Safety Analysis of Parameterised Networks with Non-Blocking Rendez-Vous

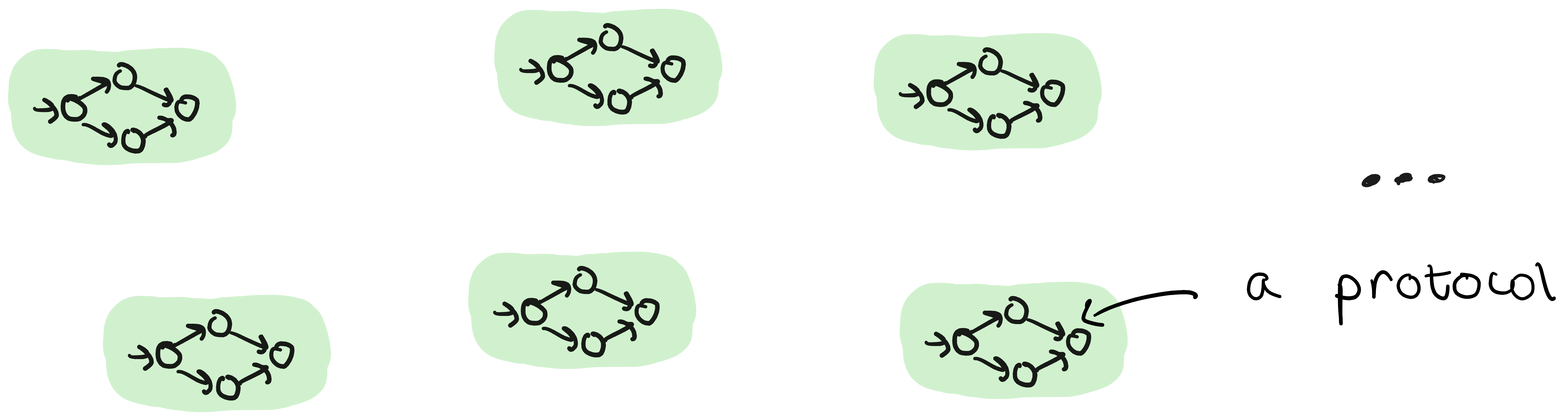
Lucie Guillou, Arnaud Sangnier, Nathalie Sznajder

IRIF, Université Paris Cité — IRIF, Université Paris Cité — LIP6, Sorbonne Université

