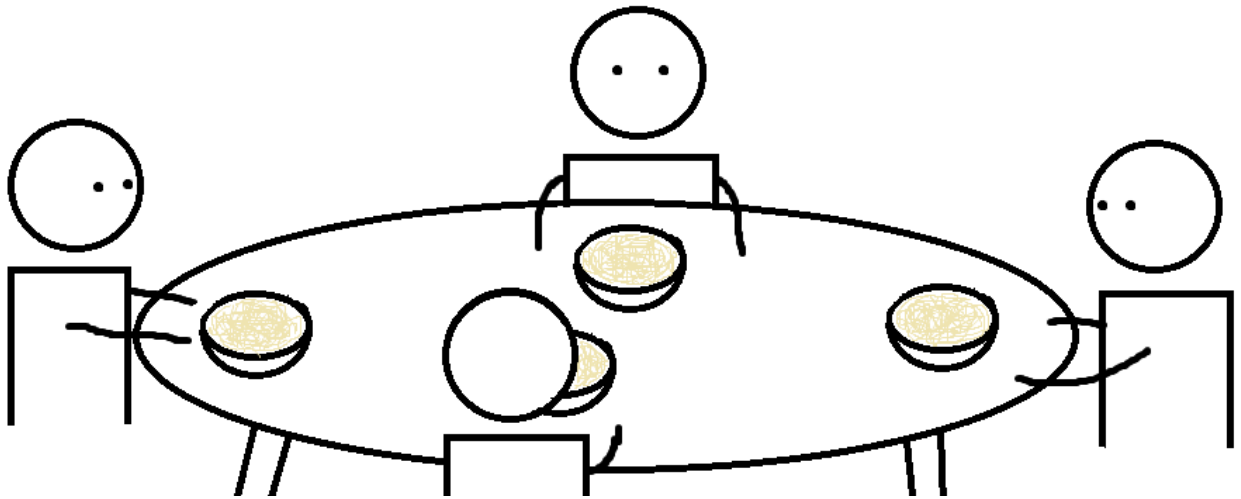
Synthesizing strategies to avoid deadlocks

Corto Mascle

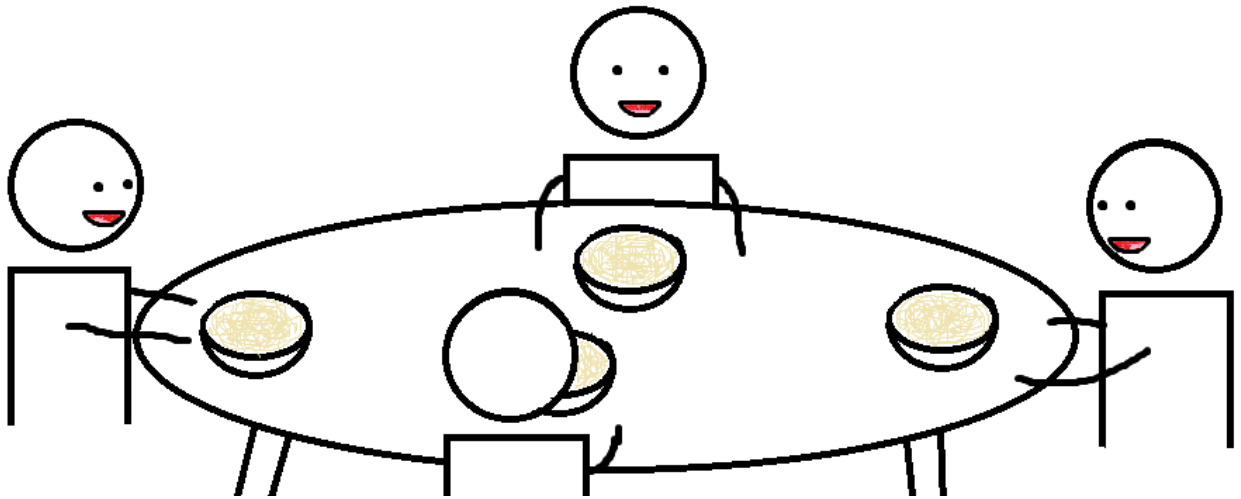
Joint work with Hugo Gimbert, Anca Muscholl and Igor Walukiewicz

