Counting Abstraction
for the Verification of
Structured Parameterized Networks

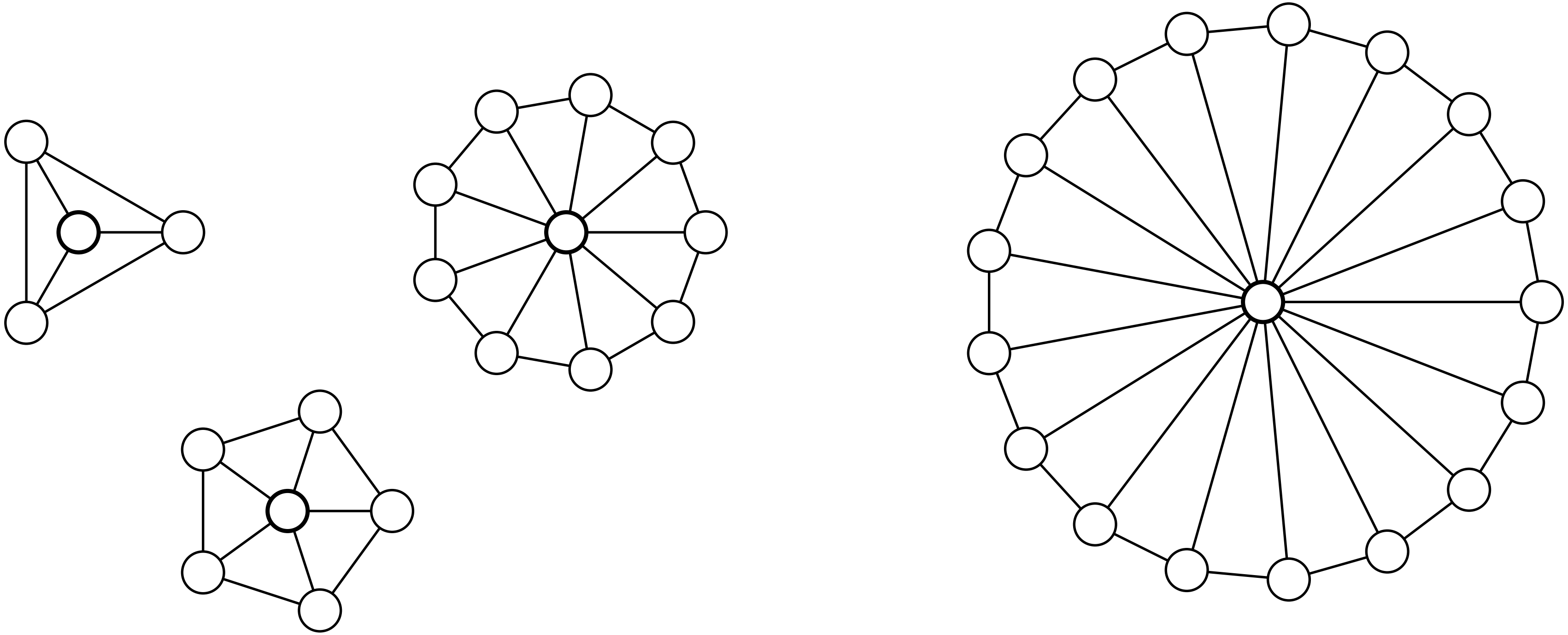
Neven Villani[1] , Marius Bozga[1], Radu Iosif[1], Arnaud Sangnier[2]

CAV'25 @ Zagreb

2025-07-25

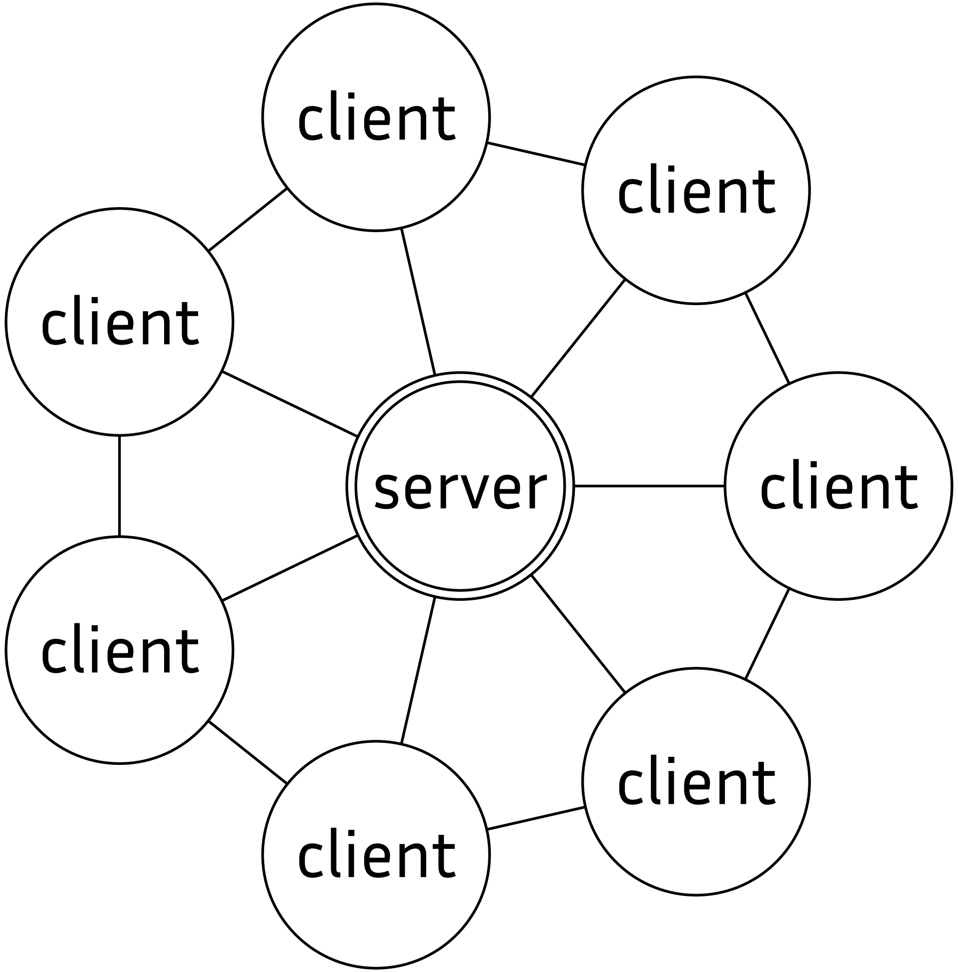
[1]VERIMAG, Univ. Grenoble Alpes, CNRS
[2]DIBRIS, Univ. di Genoa

