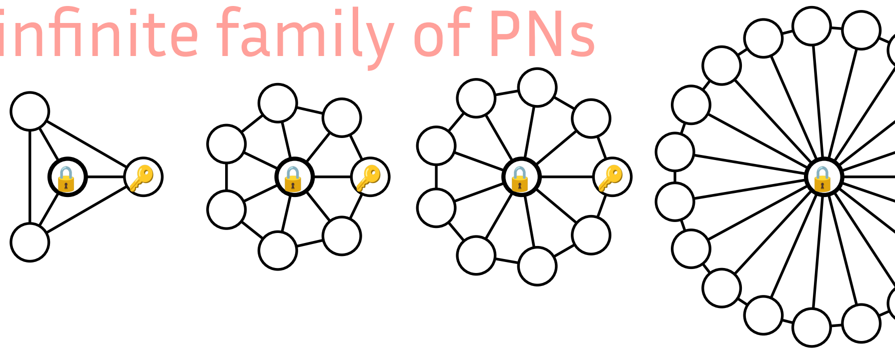
$\boxed{\#(🔑) + \#(🔐) > 1}$

Verification pipeline

$$
\begin{aligned}
\mathrm{Sys} &\longrightarrow \mathsf{compose}(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')) \\
\mathrm{X} &\longrightarrow \mathsf{compose}(\\
&\qquad \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)), \\
&\qquad \mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client}) \\
&\ \ ) \\
\mathrm{X} &\longrightarrow \mathsf{compose}(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(\mathrm{server}), \mathrm{client})
\end{aligned}
$$

language

infinite family of PNs



abstraction

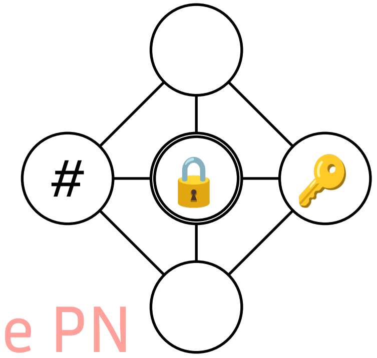
CAV Artifact Evaluation ★ ★ ★ Reusable

$\supseteq$

(soundness)

~~safety~~

undecidable!



