Verifying Parameterized Networks
Specified by
Vertex-Replacement Graph Grammars

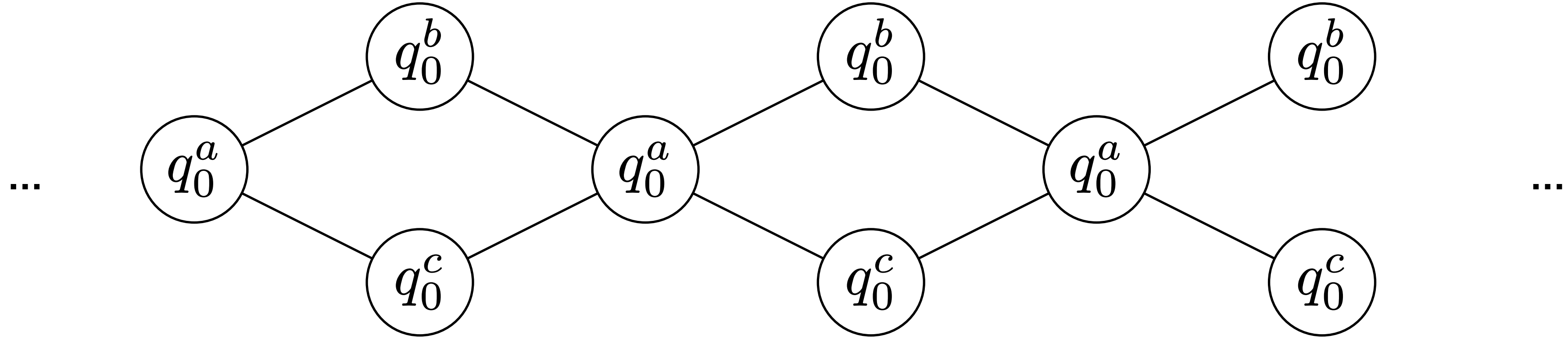
Neven Villani, Radu Iosif, Arnaud Sangnier

Univ. Grenoble Alpes, Verimag

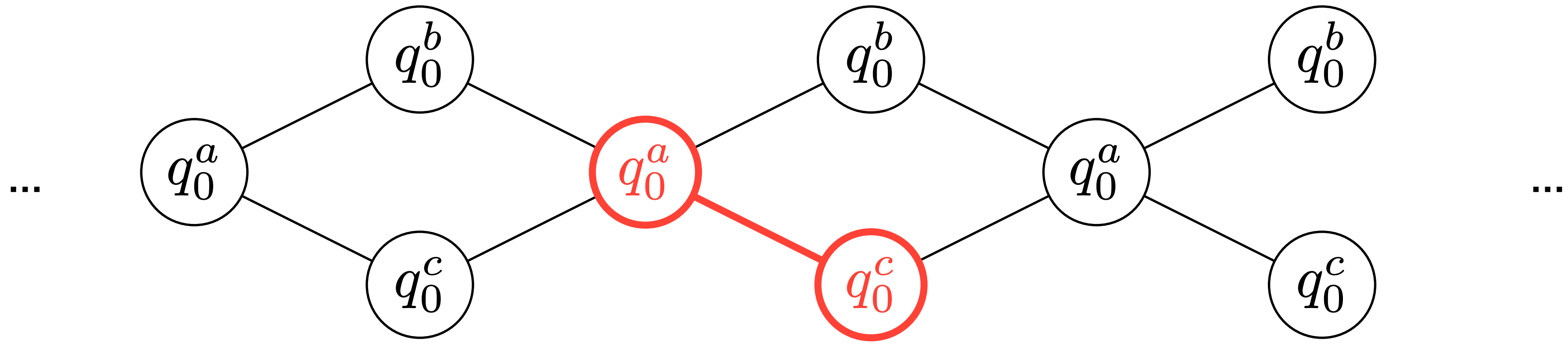
2025-05-21; NETYS (Rabat)

Context

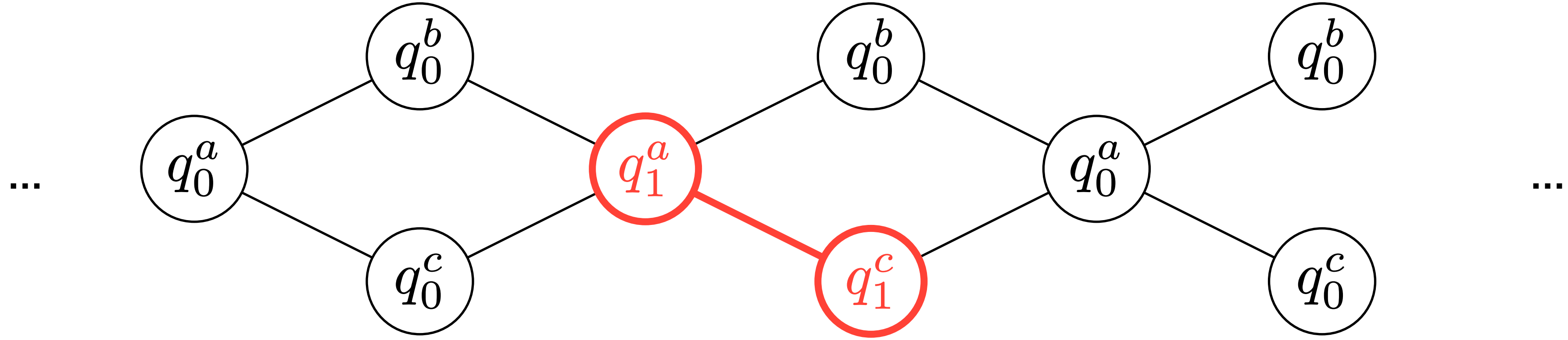
- Parameterized structured networks
- Non-homogeneous
- Binary rendezvous
- Locally finite memory
- Safety properties



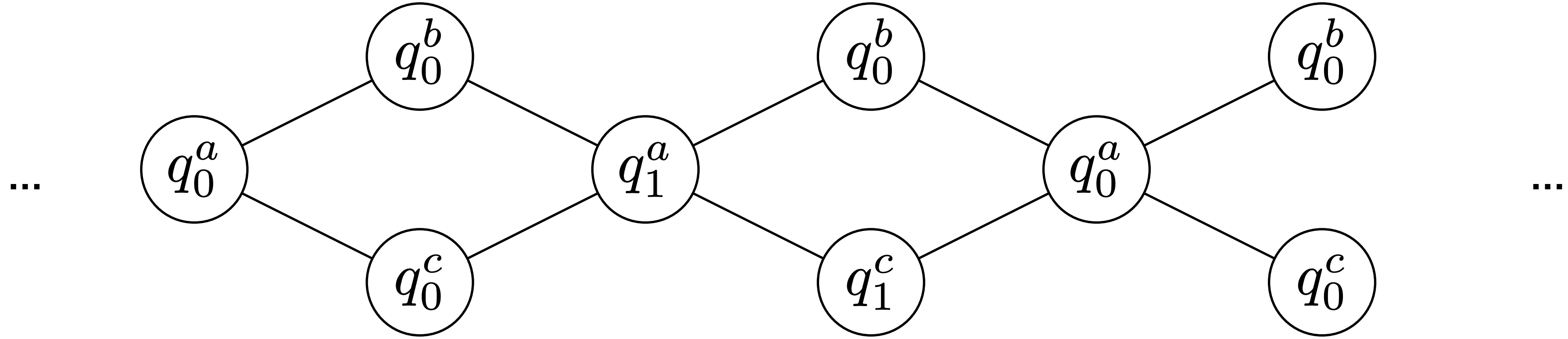
- Parameterized structured networks
- Non-homogeneous
- Binary rendezvous
- Locally finite memory
- Safety properties



- Parameterized structured networks
- Non-homogeneous
- Binary rendezvous
- Locally finite memory
- Safety properties



- Parameterized structured networks
- Non-homogeneous
- Binary rendezvous
- Locally finite memory
- Safety properties



$\#q_1^b > 1 \wedge \#q_1^c > 1$ reachable ?

processes

architecture

$$(P, \Gamma) \models ?\ \varphi$$

safety property

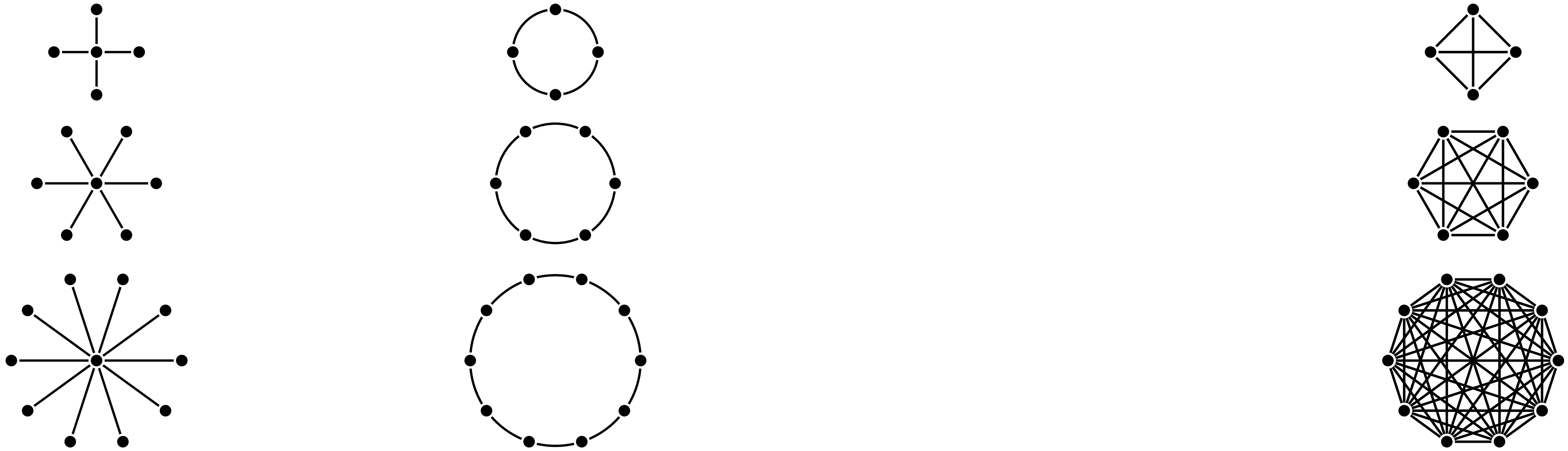
processes

architecture

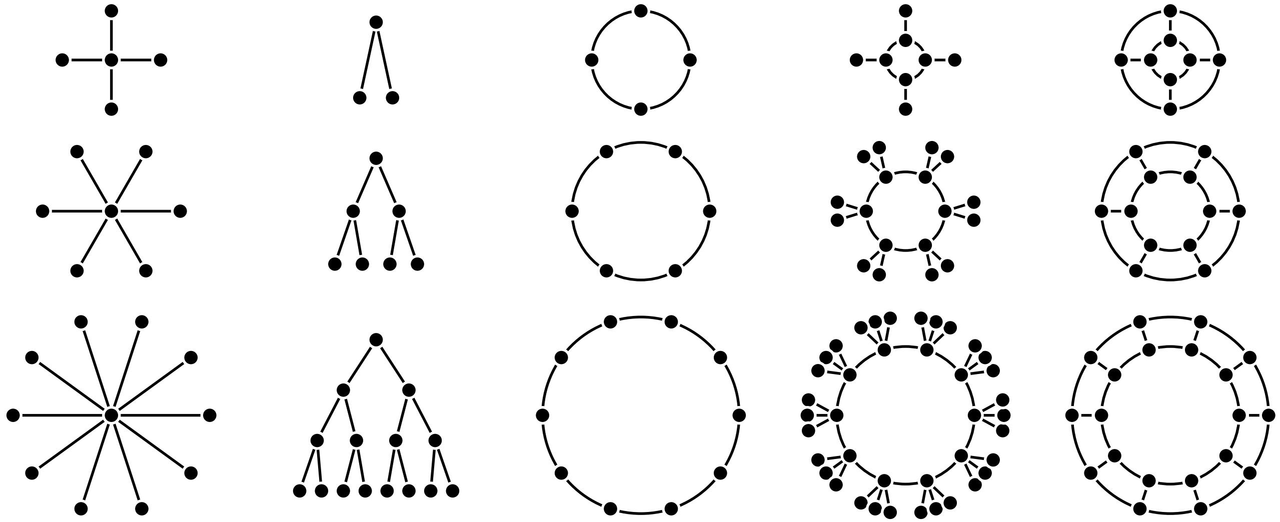
$$(P, \Gamma) \models ?\ \varphi$$

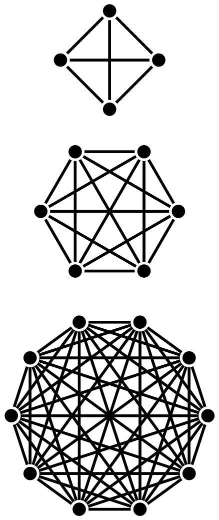
safety property

What formalism to describe architectures ?



What formalism to describe architectures ?