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

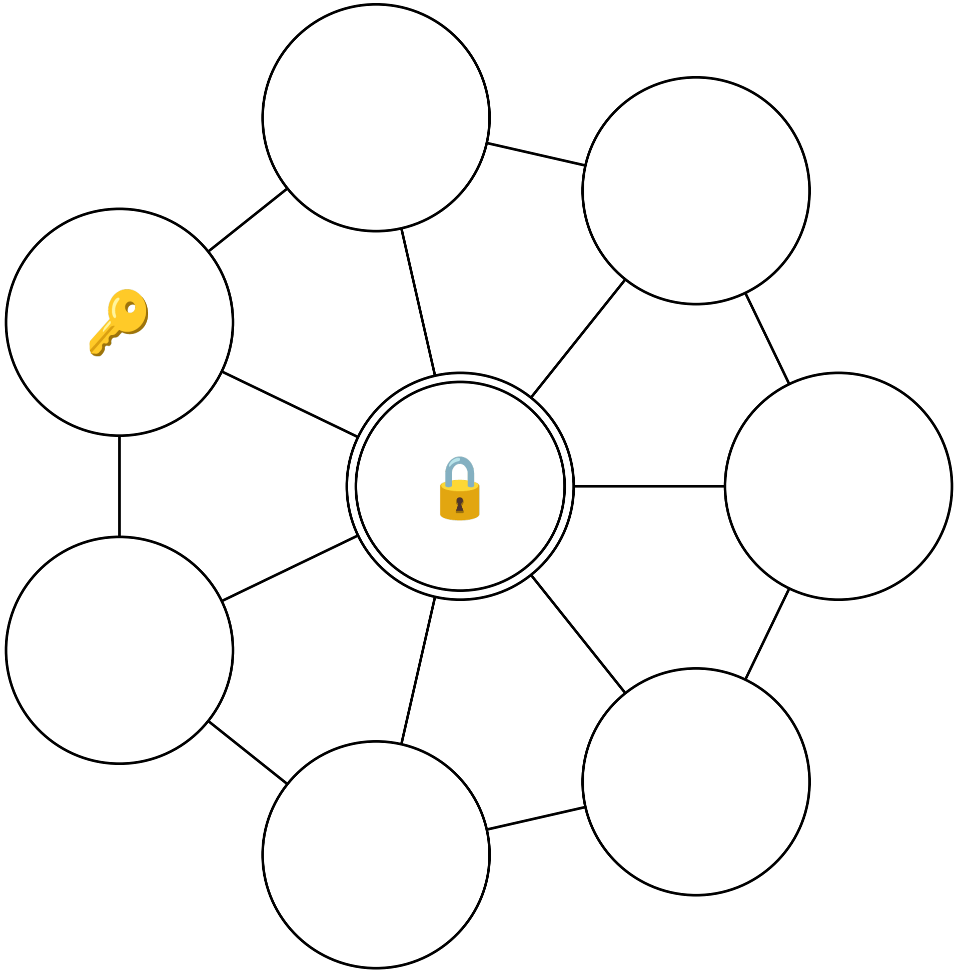


- undecidable $\Rightarrow$ abstraction

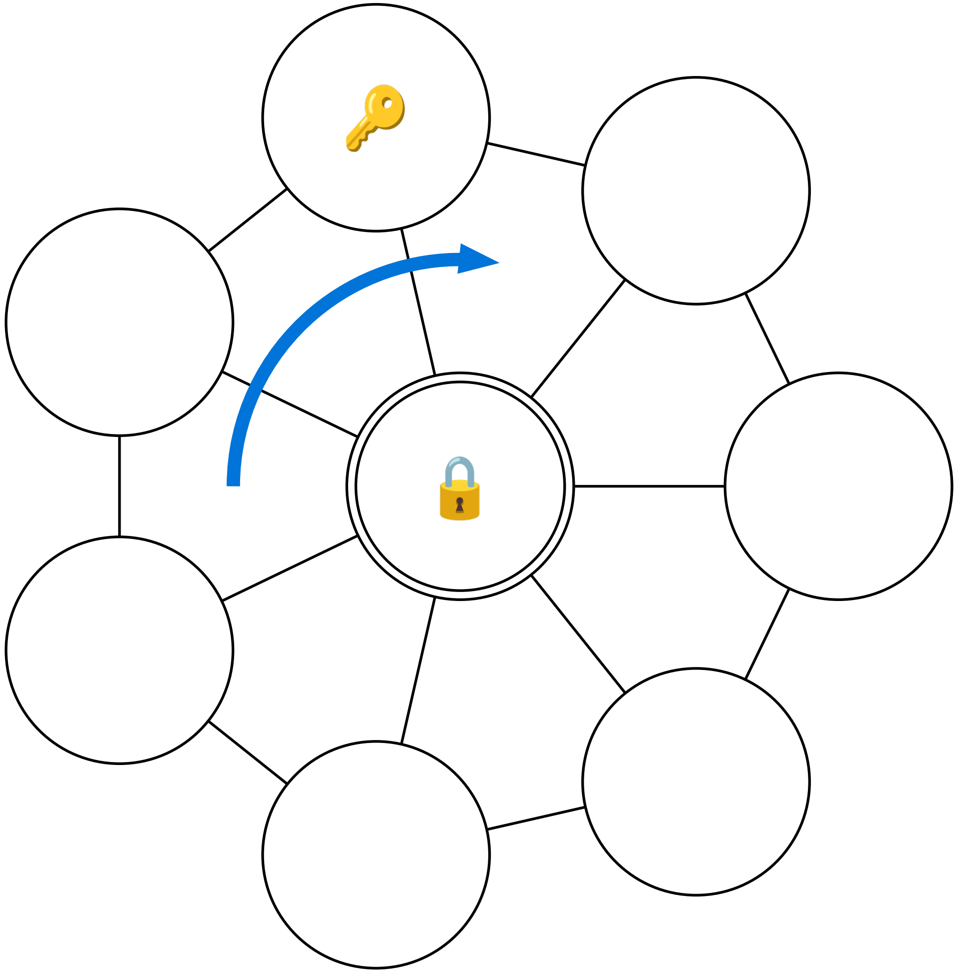
Token ring with resource



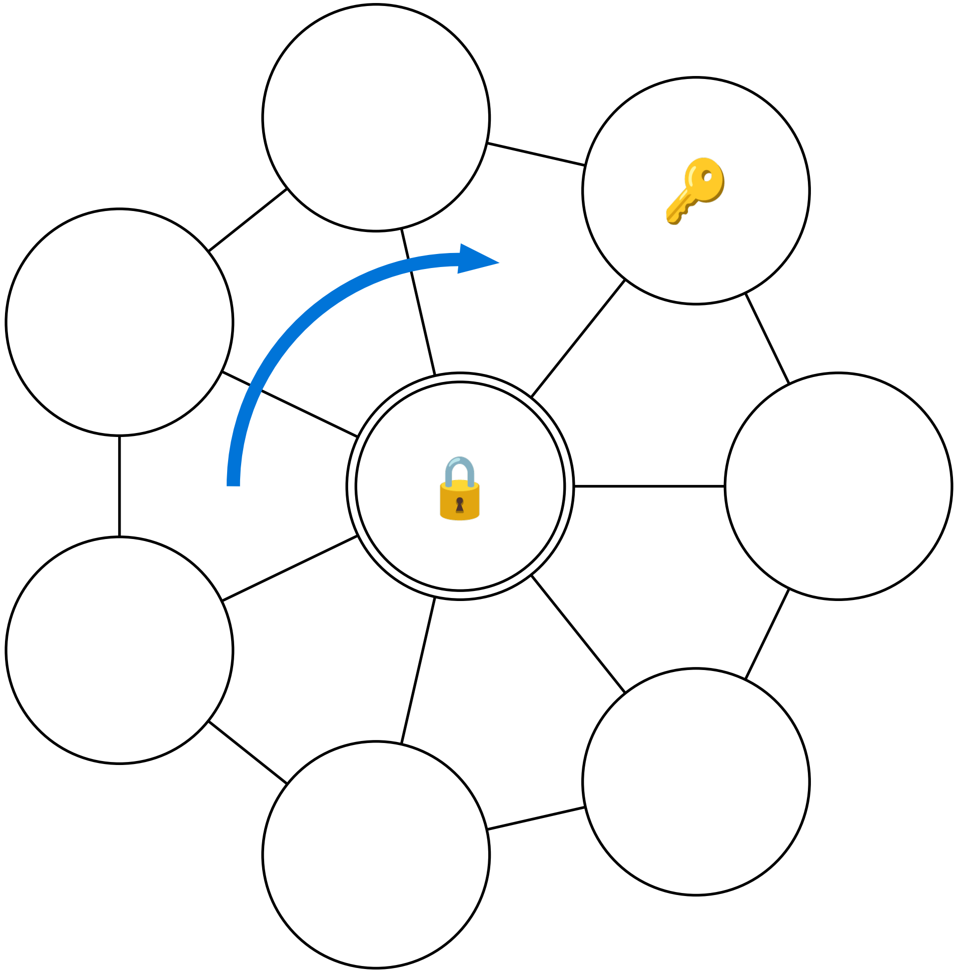
Token ring with resource



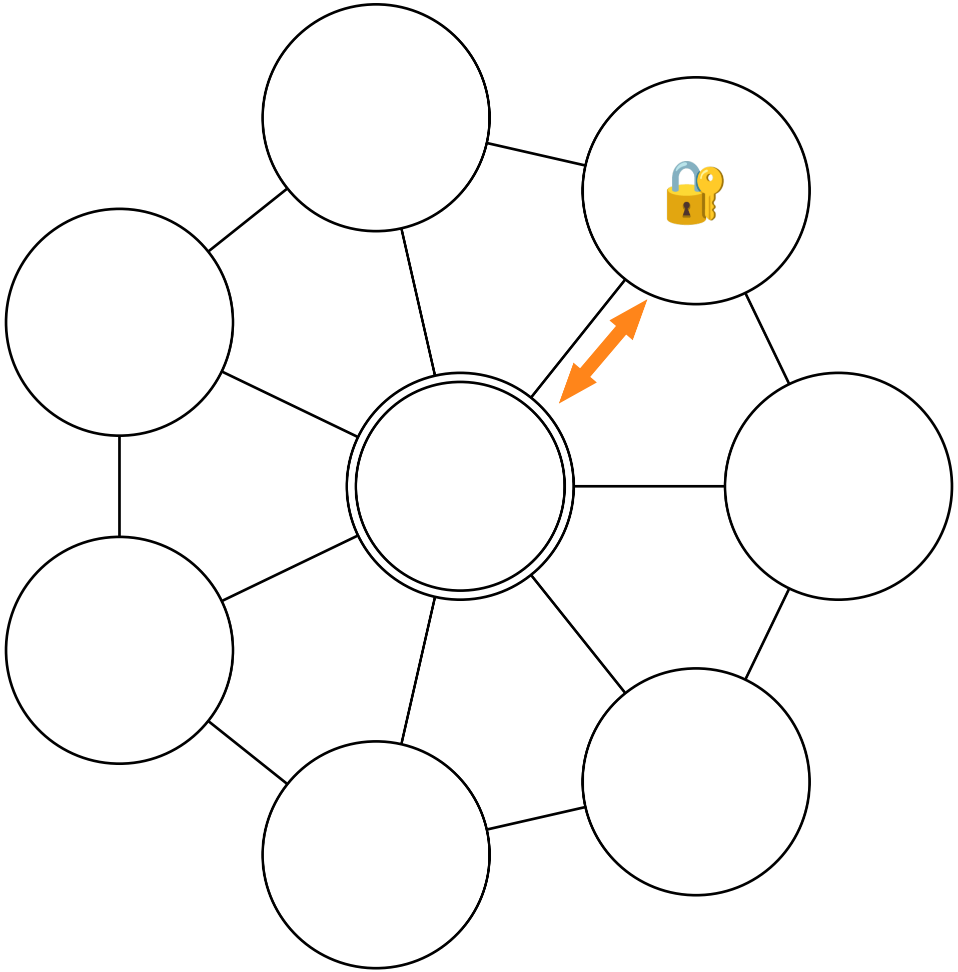
Token ring with resource



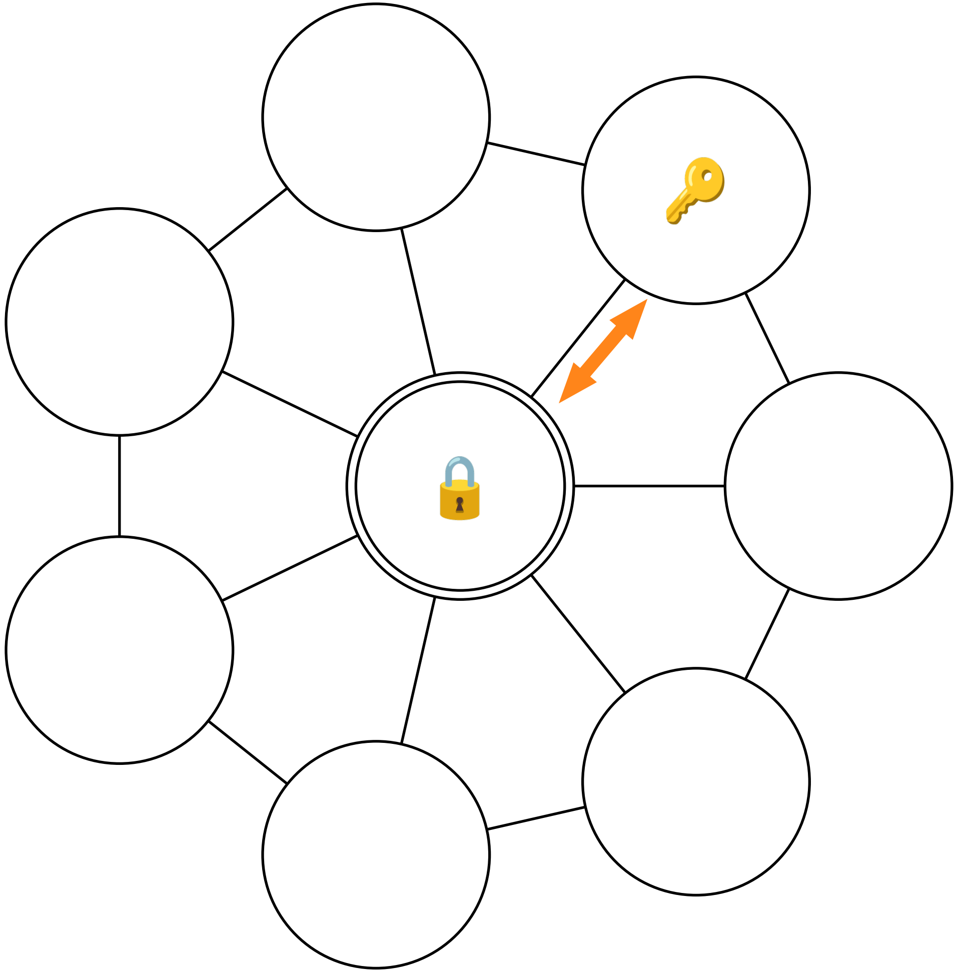
Token ring with resource



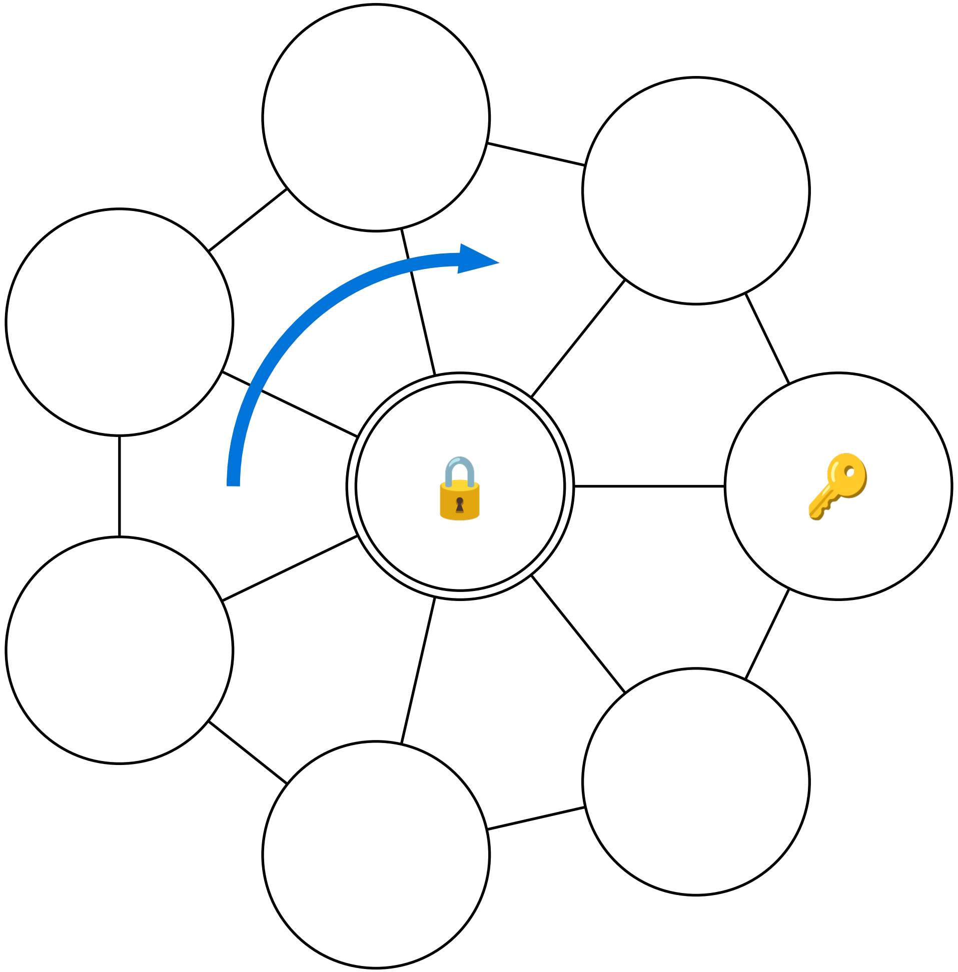
Token ring with resource



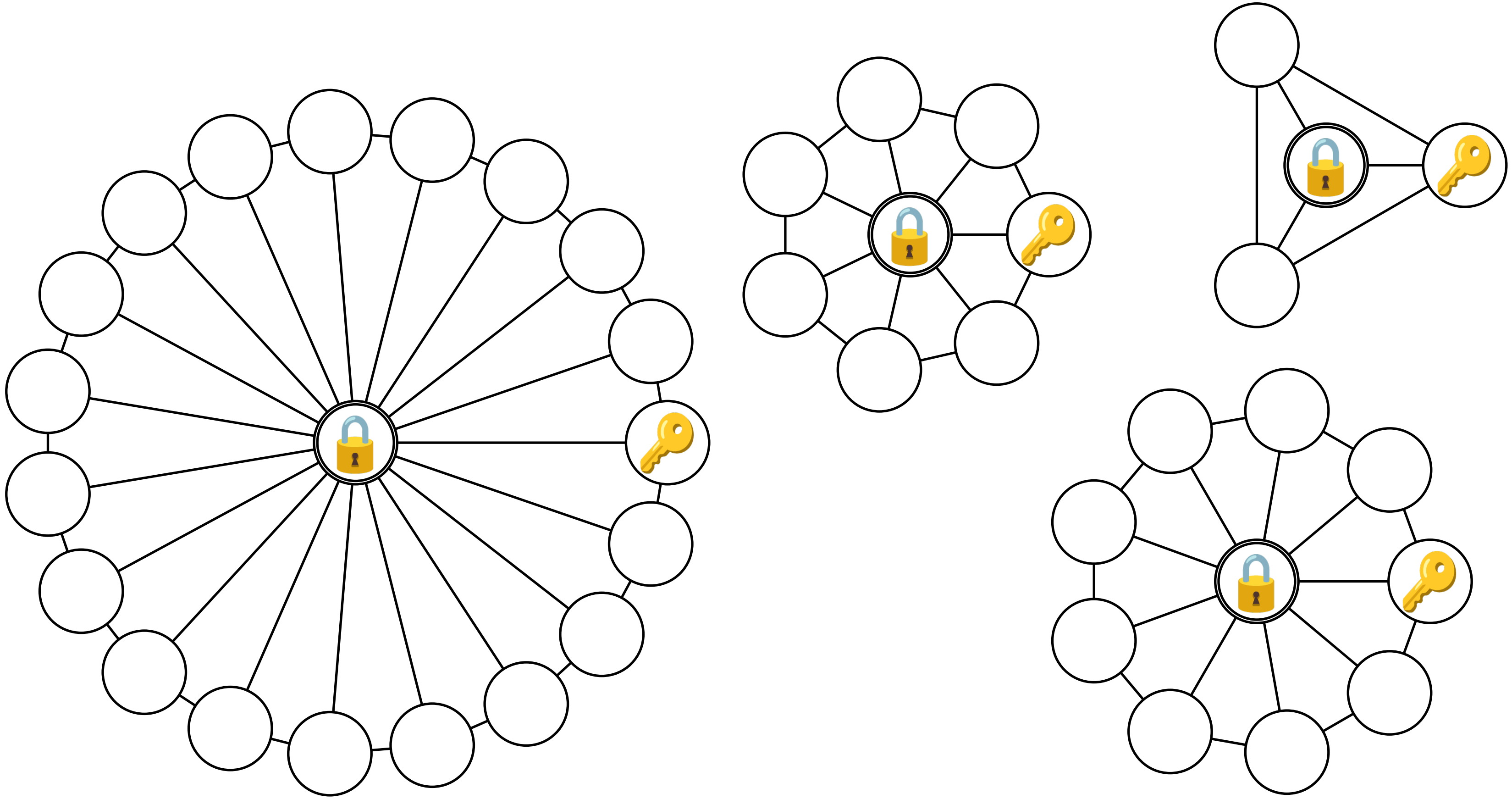
Token ring with resource



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

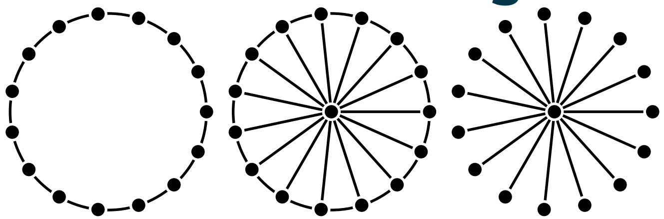
(B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith)