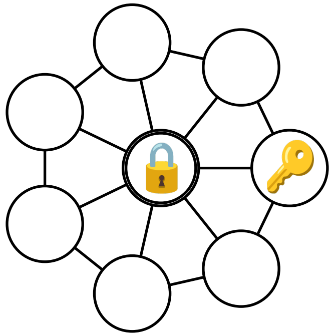
finite PN

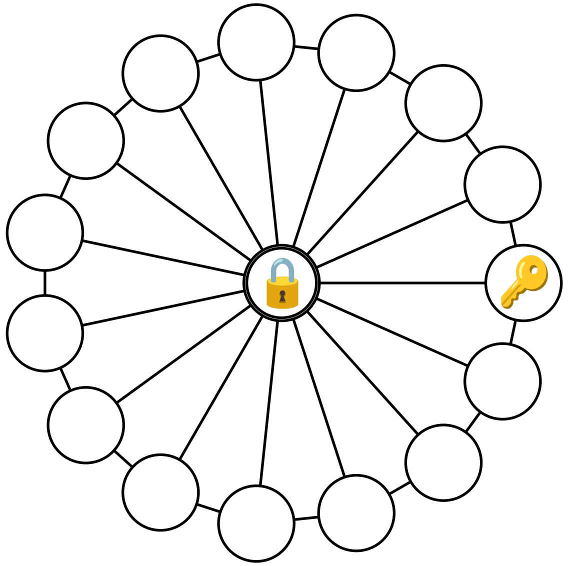
decidable

e.g. LoLA

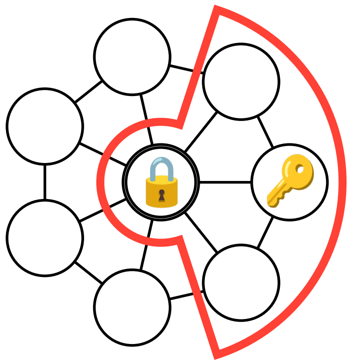
$\boxed{\#(🔑) + \#(🔐) > 1}$

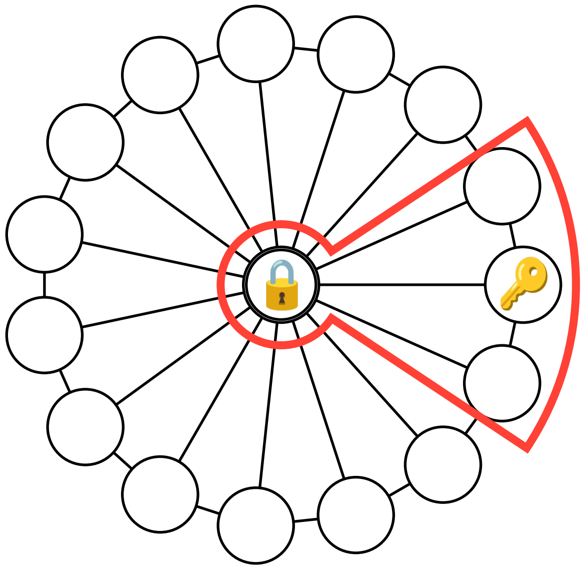
Folding abstraction



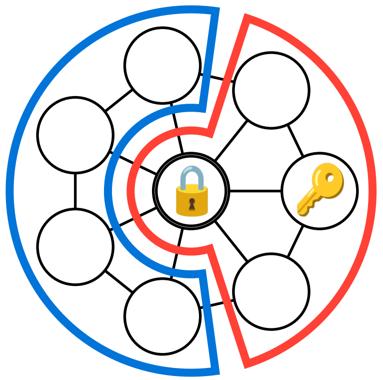


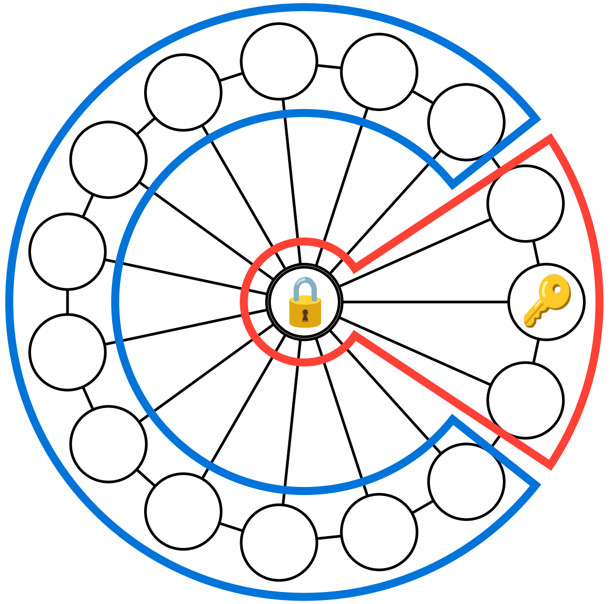
Folding abstraction



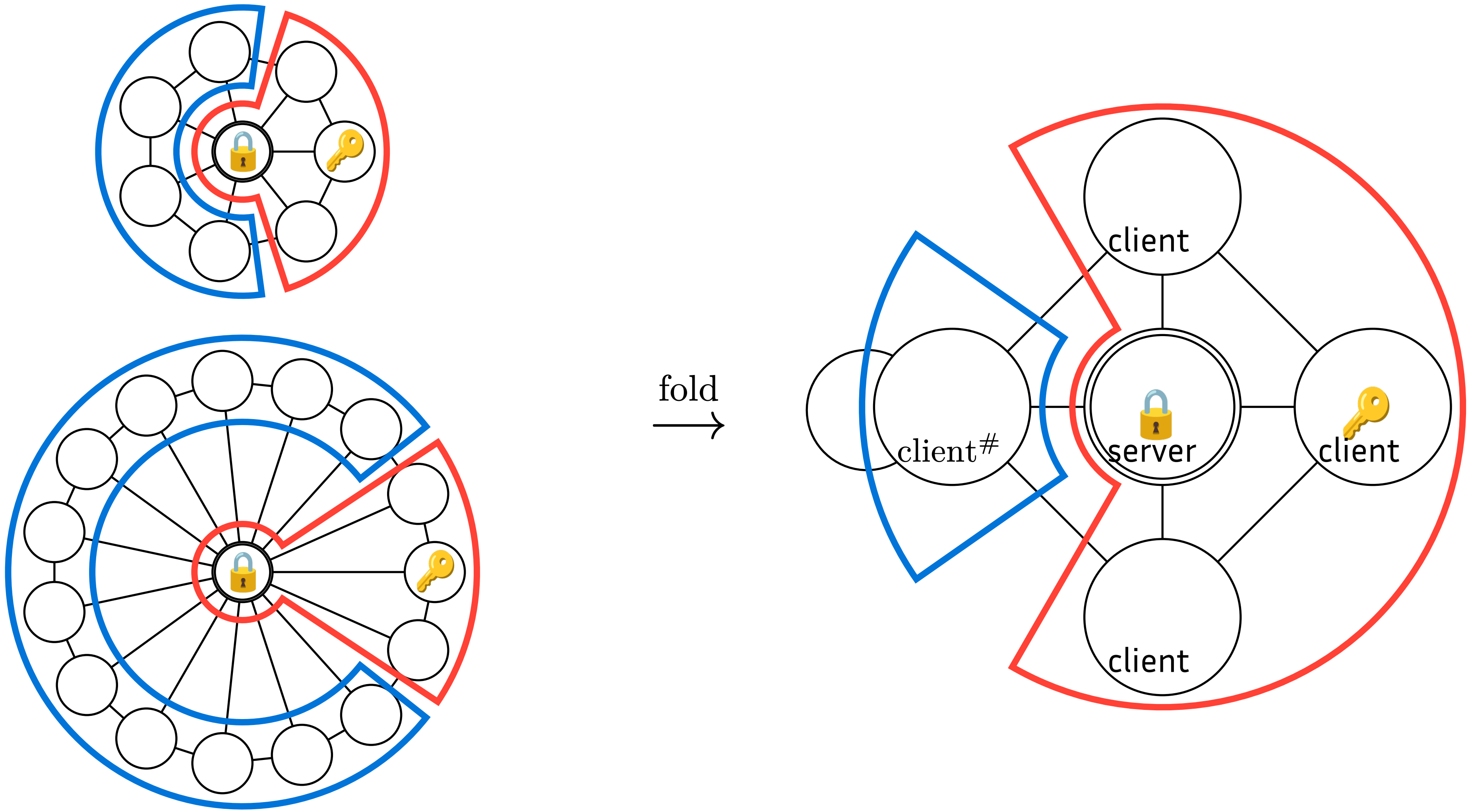


Folding abstraction

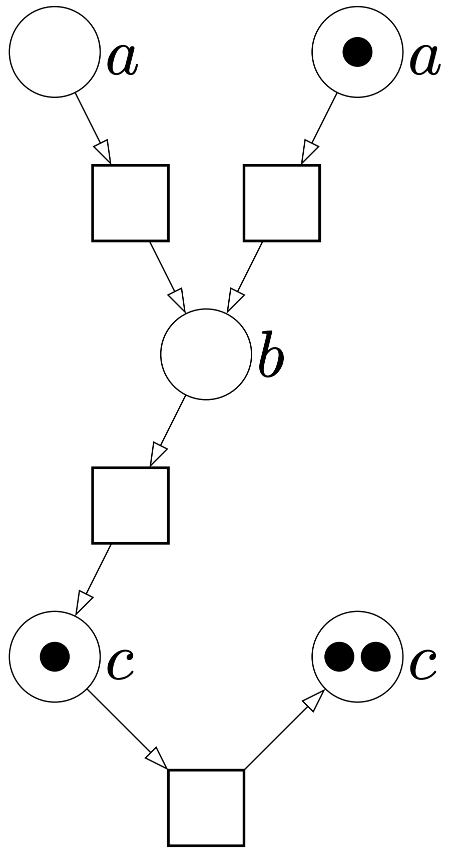




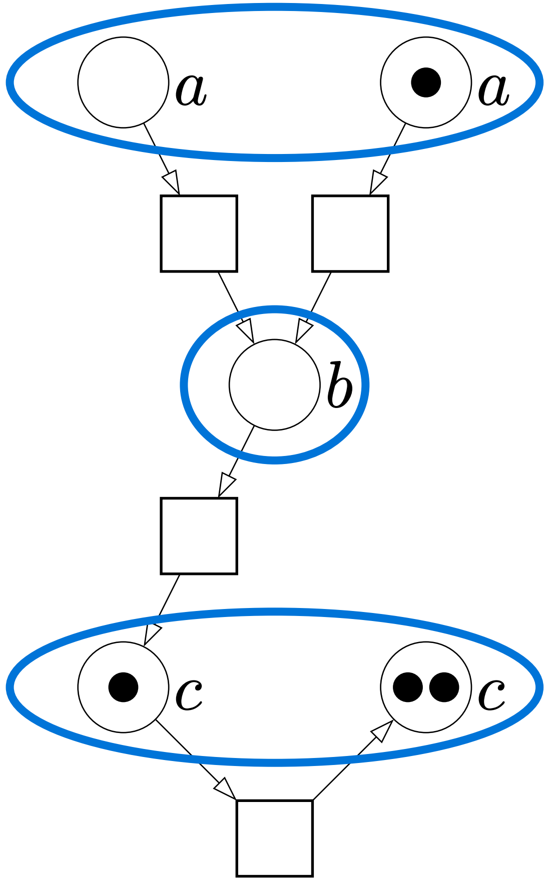
Folding abstraction



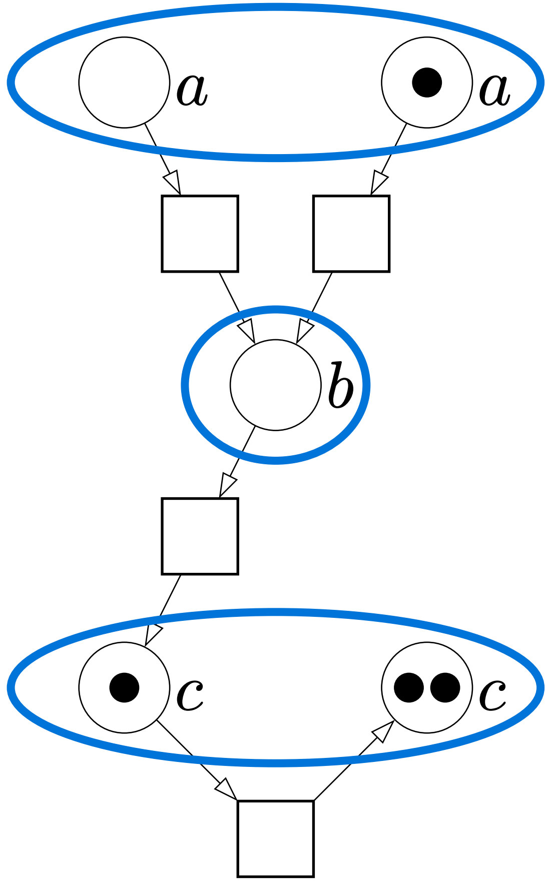
Implementing fold



Implementing fold

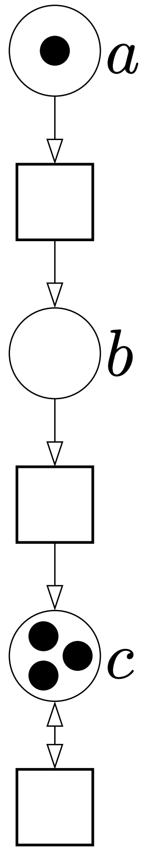


Implementing fold

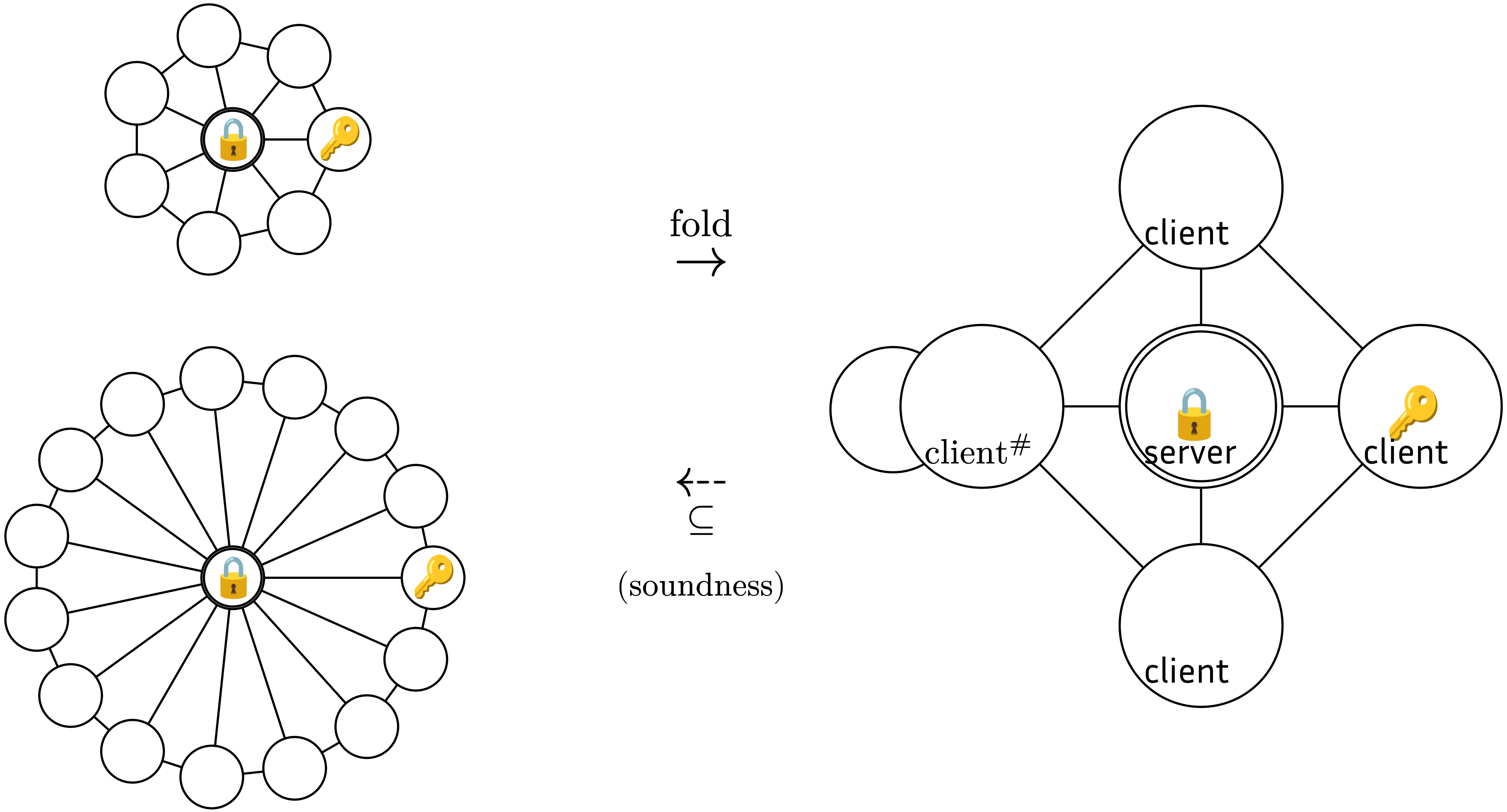


$$\xrightarrow{\text{fold}}$$

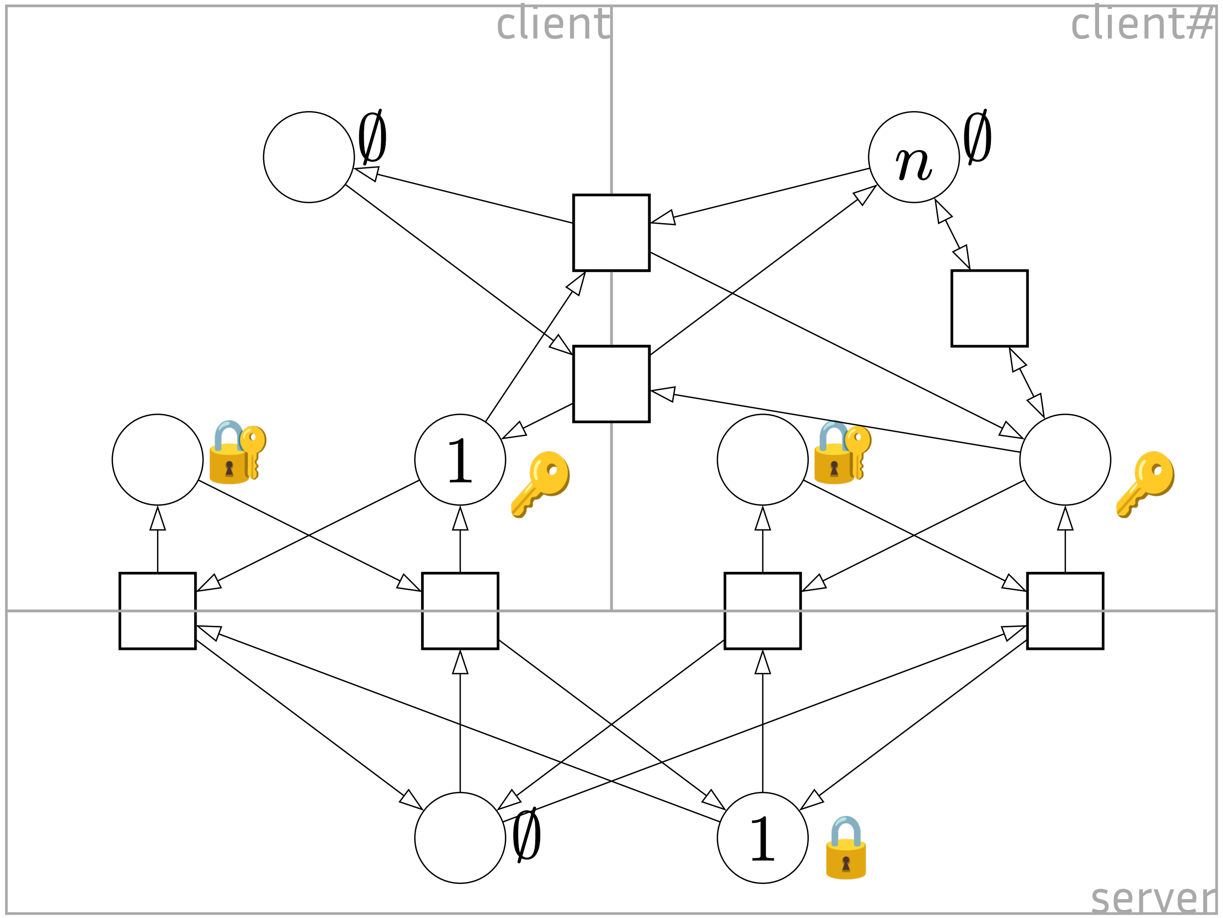
$$\dashleftarrow$$
$$\subseteq$$
(soundness)