RP 2022

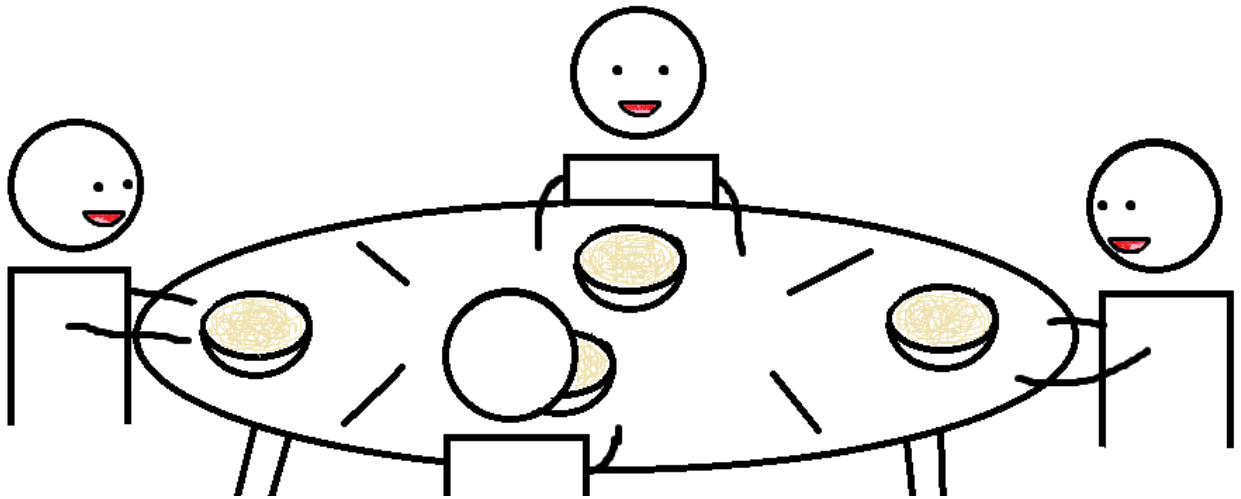
Dining philosophers



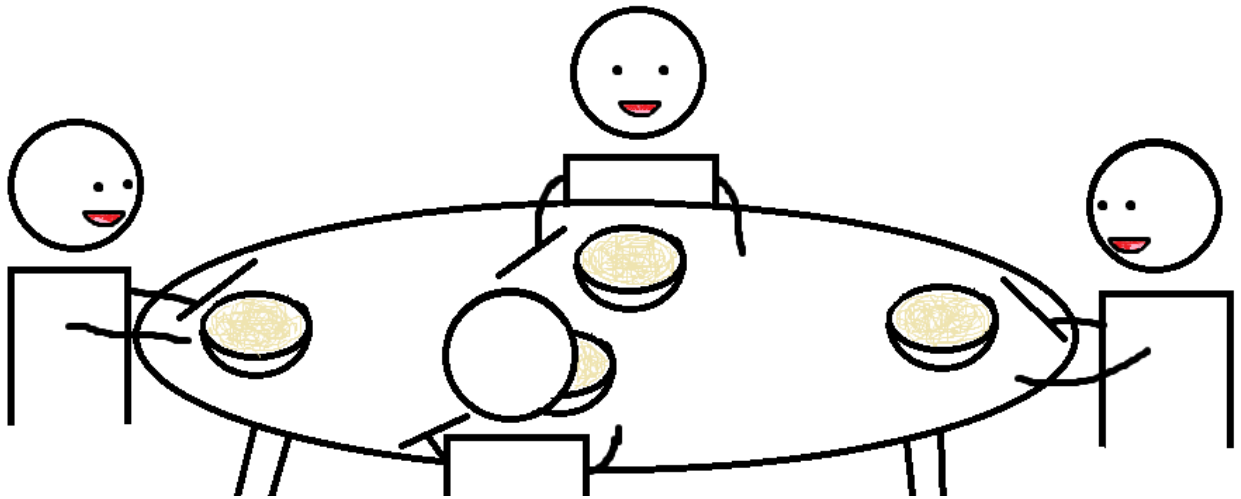
Dining philosophers



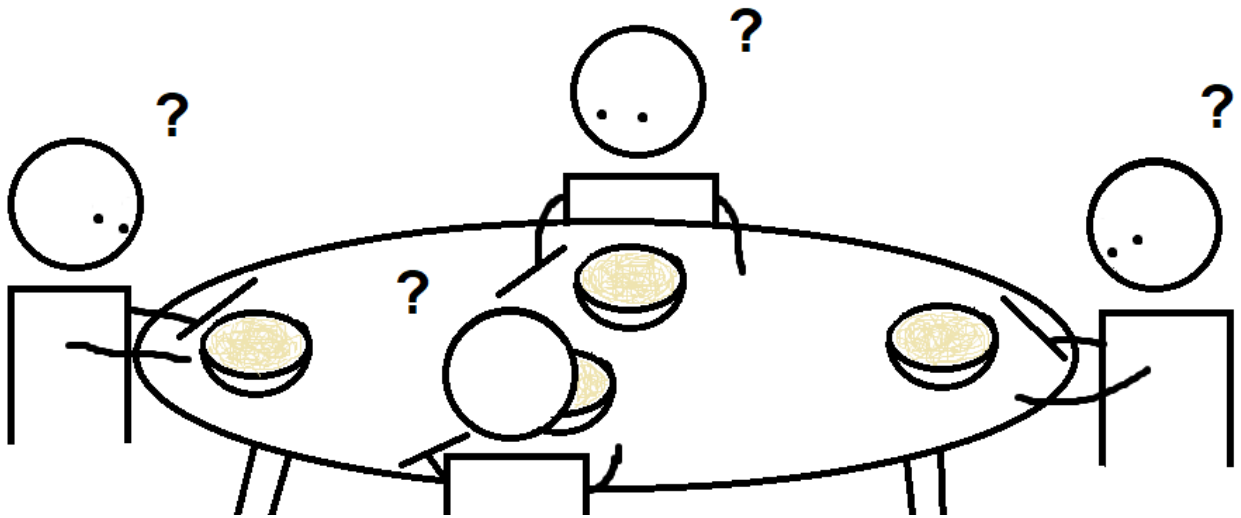
Dining philosophers



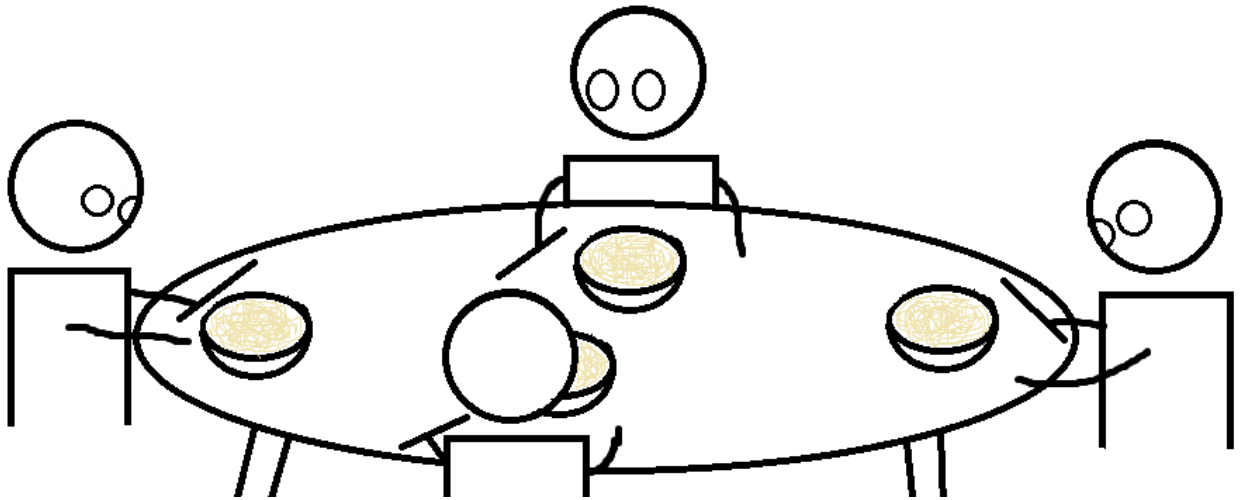
Dining philosophers



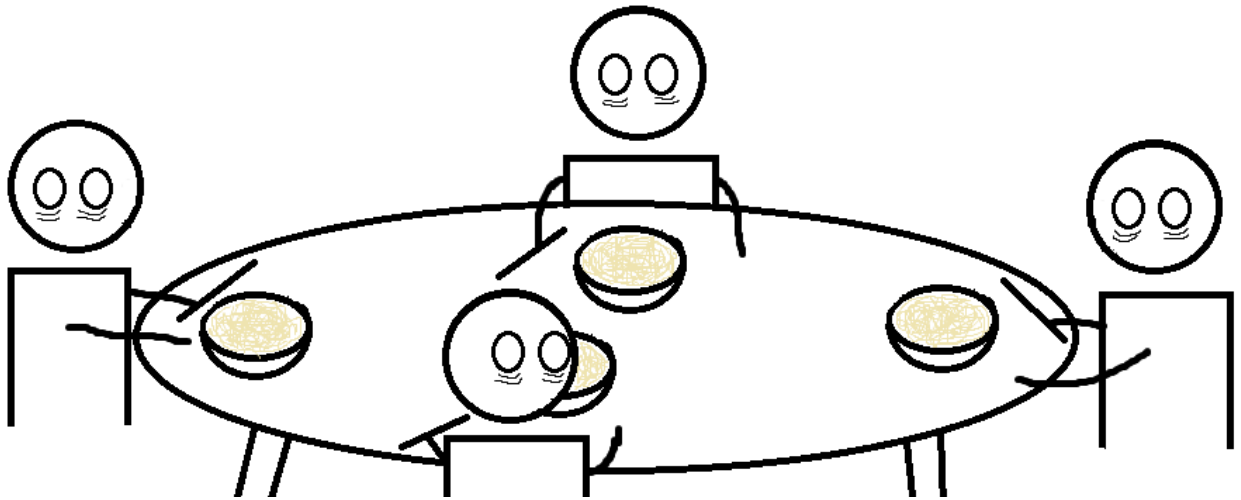
Dining philosophers



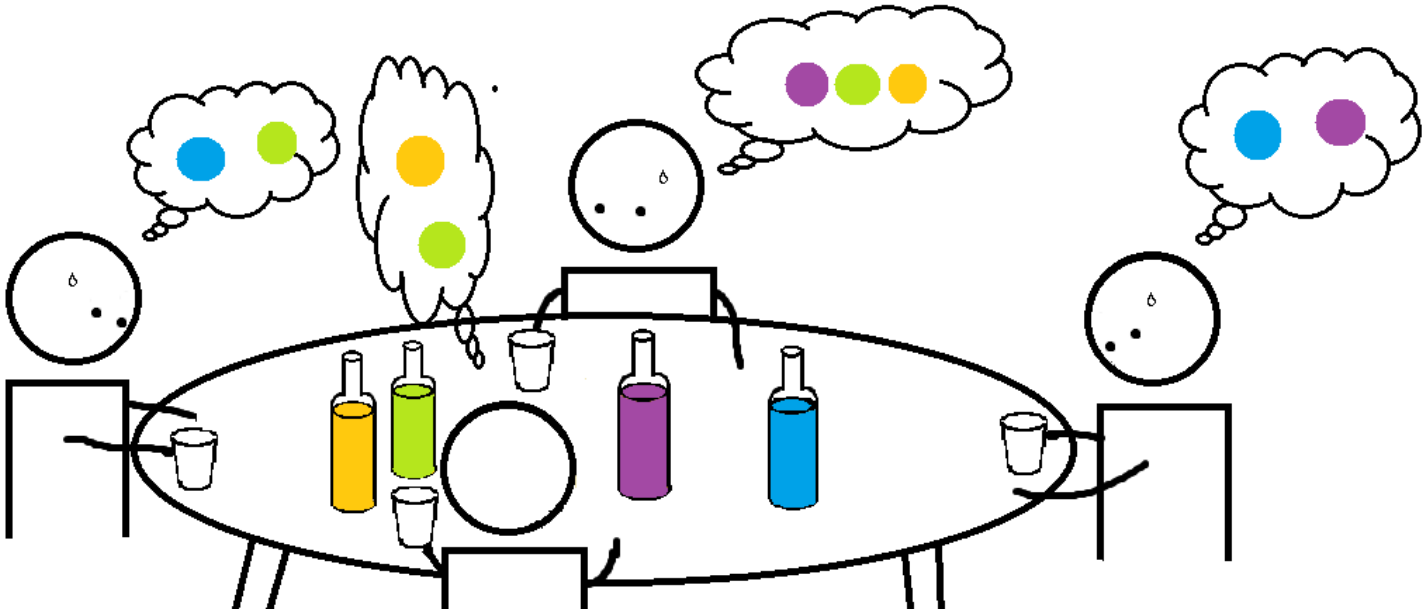
Dining philosophers



Dining philosophers



Drinking philosophers



Lock-sharing systems (LSS)

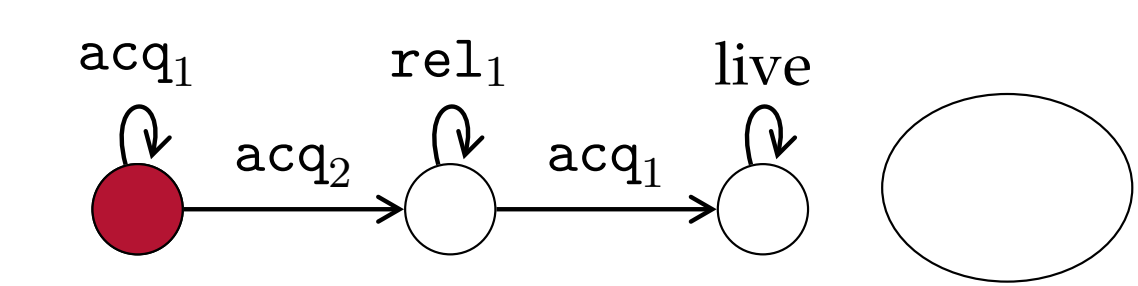
Lock-sharing system

$Proc$: set of processes T: set of locks .

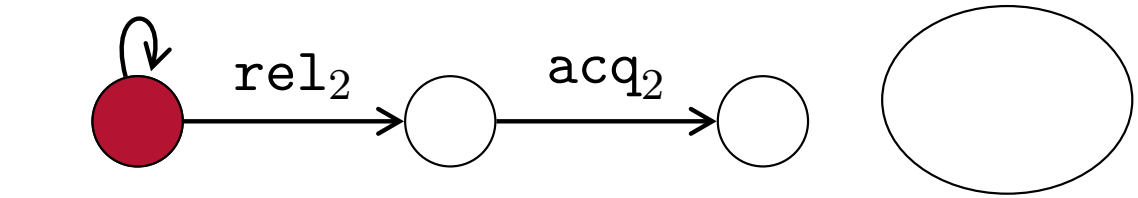
Lock-sharing system: $\mathcal{A}_p = (S_p, \Sigma_p, \delta_p, init_p)$ for each $p \in Proc$.

Transitions include operations on locks :
$\delta_p : S_p \times \Sigma_p \to Op_T \times S_p$ with $Op_T = \{acq_t, rel_t \mid t \in T\} \cup \{nop\}$.

Semantics



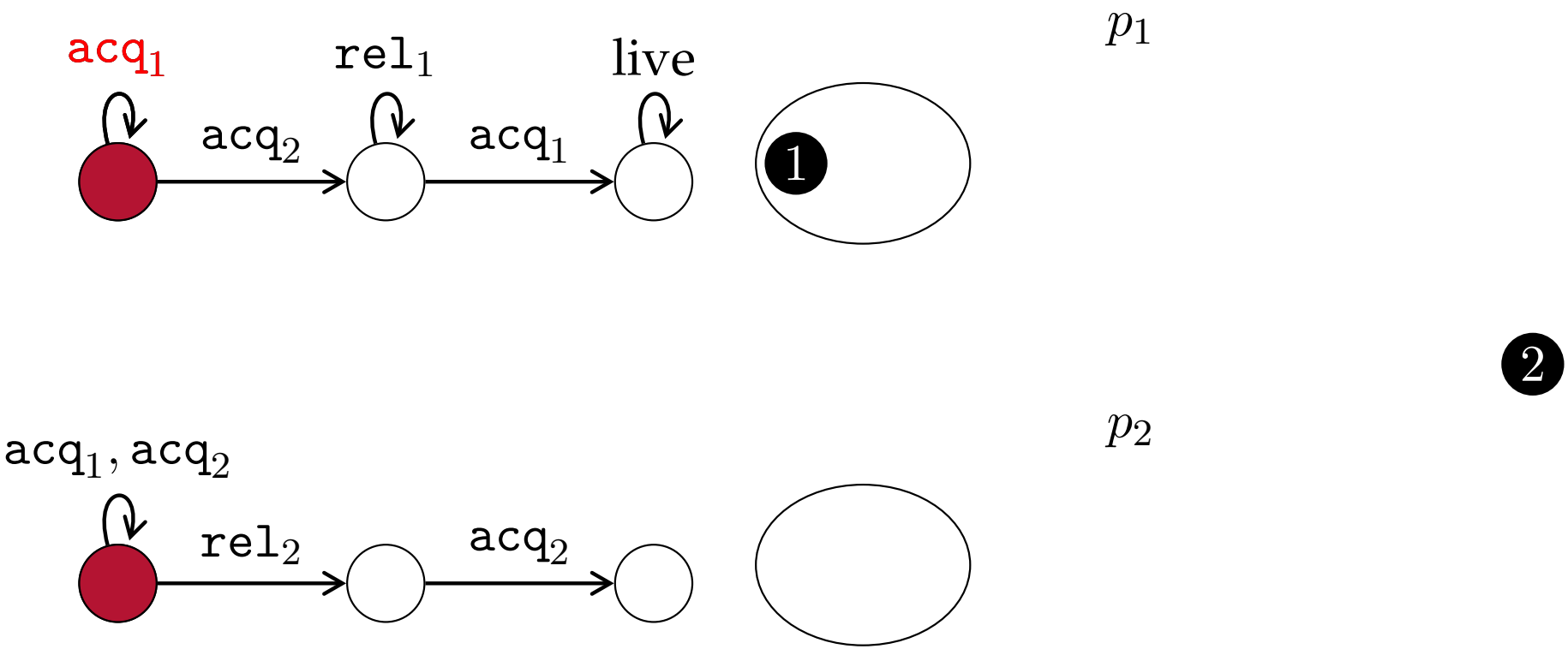
p_1



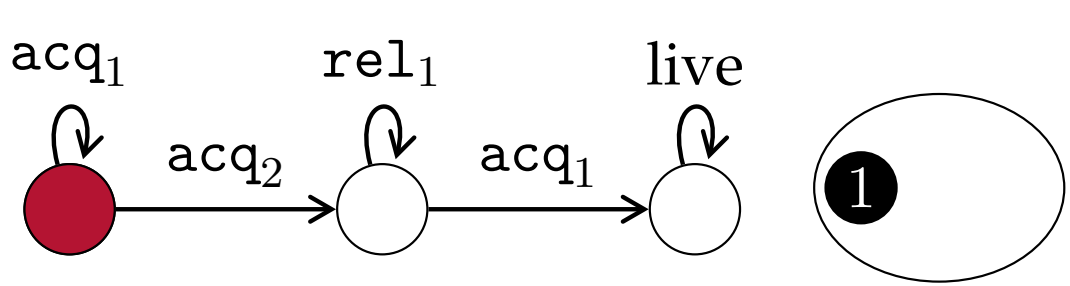
p_2

❶ ❷

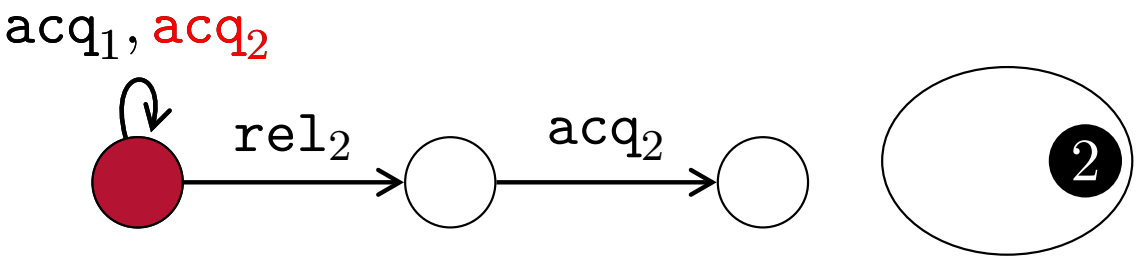
Semantics


$\mathtt{acq}_1$
$\mathtt{rel}_1$
live
$\mathtt{acq}_2$
$\mathtt{acq}_1$
1
p_1
2
$\mathtt{acq}_1, \mathtt{acq}_2$
$\mathtt{rel}_2$
$\mathtt{acq}_2$
p_2