❌ 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

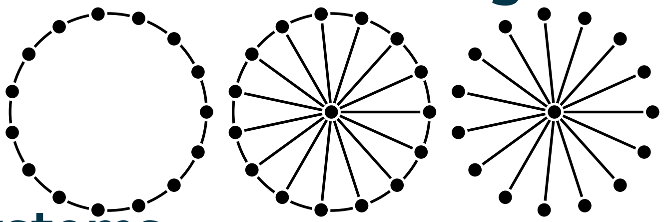
(B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith)

❌ 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)

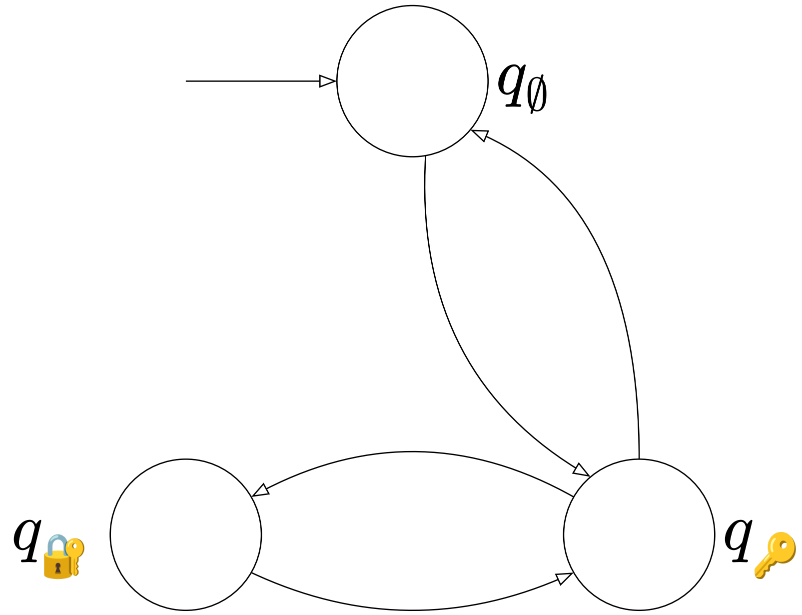
Framework requirements

We must be able to express

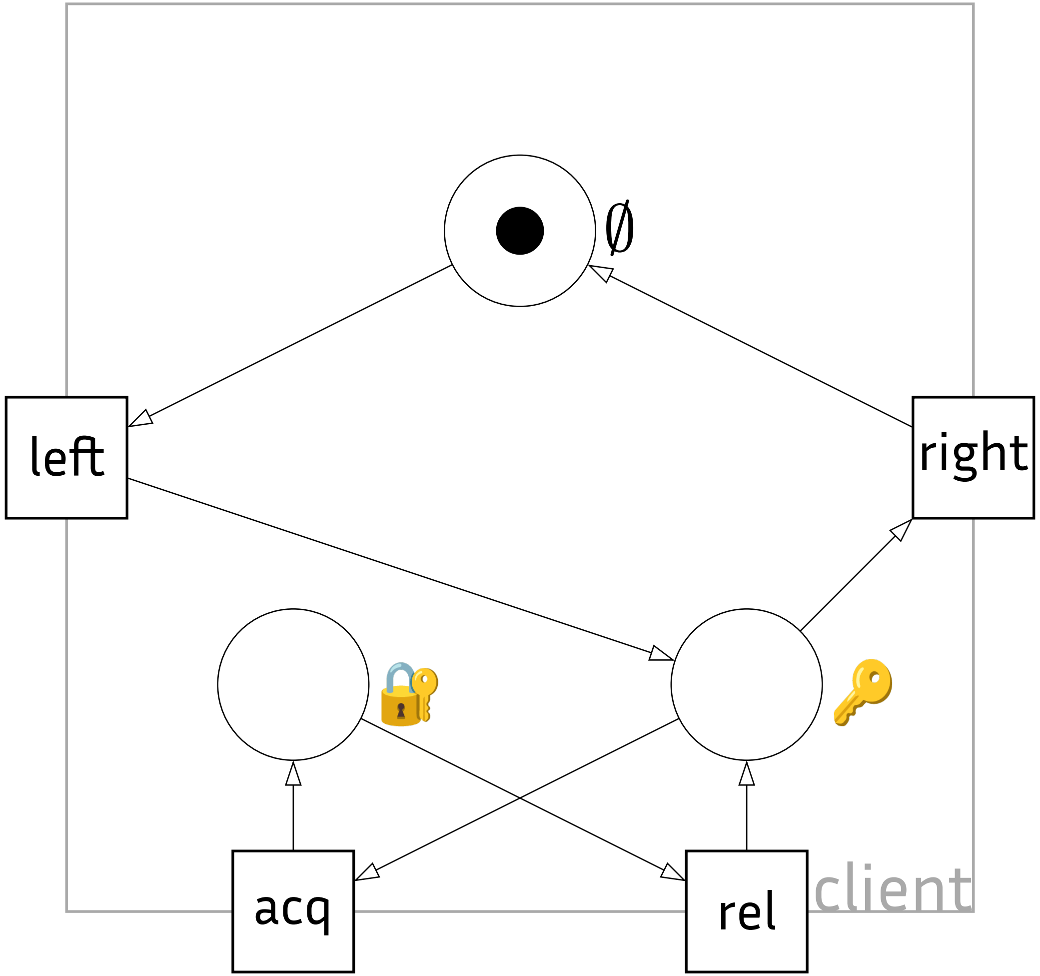
- an encoding of the **local behavior** of processes
- a description of the **interactions** and **architectures** of arbitrary size
- a **specification language** for safety properties