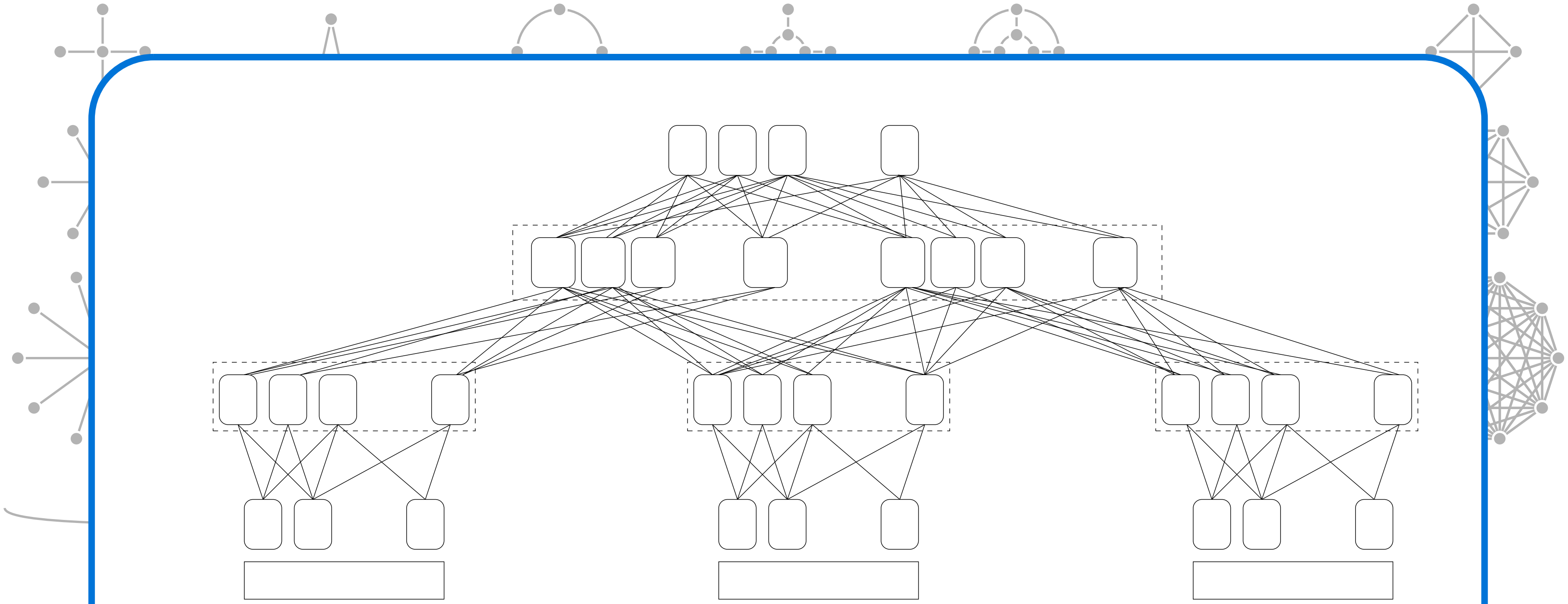




Hyperedge **R**eplacement (sparse only)

e.g., us, VDS 2025 + CAV 2025

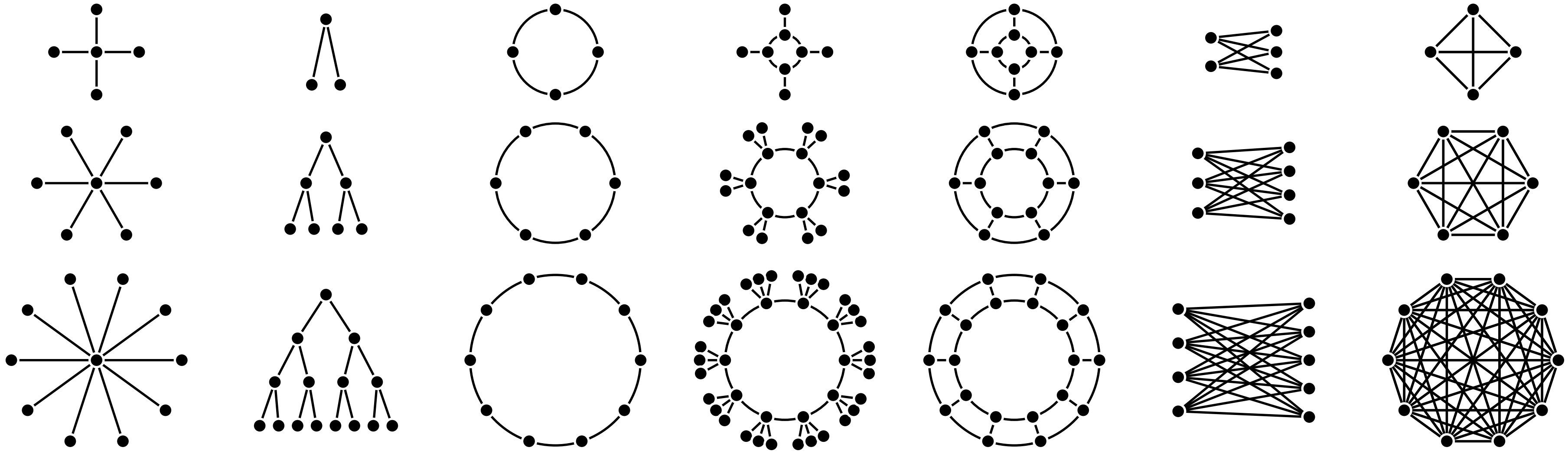
What formalism to describe architectures ?



Azure Datacenter Topology

Greenberg et al., SIGCOMM 2009

What formalism to describe architectures ?



Hyperedge **R**eplacement (sparse only)

e.g., us, VDS 2025 + CAV 2025

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

Contribution

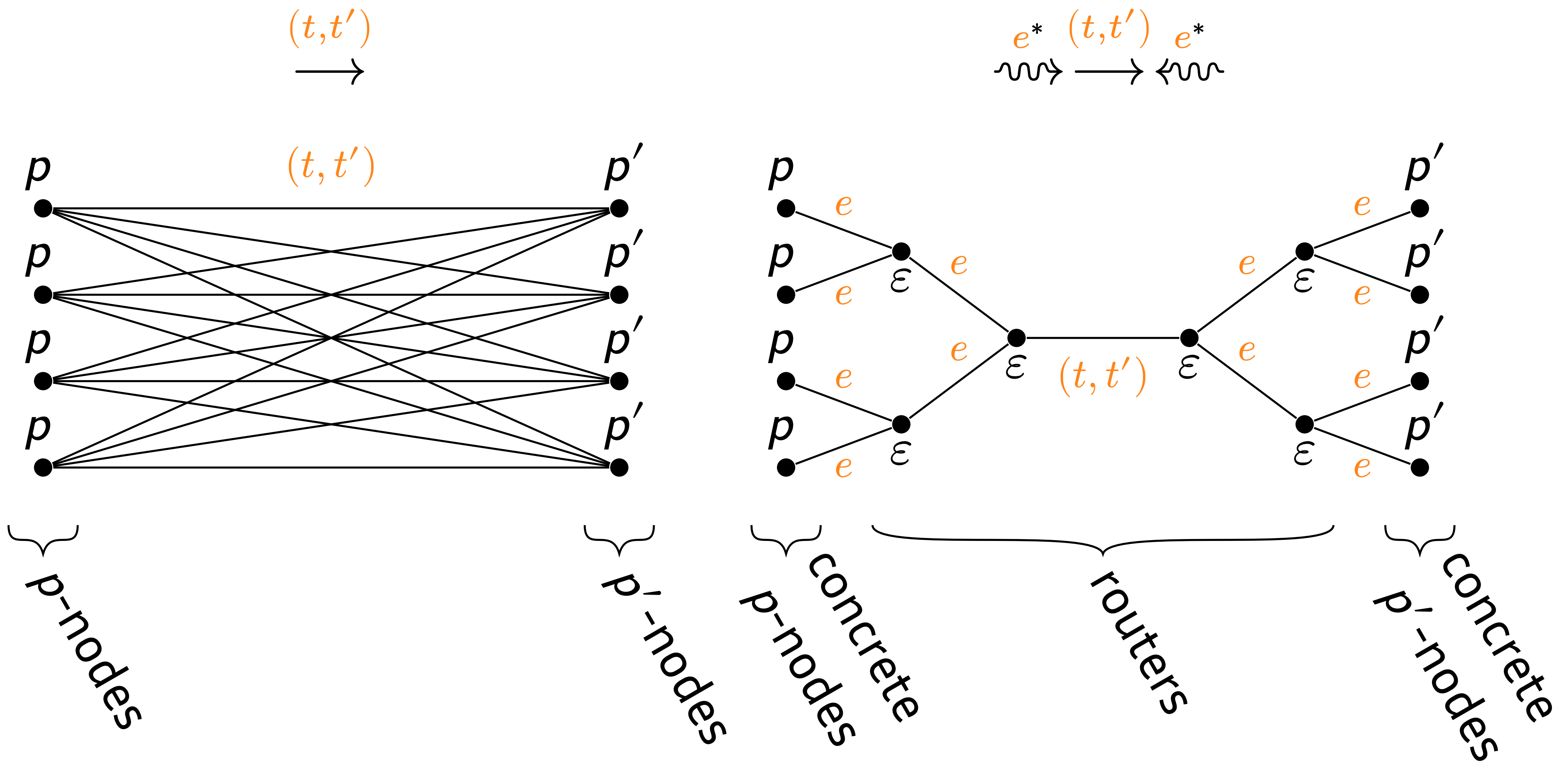
A **translation** from VR to HR architectures.

Contribution

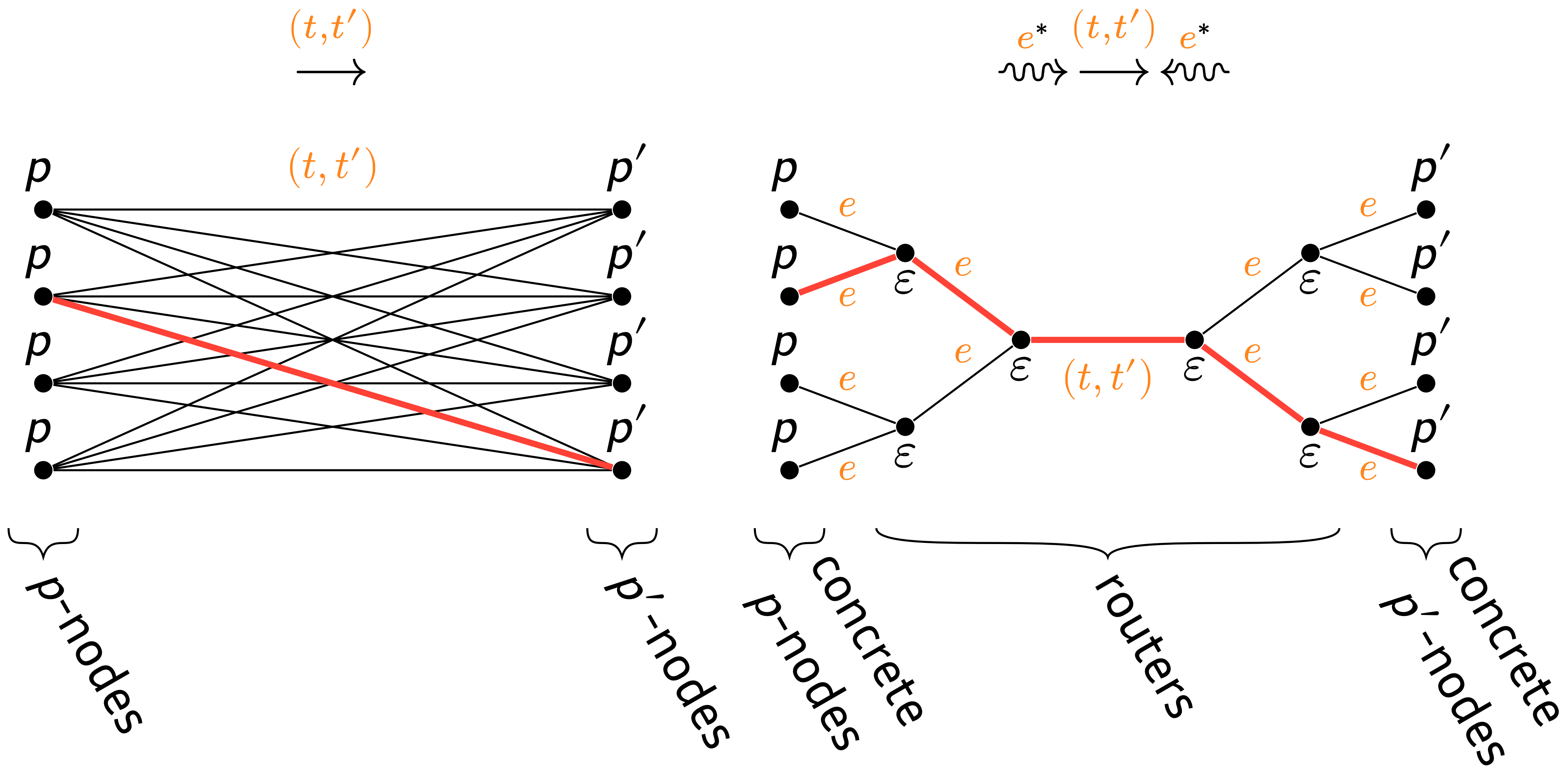
Adapting a translation from VR to HR architectures **to the case of systems**, and studying which safety properties are preserved.