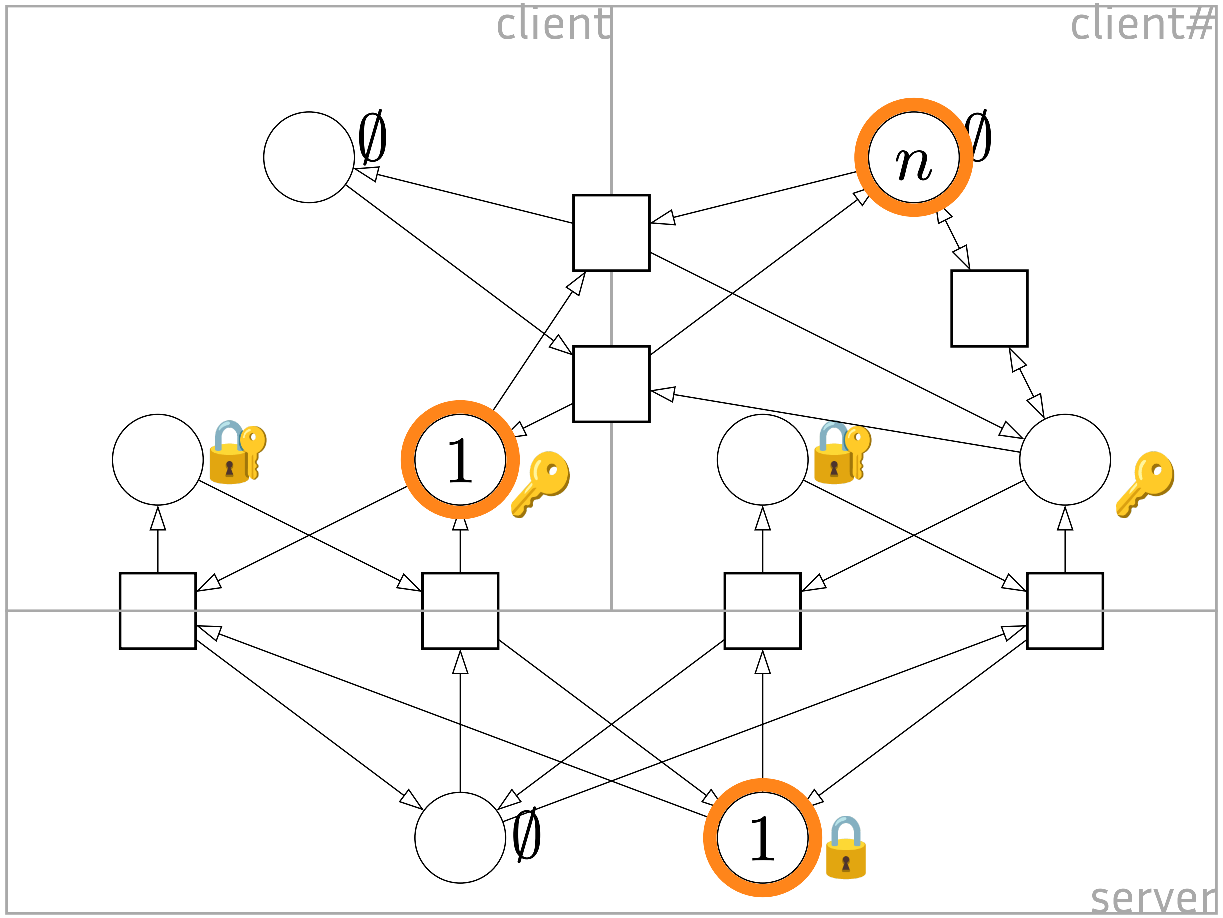
Implementing fold



Folded system



Folded system



Initial marking

$$\mathrm{Sys} \longrightarrow \mathsf{compose}\Big(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')\Big)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\Big($$

$$\mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\Big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(X)\Big),$$

$$\mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}(\mathrm{client})$$

$$\Big)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\Big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}(\mathrm{server}), \mathrm{client}\Big)$$

Initial marking

$$\mathrm{Sys} \longrightarrow \mathsf{compose}\left(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}\left(\mathrm{client}'\right)\right)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\Big($$

$$\mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\left(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}\left(\mathrm{X}\right)\right),$$

$$\mathsf{rename}_{\mathrm{right}\mapsto\mathrm{mid}}\left(\mathrm{client}\right)$$

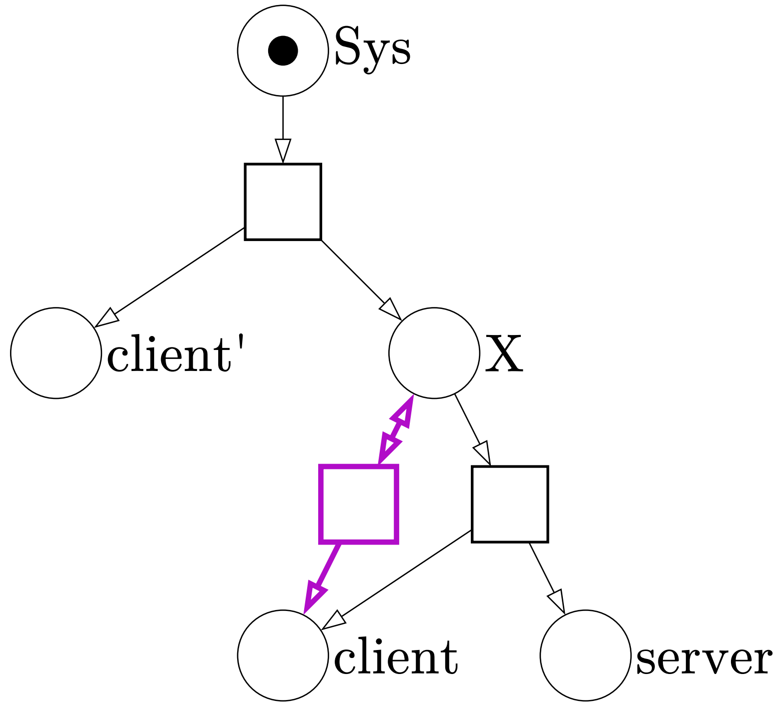
$$\Big)$$

$$\mathrm{X} \longrightarrow \mathsf{compose}\left(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{acq},\mathrm{recv}\rightsquigarrow\mathrm{rel}}\left(\mathrm{server}\right), \mathrm{client}\right)$$

Initial marking

From the grammar

$$\begin{aligned} \mathrm{Sys} &\longrightarrow X, \mathrm{client'} \\ X &\longrightarrow X, \mathrm{client} \\ X &\longrightarrow \mathrm{server}, \mathrm{client} \end{aligned}$$



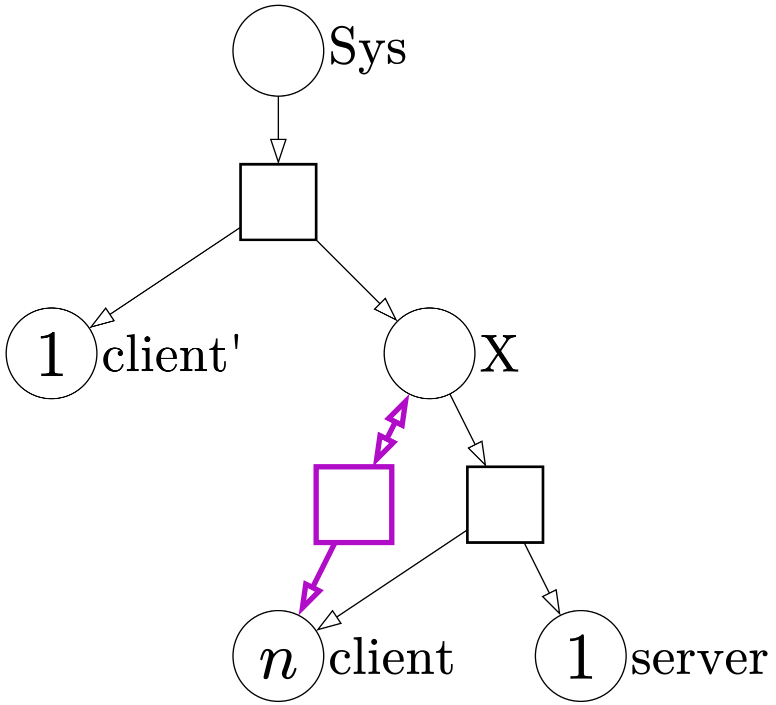
Initial marking

From the grammar

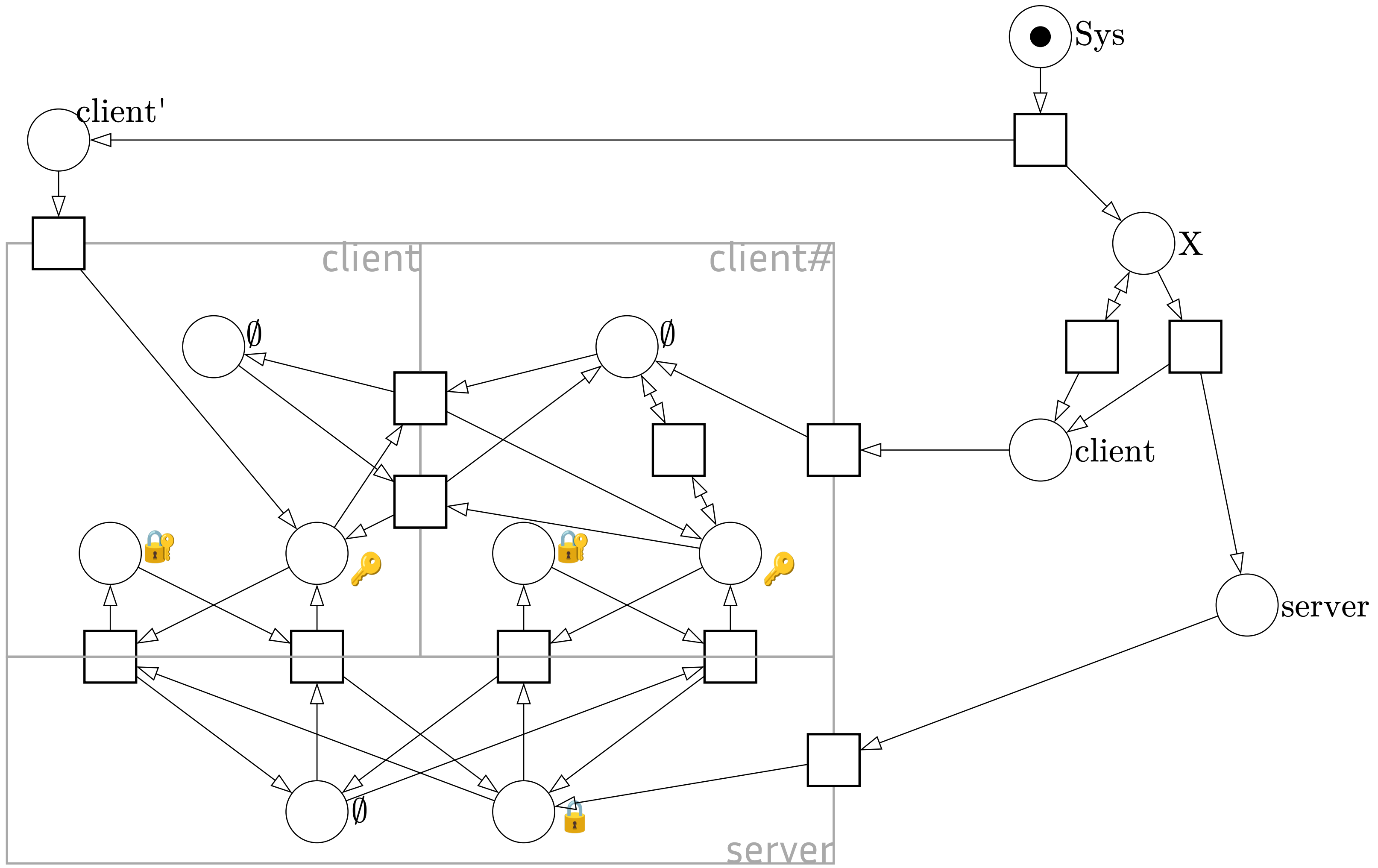
$$\text{Sys} \longrightarrow X, \text{client'}$$