Framework

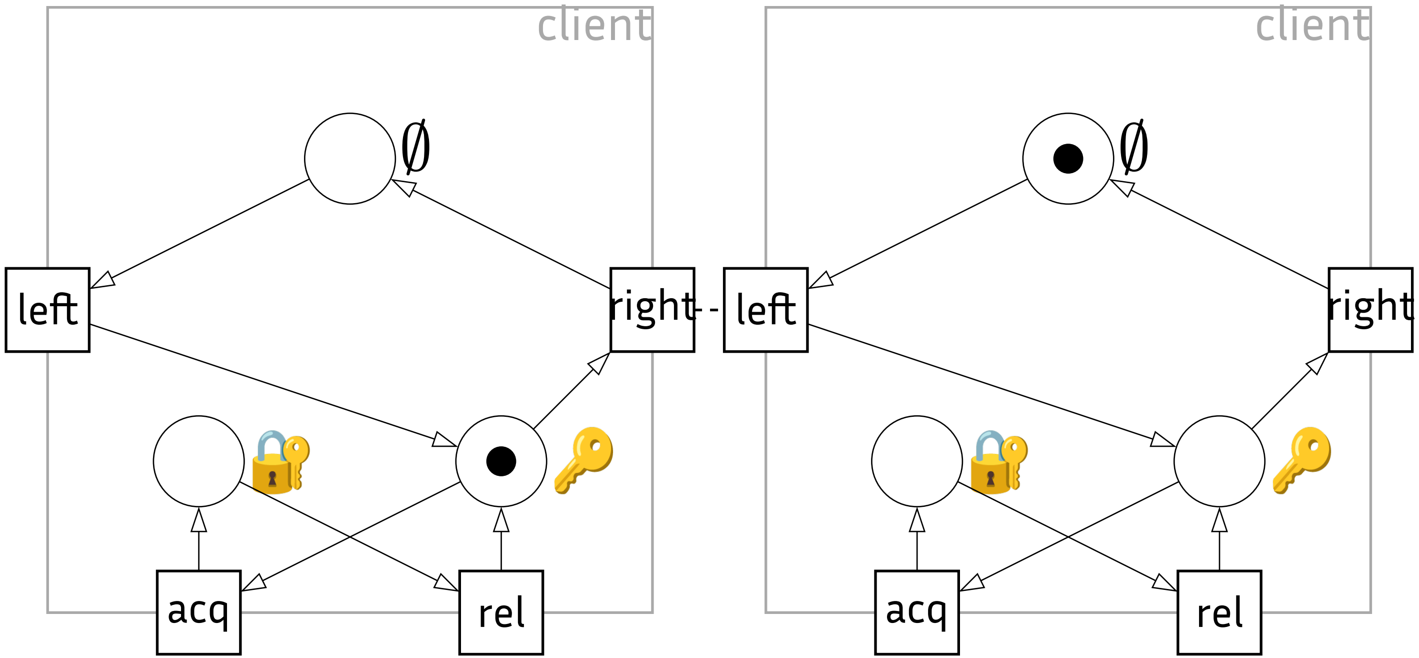
Client



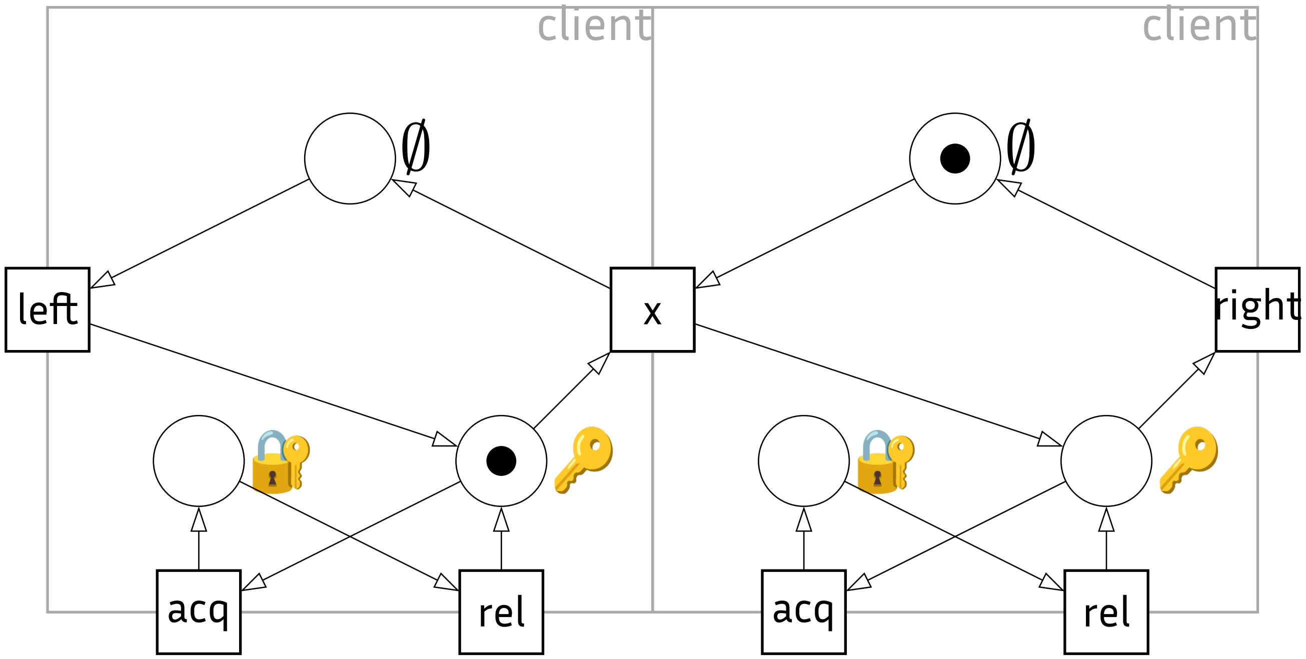
Client



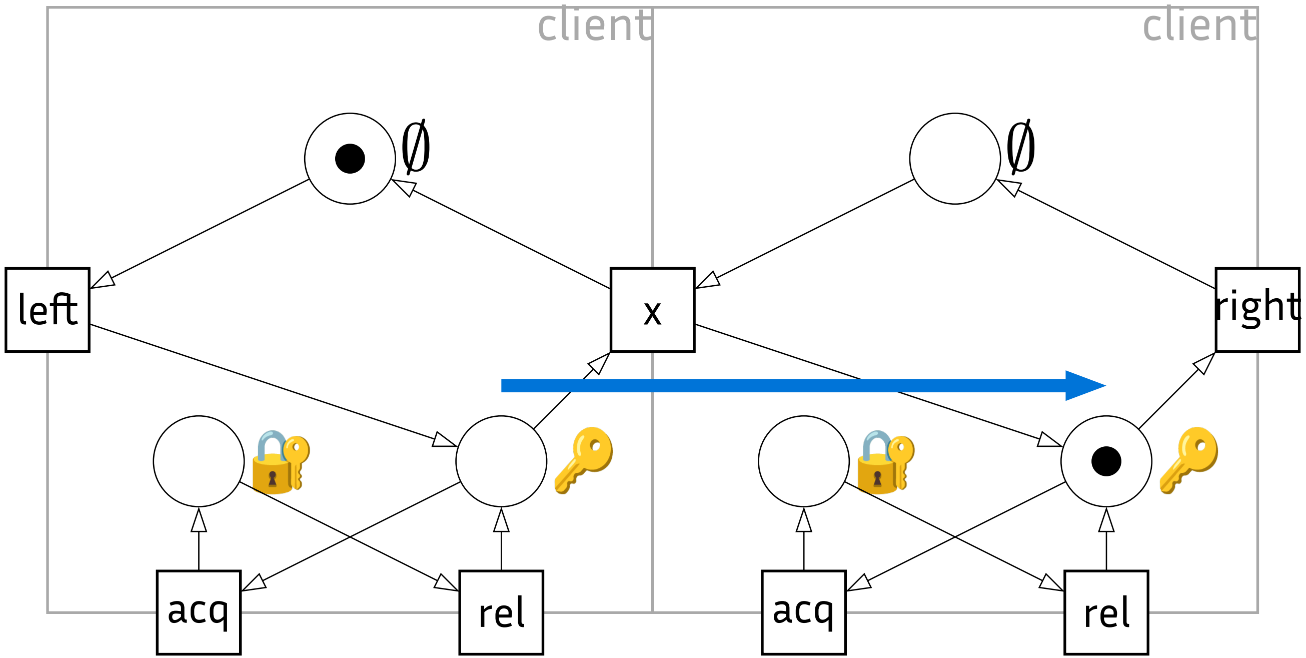
Interactions through composition



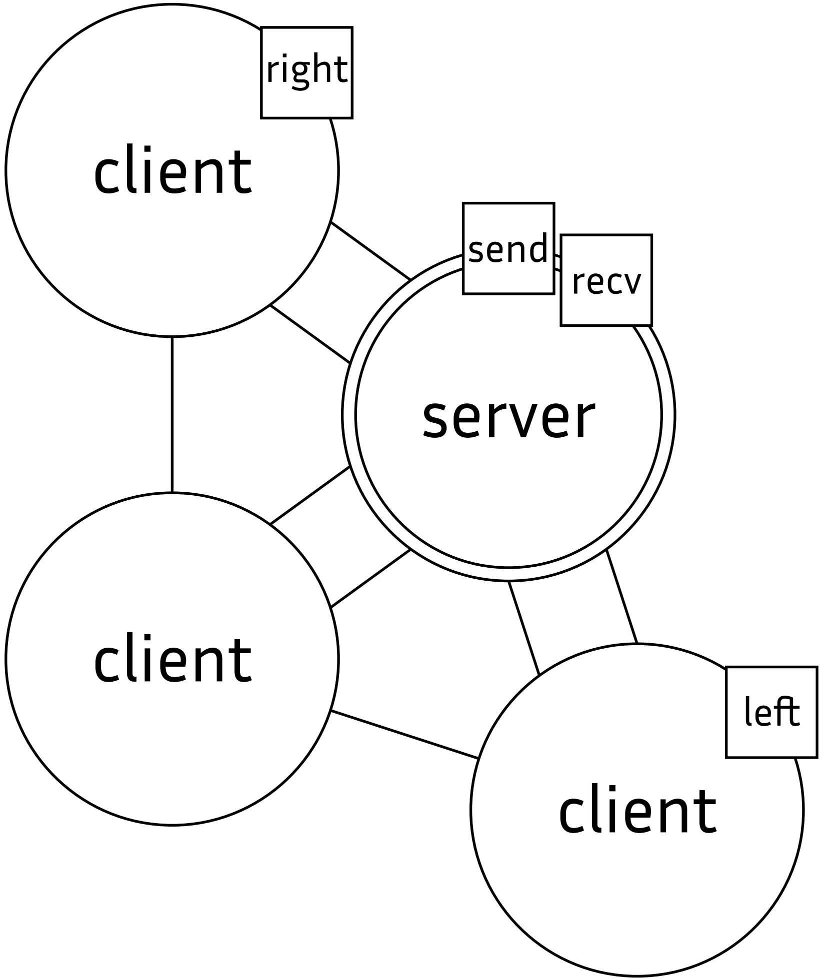
Interactions through composition



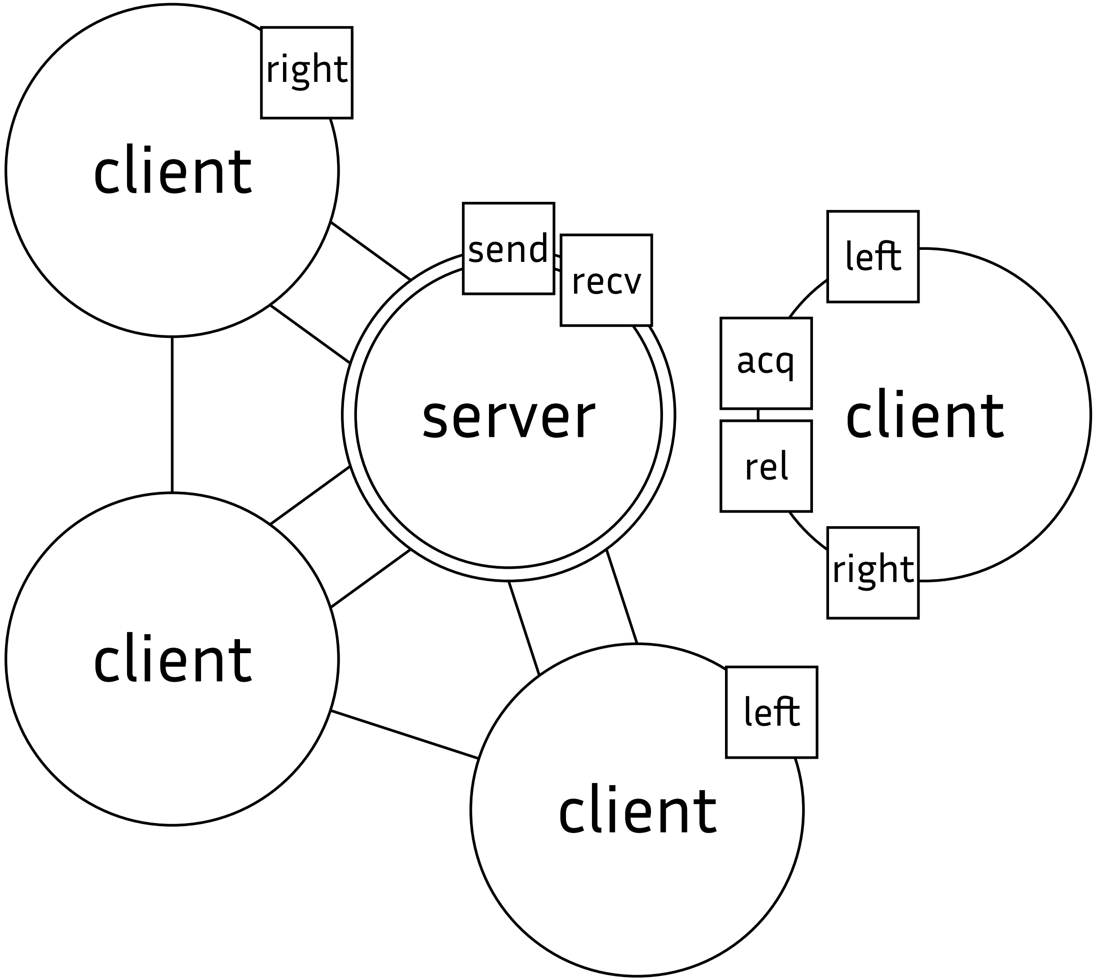
Interactions through composition



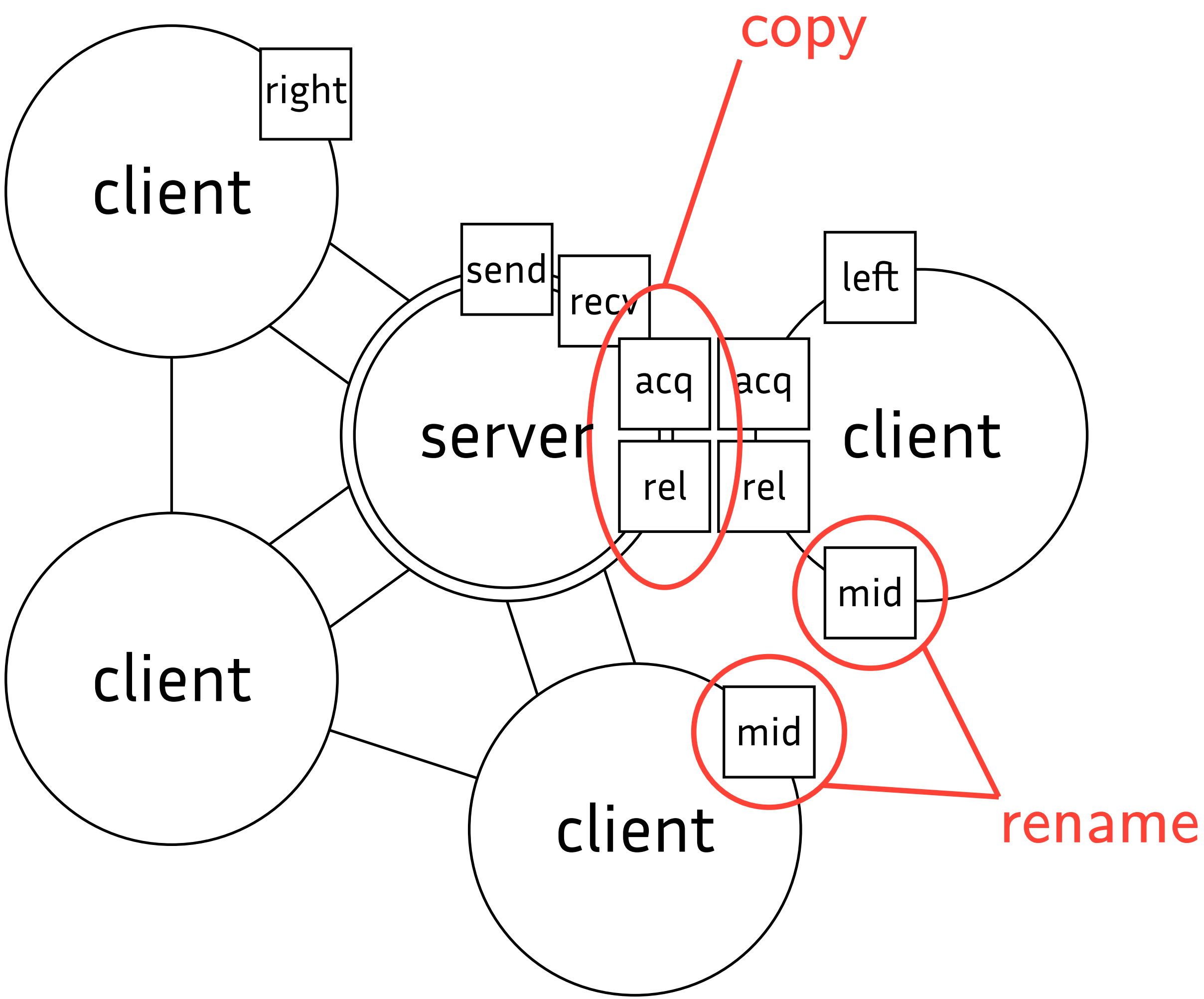
Structure: inductive step



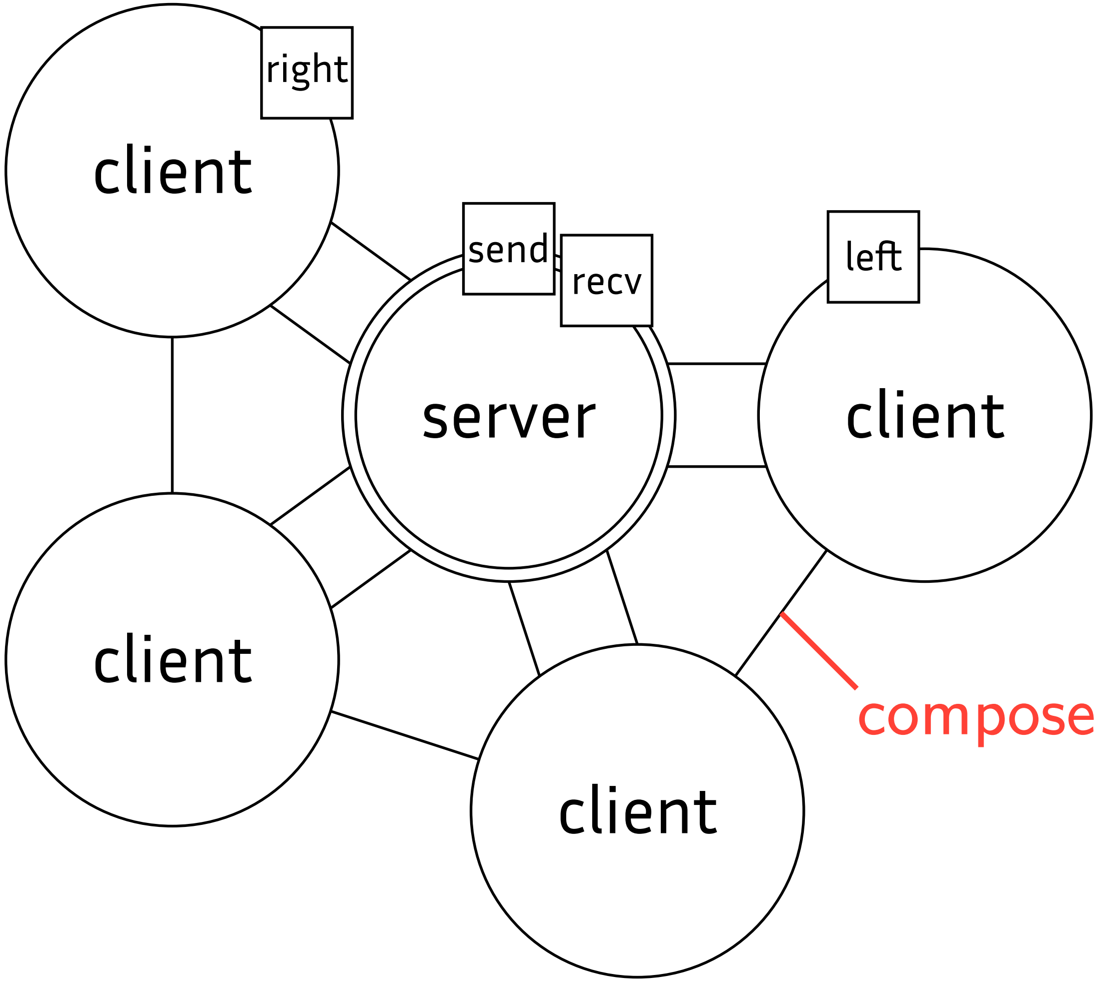
Structure: inductive step



Structure: inductive step



Structure: inductive step

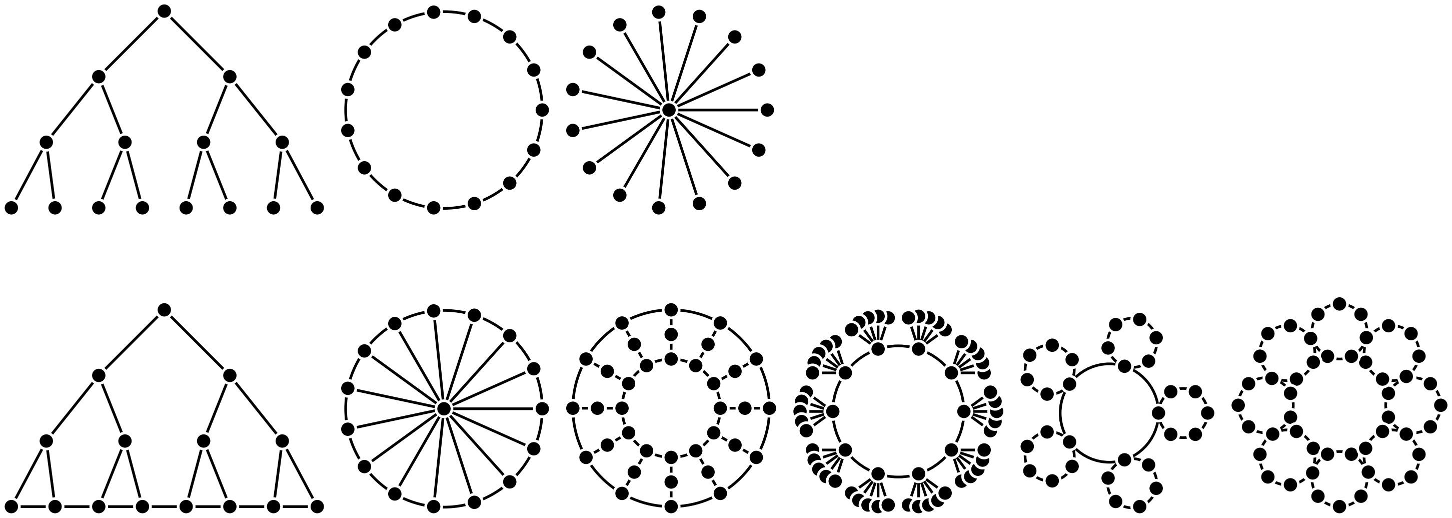


Structure represented by a grammar

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

Representable architectures

Encoded as a CFG for graphs[1] $\implies$ families of bounded TW are representable (missing: ~~square grids~~, cliques[2])



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

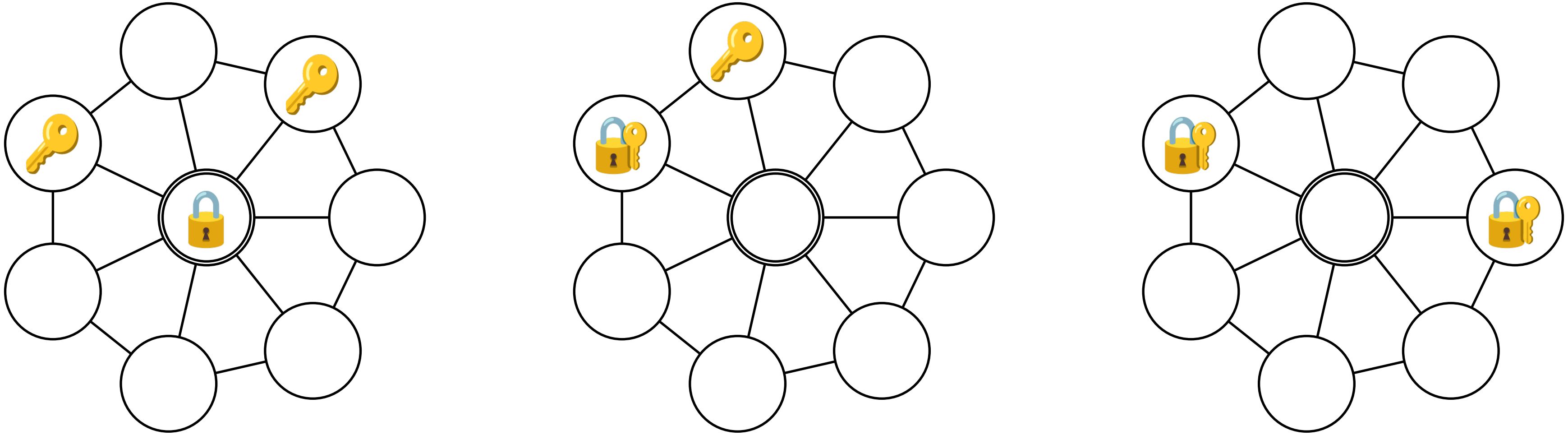
[2]Verifying Parameterized Networks Specified by Vertex Replacement Graph Grammars; by A. Sangnier, R. Iosif, N. Villani @ NETYS'25

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

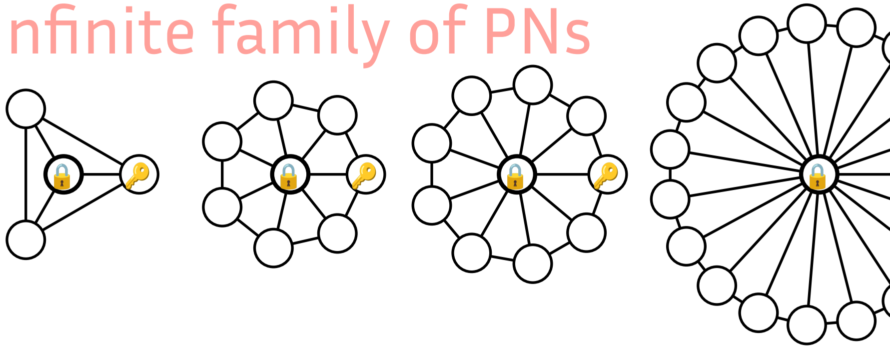
$$
\begin{aligned}
\mathrm{Sys} &\longrightarrow \mathsf{compose}\big(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')\big)\\
\mathrm{X} &\longrightarrow \mathsf{compose}\big(\\