B. Courcelle, Structural Properties of Context-Free Sets of Graphs Generated by Vertex Replacement, *Information and Computation*, 1995

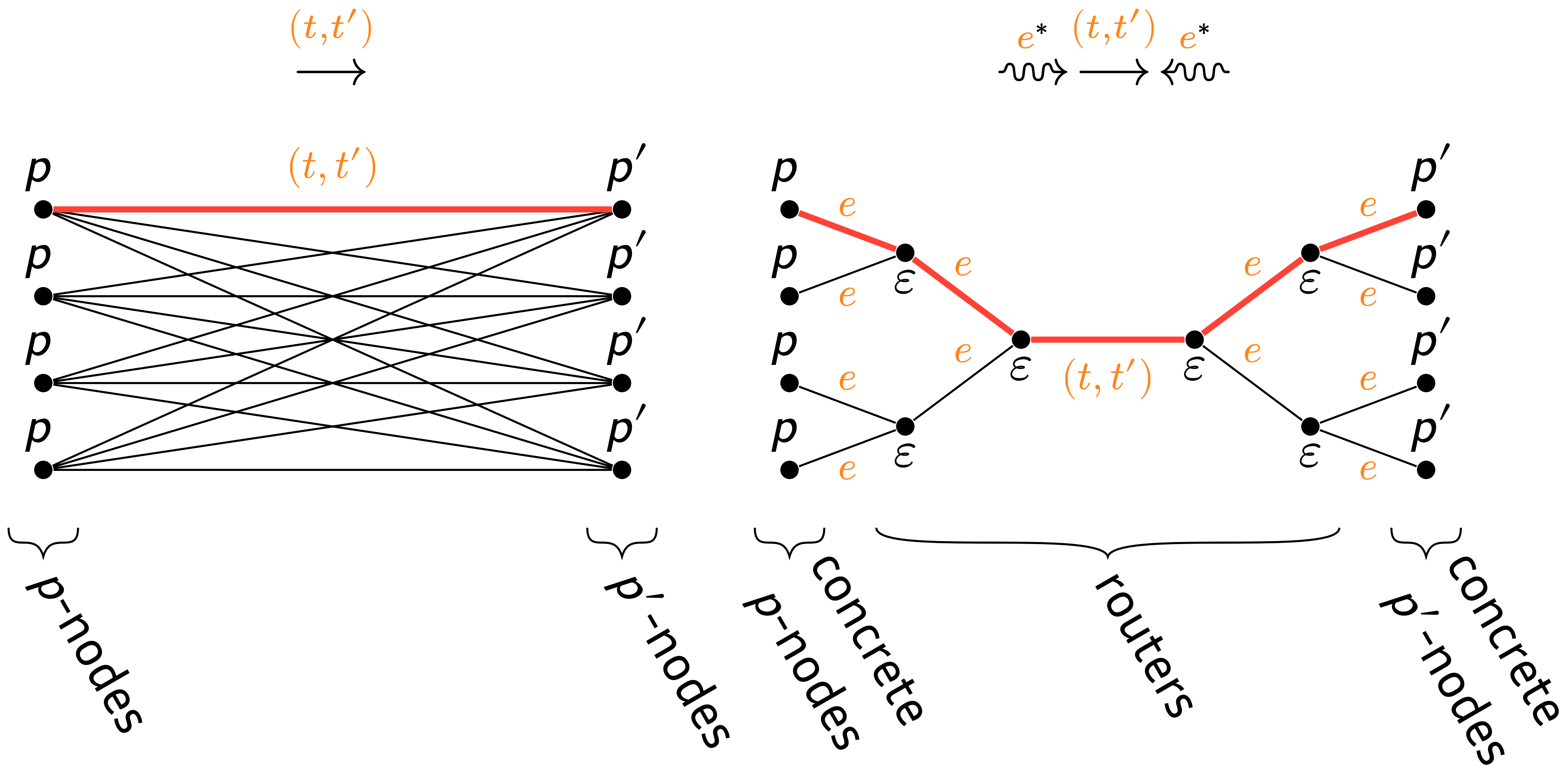
Key idea


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

Key idea

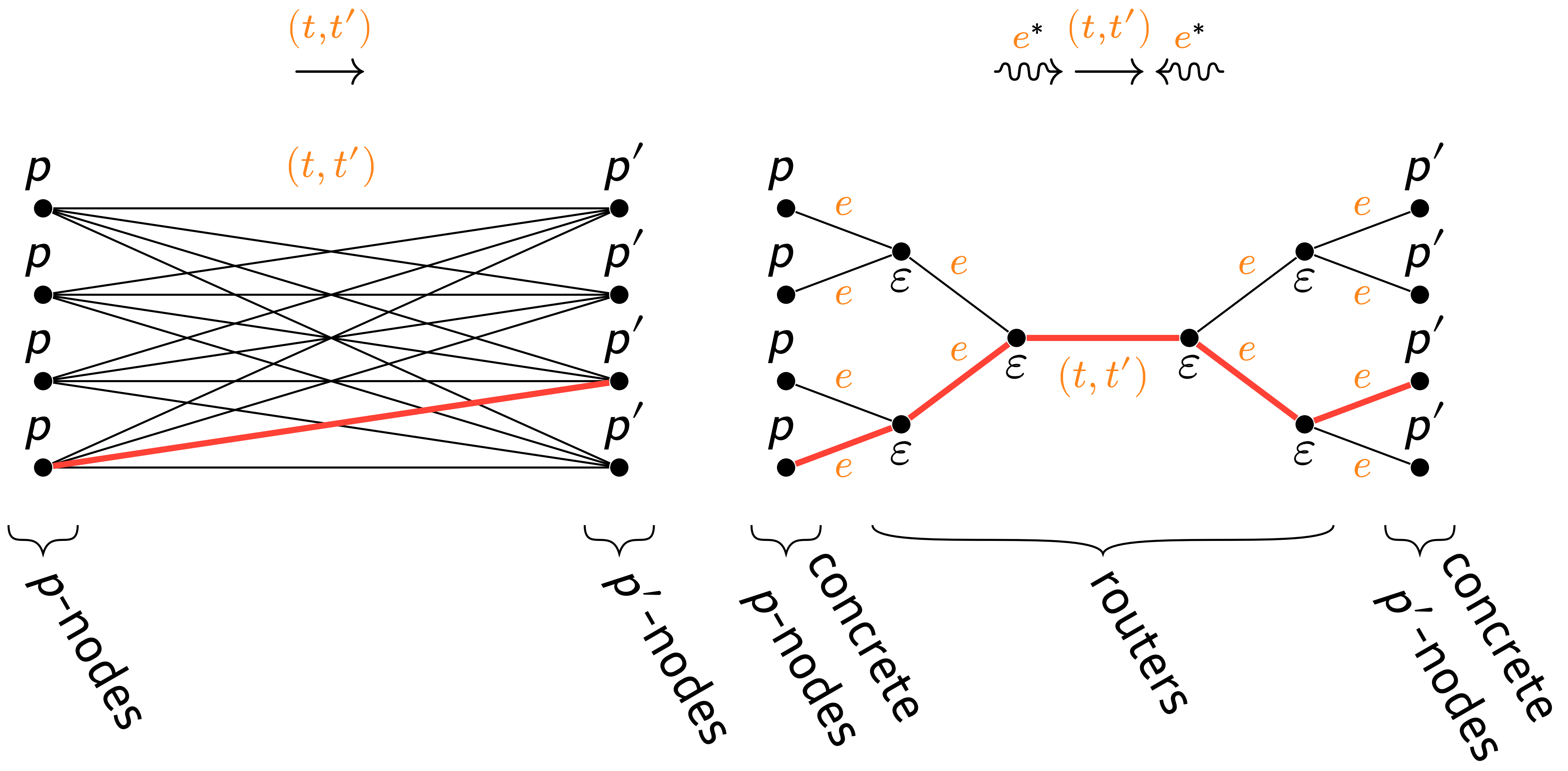

$\xrightarrow{(t,t')}$
$\overset{e^*}{\rightsquigarrow} \xrightarrow{(t,t')} \overset{e^*}{\leftsquigarrow}$
p
p'
(t, t')
e
ε
p-nodes
p'-nodes
concrete p-nodes
routers
concrete p'-nodes

Key idea



(t,t')
p
p'
(t,t')
p-nodes
p'-nodes
e^*
(t,t')
e^*
e
ε
(t,t')
concrete p-nodes
routers
concrete p'-nodes

Key idea

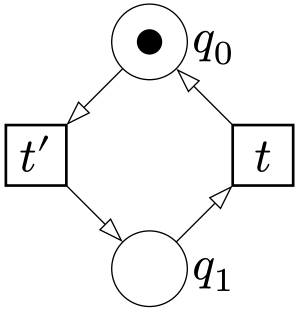

$\xrightarrow{(t,t')}$
(t,t')
p
p'
p-nodes
p'-nodes
e^* (t,t') e^*
p
p'
e
ε
(t,t')
concrete p-nodes
routers
concrete p'-nodes

Encoding of Networks

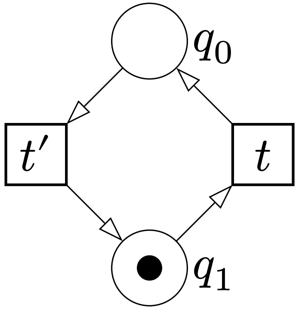
Processes and systems

- process types p_1, p_2, … = Petri nets (PN) with observable transitions

$p_1 =$

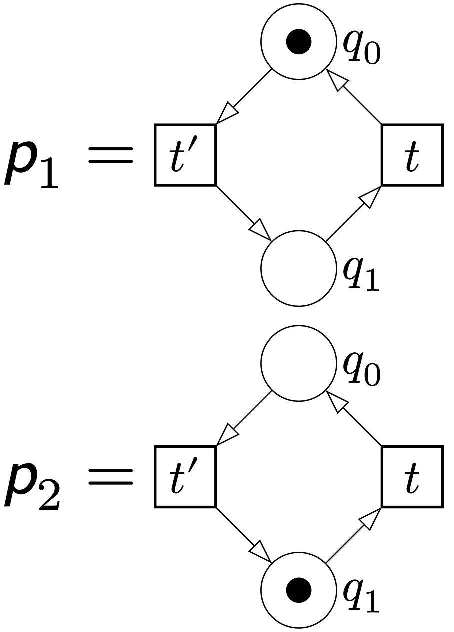


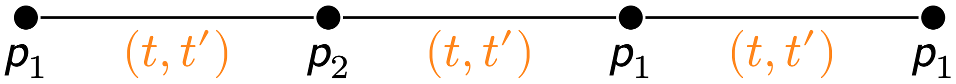
$p_2 =$



Processes and systems

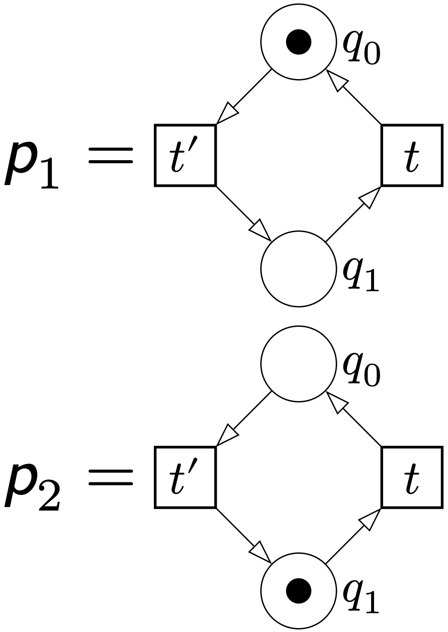
- process types $p_1, p_2, \ldots$ = Petri nets (PN) with observable transitions
- system: graph with
  - vertices labeled by a process type
  - edges labeled by pairs of observable transitions