Semantics

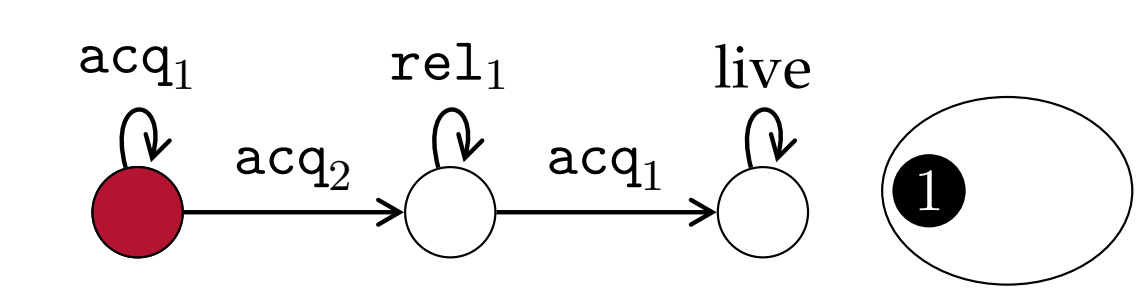


p_1



p_2

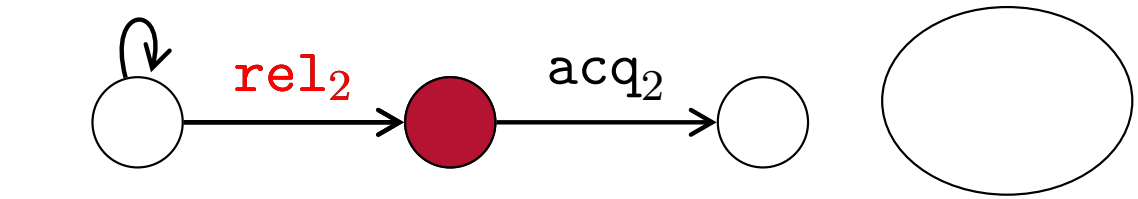
Semantics



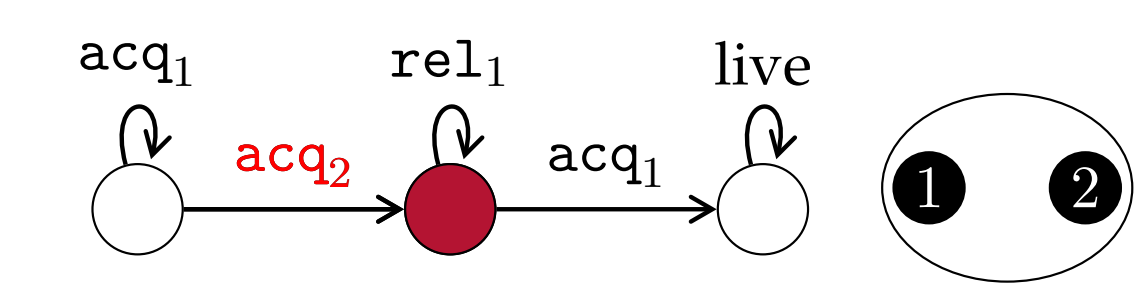
p_1

2

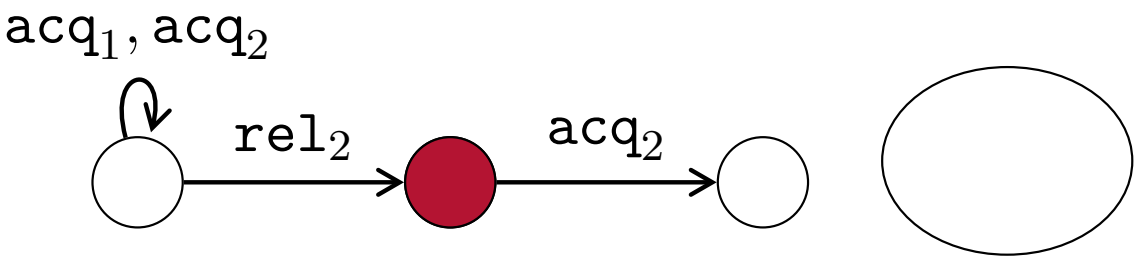
p_2



Semantics

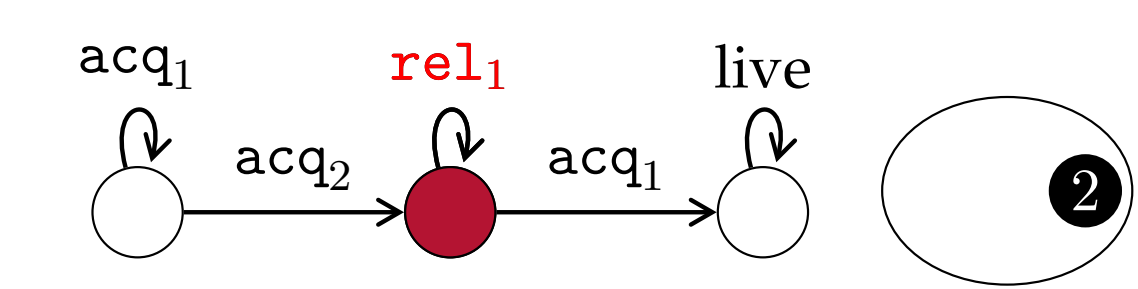


p_1



p_2

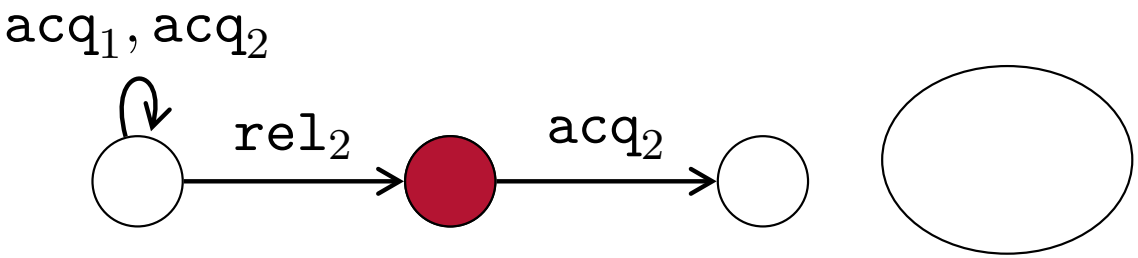
Semantics



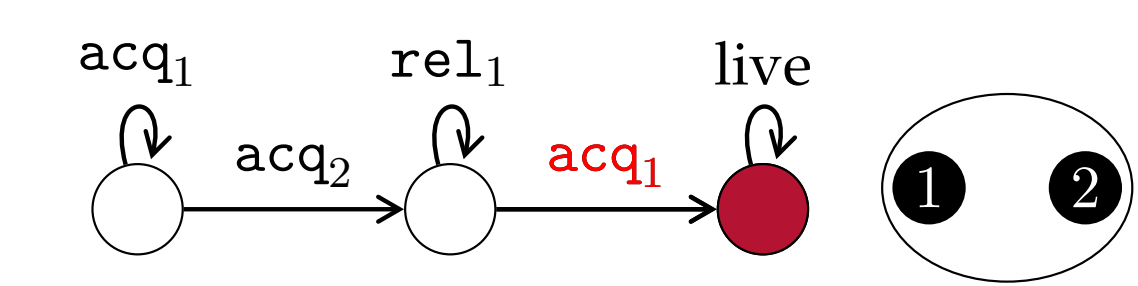
p_1

1

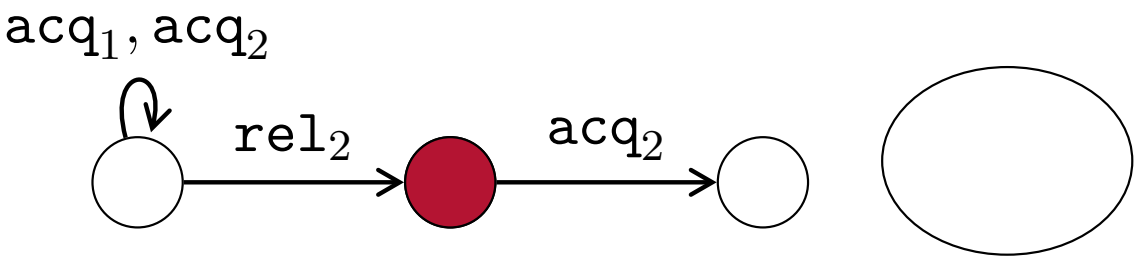
p_2



Semantics



p_1



p_2

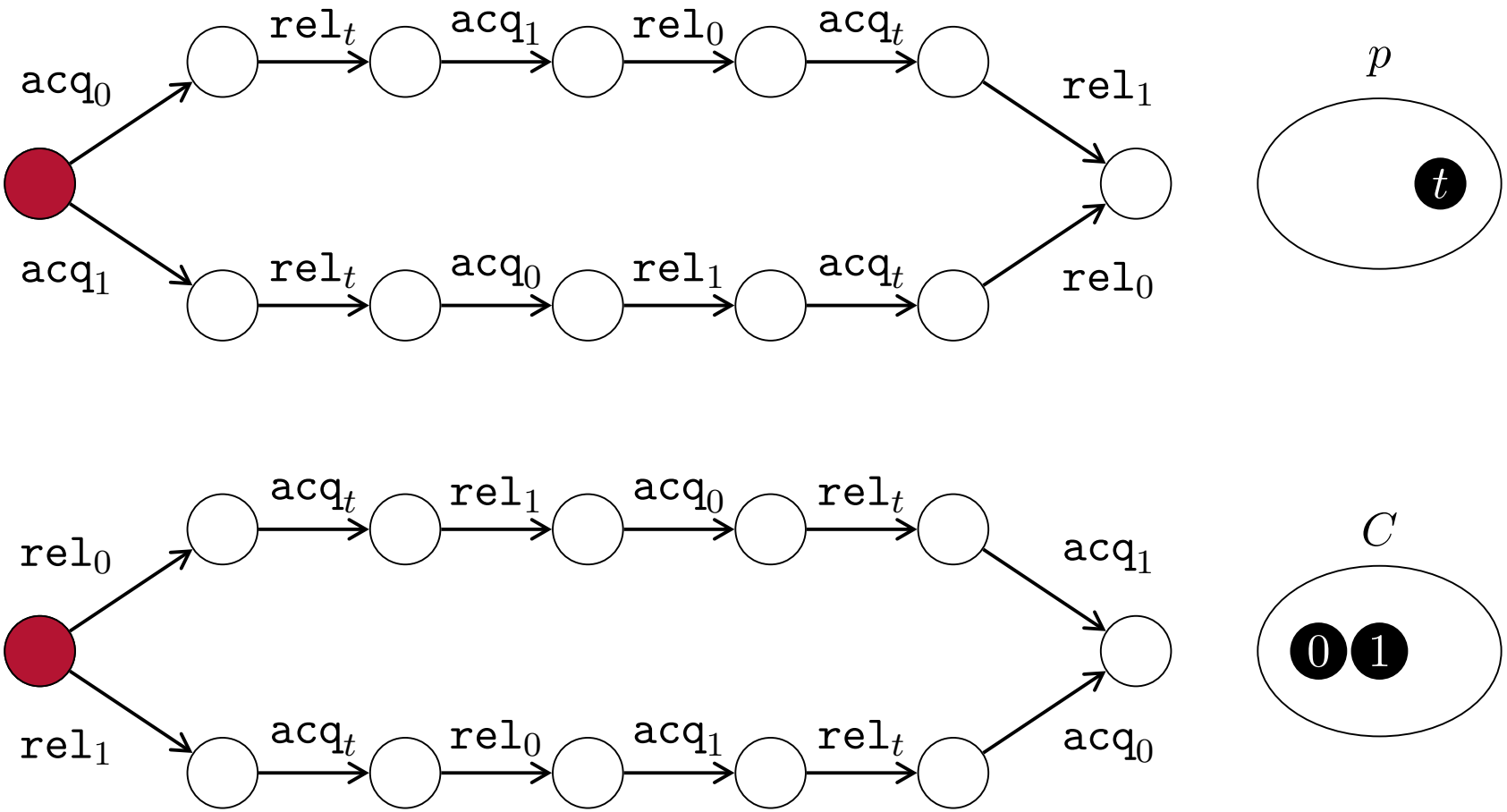
A problem

Model-checking problem

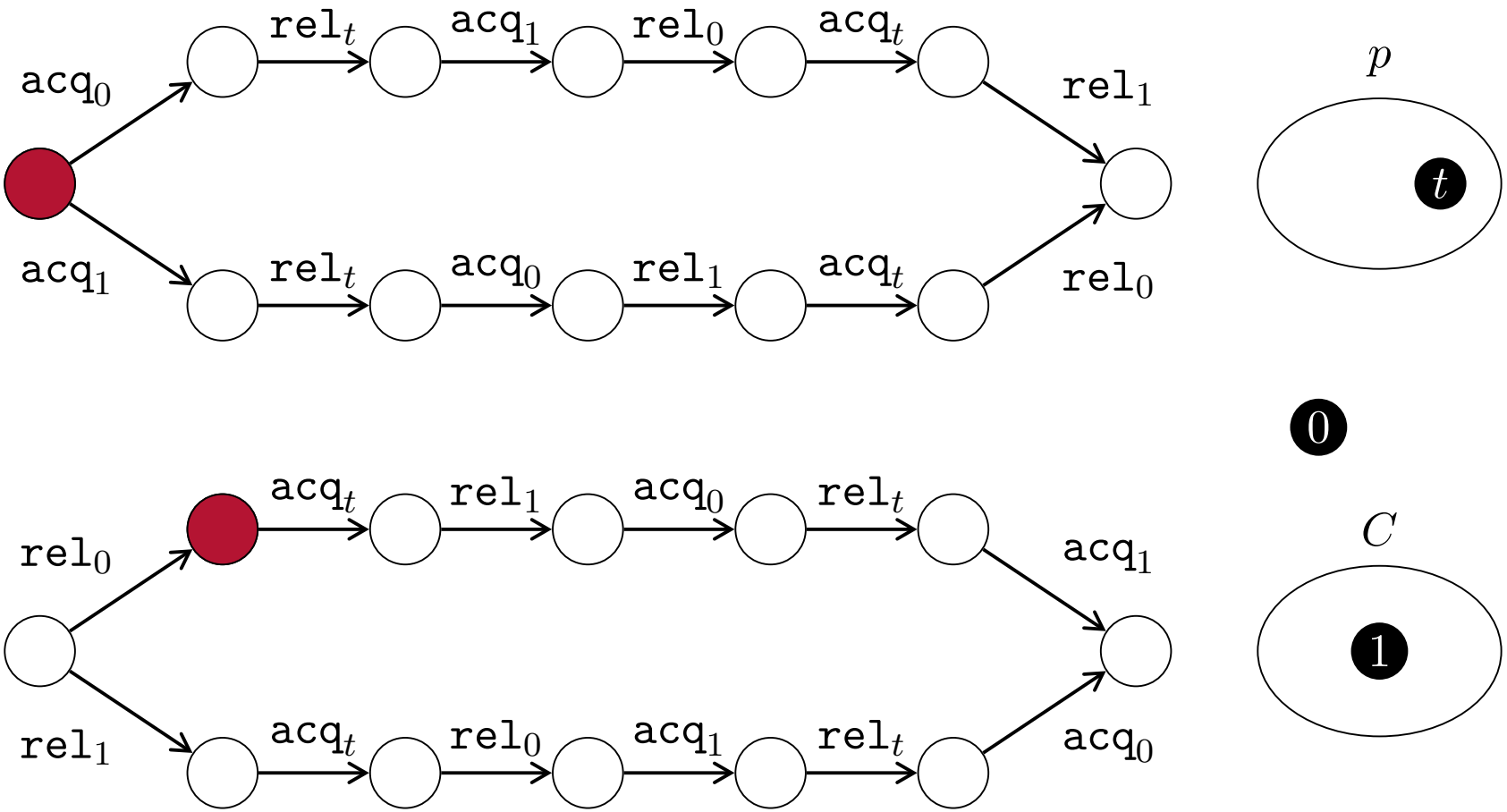
Input: A set of processes $Proc$, a set of locks T, an LSS $(\mathcal{A}_p)_{p \in Proc}$, a property $\mathcal{P}$ (for instance $\mathcal{P}$ =“the run ends in a deadlock”)

Output: Is there a run satisfying $\mathcal{P}$?