&\qquad \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{mid}}\big(\mathsf{copy}_{\mathrm{send}\rightsquigarrow\mathrm{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

Verification pipeline

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

$\xrightarrow{\text{language}}$

infinite family of PNs

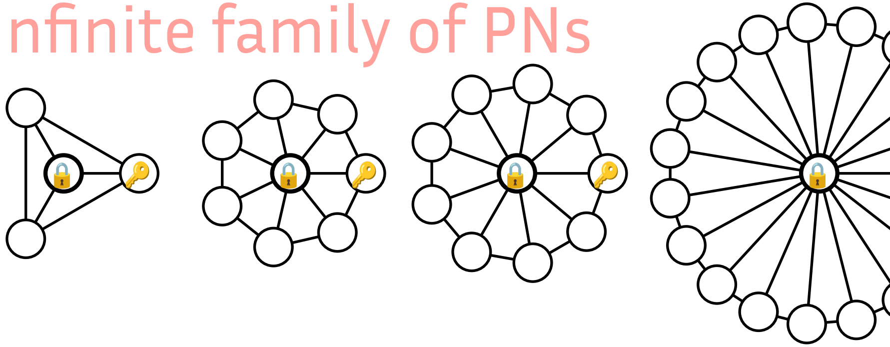


Verification pipeline

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

language

infinite family of PNs



safety

$\#(🔑) + \#(🔐) > 1$

Verification pipeline

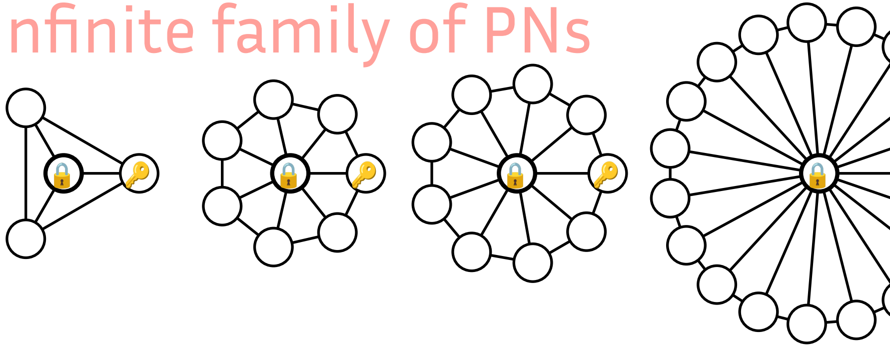
$$\mathrm{Sys} \longrightarrow \mathsf{compose}\big(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}')\big)$$

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

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

language

infinite family of PNs



~~safety~~

undecidable!

$$\boxed{\#(🔑) + \#(🔐) > 1}$$

Verification pipeline

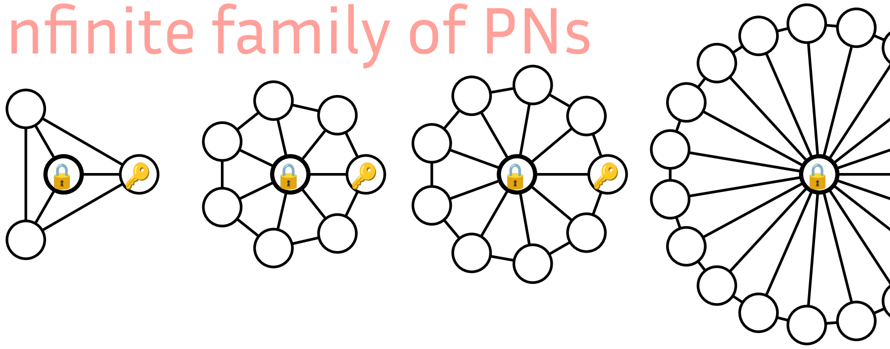
$$\mathrm{Sys} \longrightarrow \mathsf{compose}(\mathrm{X}, \mathsf{rename}_{\mathrm{left}\mapsto\mathrm{right},\ \mathrm{right}\mapsto\mathrm{left}}(\mathrm{client}'))$$

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

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

language

infinite family of PNs

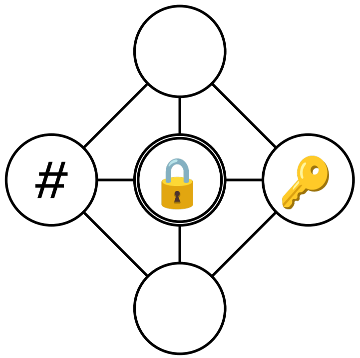


abstraction

$\supseteq$

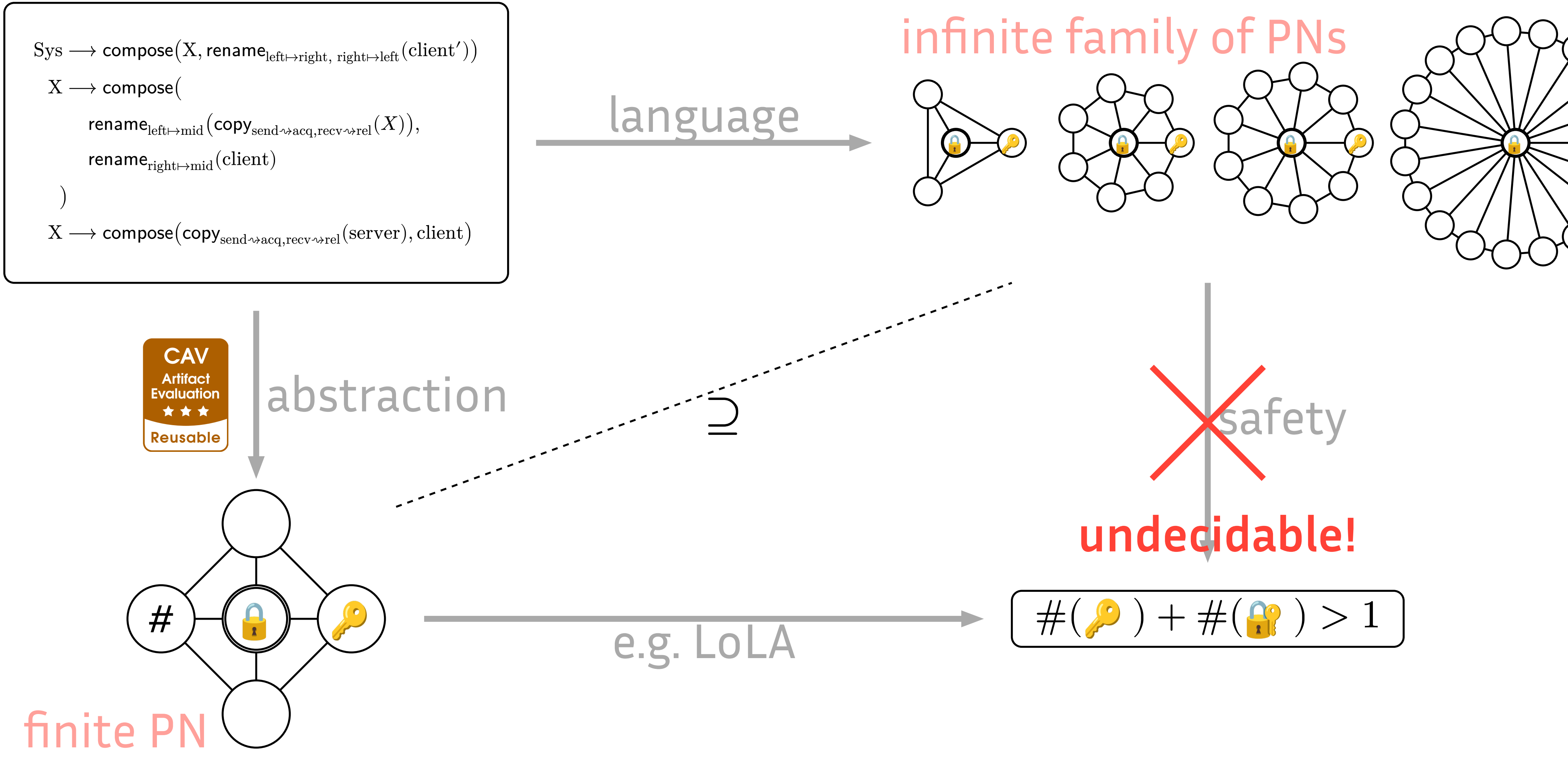
~~safety~~

undecidable!