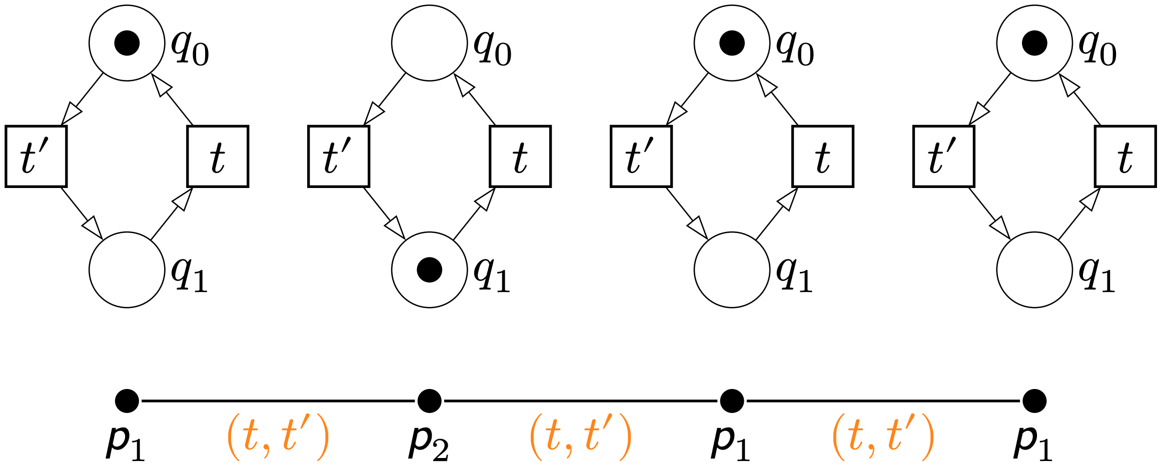




Processes and systems

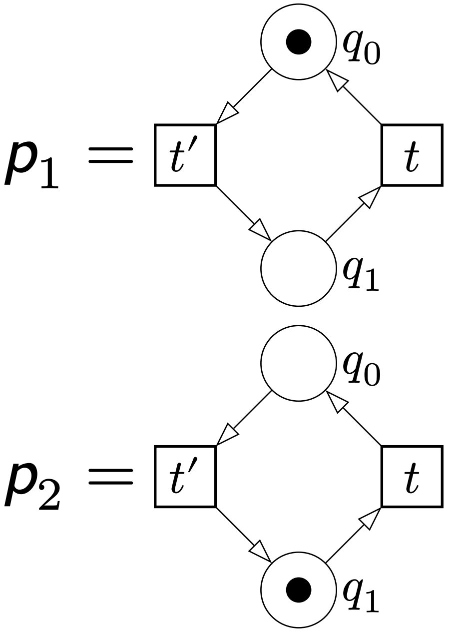
- process types $p_1, p_2, \ldots$ = Petri nets (PN) with observable transitions
- system: graph with
  - vertices labeled by a process type
  - edges labeled by pairs of observable transitions

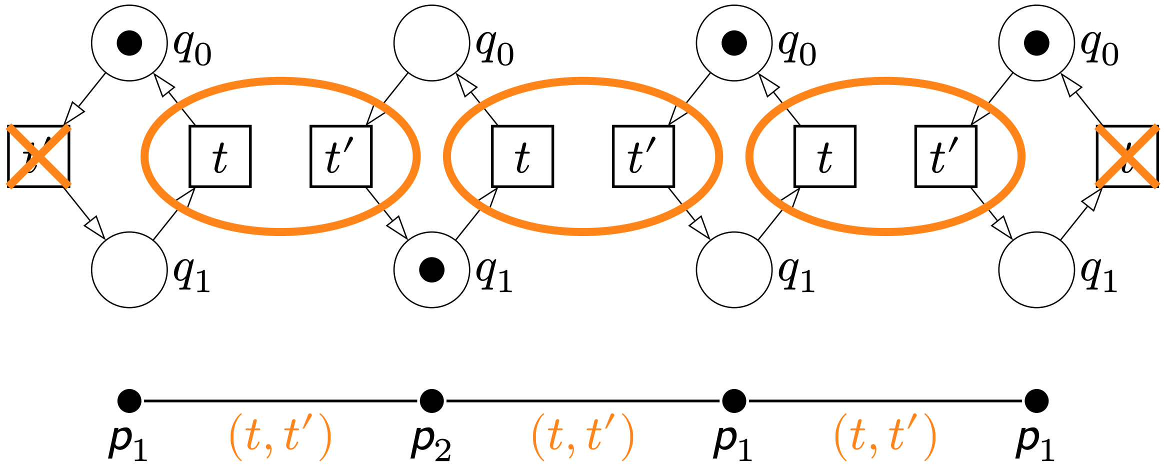




Processes and systems

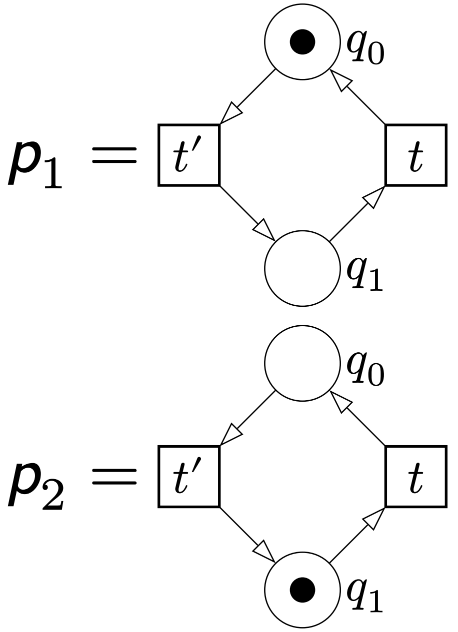
- process types p_1, p_2, ... = Petri nets (PN) with observable transitions
- system: graph with
  - vertices labeled by a process type
  - edges labeled by pairs of observable transitions

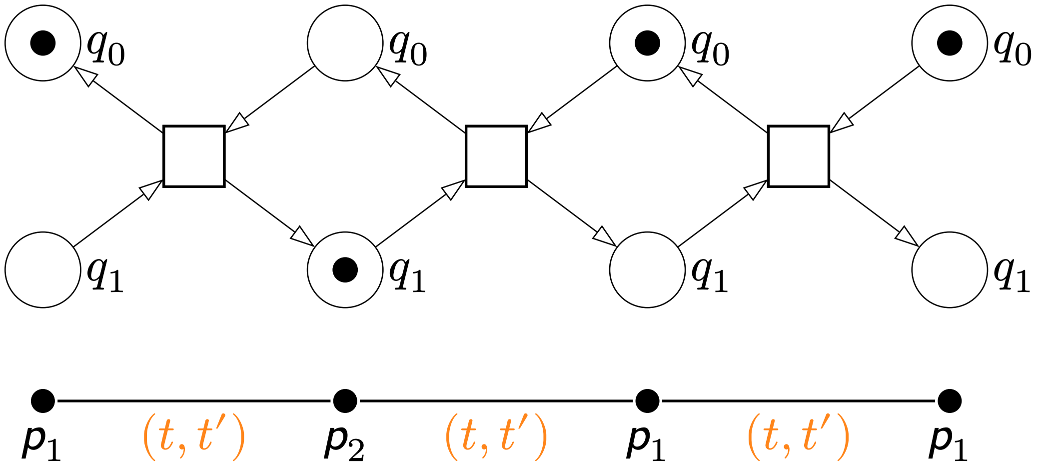




Processes and systems

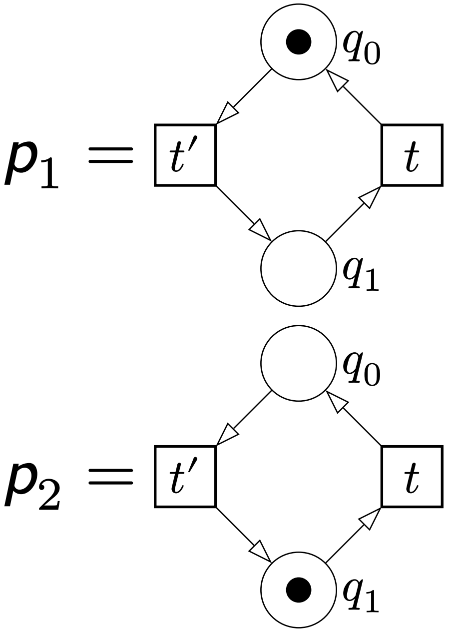
- process types $p_1, p_2, \ldots$ = Petri nets (PN) with observable transitions
- system: graph with
  - vertices labeled by a process type
  - edges labeled by pairs of observable transitions

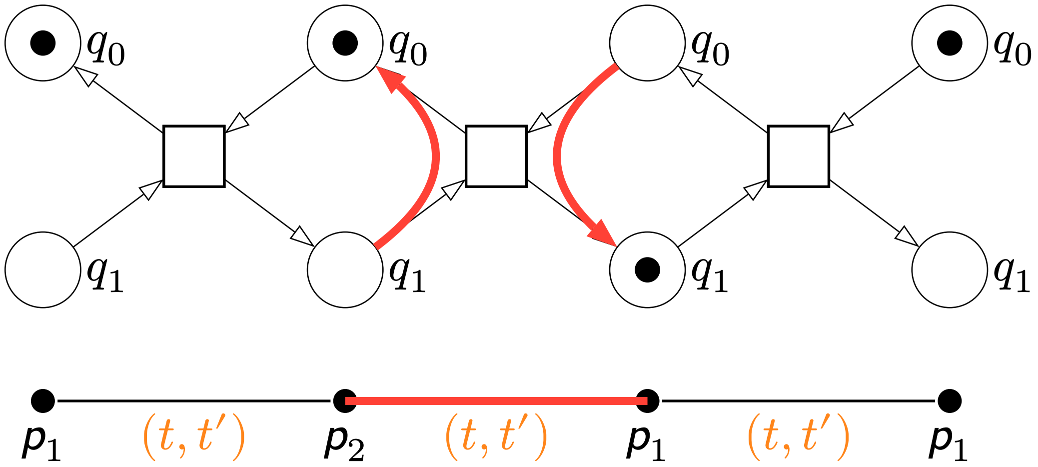




Processes and systems

- process types $p_1, p_2, \ldots$ = Petri nets (PN) with observable transitions
- system: graph with
  - vertices labeled by a process type
  - edges labeled by pairs of observable transitions





Properties

$\#q^{p}$ total number of tokens in place q of processes of type p.

α any arithmetic formula on $\{\#q^{p} \mid q, p\}$

φ reachability/coverability/sequence/etc on α

$\rightarrow$ interpreted over the firing sequences of the PN

HR & VR

Architecture

Given by a term in HR/VR.

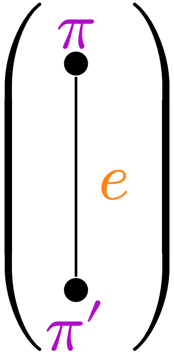
Context-free grammars generate infinite families of terms, thus infinite families of architectures.

HR: Single edge

$$\vec{e}_{\pi,\pi'} = \left(\begin{tikzcd} \overset{\pi}{\bullet} \arrow[d, dash, "e"] \\ \underset{\pi'}{\bullet} \end{tikzcd} \right)$$