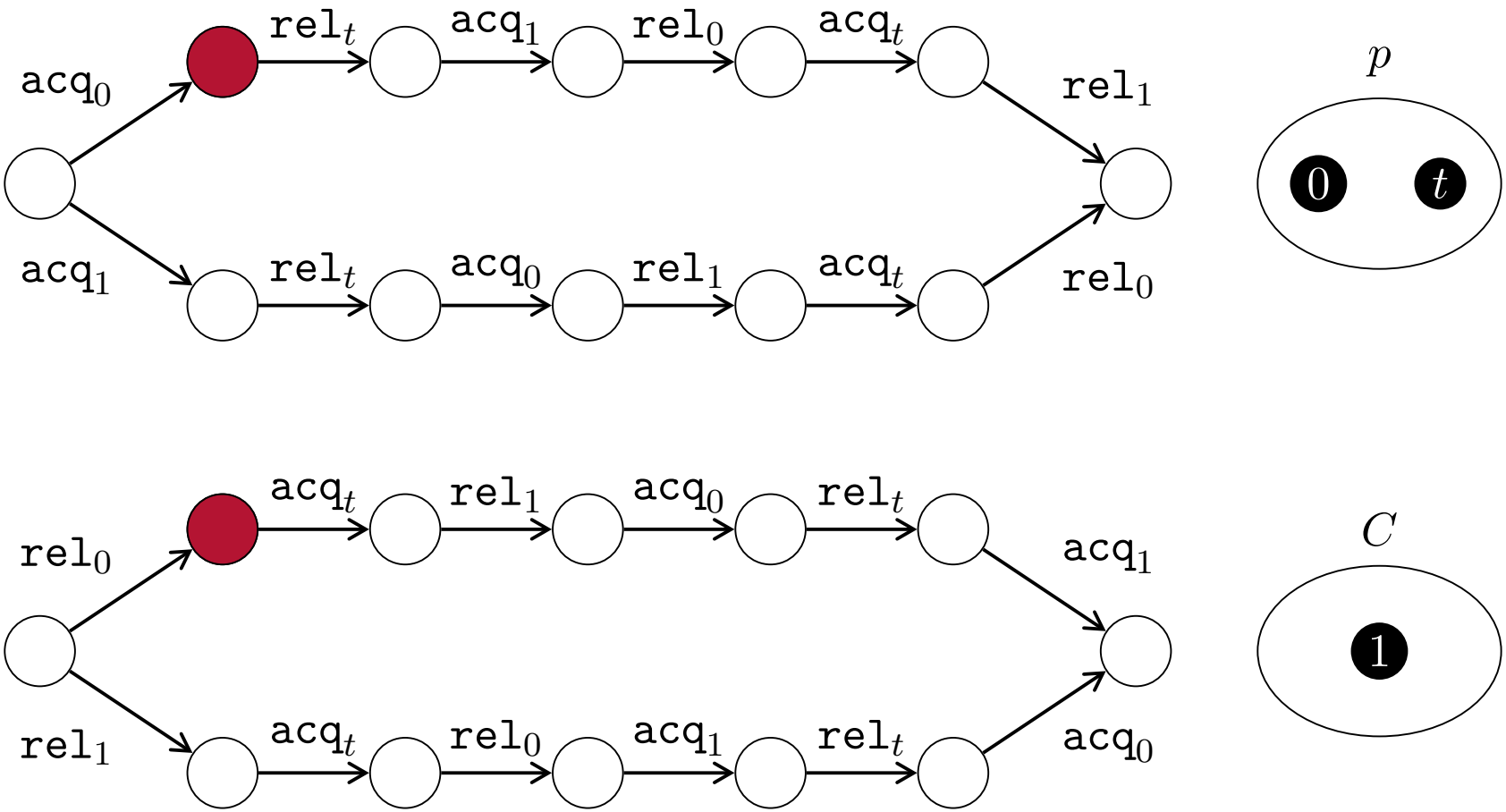
Passing information



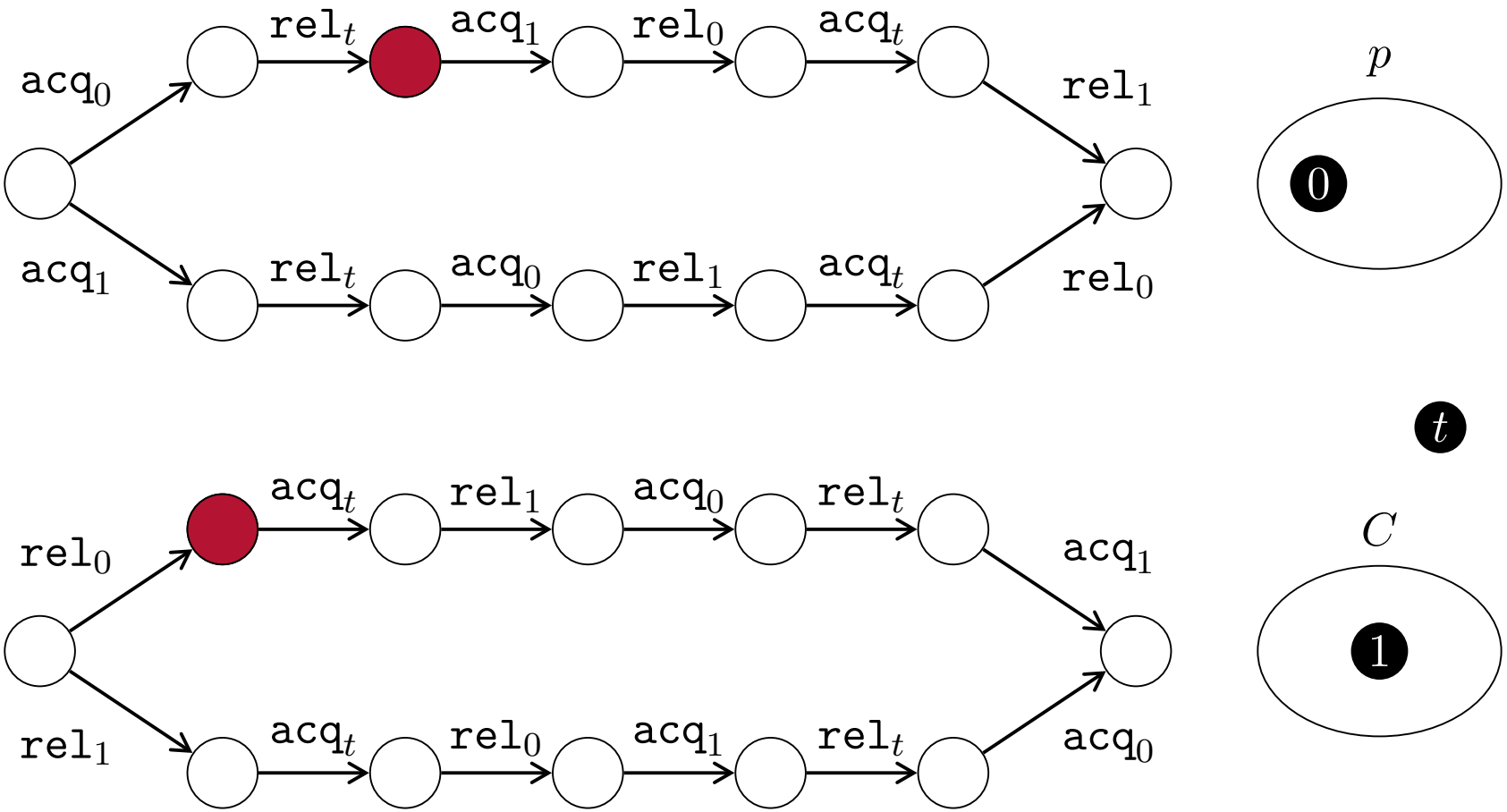
Passing information



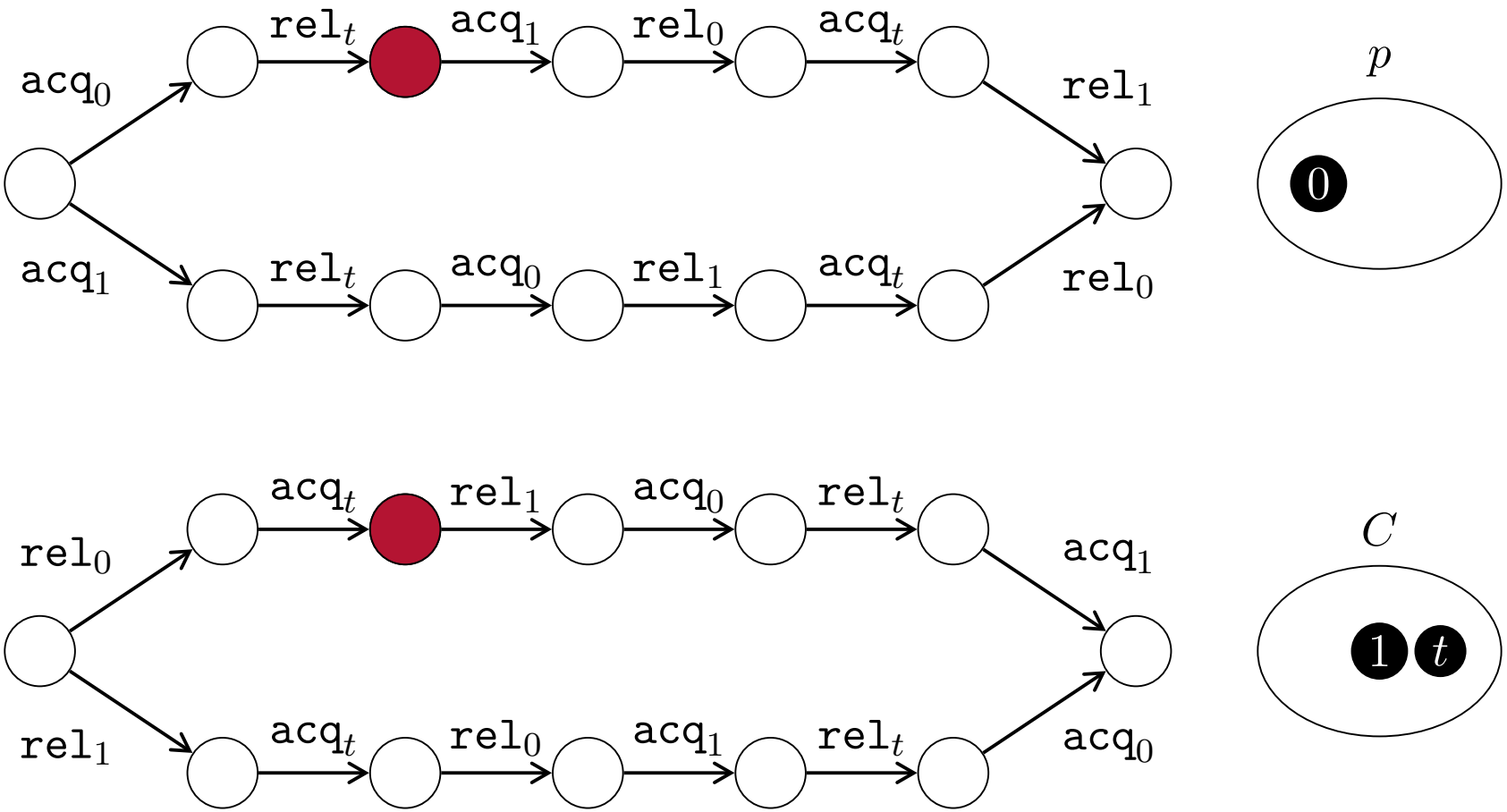
Passing information



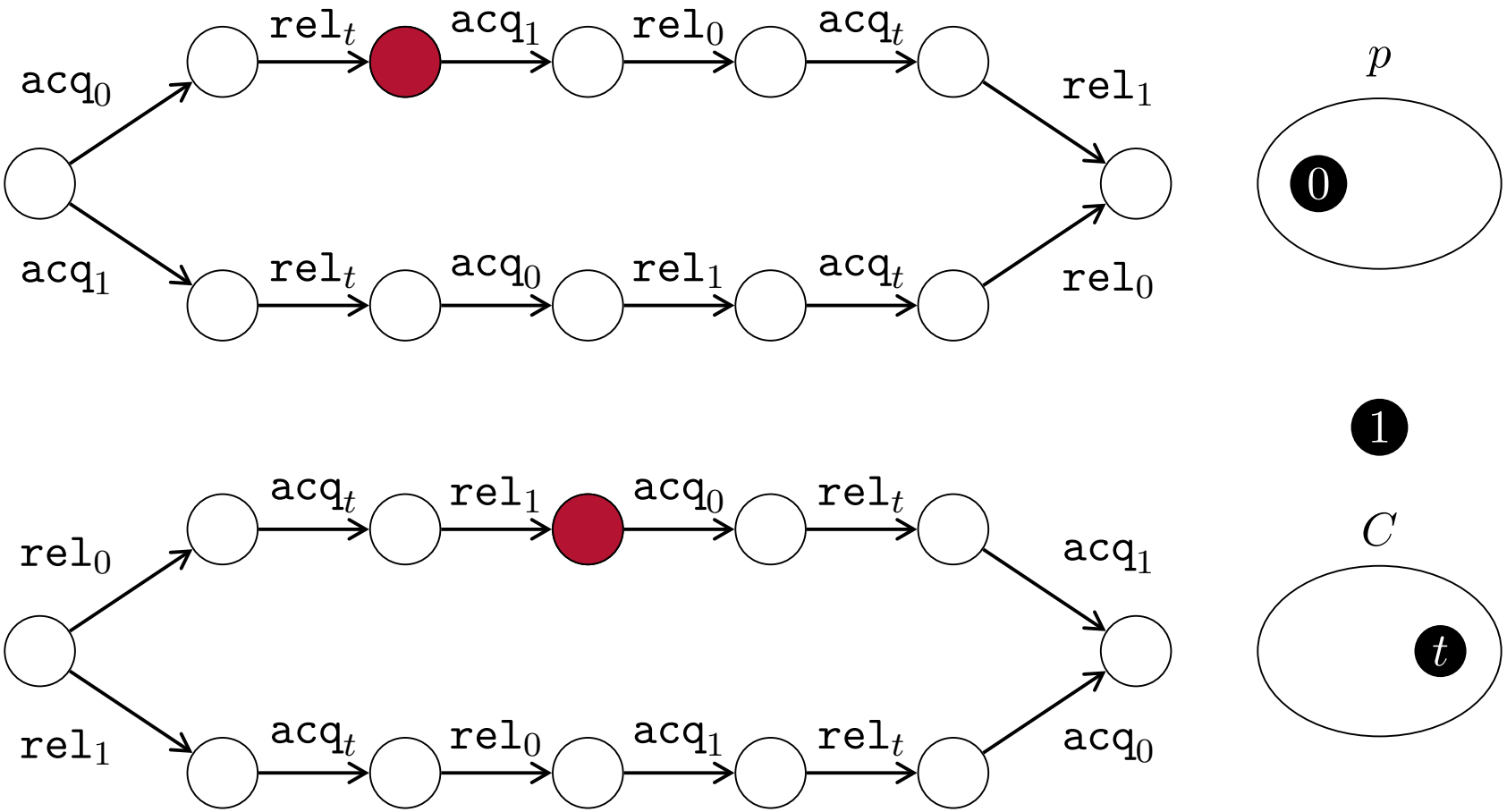
Passing information



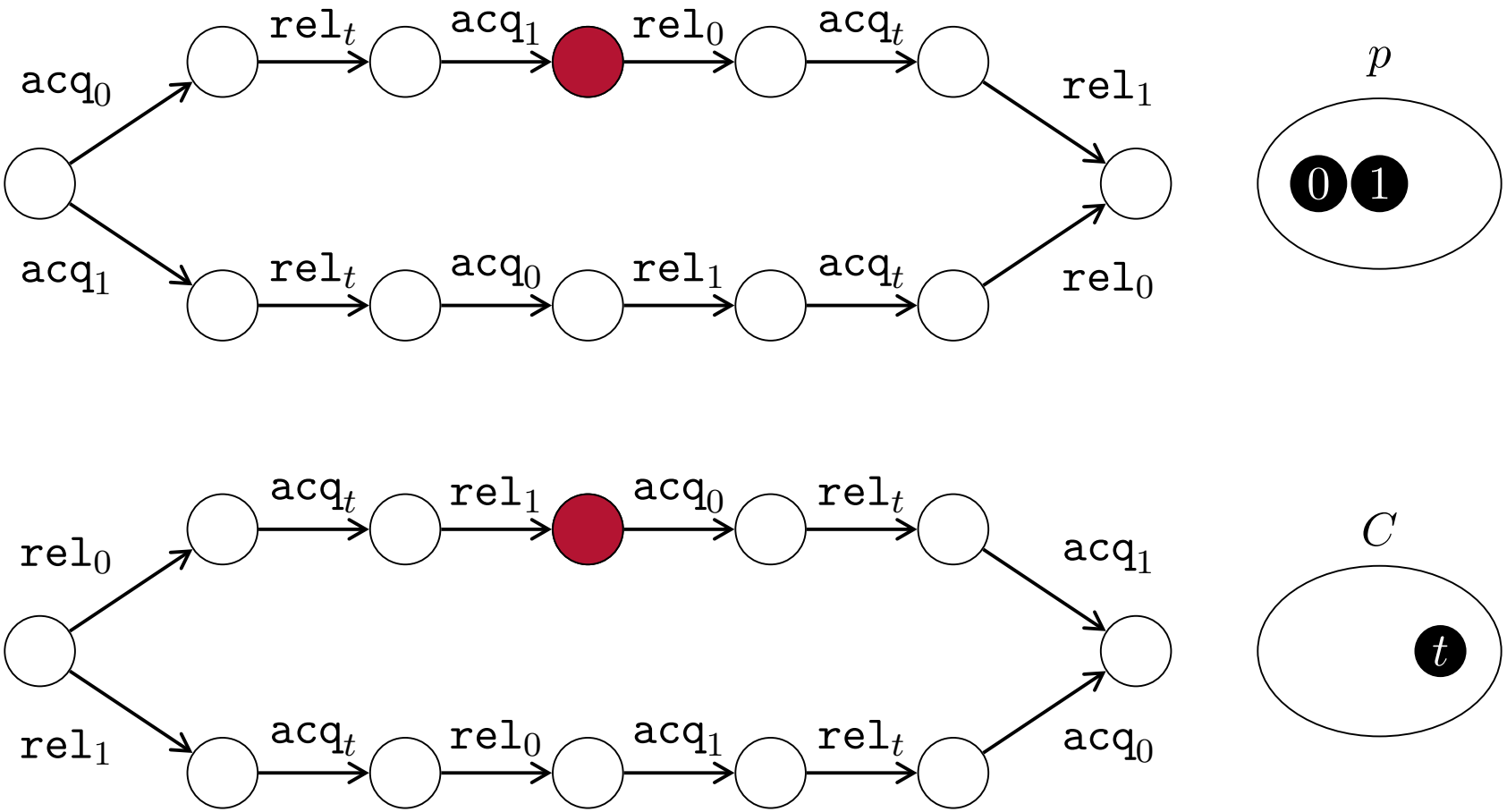
Passing information



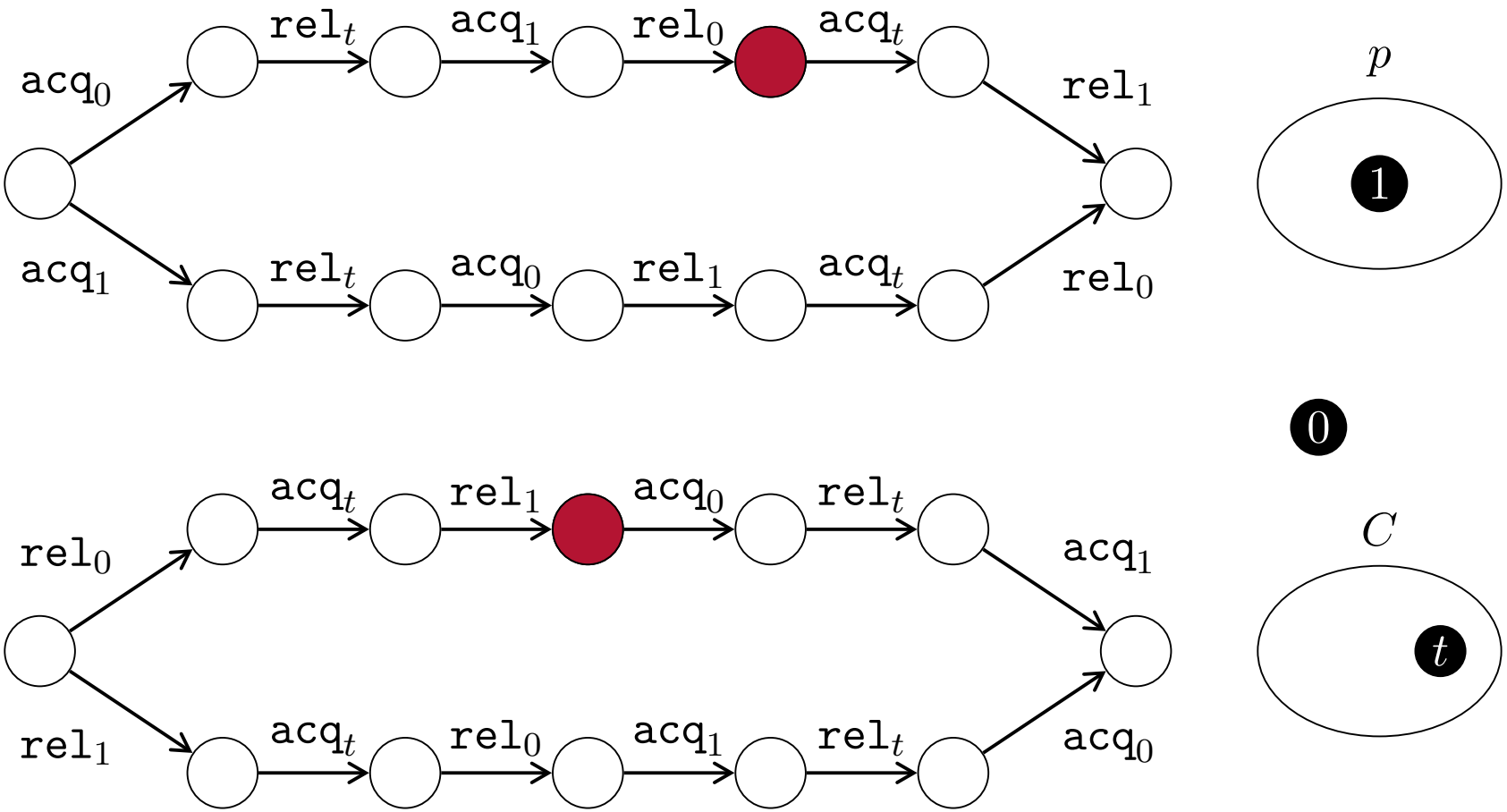
Passing information



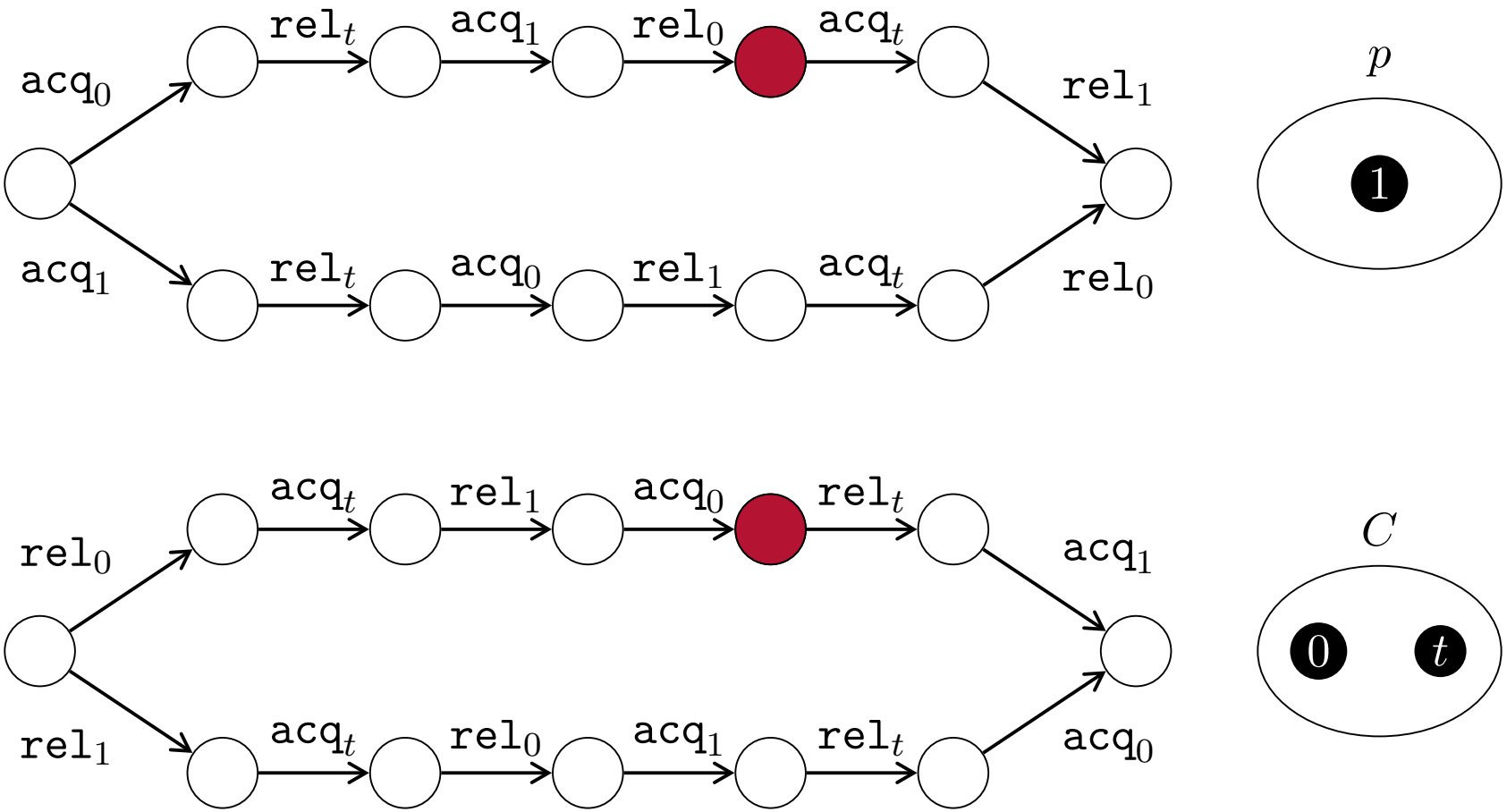
Passing information



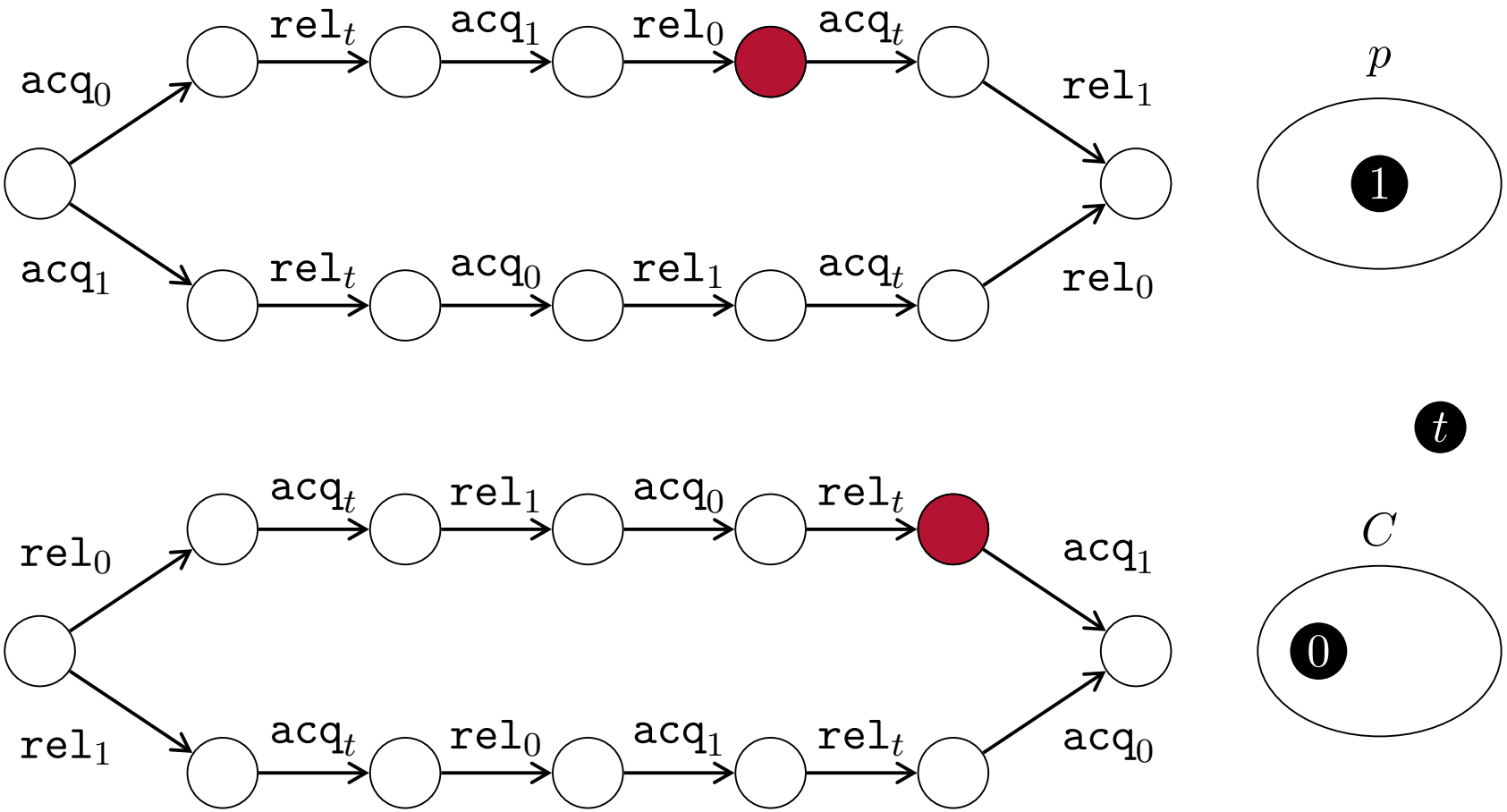
Passing information



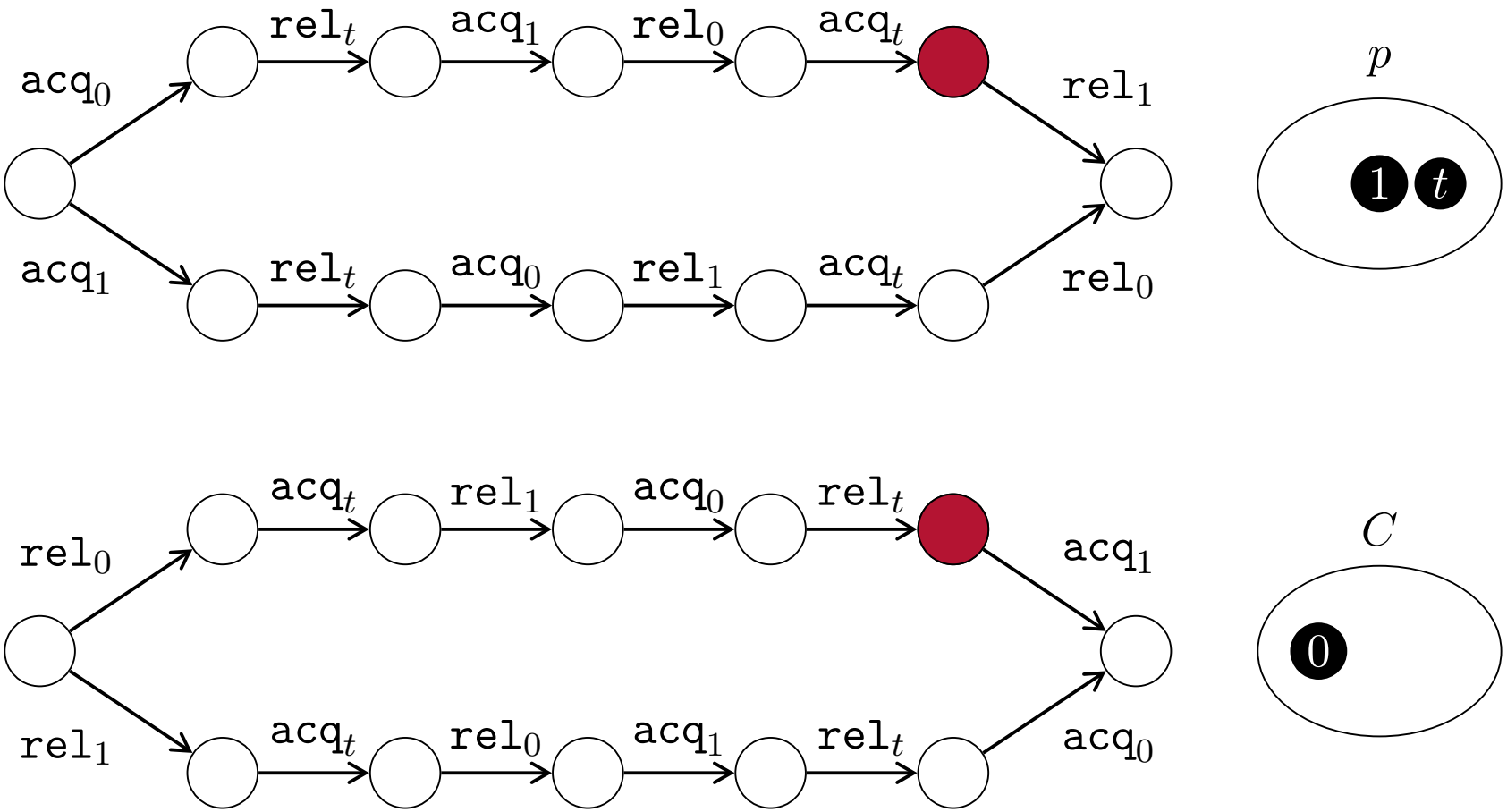
Passing information



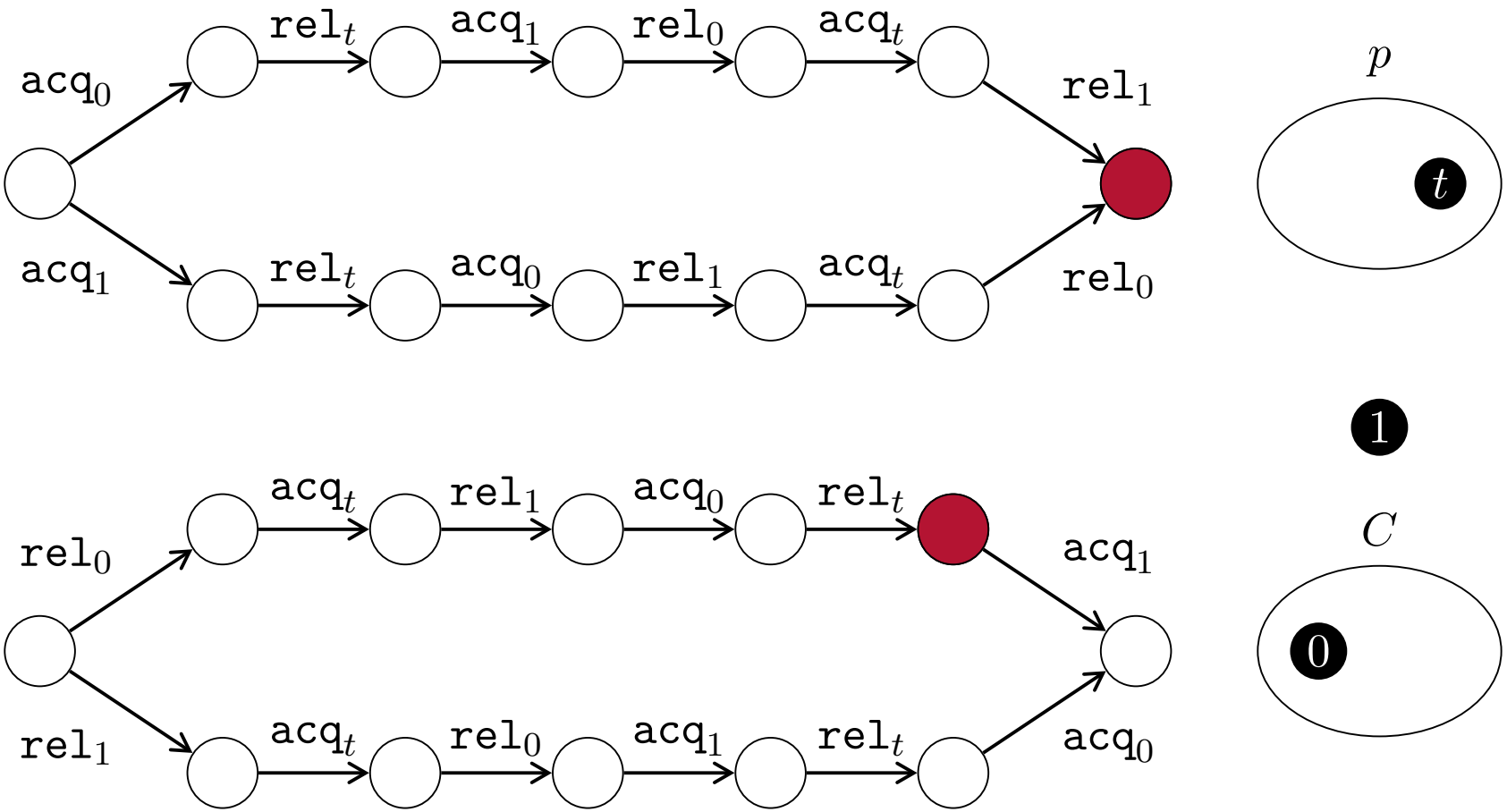
Passing information



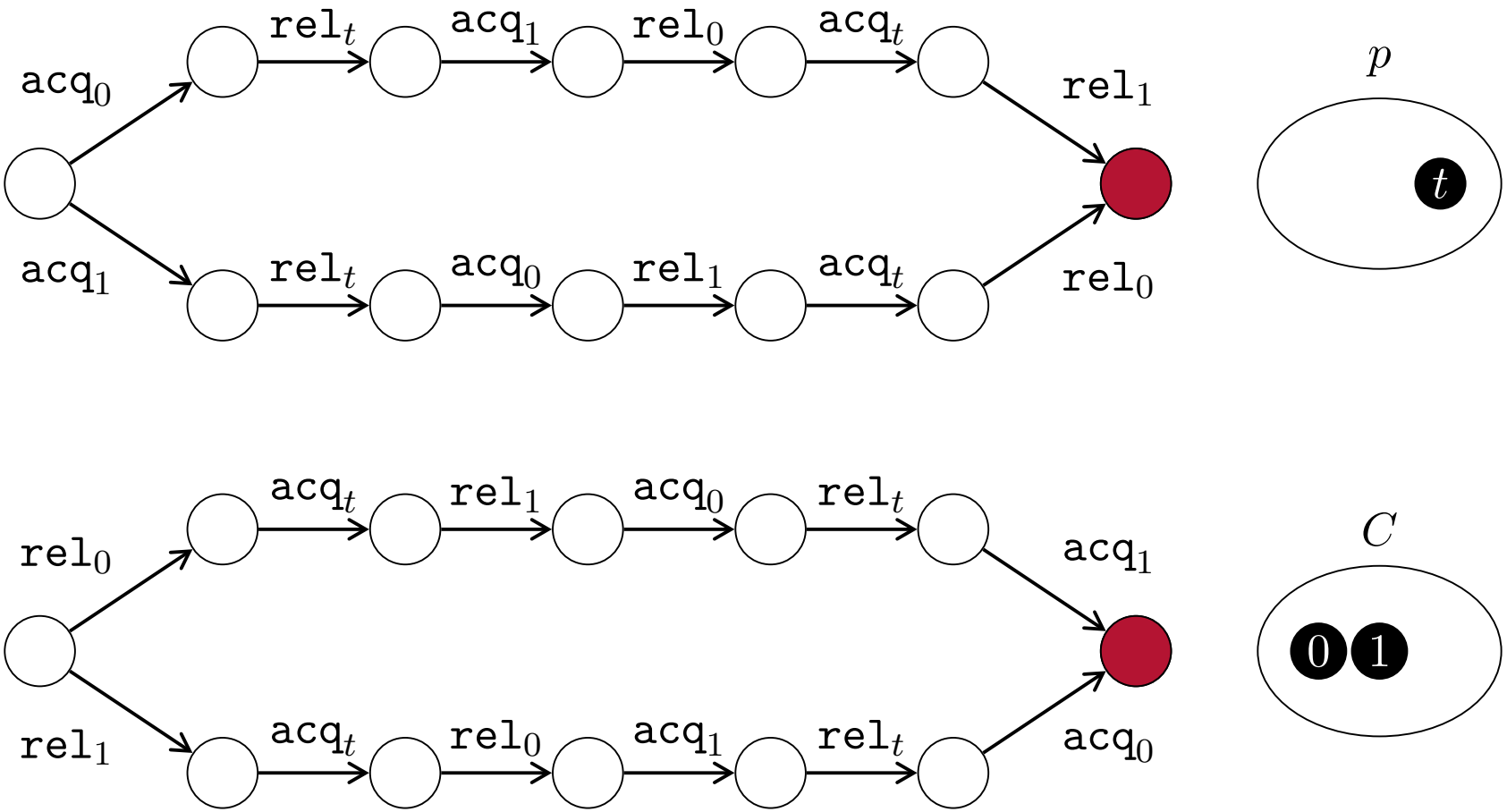
Passing information



Passing information



Passing information



Useful restrictions