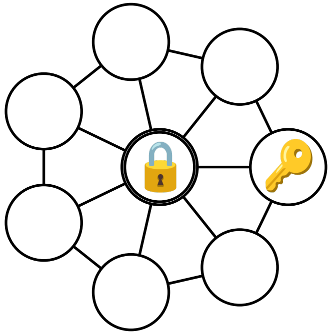


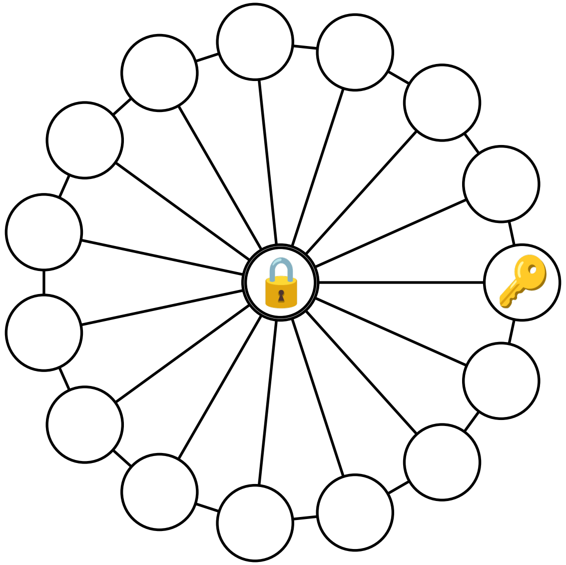
$$\boxed{\#(🔑) + \#(🔐) > 1}$$

Verification pipeline

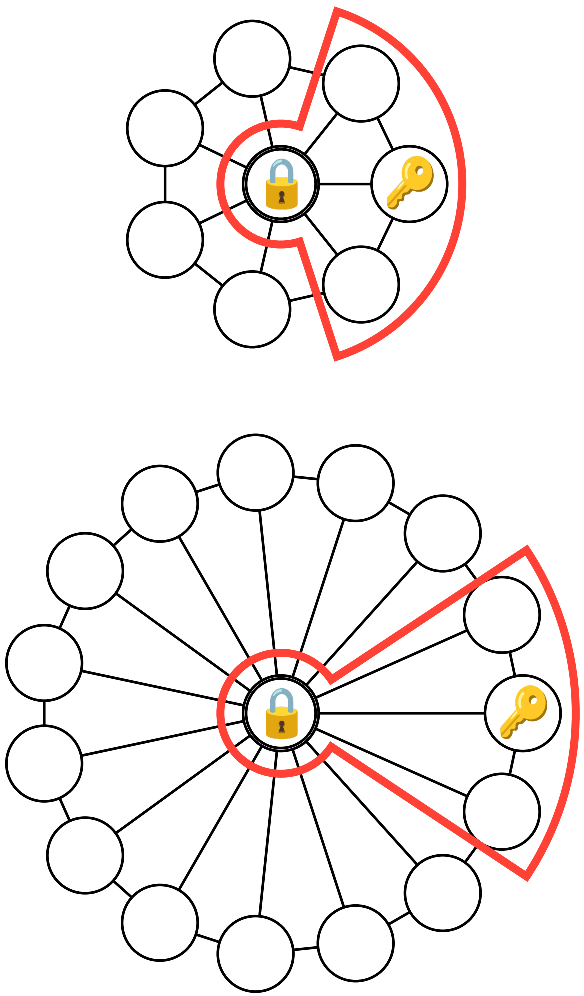


Folding abstraction

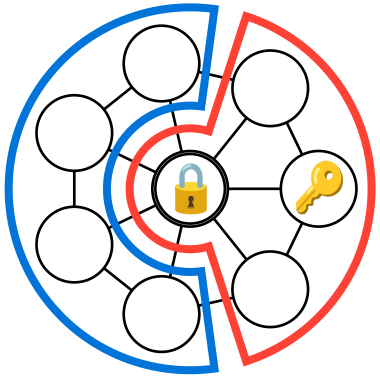


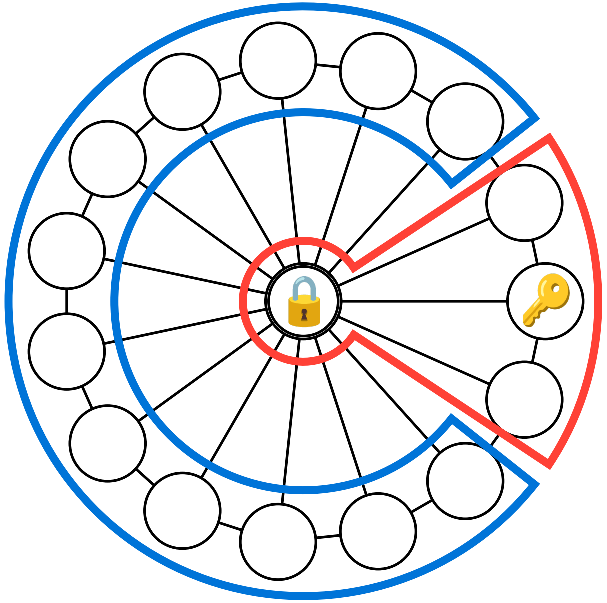


Folding abstraction

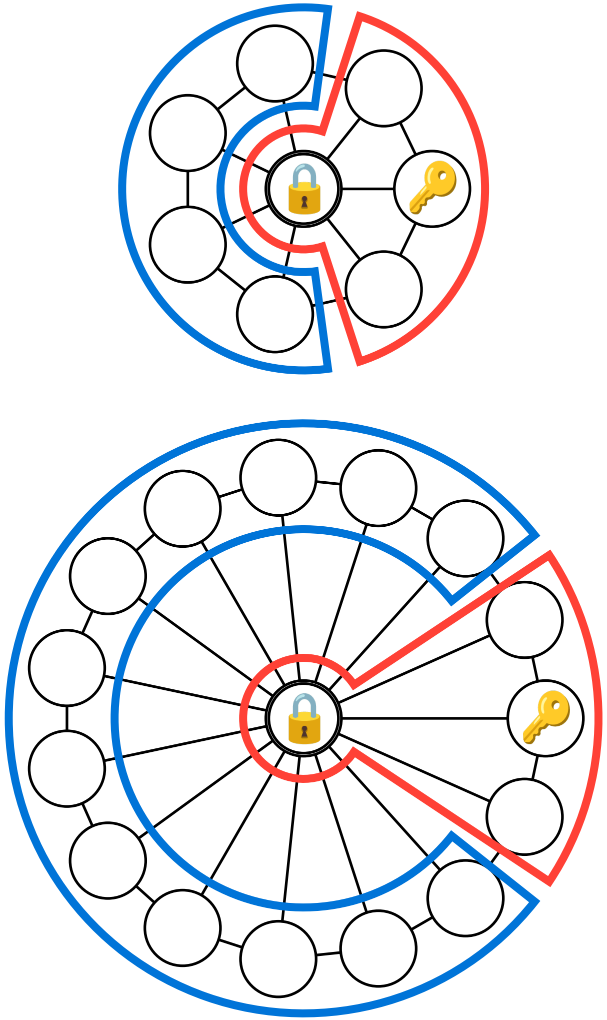


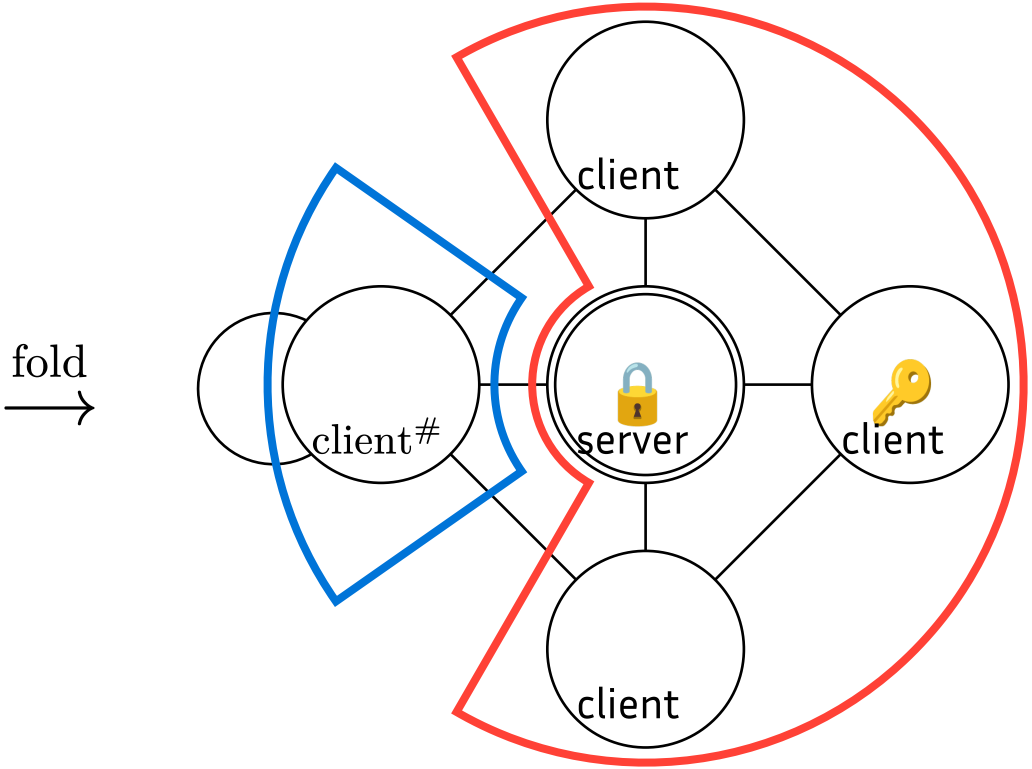
Folding abstraction



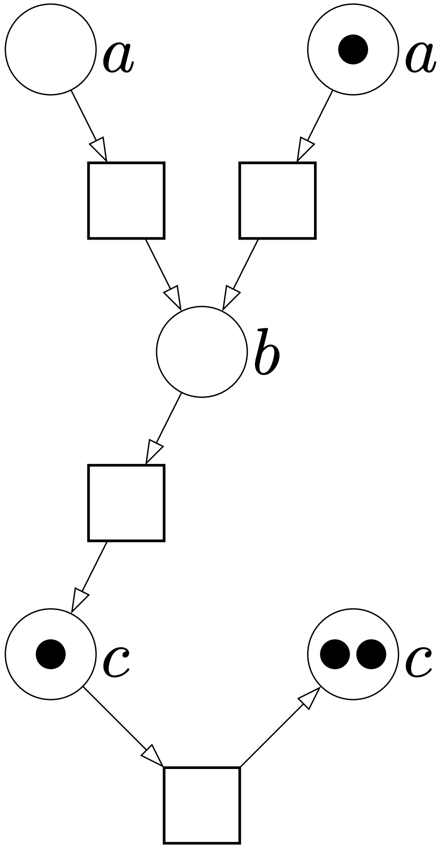


Folding abstraction

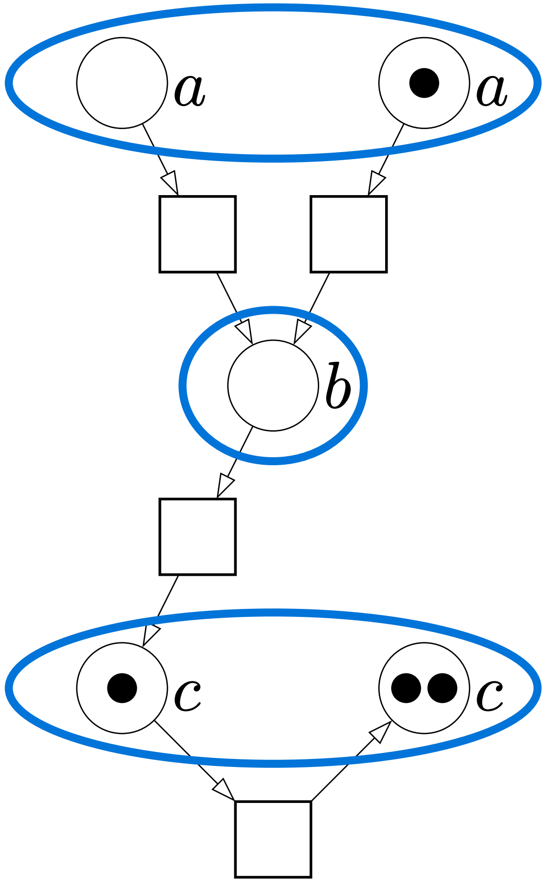




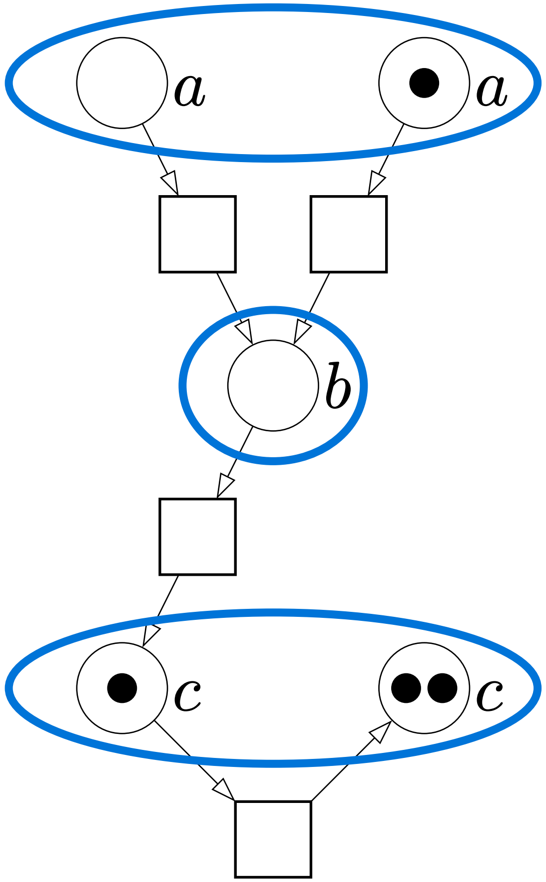
Implementing fold



Implementing fold

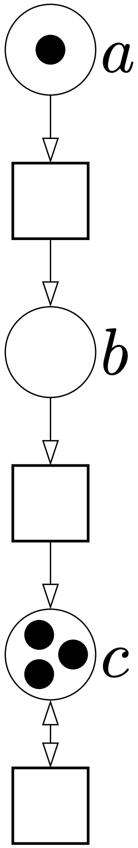


Implementing fold

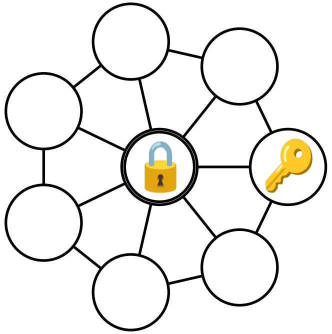


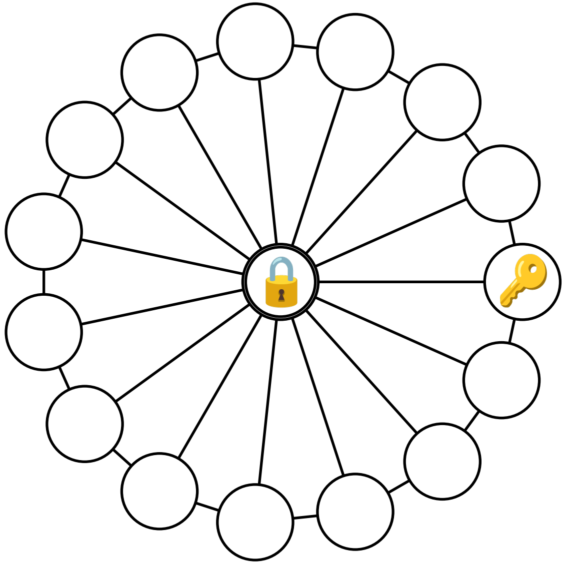
$\xrightarrow{\text{fold}}$

$\dashleftarrow$
$\subseteq$