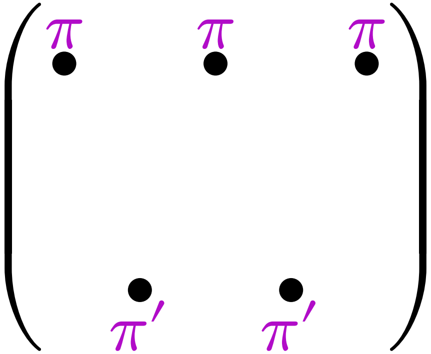
HR: Single edge

$$\vec{e}_{\pi,\pi'} =$$

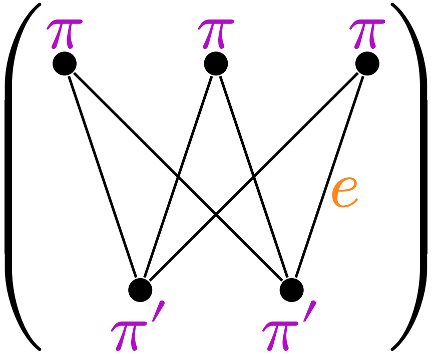


VR: All-pairs edges

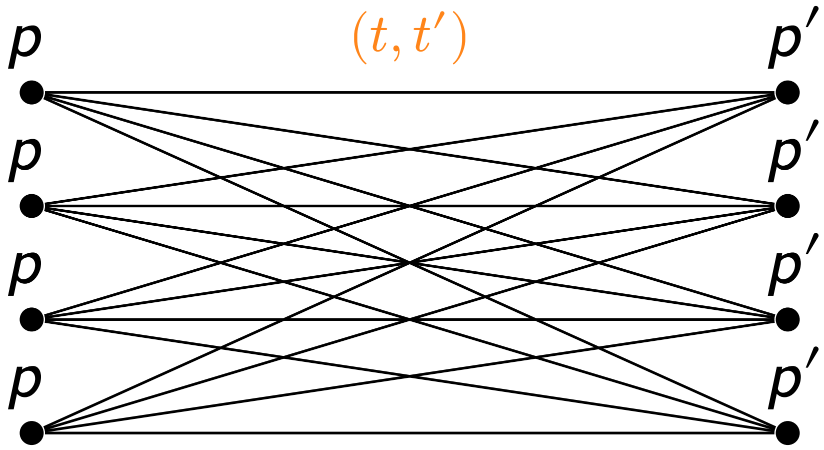
$$\mathrm{add}^{e}_{\pi,\pi'}$$



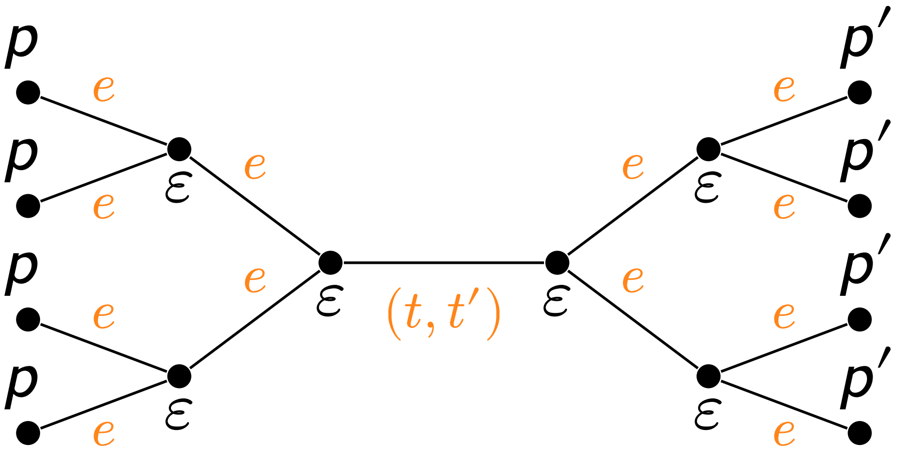
$$=$$



VR

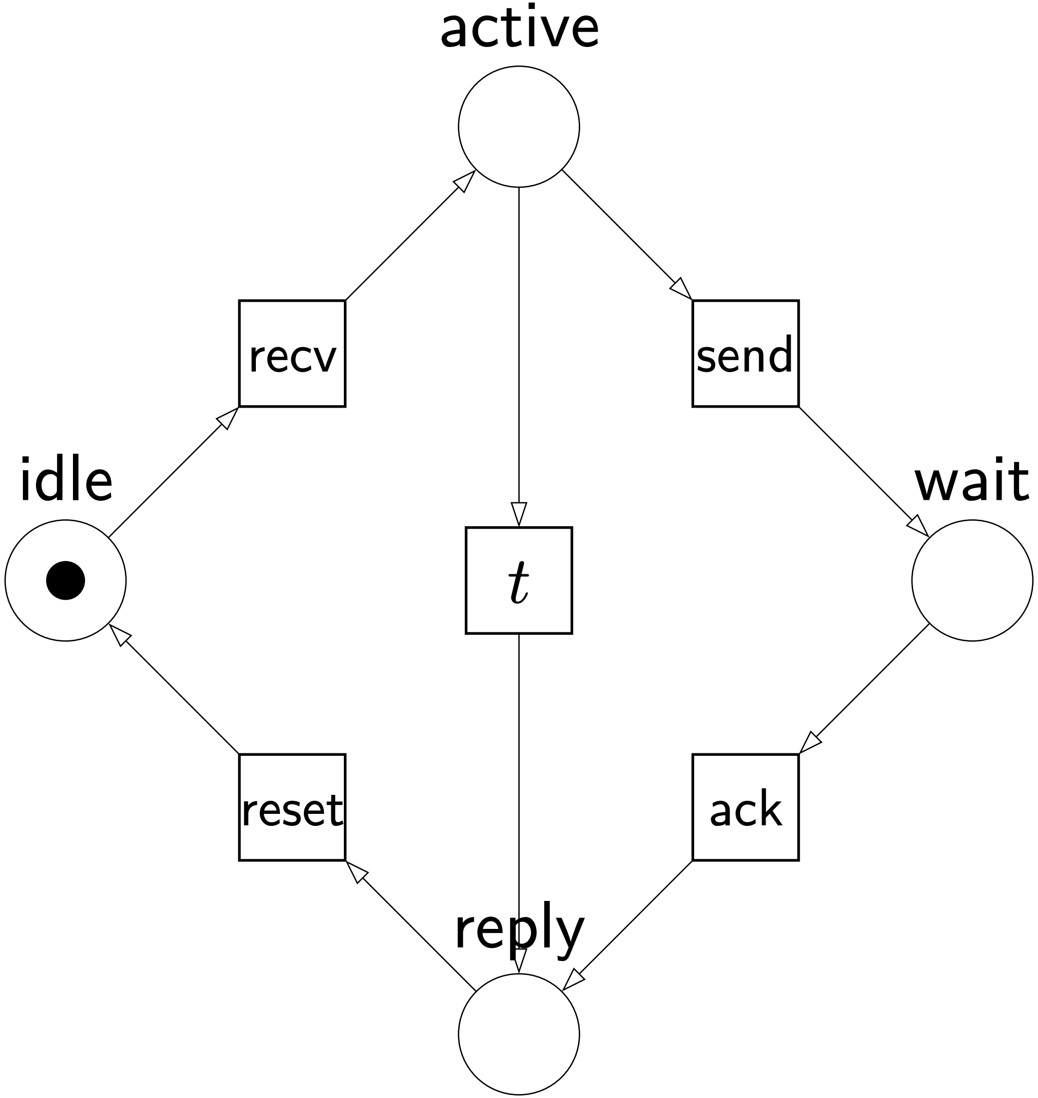


HR

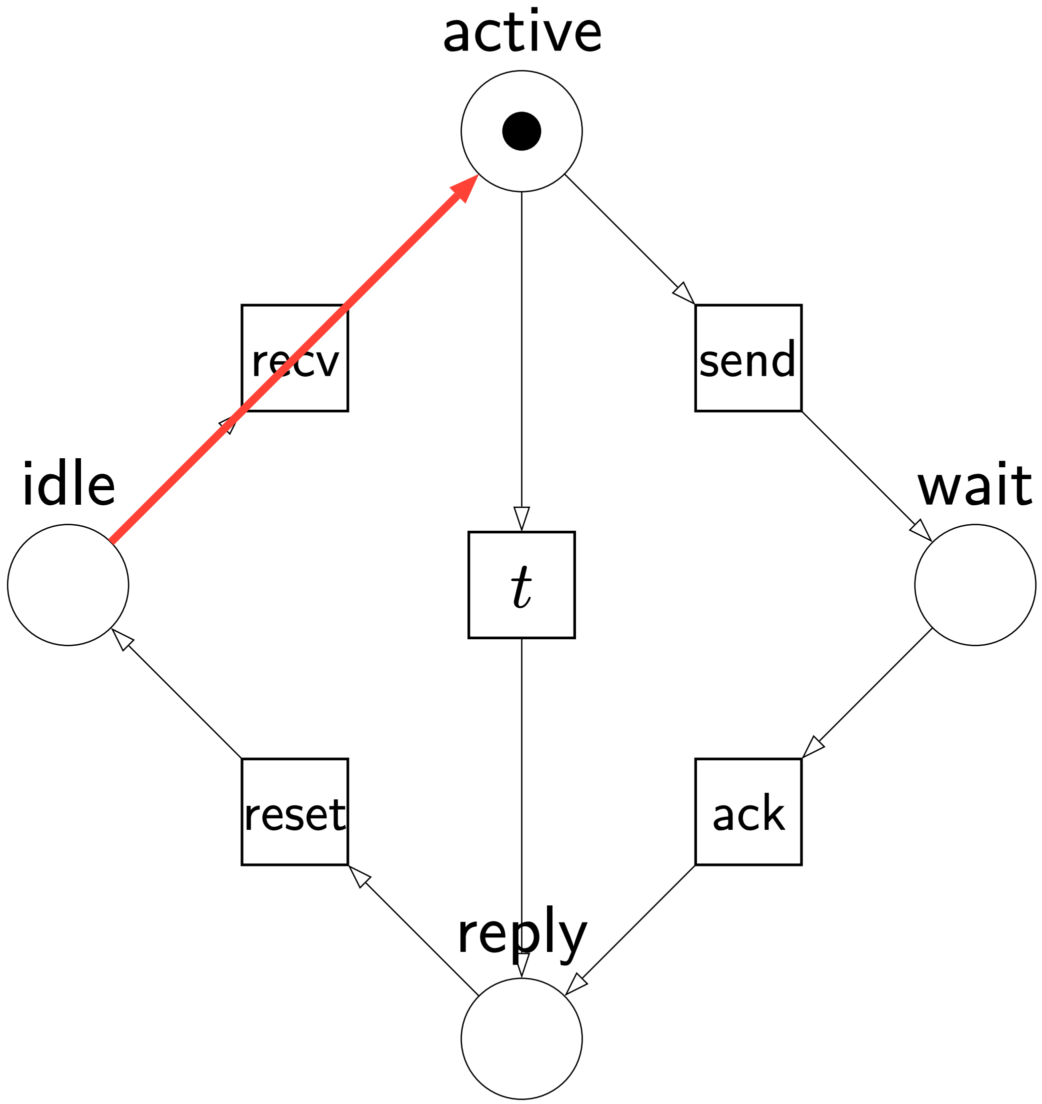


Routing

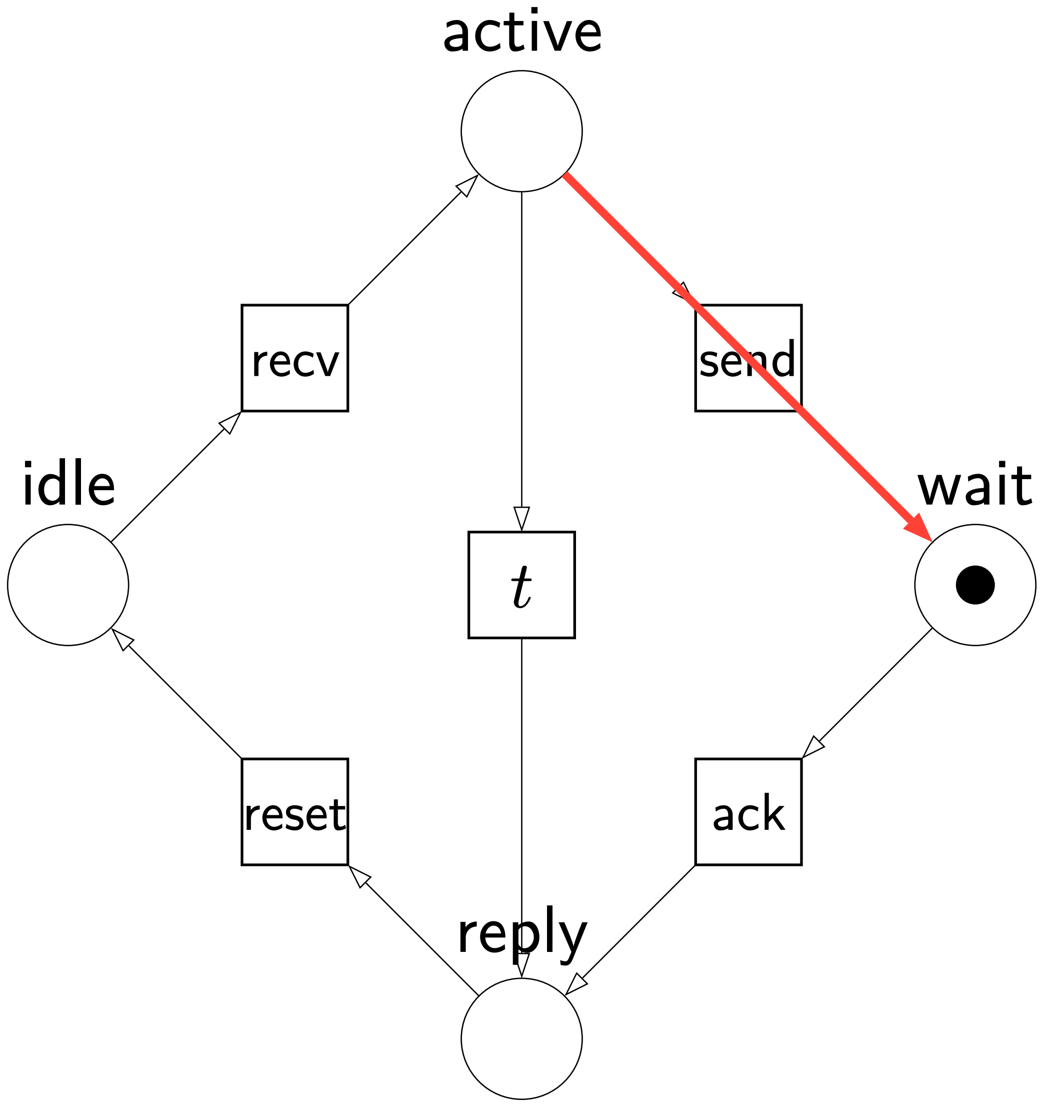
Construction of a router