$$X \longrightarrow X, \text{client}$$

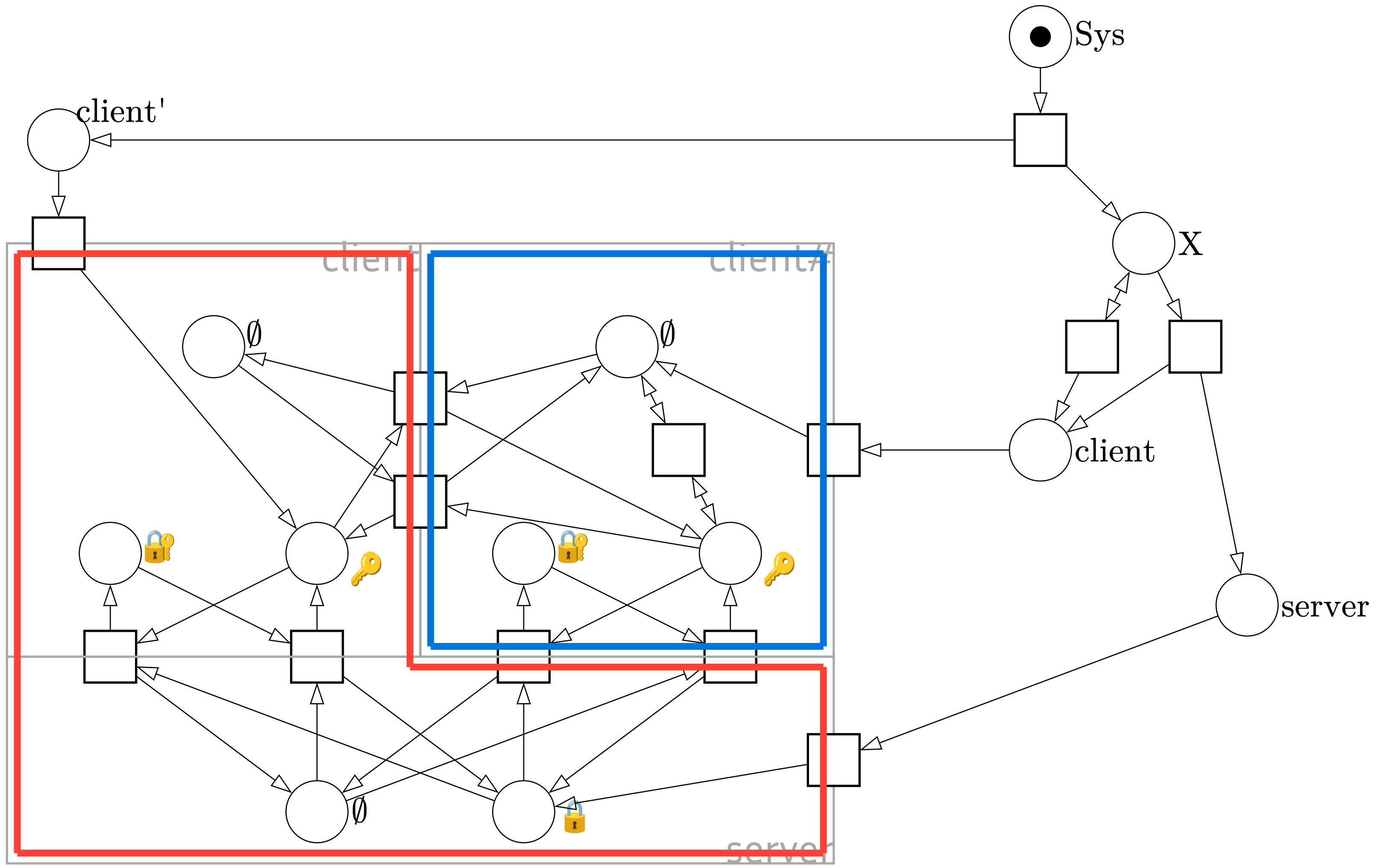
$$X \longrightarrow \text{server}, \text{client}$$



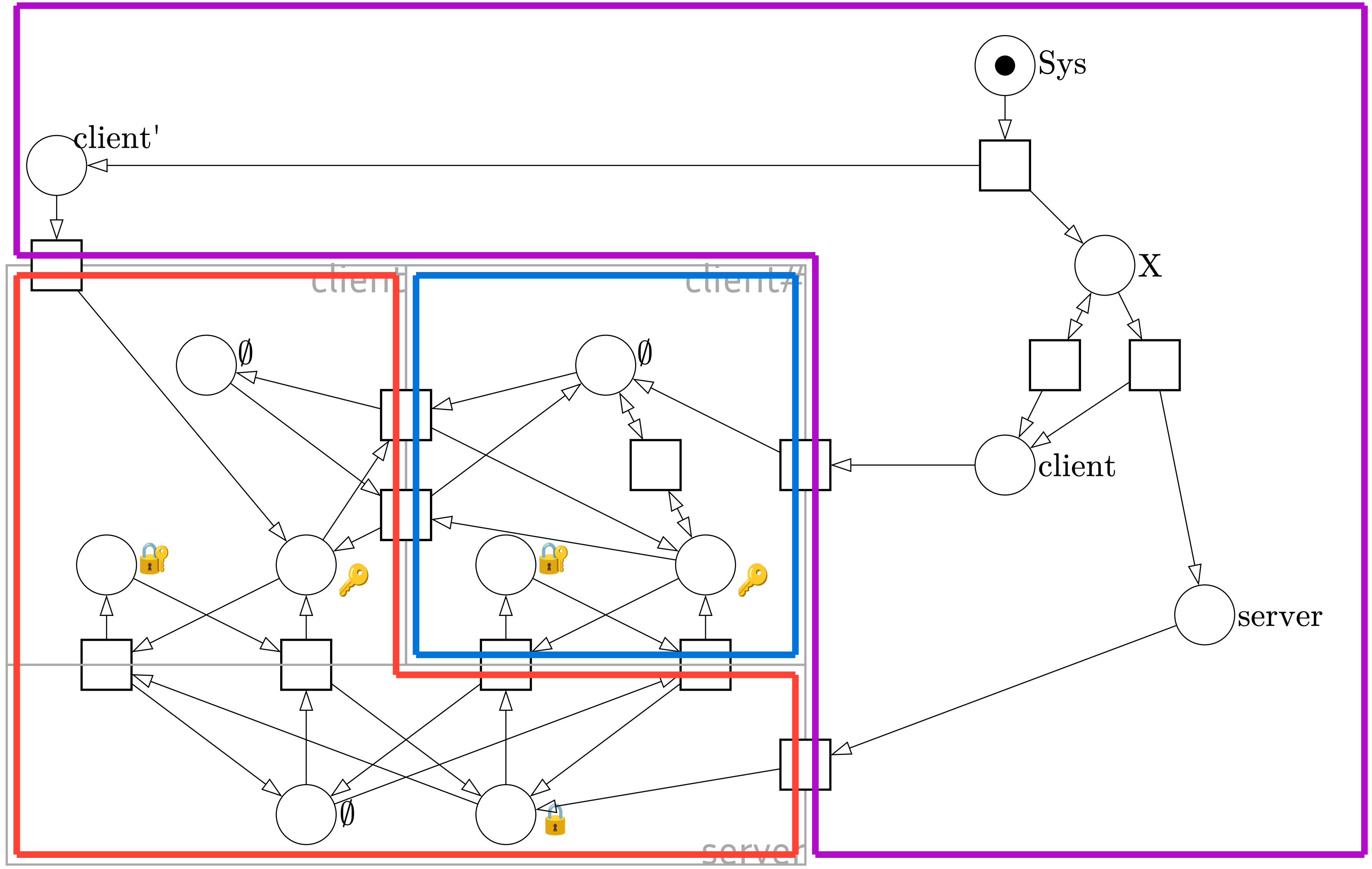
Counting abstraction



Counting abstraction

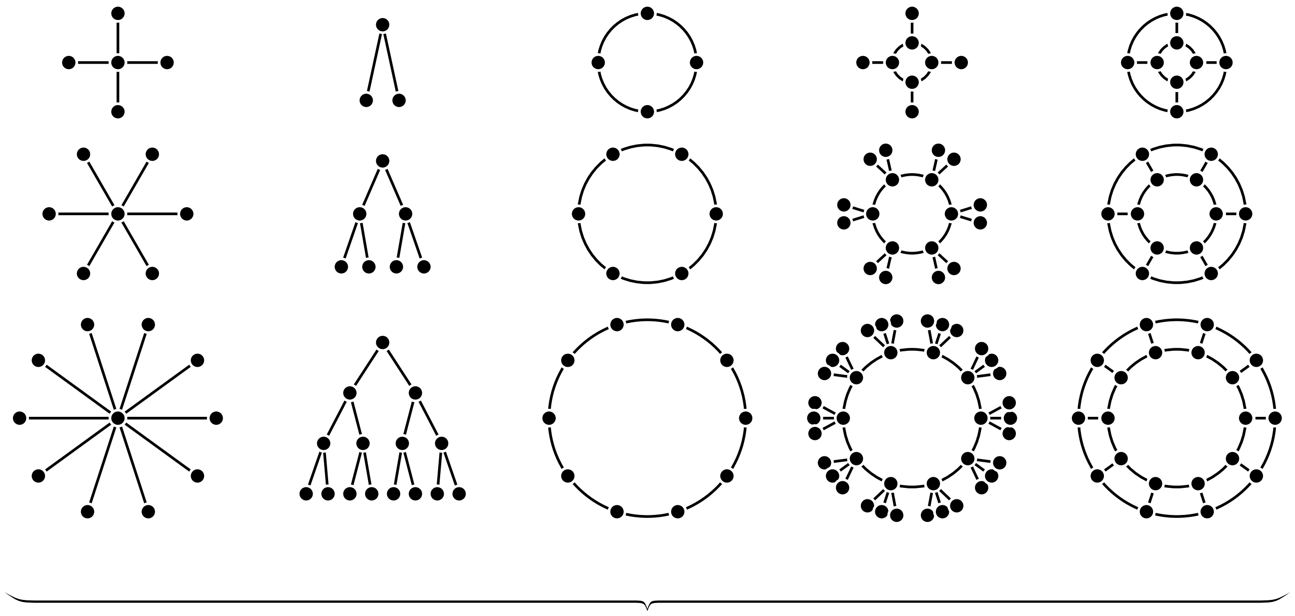


Counting abstraction



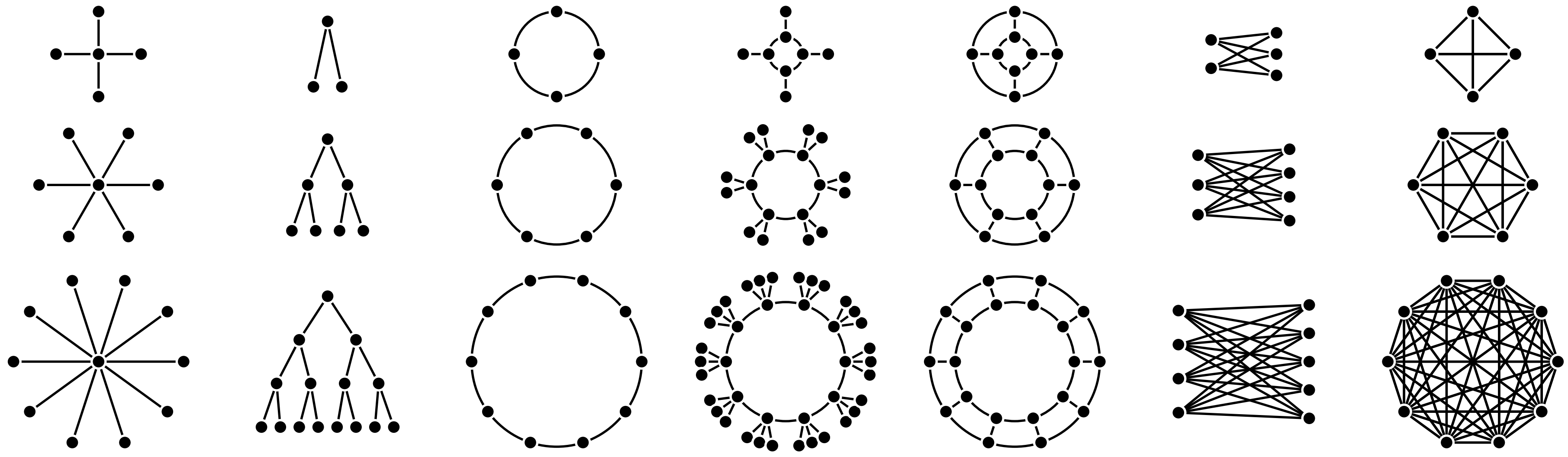
Verifying Dense Graphs

What formalism for architectures ?