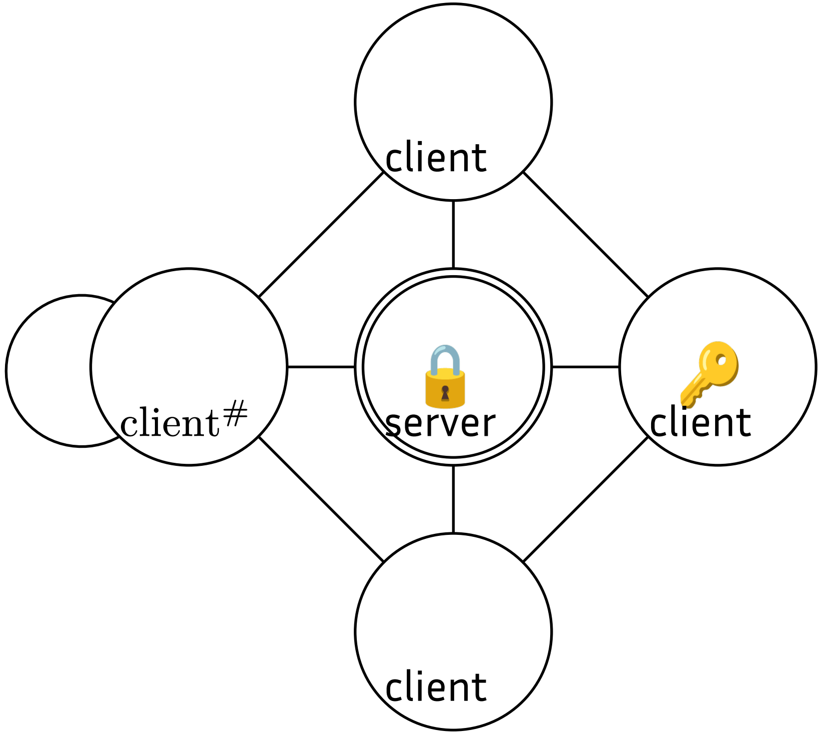
Implementing fold



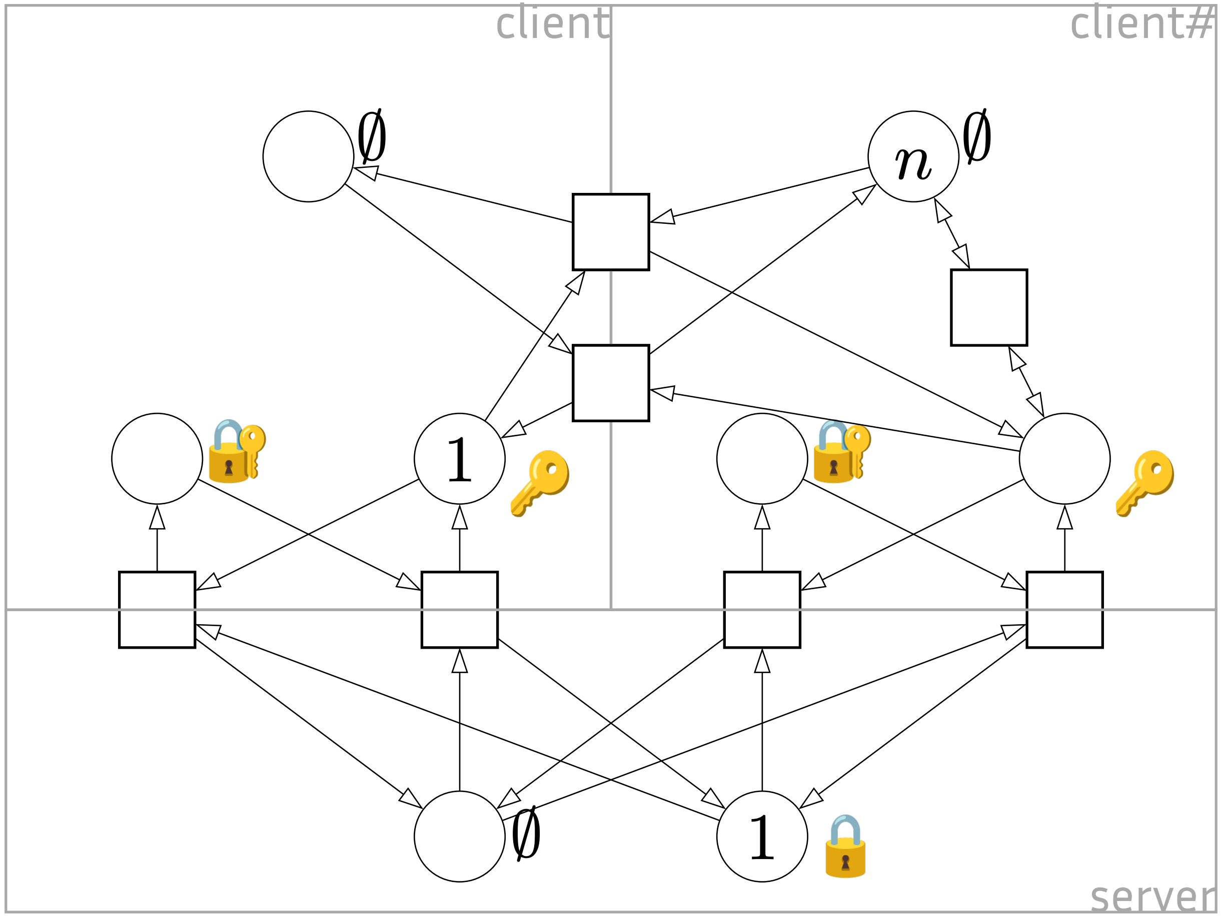


$\xrightarrow{\text{fold}}$

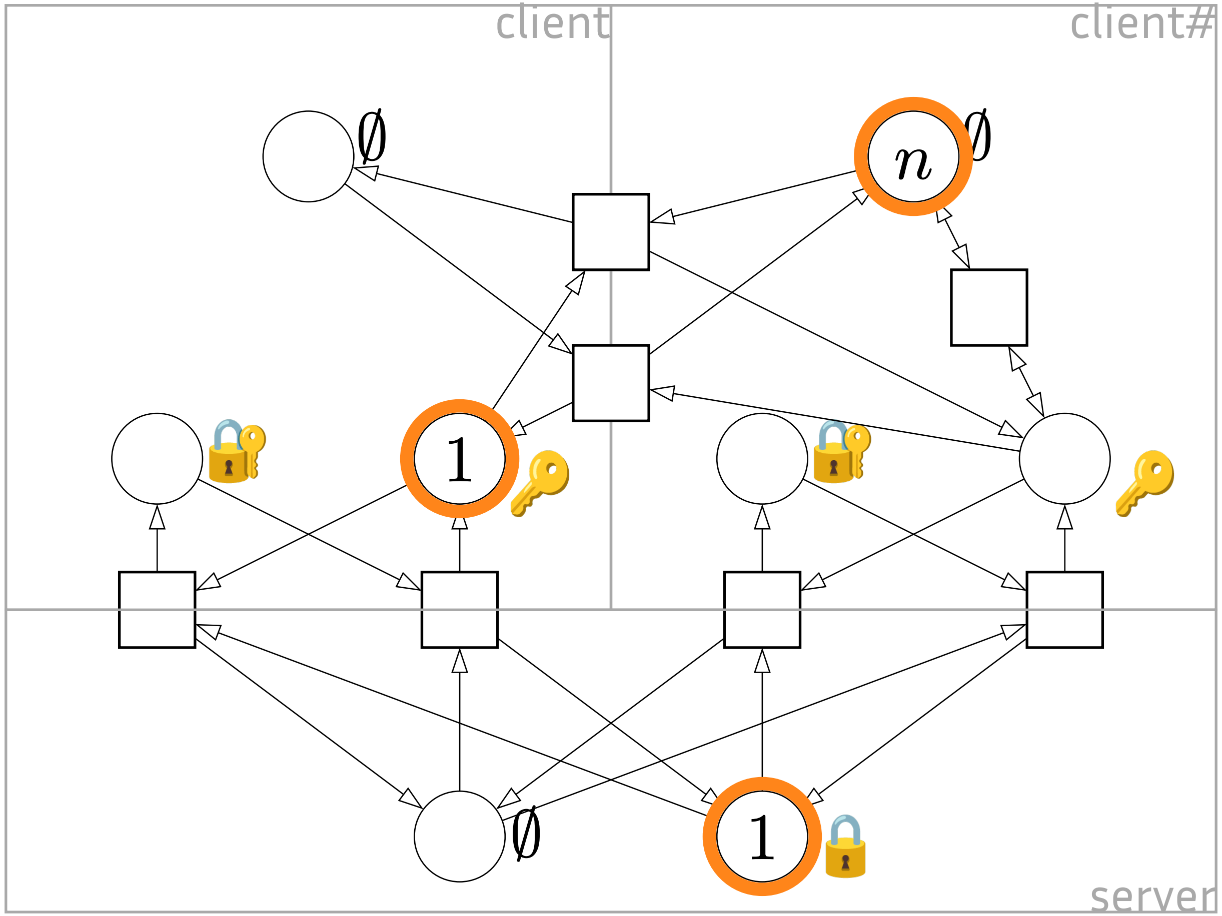
$\overset{\dashleftarrow}{\subseteq}$



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

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

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

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

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

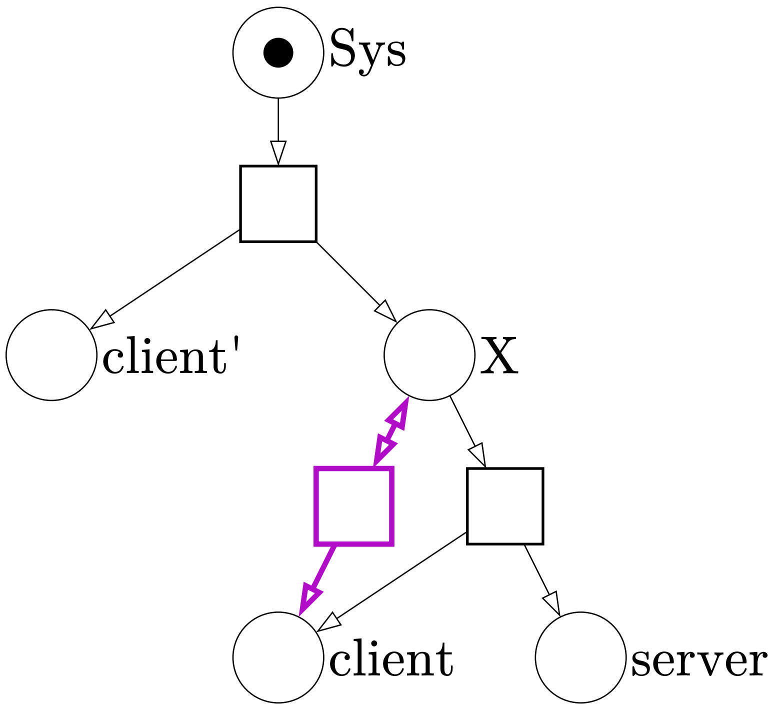
$$\big)$$

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

Initial marking

From the grammar

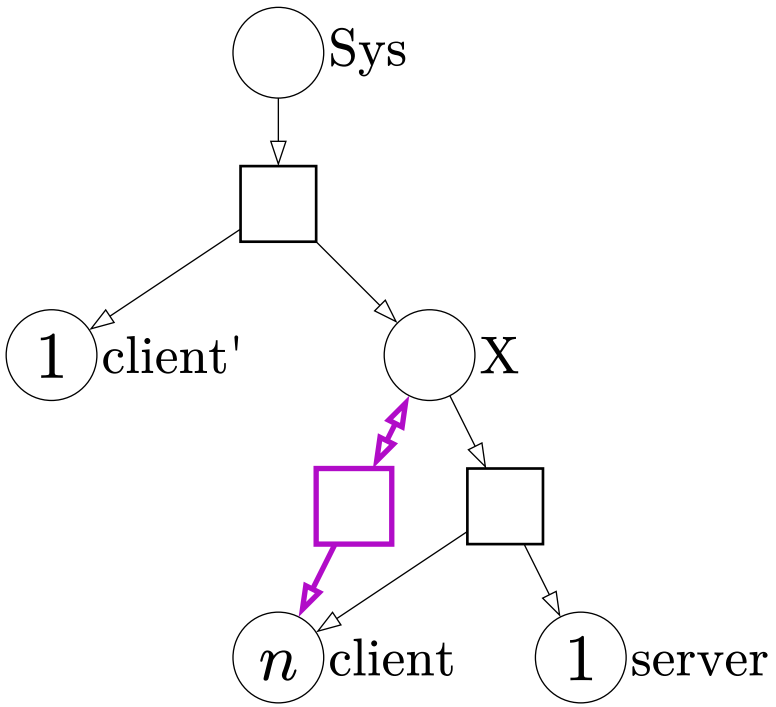
$$\mathrm{Sys} \longrightarrow X, \mathrm{client'}$$
$$X \longrightarrow X, \mathrm{client}$$
$$X \longrightarrow \mathrm{server}, \mathrm{client}$$