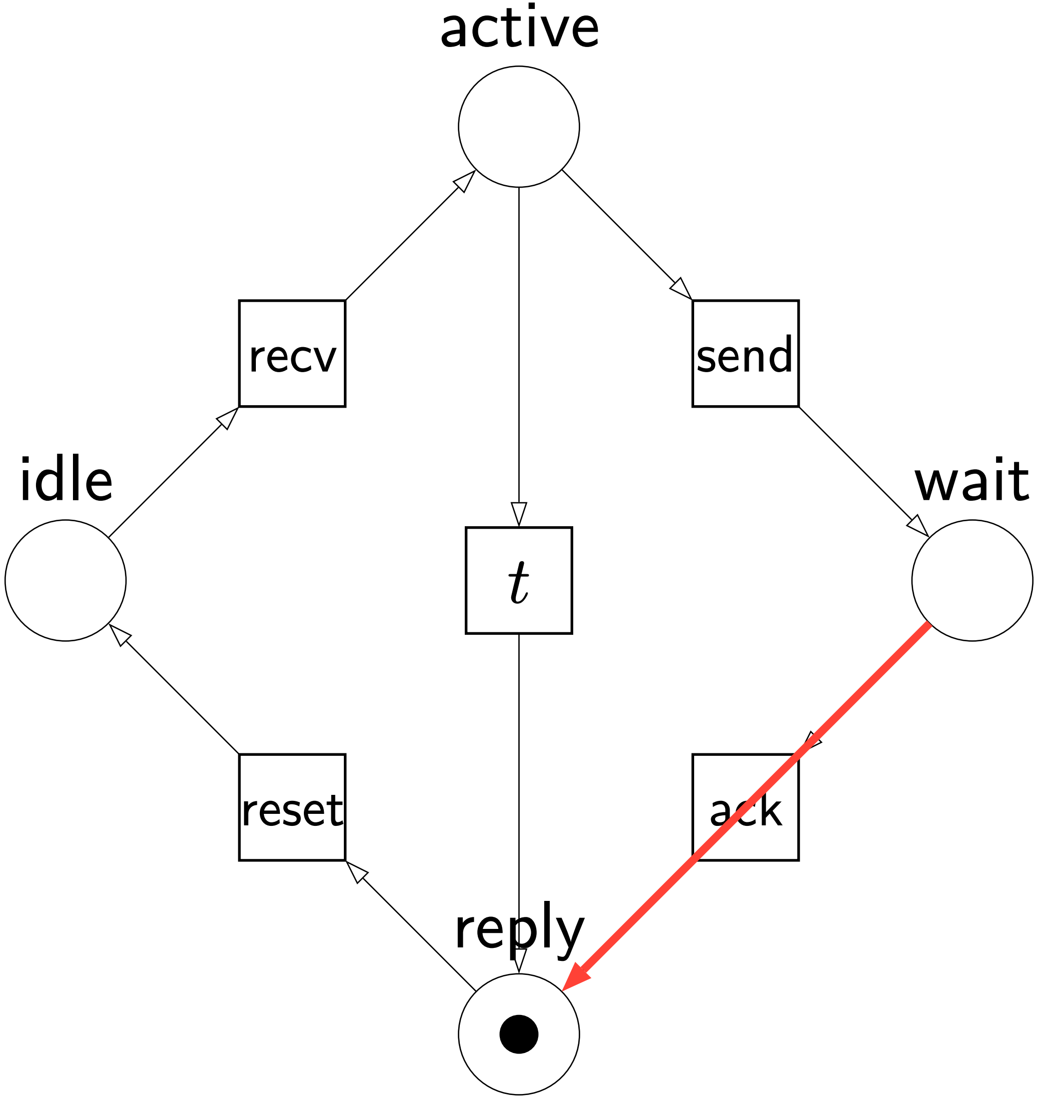
Construction of a router



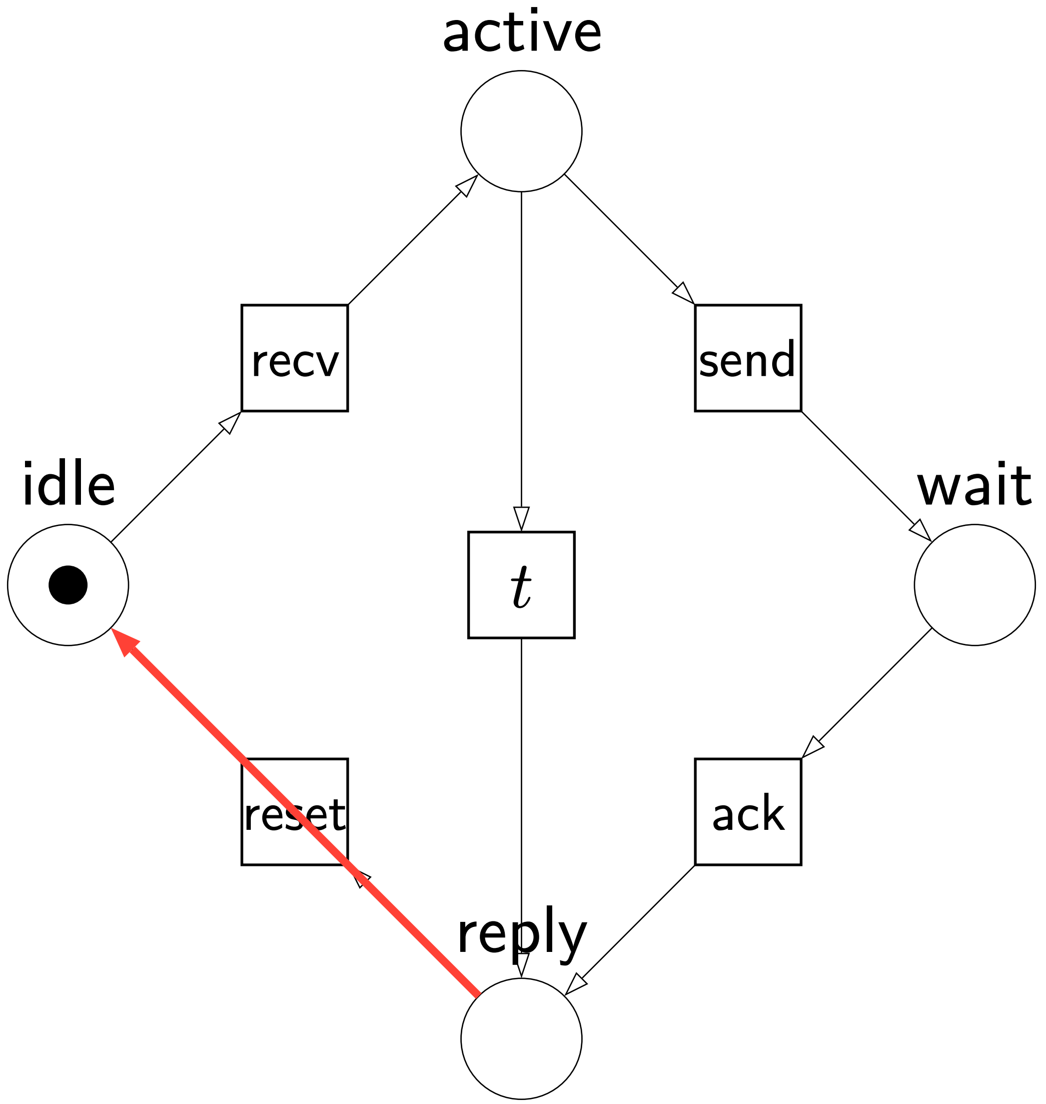
Construction of a router



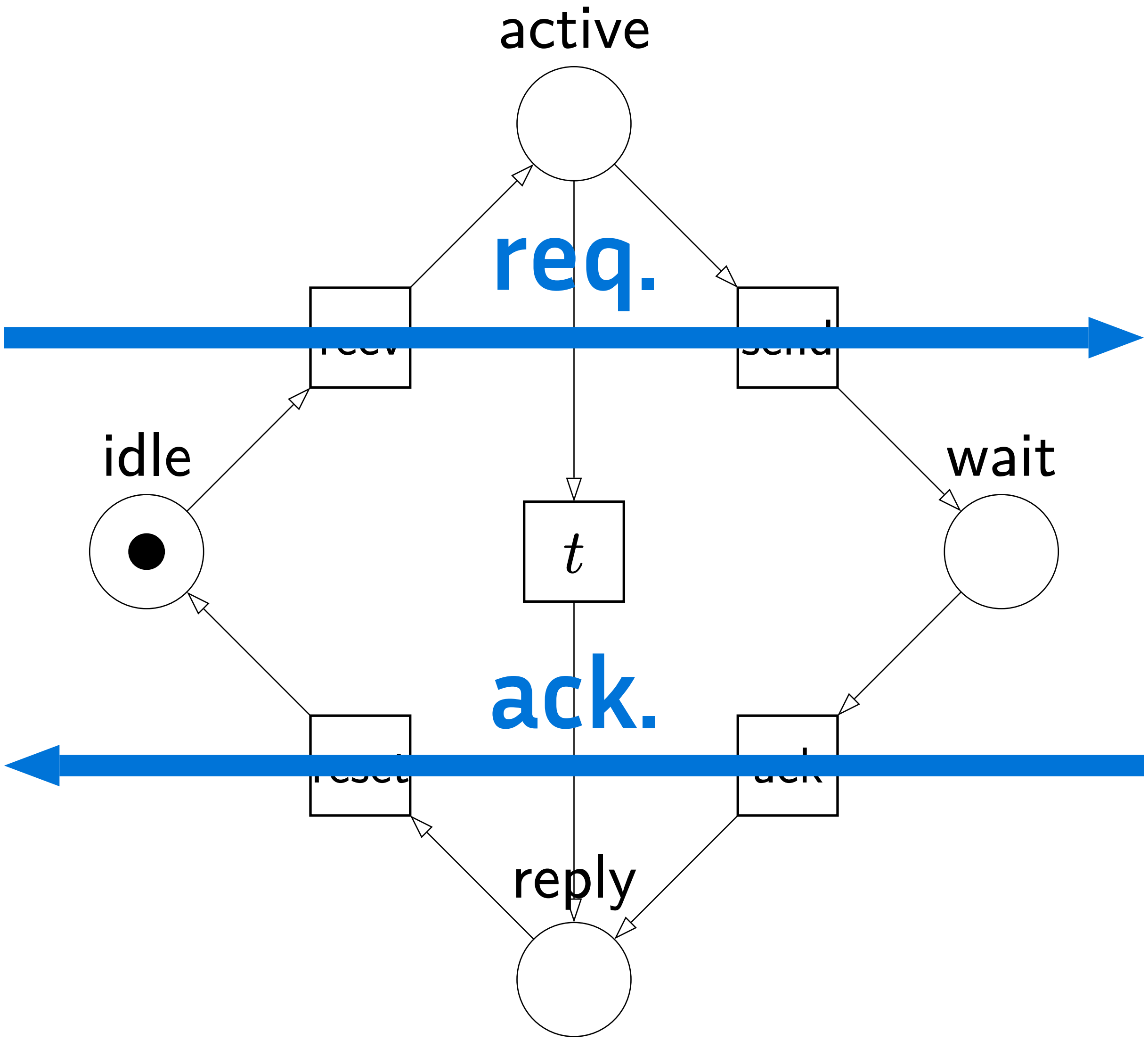
Construction of a router



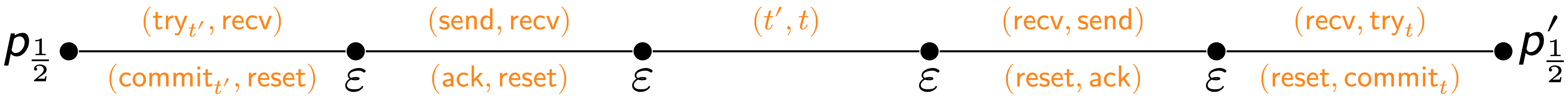
Construction of a router

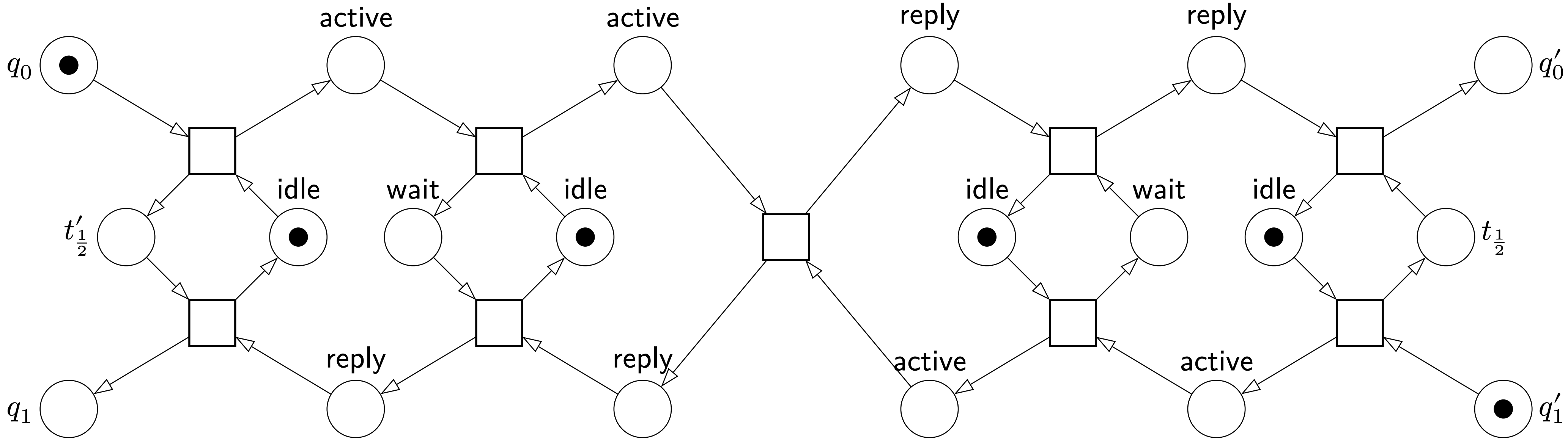


Construction of a router

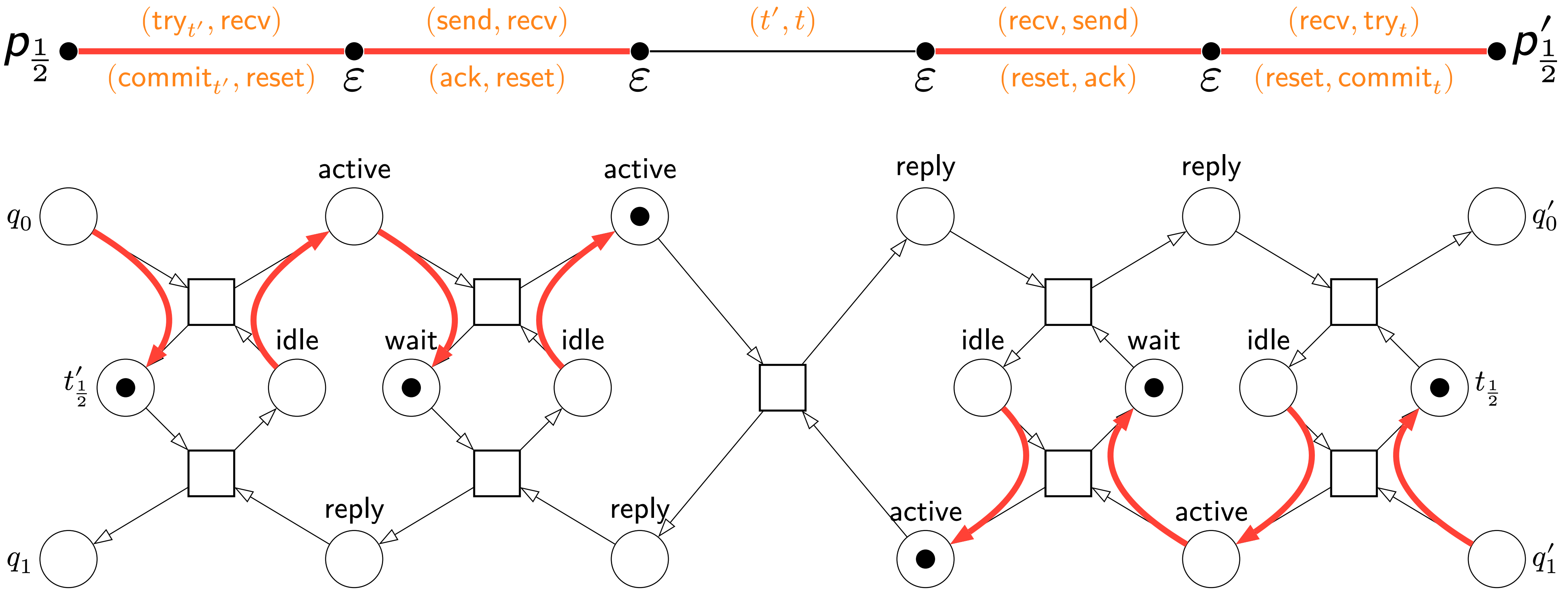


Communication through routers

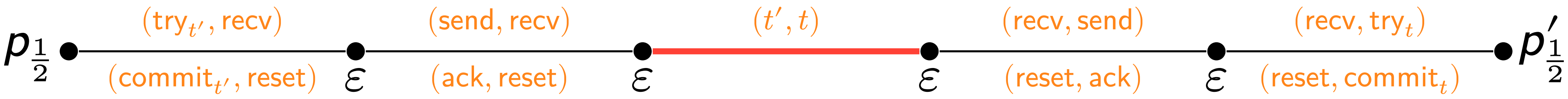