Hyperedge **R**eplacement (sparse only)

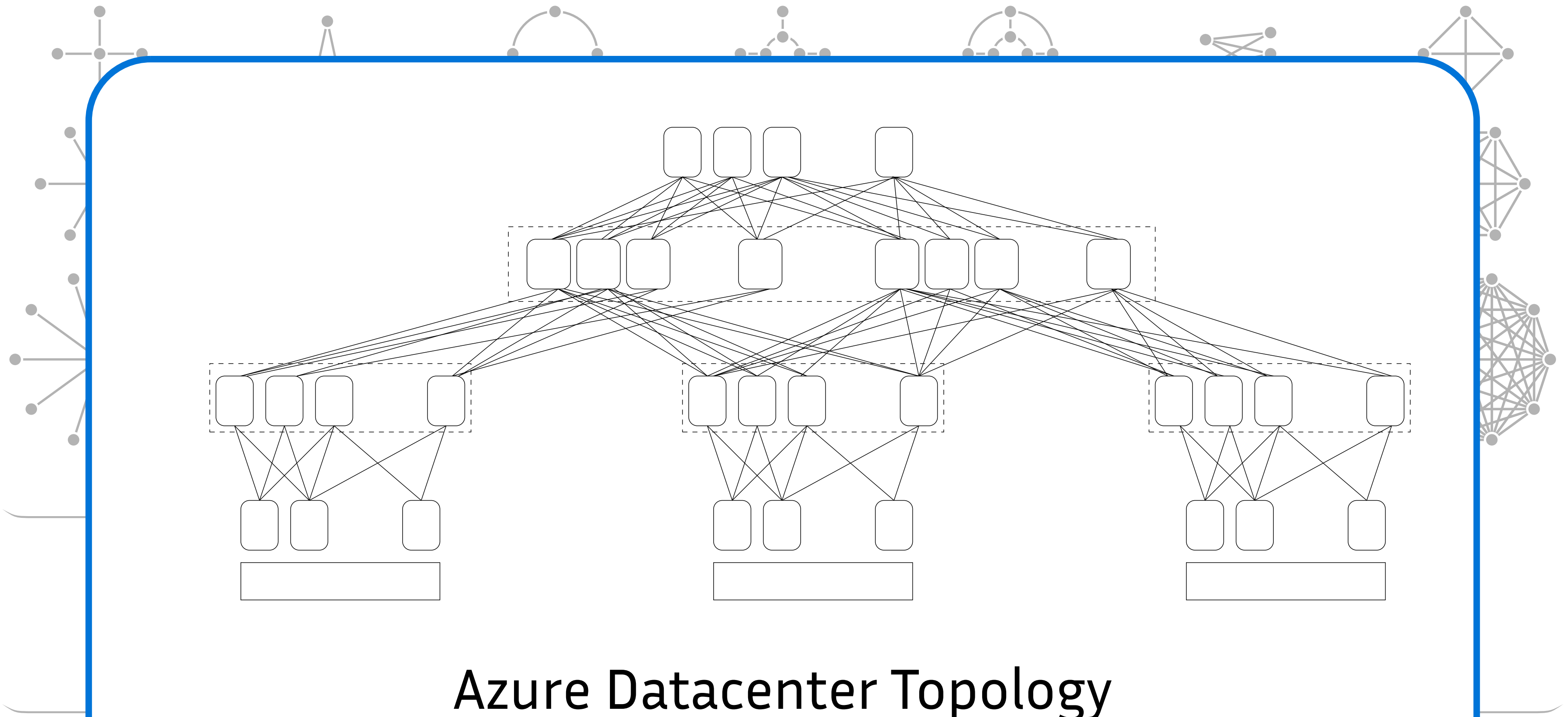
What formalism for architectures ?

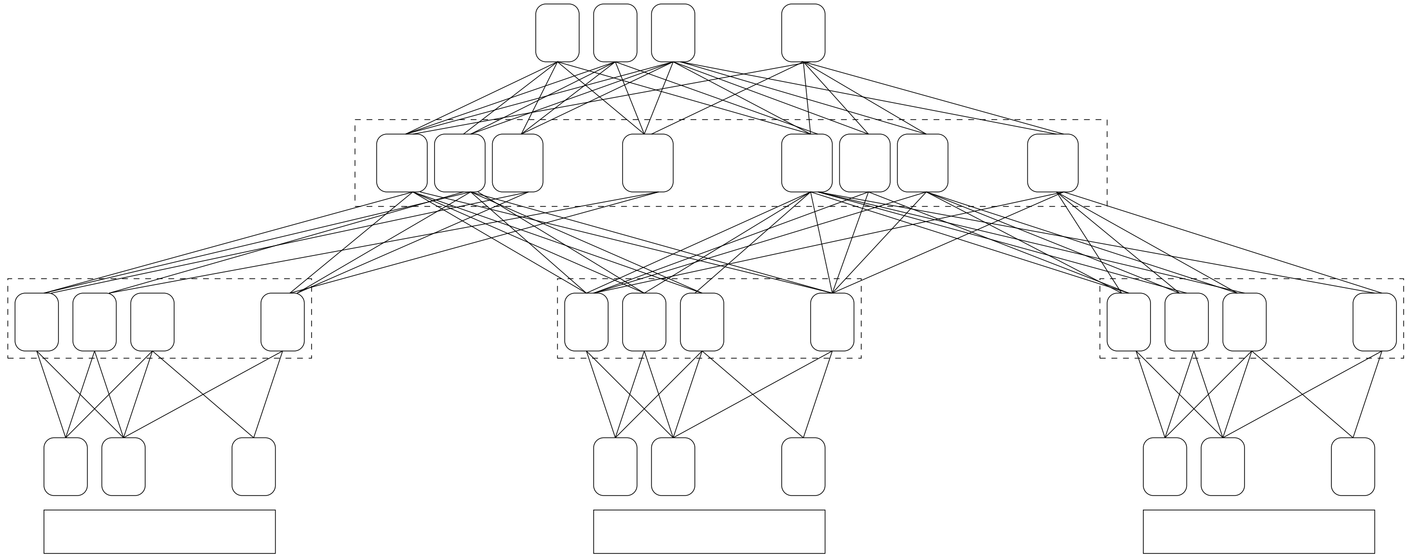


Hyperedge **R**eplacement (sparse only)

Vertex **R**eplacement (incl. some dense)

What formalism for architectures ?



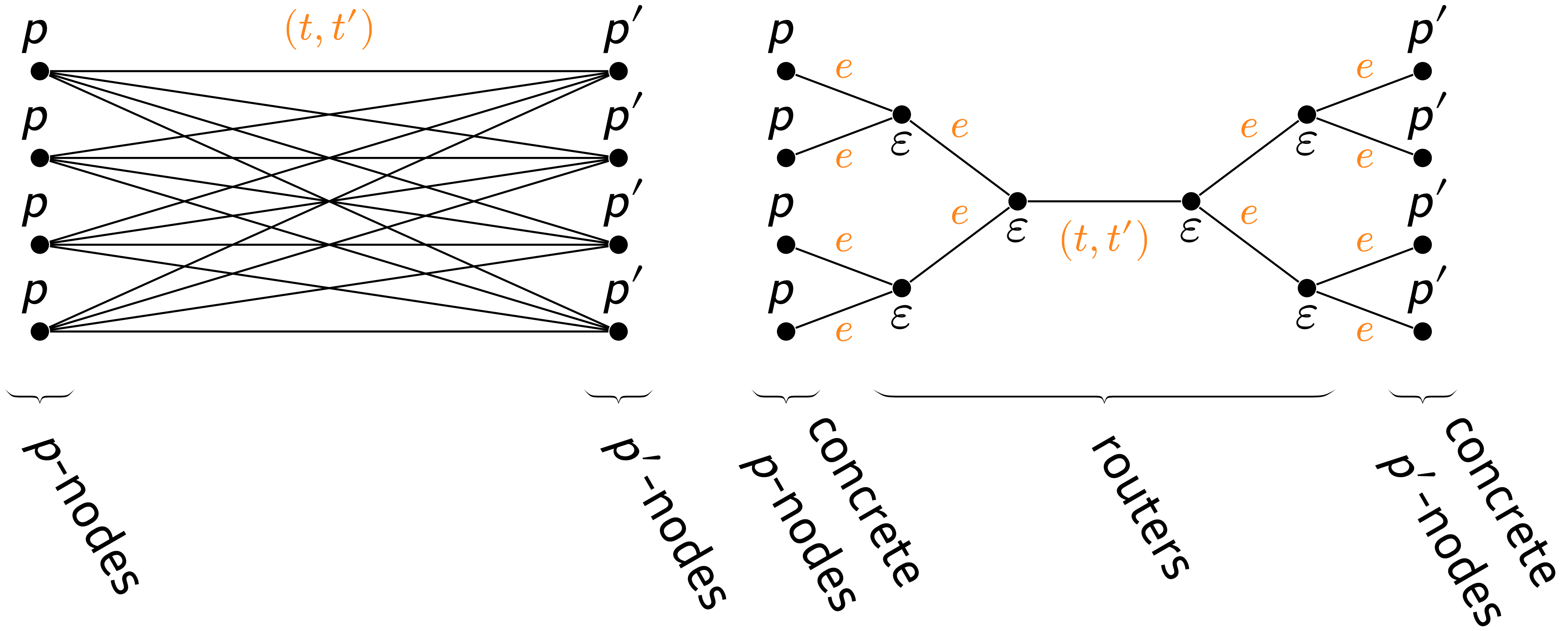


Azure Datacenter Topology
Greenberg et al., SIGCOMM 2009

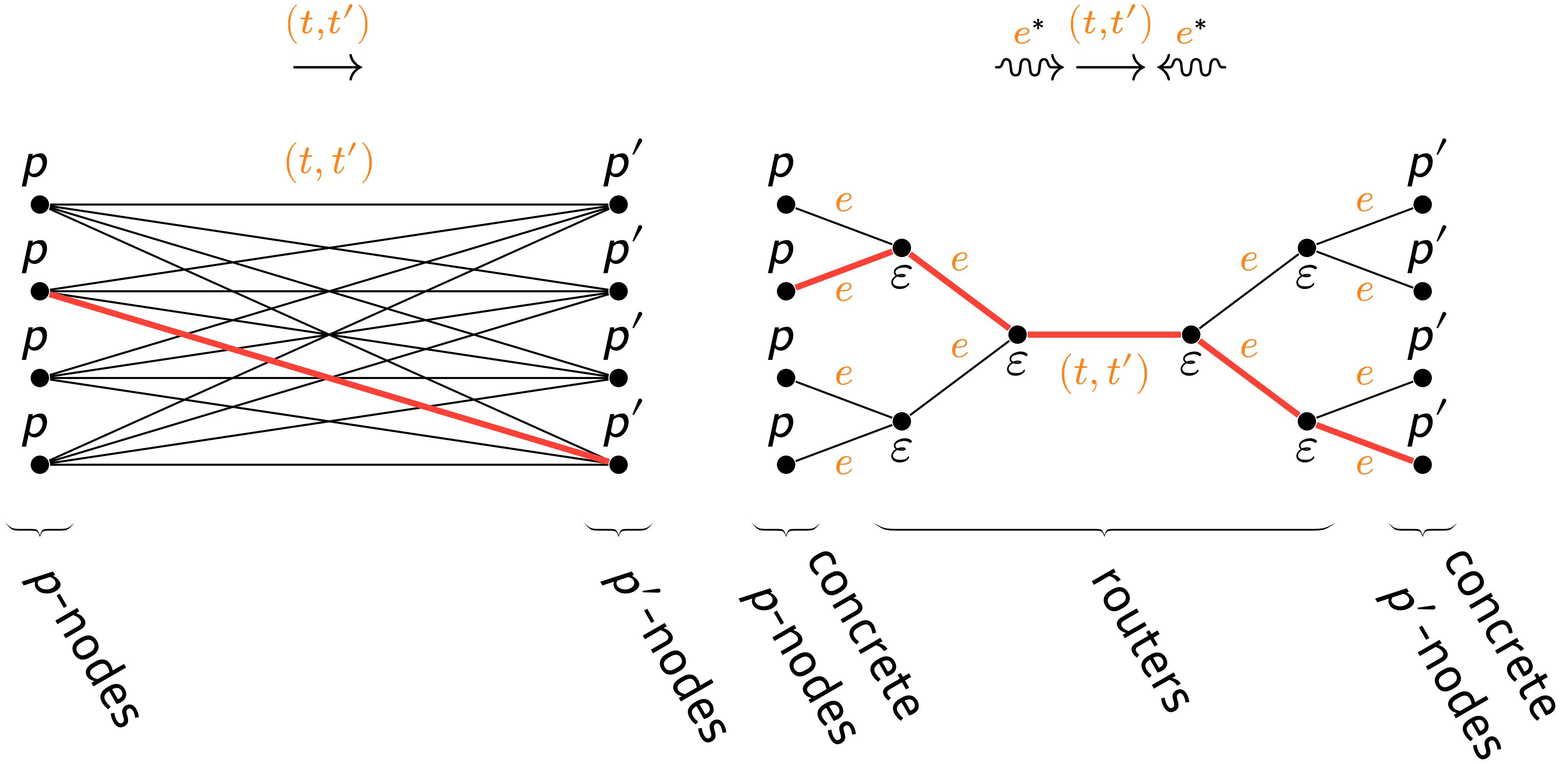
Key idea

$\xrightarrow{(t,t')}$

$\overset{e^*}{\rightsquigarrow} \xrightarrow{(t,t')} \overset{e^*}{\leftsquigarrow}$