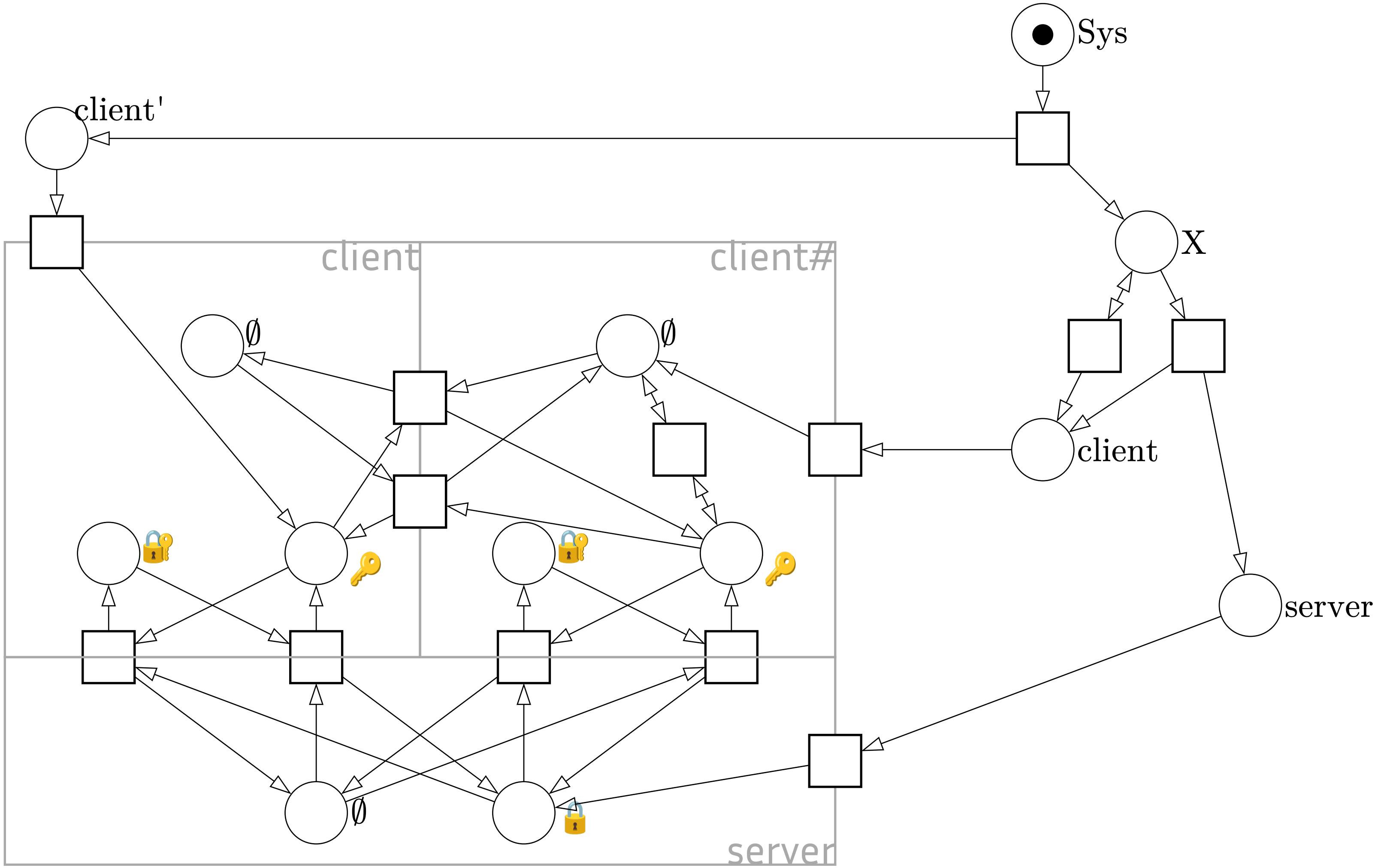
Initial marking

From the grammar

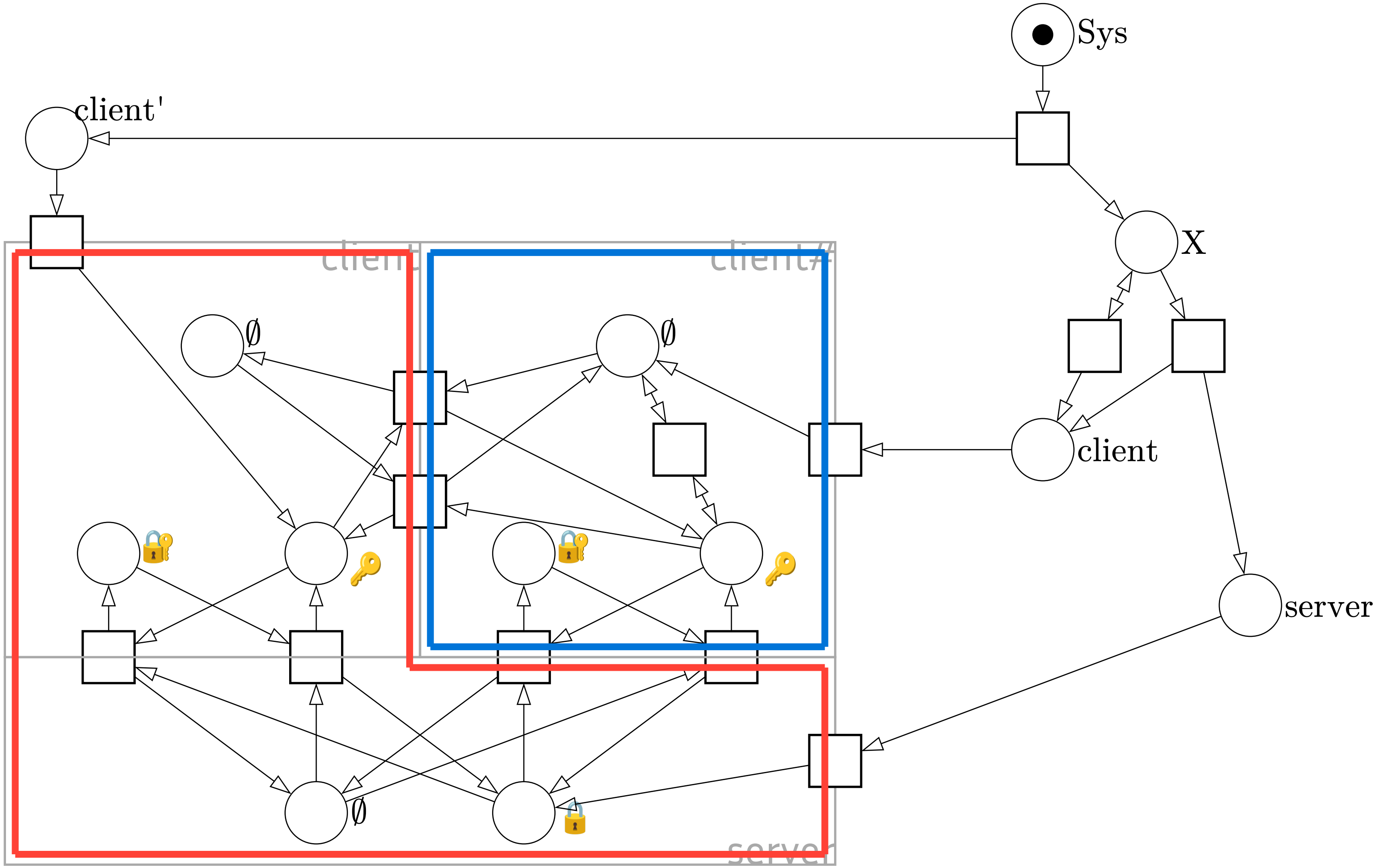
$$\mathrm{Sys} \longrightarrow X, \mathrm{client'}$$
$$X \longrightarrow X, \mathrm{client}$$
$$X \longrightarrow \mathrm{server}, \mathrm{client}$$



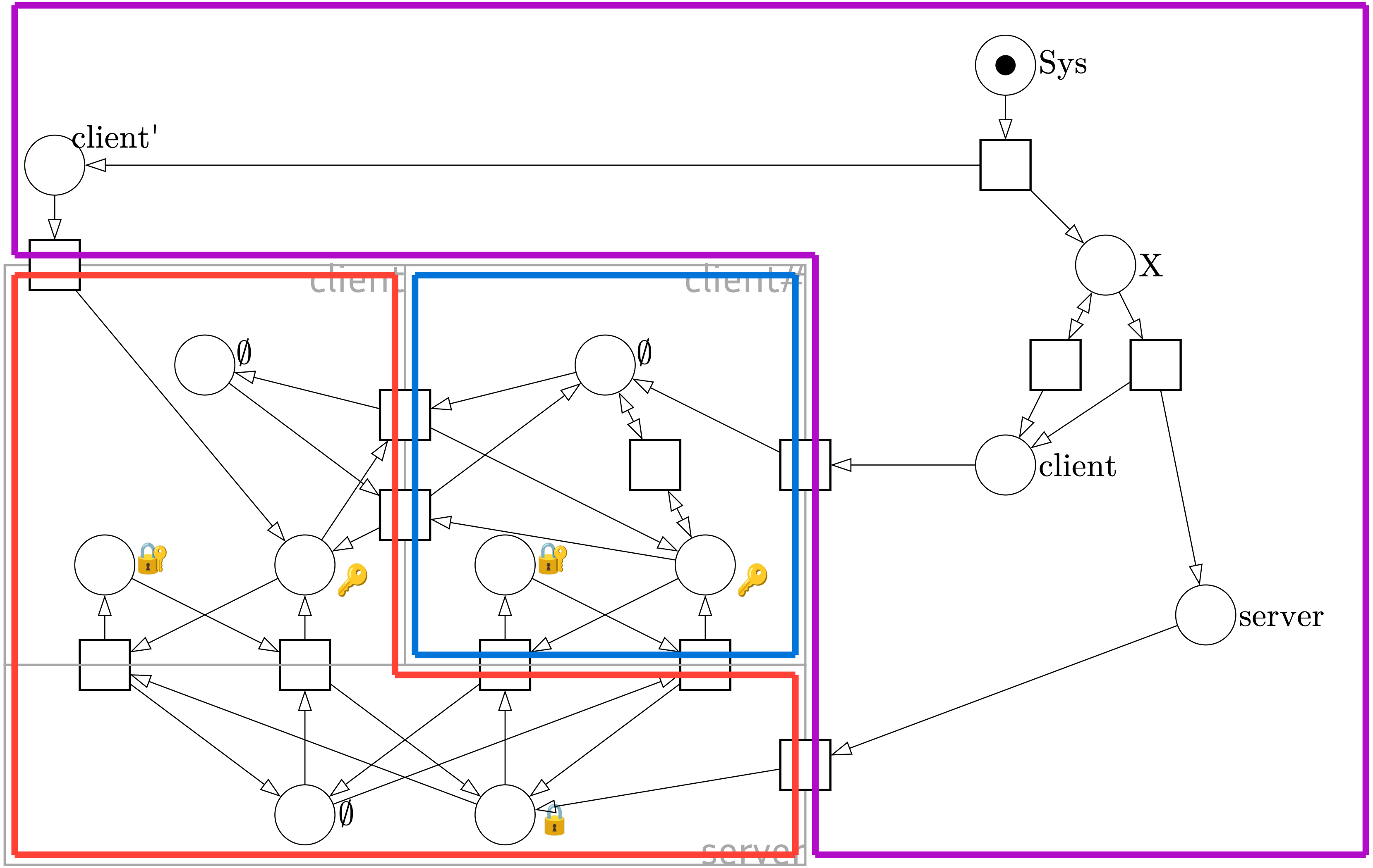
Finite abstract system



Finite abstract system



Finite abstract system



Conclusion

Refinements

Partial unfolding, boolean contracts

A decidable restriction

Pebble-Passing Systems

- multiple process types, multiple tokens
- processes only record the presence of a token

(similar to but incomparable with Token-Passing Systems)

Coverability of a control state is **2EXPTIME** and **PSPACE-hard**