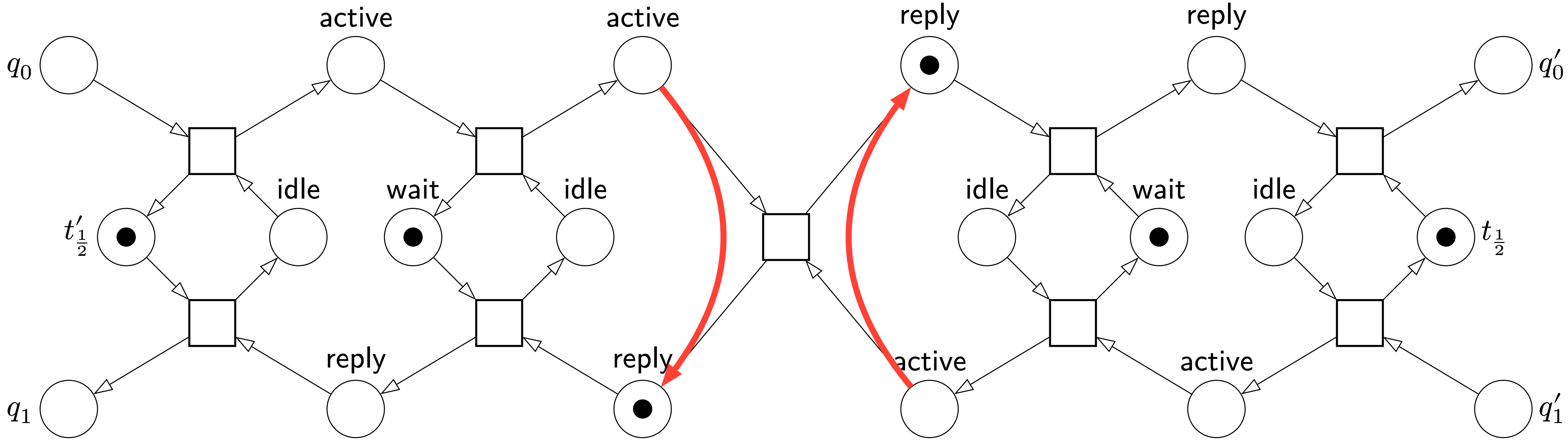


Communication through routers

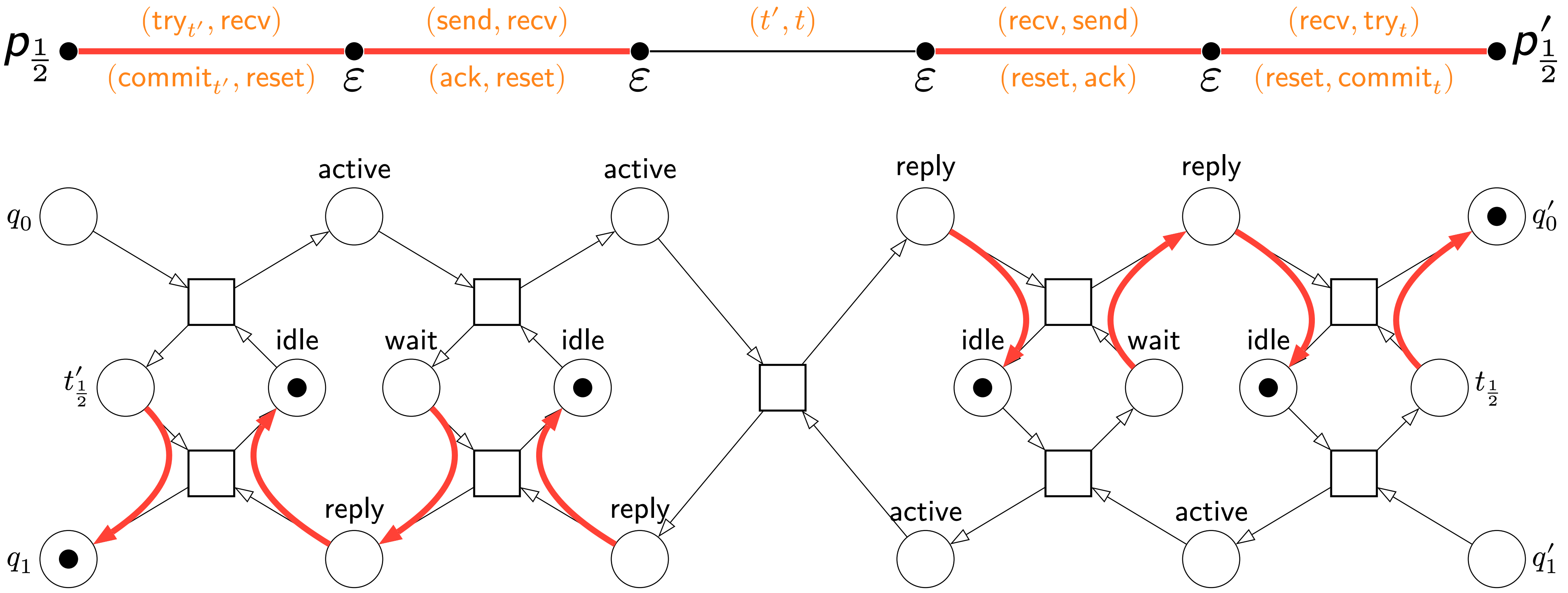


Communication through routers

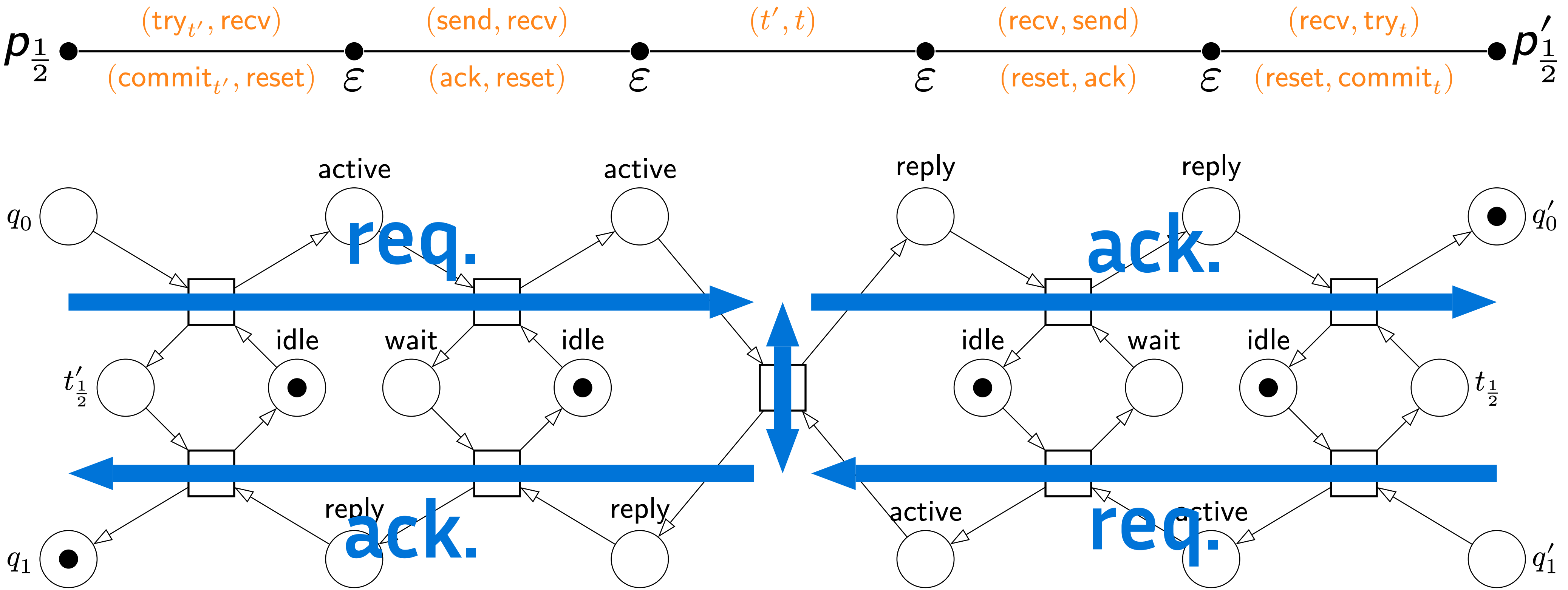




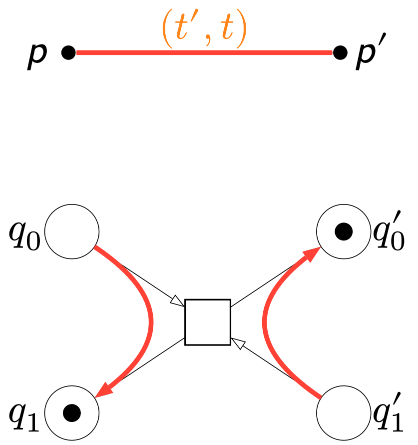
Communication through routers



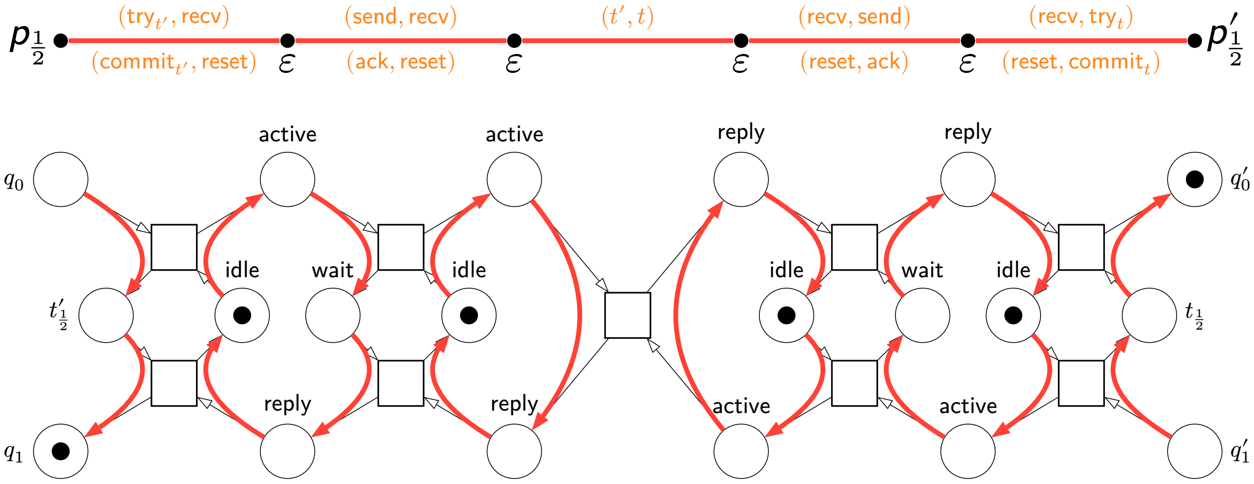
Communication through routers



Stuttering



$$s_1 s_2$$



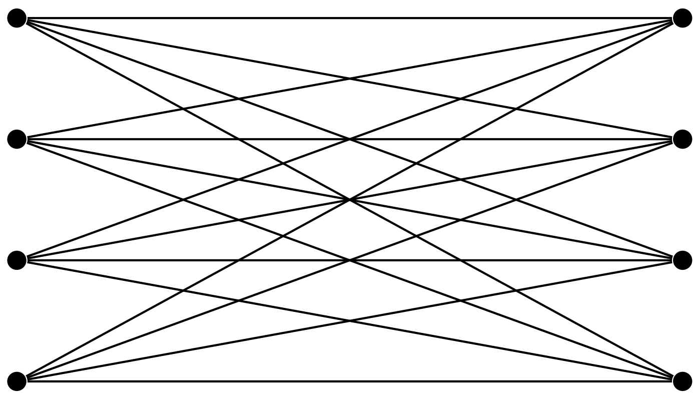
$$s_1 s_1 s_1 s_1 s_1 s_2 s_2 s_2 s_2 s_2$$