Key idea

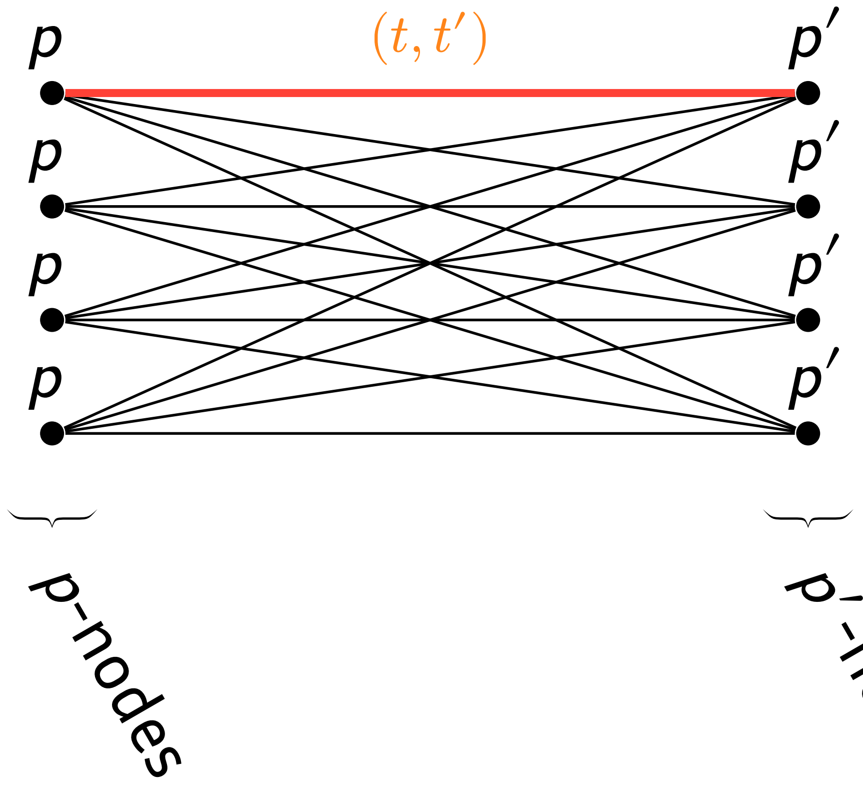


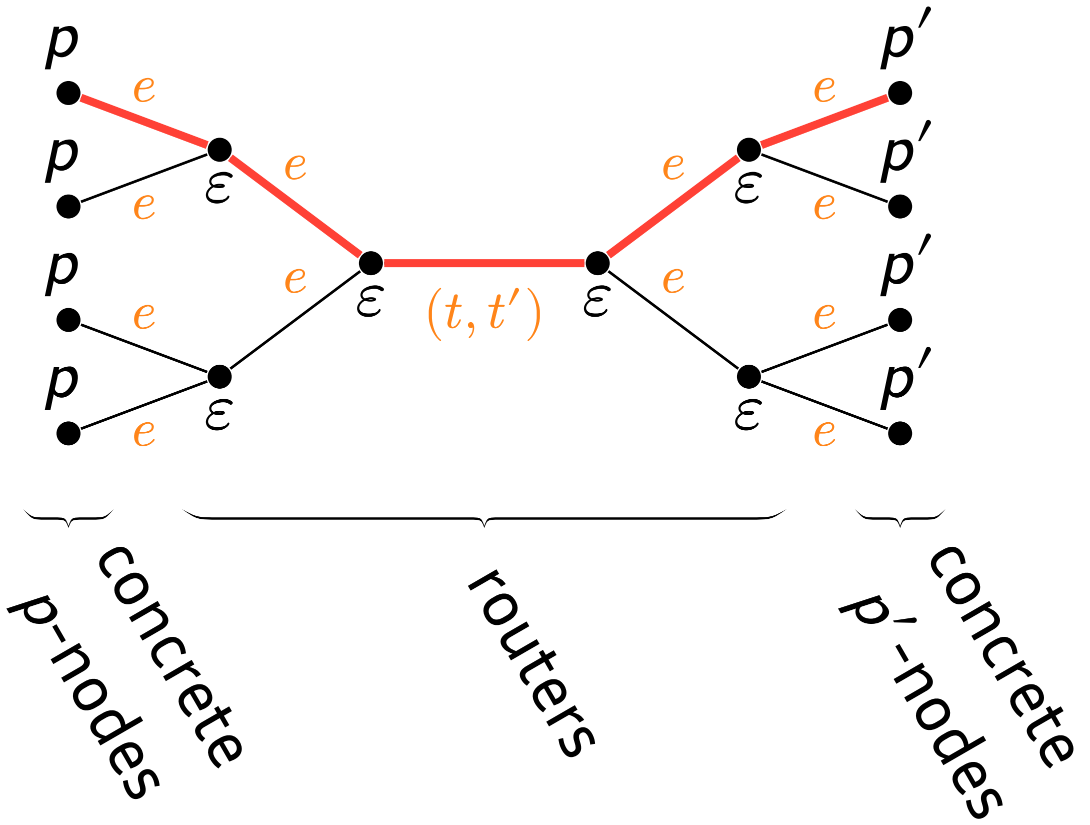
$\xrightarrow{(t,t')}$
$\overset{e^*}{\rightsquigarrow} \xrightarrow{(t,t')} \overset{e^*}{\leftsquigarrow}$
p
p'
(t,t')
e
ε
p-nodes
p'-nodes
concrete p-nodes
routers
concrete p'-nodes

Key idea

$\xrightarrow{(t,t')}$

$\overset{e^*}{\rightsquigarrow} \xrightarrow{(t,t')} \overset{e^*}{\leftsquigarrow}$

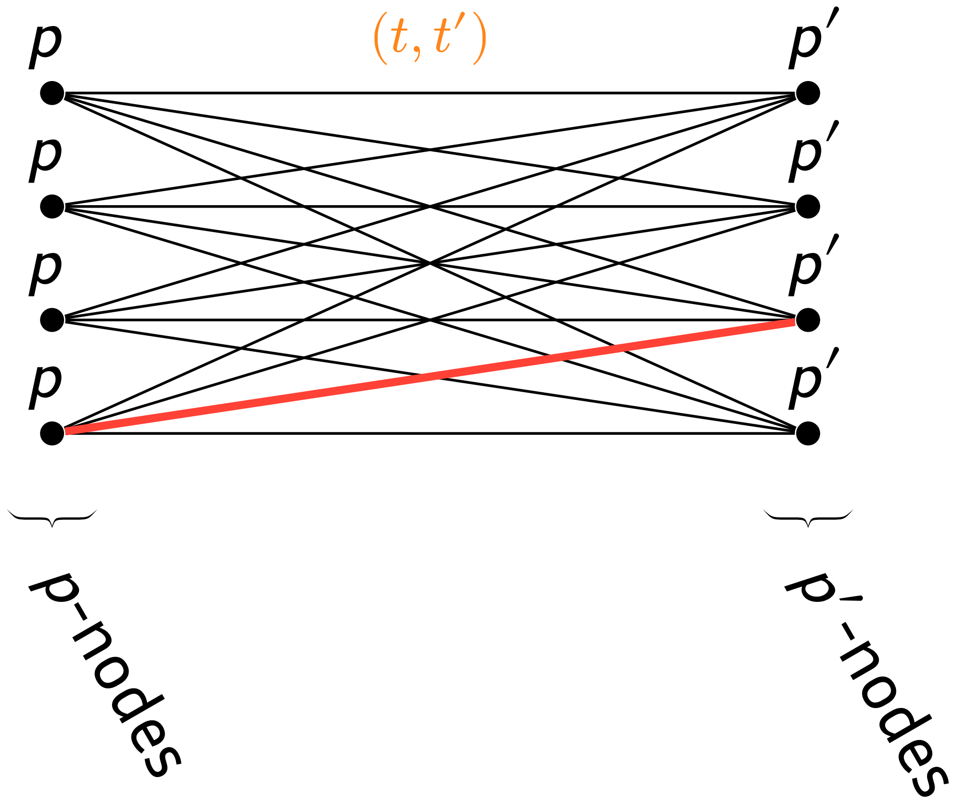


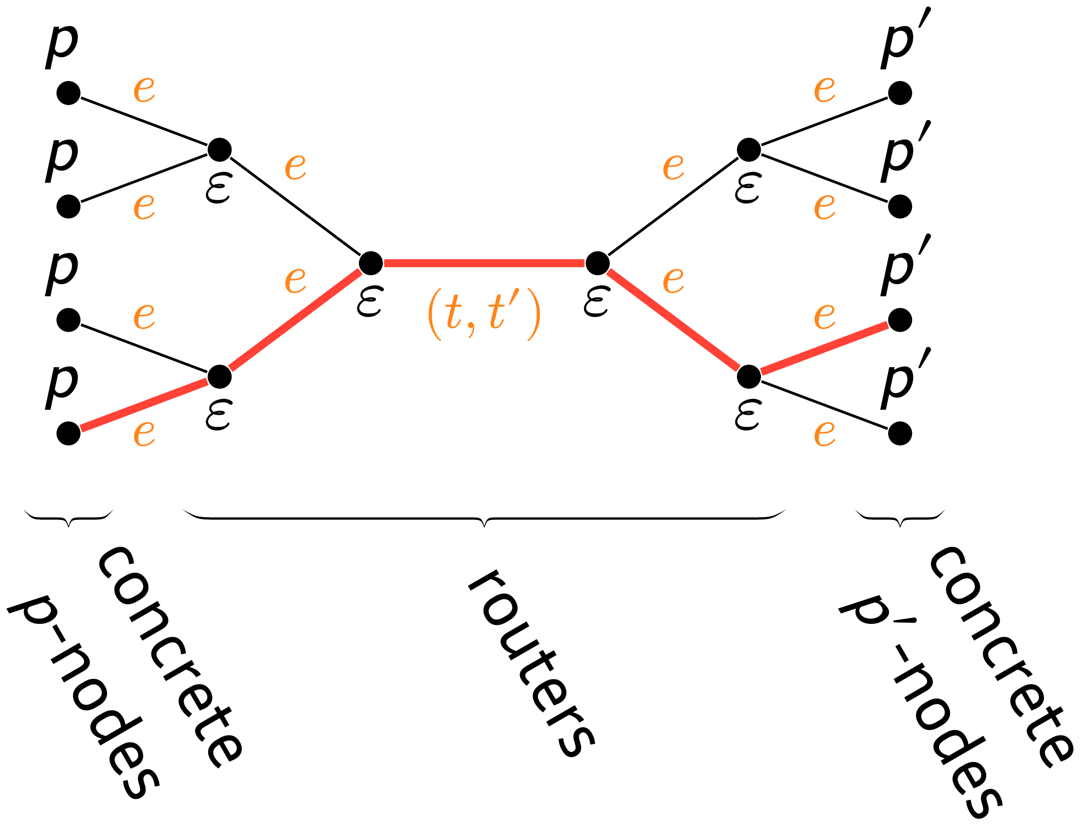


Key idea

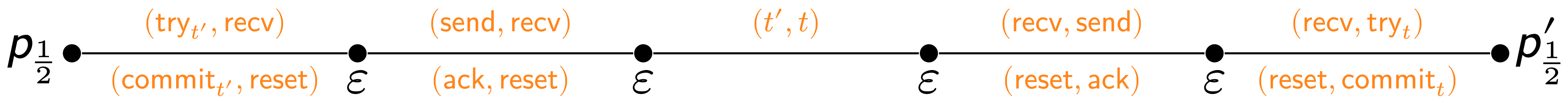
$\xrightarrow{(t,t')}$

$\overset{e^*}{\rightsquigarrow} \xrightarrow{(t,t')} \overset{e^*}{\leftsquigarrow}$