If we can pass unlimited information, **PSPACE-complete**

Useful restrictions

If we can pass unlimited information, **PSPACE-complete**

2LSS

Each process accesses **at most two** different locks.

Useful restrictions

If we can pass unlimited information, **PSPACE-complete**

2LSS

Each process accesses **at most two** different locks.

Nested LSS

All processes acquire and release locks in a **stack-like order**, i.e., a process can only release the lock it acquired the latest.

Useful restrictions

If we can pass unlimited information, **PSPACE-complete**

2LSS

Each process accesses **at most two** different locks.

Nested LSS

All processes acquire and release locks in a **stack-like order**, i.e., a process can only release the lock it acquired the latest.

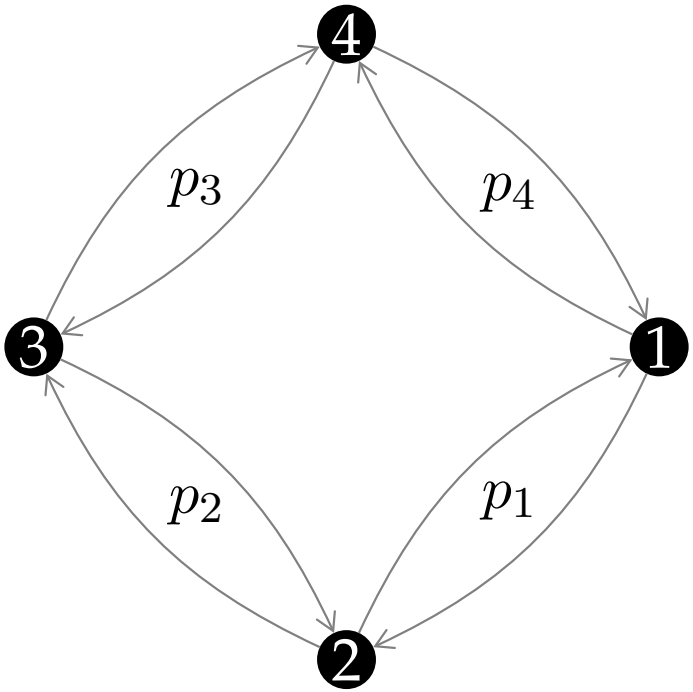
Those cases are generally **NP-complete**

Key property

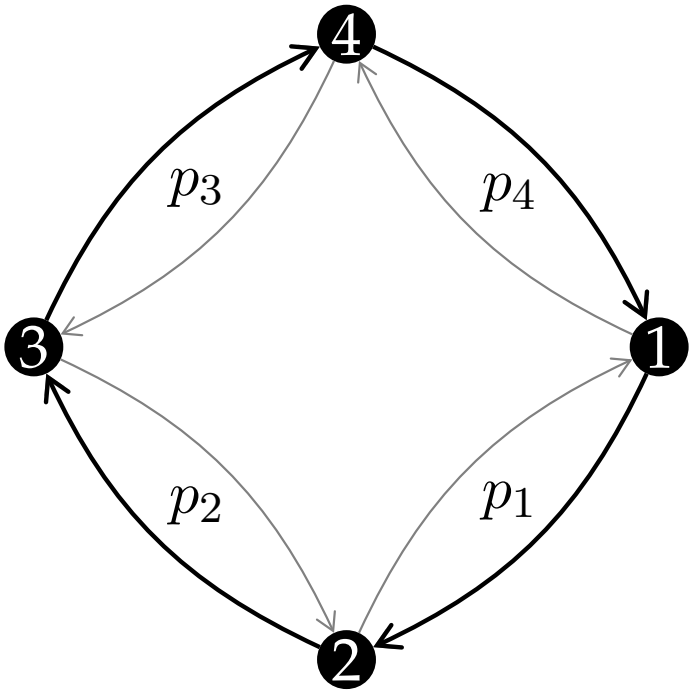
We summarize runs as short patterns .

Patterns tell us if some local runs can be interleaved into a global run leading to a deadlock.

Solving the dining philosophers

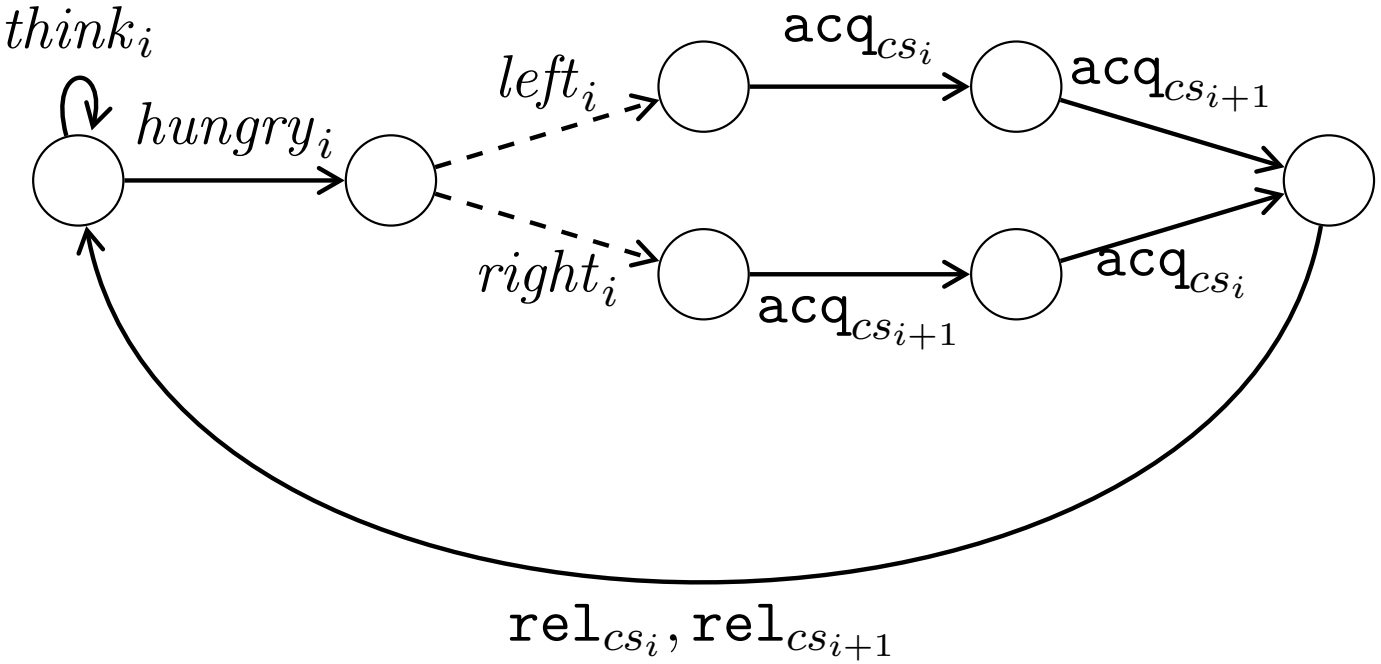


Solving the dining philosophers



Synthesis

Processes can forbid some transitions



If σ_{p_i} always selects $left_i$ and σ_{p_j} $right_j$ for some processes $i \neq j$ then it avoids deadlocks.

Strategies

System and Environment actions $\Sigma_p = \Sigma_p^s \sqcup \Sigma_p^e$.

Strategy

$(\sigma_p)_{p \in Proc}$ with $\sigma_p : \Sigma_p^* \to 2^{\Sigma_p}$ such that $\Sigma_p^e \subseteq \sigma_p(u)$ for all u.

Strategies

System and Environment actions $\Sigma_p = \Sigma_p^s \sqcup \Sigma_p^e$.

Strategy

$(\sigma_p)_{p \in Proc}$ with $\sigma_p : \Sigma_p^* \to 2^{\Sigma_p}$ such that $\Sigma_p^e \subseteq \sigma_p(u)$ for all u.

Locally live strategy

σ is *locally live* if all processes always have an available local action.

Strategies

System and Environment actions $\Sigma_p = \Sigma_p^s \sqcup \Sigma_p^e$.

Strategy

$(\sigma_p)_{p \in Proc}$ with $\sigma_p : \Sigma_p^* \to 2^{\Sigma_p}$ such that $\Sigma_p^e \subseteq \sigma_p(u)$ for all u.

Locally live strategy

σ is *locally live* if all processes always have an available local action.

Global deadlock

A σ-run u reaches a *global deadlock* if all processes are blocked at the end.