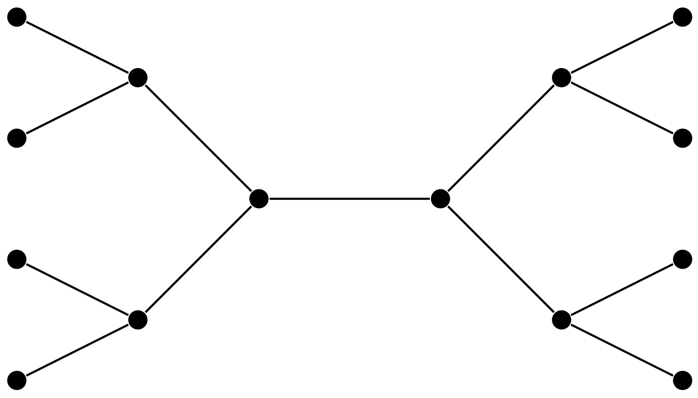
Simulation

- Stuttering-invariant properties are preserved
  - (un)reachability, (un)coverability
  - mutual exclusion
  - reachability in a specific order
- Linear transformation
  - $|T| \cdot \mathrm{cw} \cdot \Theta(n)$ router nodes
  - sparse graph

Downsides

- Lose properties sensitive to stuttering
  - ~~next-step~~ ($s_1 s_2$ vs $s_1 s_1 s_1 s_1 s_1 s_2$)
  - ~~deadlock~~ ($s_1 \bot$ vs $s_1 s_1 s_1 s_1 \bot$)
  - related: LTL $\setminus$ X
- Increased trace length
  - $\times\ \Theta(n)$ worst-case
  - $\times\ \Theta(\lg n)$ average-case
- Loss of parallelism
  - finite throughput





Translation

Invariants

VR

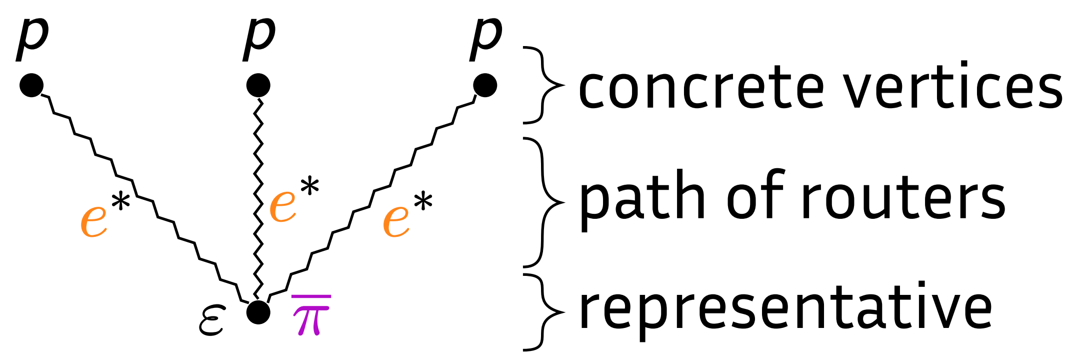
$\xrightarrow{(t,t')}$

p π $\quad$ p π $\quad$ p π

$\xrightarrow{H}$

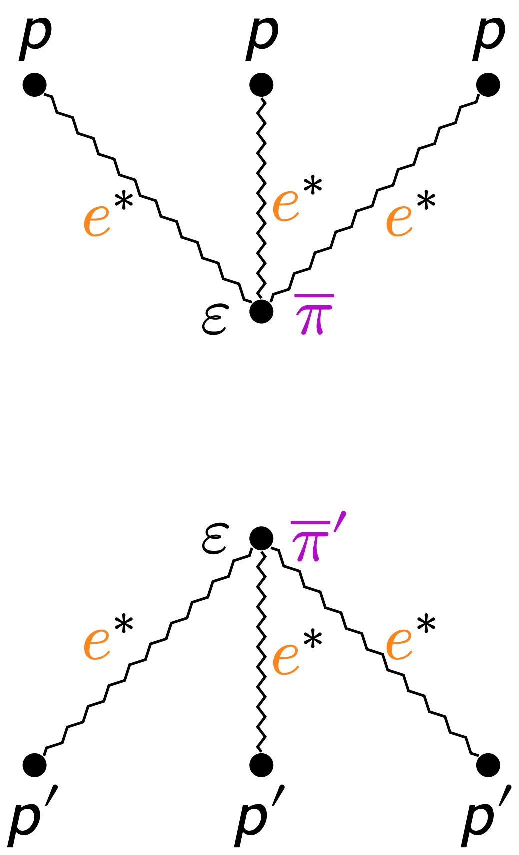
HR

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



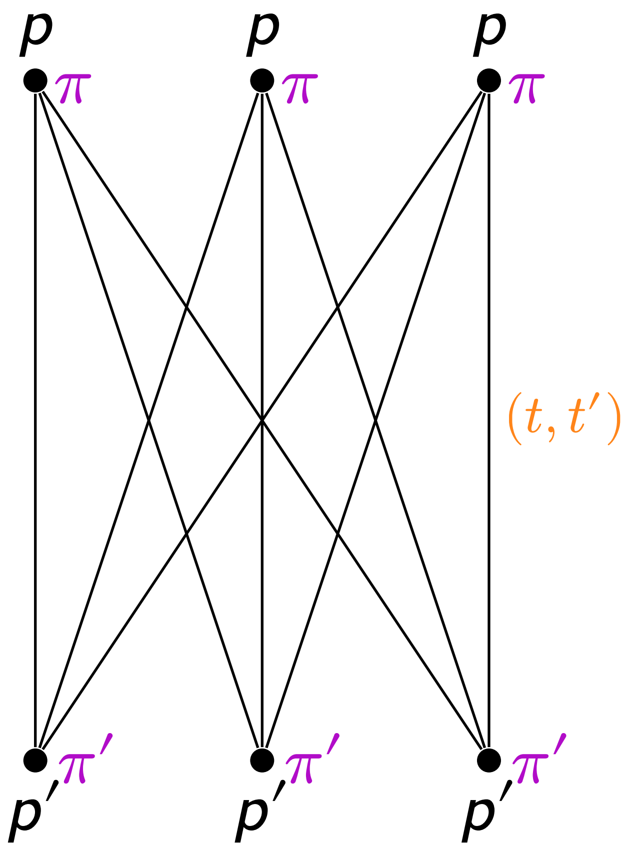
Edge creation

θ

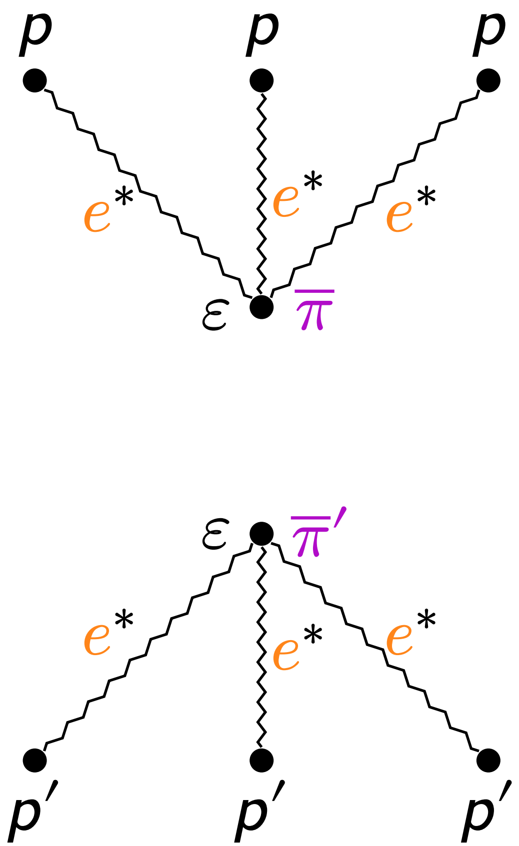


$H(\theta)$

Edge creation

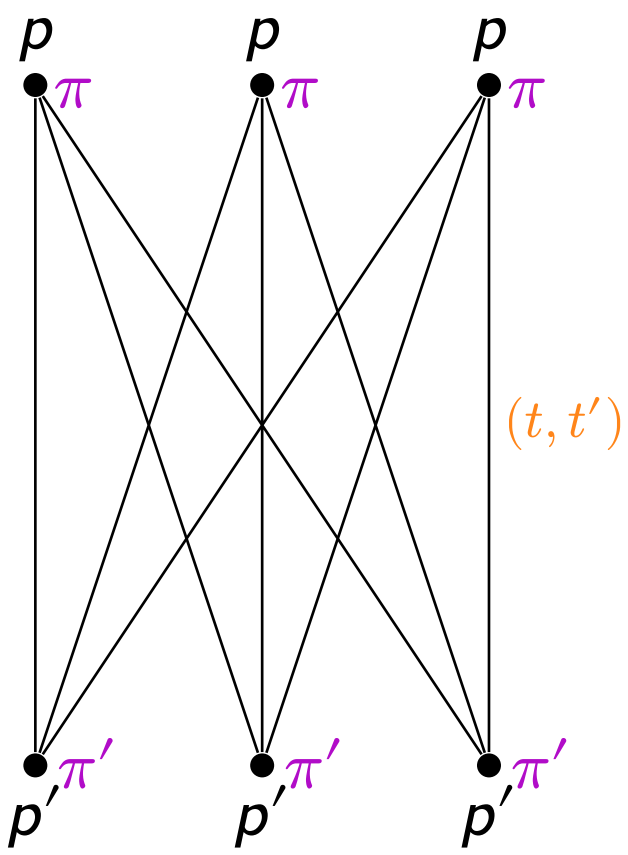


$\mathsf{add}_{\pi,\pi'}^{(t,t')}(\theta)$

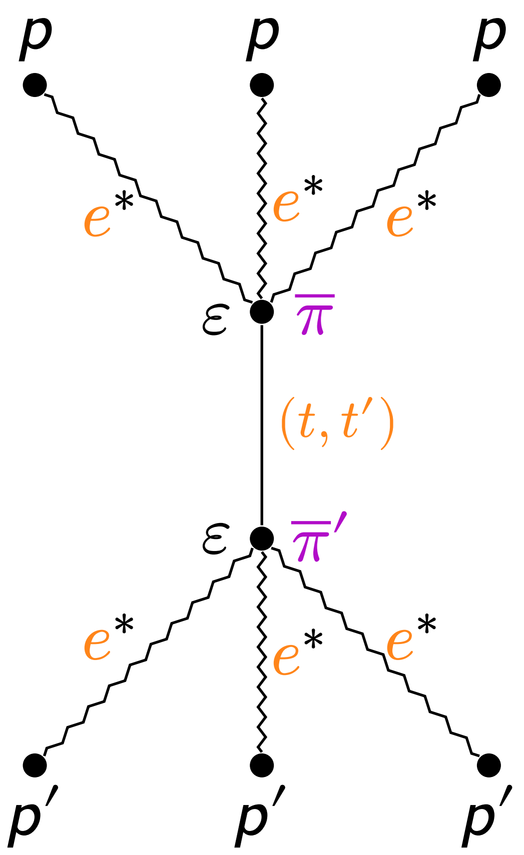


$H(\theta)$

Edge creation



$\mathsf{add}_{\pi,\pi'}^{(t,t')}(\theta)$



$H(\theta) \parallel \overrightarrow{(t, t')}_{\overline{\pi},\overline{\pi}'}$

Conclusion

- Translation of systems from **VR to HR**
- Preserves **stuttering-invariant** properties
- Enables applying semi-algorithms for HR to **dense families**