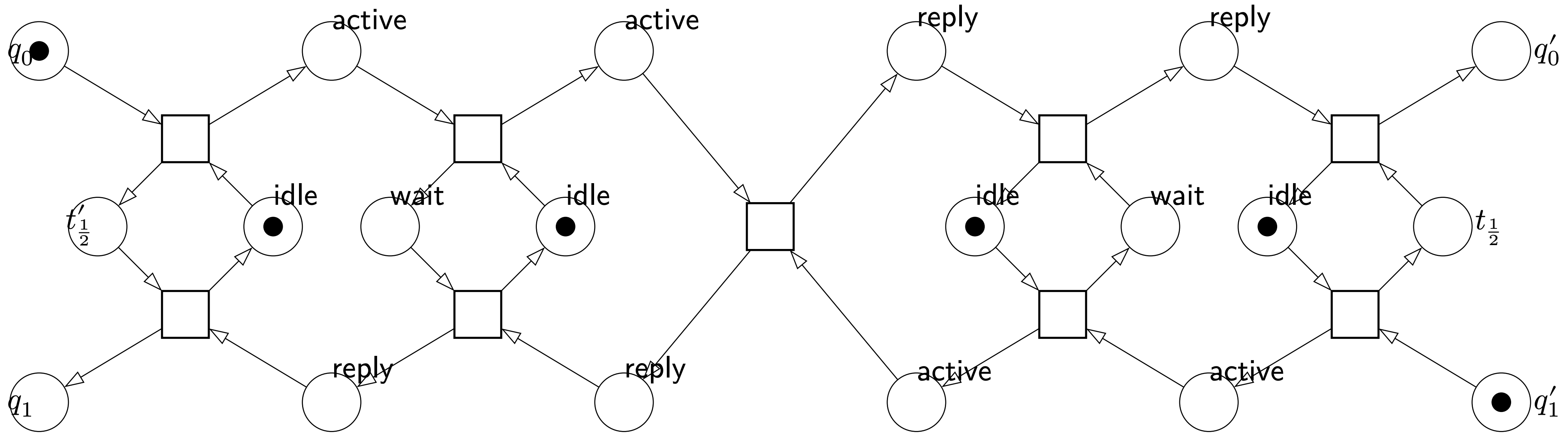




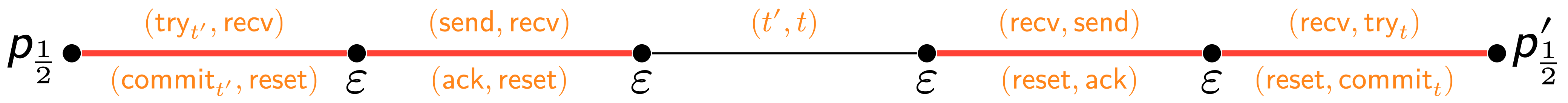
Communication through routers

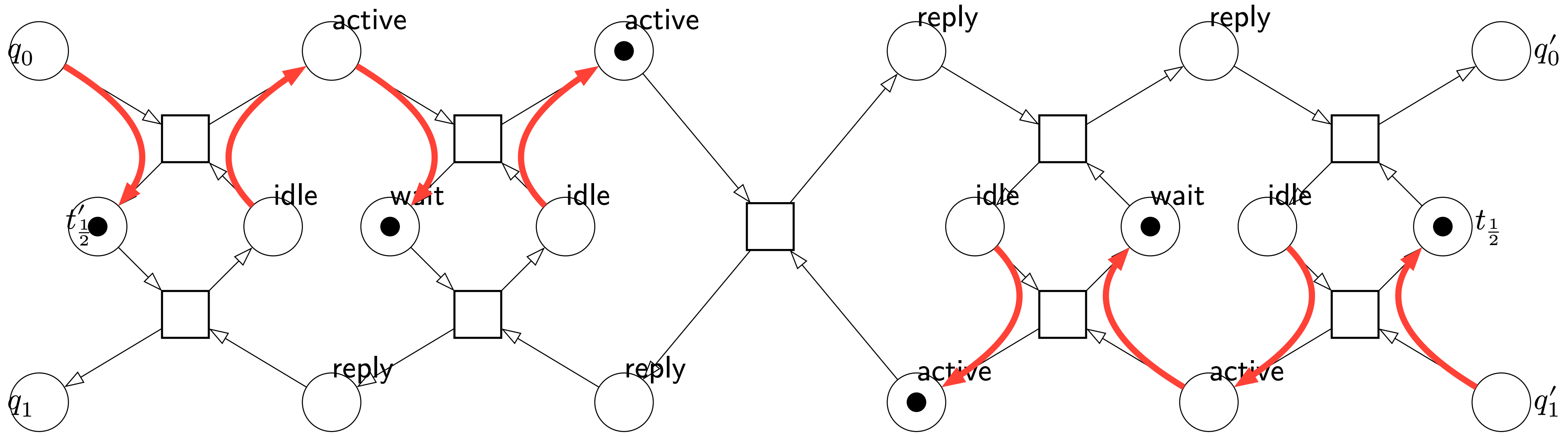


$p_{\frac{1}{2}}$
$(\text{try}_{t'}, \text{recv})$
$(\text{commit}_{t'}, \text{reset})$
ε
$(\text{send}, \text{recv})$
$(\text{ack}, \text{reset})$
ε
(t', t)
ε
$(\text{recv}, \text{send})$
$(\text{reset}, \text{ack})$
ε
$(\text{recv}, \text{try}_t)$
$(\text{reset}, \text{commit}_t)$
$p'_{\frac{1}{2}}$

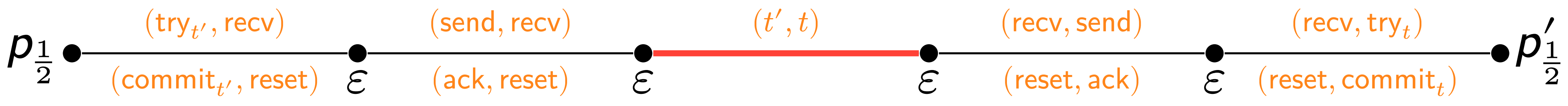


Communication through routers

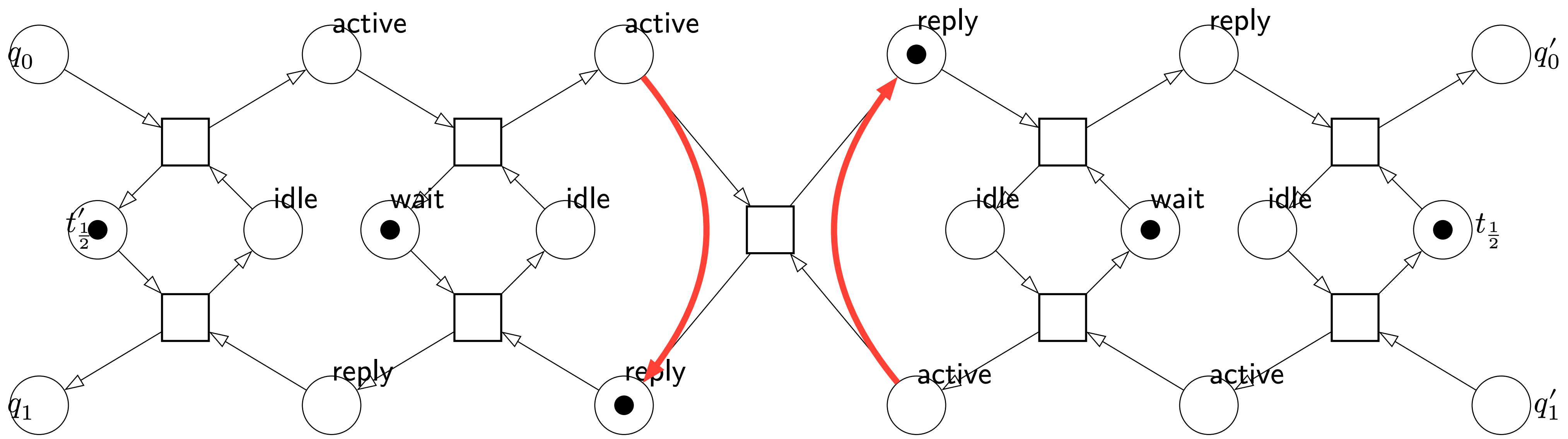

$p_{\frac{1}{2}}$
$(\mathsf{try}_{t'}, \mathsf{recv})$
$(\mathsf{commit}_{t'}, \mathsf{reset})$
ε
$(\mathsf{send}, \mathsf{recv})$
$(\mathsf{ack}, \mathsf{reset})$
ε
(t', t)
ε
$(\mathsf{recv}, \mathsf{send})$
$(\mathsf{reset}, \mathsf{ack})$
ε
$(\mathsf{recv}, \mathsf{try}_t)$
$(\mathsf{reset}, \mathsf{commit}_t)$
$p'_{\frac{1}{2}}$



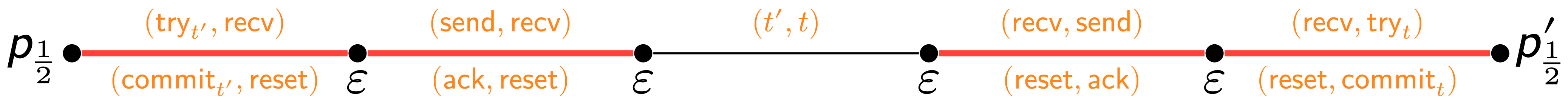
Communication through routers

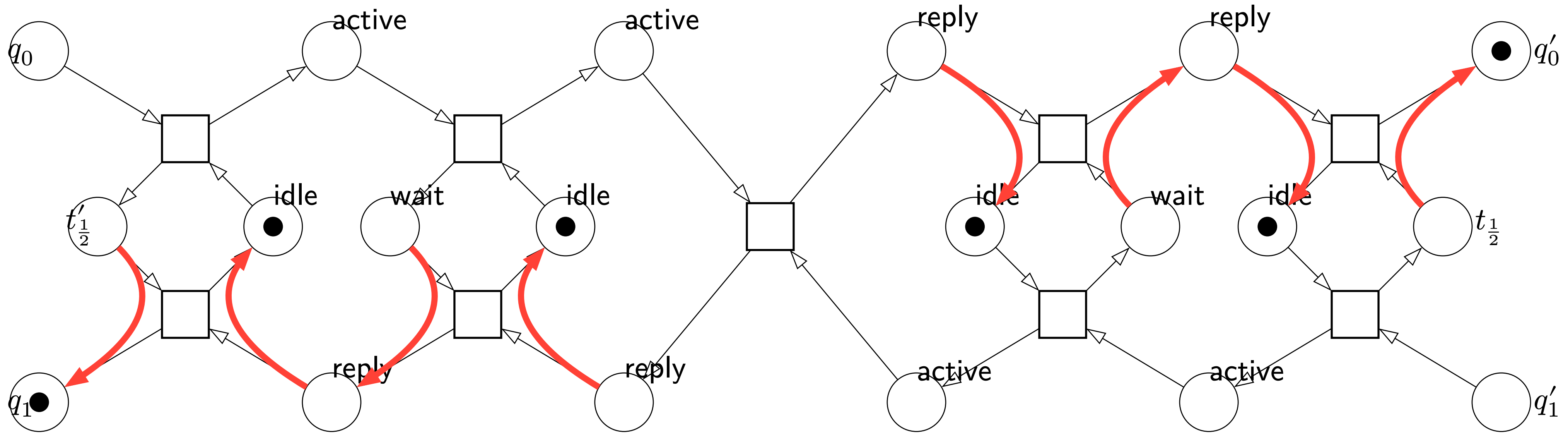


$p_{\frac{1}{2}}$
$(\text{try}_{t'}, \text{recv})$
$(\text{commit}_{t'}, \text{reset})$
ε
$(\text{send}, \text{recv})$
$(\text{ack}, \text{reset})$
ε
(t', t)
ε
$(\text{recv}, \text{send})$
$(\text{reset}, \text{ack})$
ε
$(\text{recv}, \text{try}_t)$
$(\text{reset}, \text{commit}_t)$
$p'_{\frac{1}{2}}$