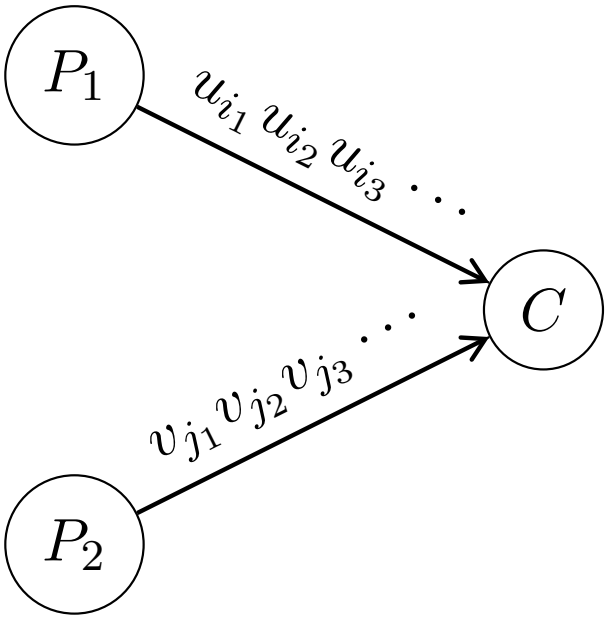
Undecidability

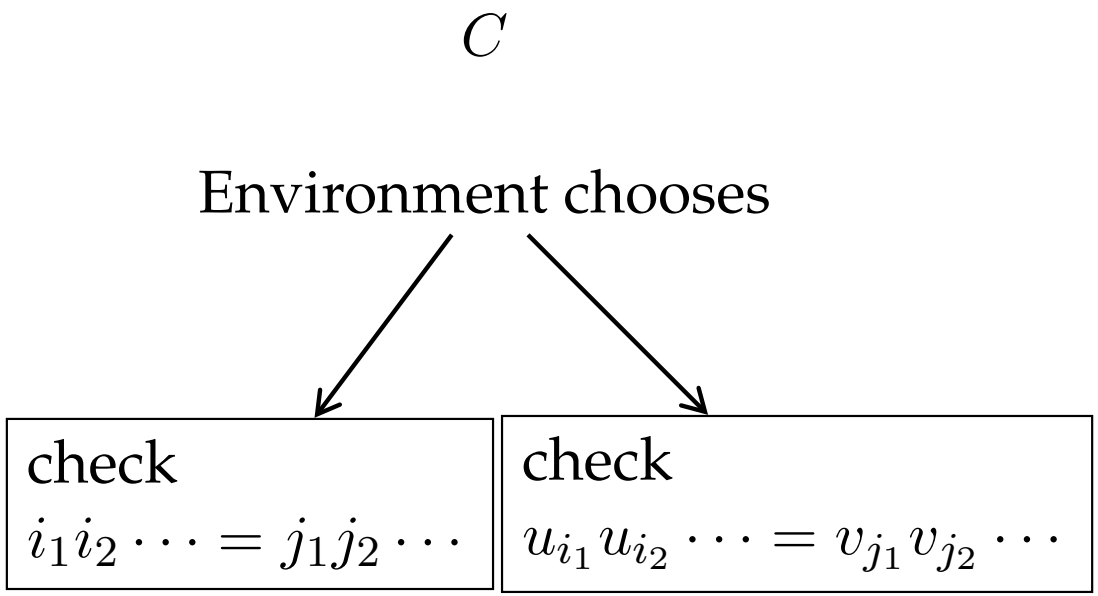
Theorem

The existence of a strategy avoiding global deadlocks is undecidable, even with 3 processes and 4 locks in total.

Proof scheme: PCP encoding

Let $(u_1, v_1), \ldots, (u_n, v_n)$ be a PCP instance.



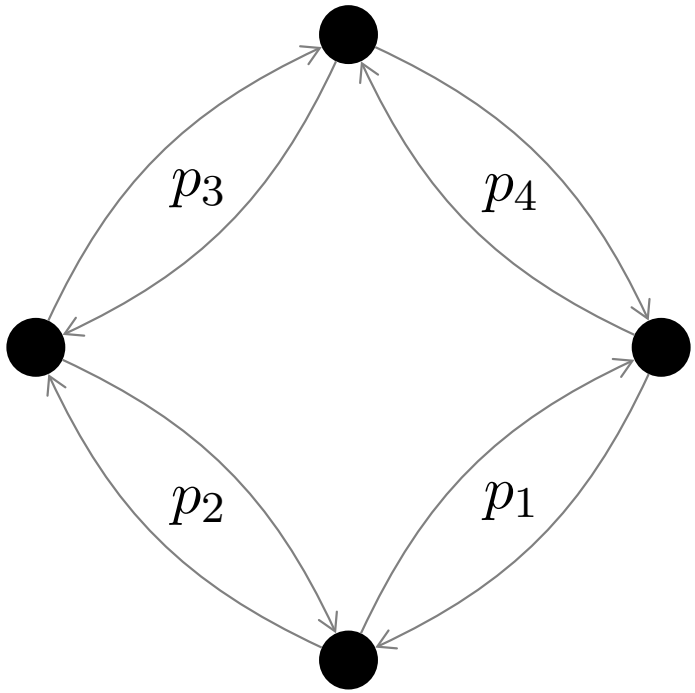


Decidability

Theorem

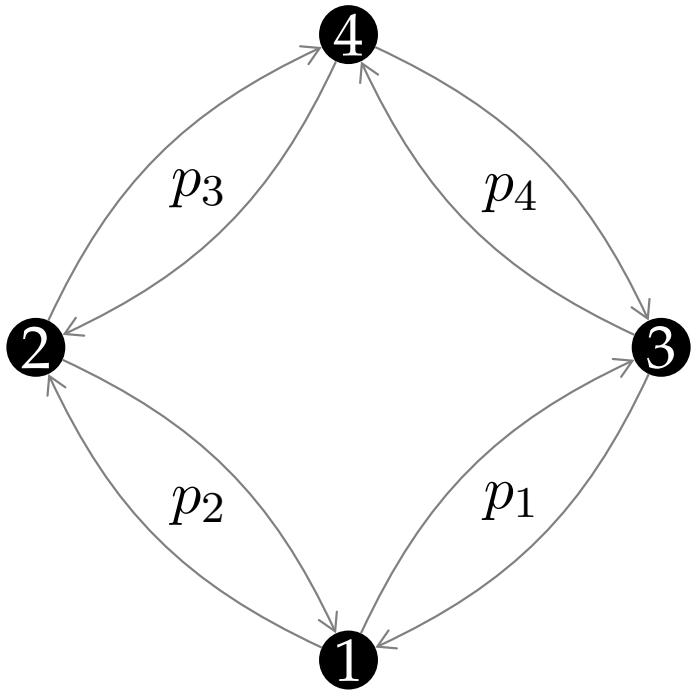
The existence of a strategy avoiding global deadlocks is Σ_2^P-complete for 2LSS
and NExpTime-complete for LSS respecting the nested lock condition.

Solving the dining philosophers



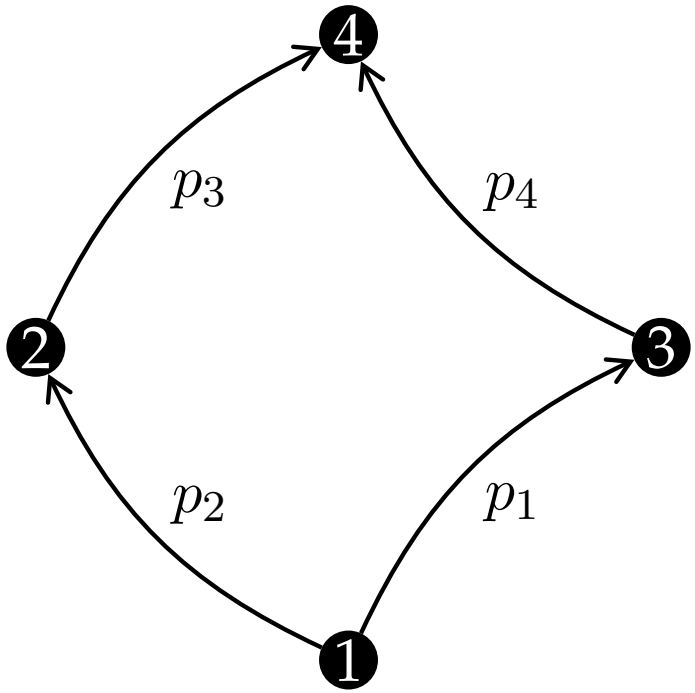
$\rightarrow$ Just pick an order on chopsticks and have all philosophers take them accordingly!

Solving the dining philosophers



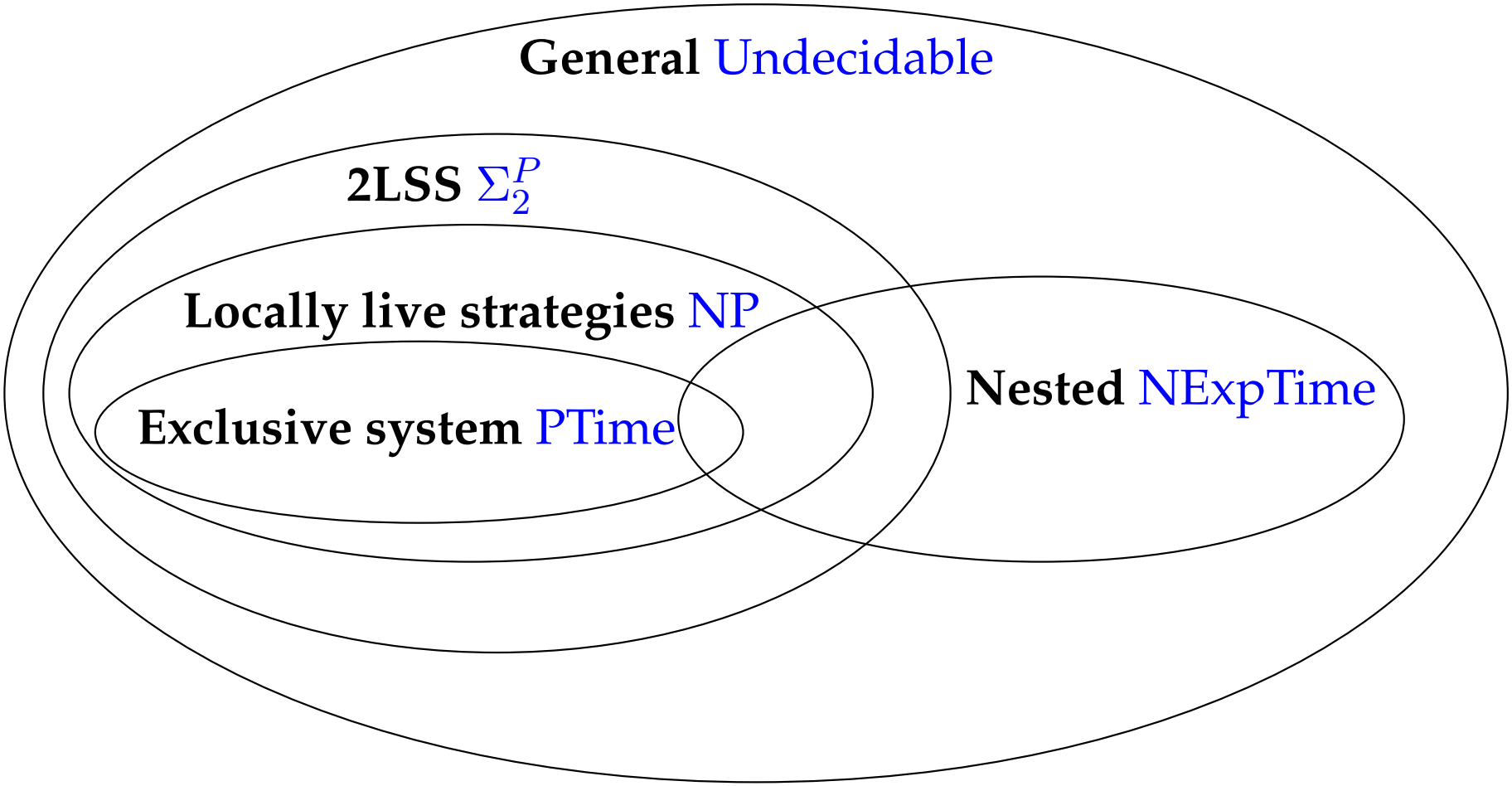
$\rightarrow$ Just pick an order on chopsticks and have all philosophers take them accordingly!

Solving the dining philosophers



$\rightarrow$ Just pick an order on chopsticks and have all philosophers take them accordingly!

Results


General Undecidable
2LSS Σ_2^P
Locally live strategies NP
Exclusive system PTime
Nested NExpTime

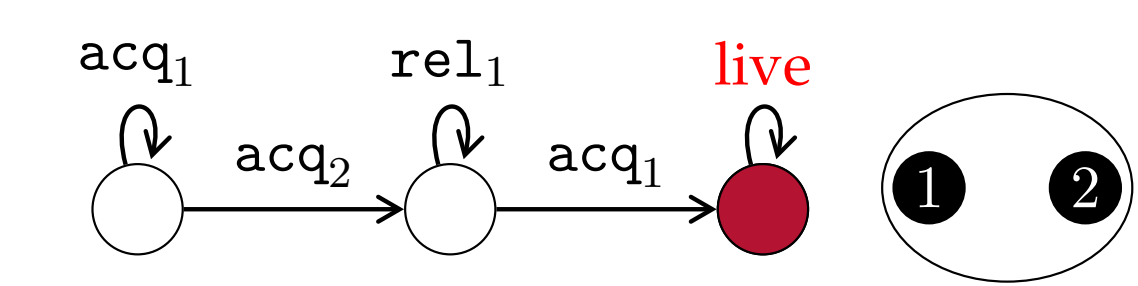
An open problem

Can the following problem be solved in PTIME ?

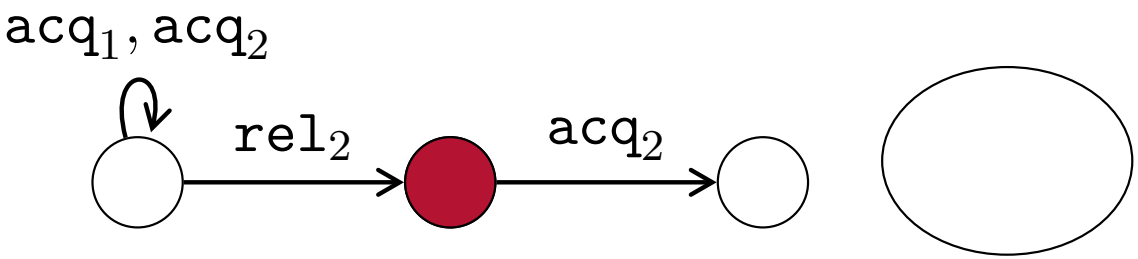
Mortality problem

Input: A 2LSS (each process uses at most 2 locks).
Output: Is there a run such that one of the processes cannot execute any action ever from some point on?

Example



p_1



p_2

Thank you for your attention!