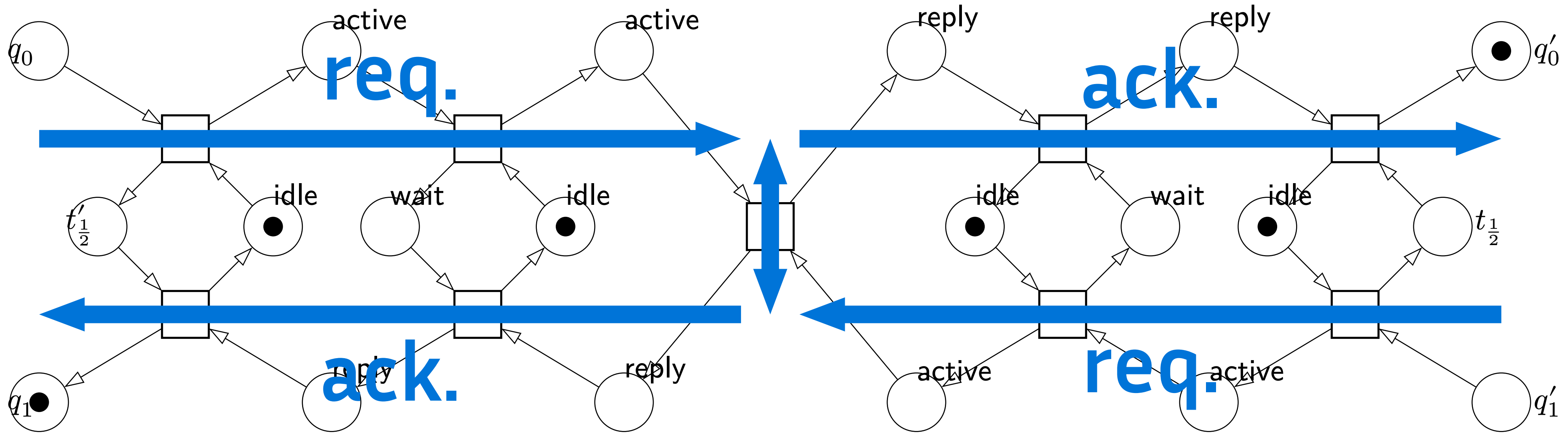
Communication through routers


$p_{\frac{1}{2}}$
$(\mathsf{try}_{t'}, \mathsf{recv})$
$(\mathsf{commit}_{t'}, \mathsf{reset})$
ε
$(\mathsf{send}, \mathsf{recv})$
$(\mathsf{ack}, \mathsf{reset})$
ε
(t', t)
ε
$(\mathsf{recv}, \mathsf{send})$
$(\mathsf{reset}, \mathsf{ack})$
ε
$(\mathsf{recv}, \mathsf{try}_t)$
$(\mathsf{reset}, \mathsf{commit}_t)$
$p'_{\frac{1}{2}}$

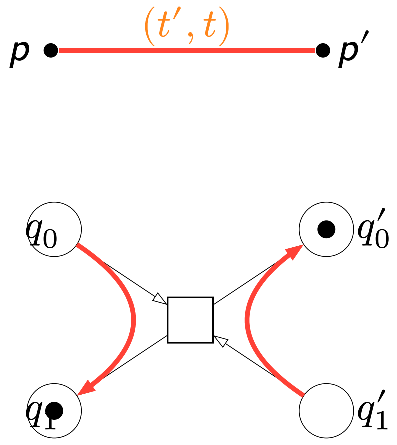


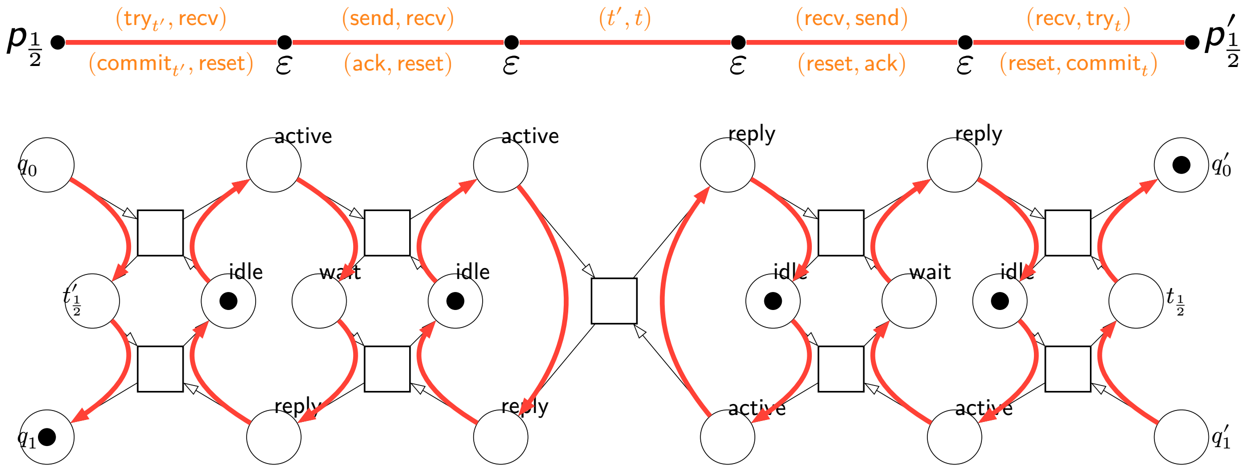
Communication through routers

$$\begin{tikzcd}
p_{\frac{1}{2}} \arrow[r, dash, "(\mathsf{try}_{t'}, \mathsf{recv})", "(\mathsf{commit}_{t'}, \mathsf{reset})"'] & \varepsilon \arrow[r, dash, "(\mathsf{send}, \mathsf{recv})", "(\mathsf{ack}, \mathsf{reset})"'] & \varepsilon \arrow[r, dash, "(t', t)"] & \varepsilon \arrow[r, dash, "(\mathsf{recv}, \mathsf{send})", "(\mathsf{reset}, \mathsf{ack})"'] & \varepsilon \arrow[r, dash, "(\mathsf{recv}, \mathsf{try}_t)", "(\mathsf{reset}, \mathsf{commit}_t)"'] & p'_{\frac{1}{2}}
\end{tikzcd}$$



Stuttering





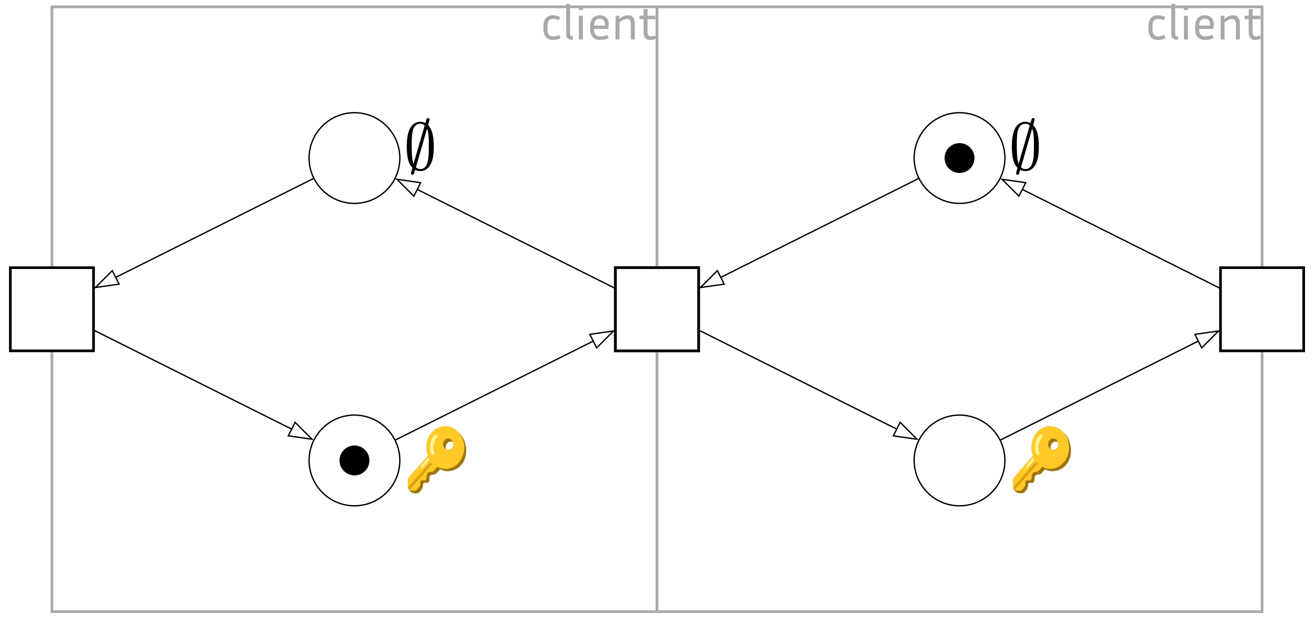
$p_{\frac{1}{2}}$
$(\text{try}_{t'}, \text{recv})$
$(\text{commit}_{t'}, \text{reset})$
ε
$(\text{send}, \text{recv})$
$(\text{ack}, \text{reset})$
ε
(t', t)
ε
$(\text{recv}, \text{send})$
$(\text{reset}, \text{ack})$
ε
$(\text{recv}, \text{try}_t)$
$(\text{reset}, \text{commit}_t)$
$p'_{\frac{1}{2}}$
q_0
active
active
reply
reply
q'_0
$t'_{\frac{1}{2}}$
idle
wait
idle
idle
wait
idle
$t_{\frac{1}{2}}$
q_1
reply
reply
active
active
q'_1

$$s_1 s_2$$

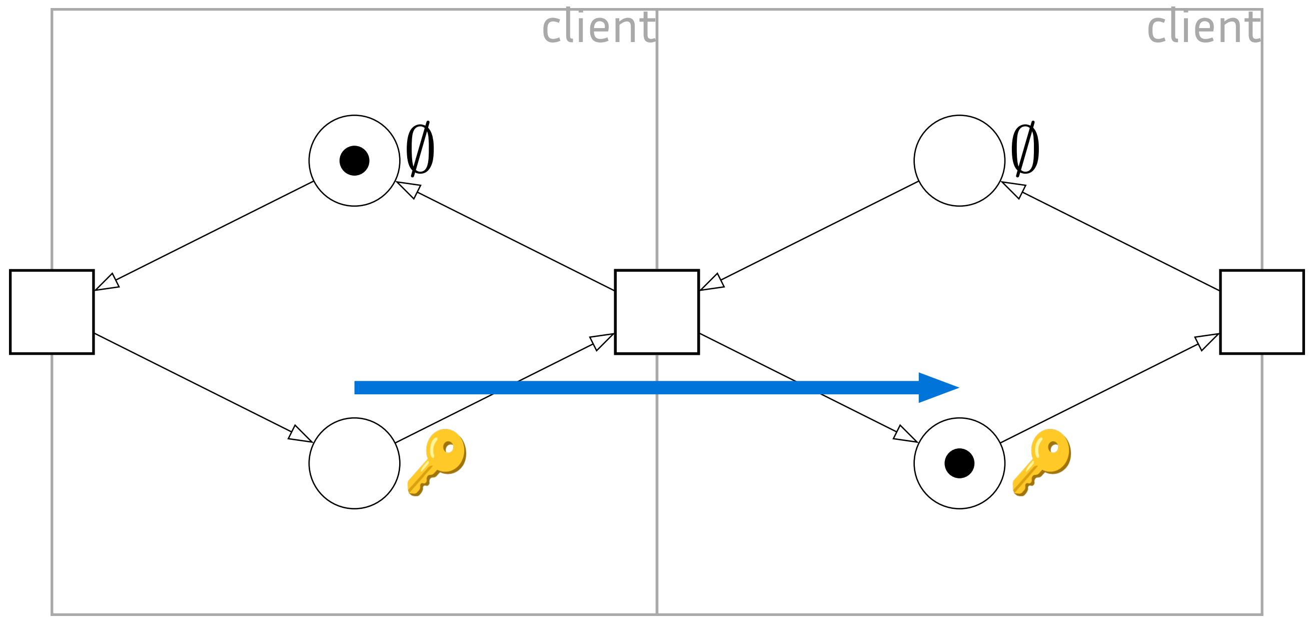
$$s_1 s_1 s_1 s_1 s_1 s_2 s_2 s_2 s_2 s_2$$

Decidable Restrictions

Token Passing



Token Passing



Decidability

	Known	***New***	***WIP***
communication	token passing		
processes	arbitrary	≤ 2 states	≤ 1 non-token state
tokens	exactly 1	arbitrary	exactly 1
architecture	MSO	HR	MSO
property	$\mathrm{CTL}^* \setminus \mathrm{X}$	cover	μ-calculus

Parameterized model checking of rendezvous systems;
by B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith; in Distributed Computing (2017)

Conclusion

- semi-algorithm + implementation for HR
- translation procedure for VR
- decidable classes inspired by literature

More details:

- Counting Abstraction and Decidability for the Verification of Structured Parameterized Networks; in CAV'25
- Verifying Parameterized Networks Specified by Vertex-Replacement Graph Grammars; in NETYS'25

Future work

- refinements
- extend decidable classes
- other communication models