16th January 2024

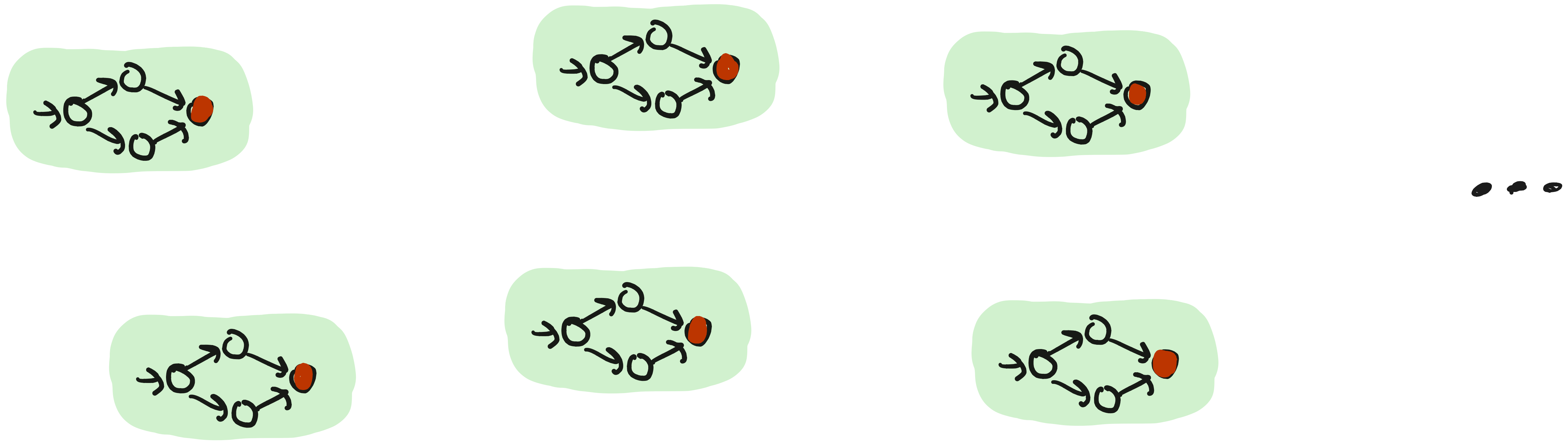
Paredys

Parameterised Distributed Networks



- Unknown number of agents
- Each agent follows a protocol given as a finite-state machine
- Synchronous Communication
- Interleaving Semantics

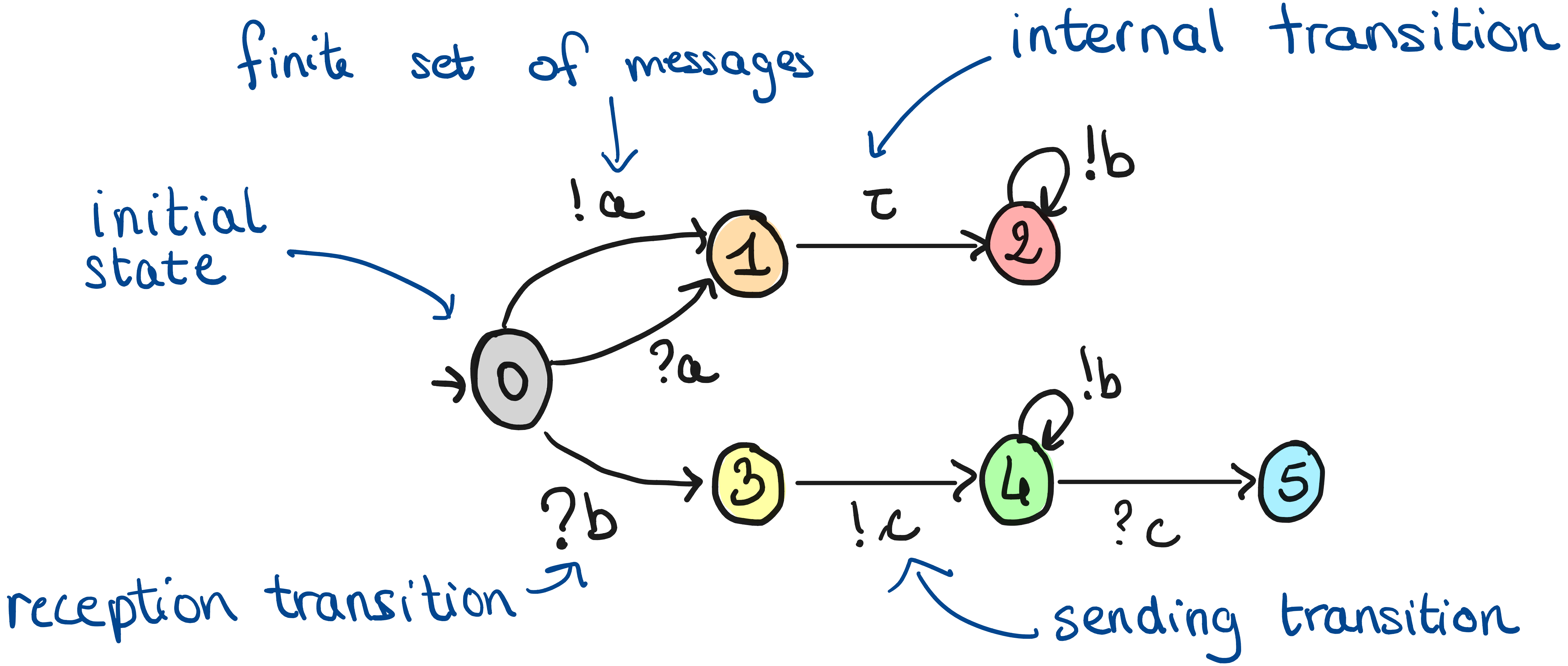
Verification of Parameterised Distributed Networks



Is there a number of agents such that there exists a run leading to a bad configuration?

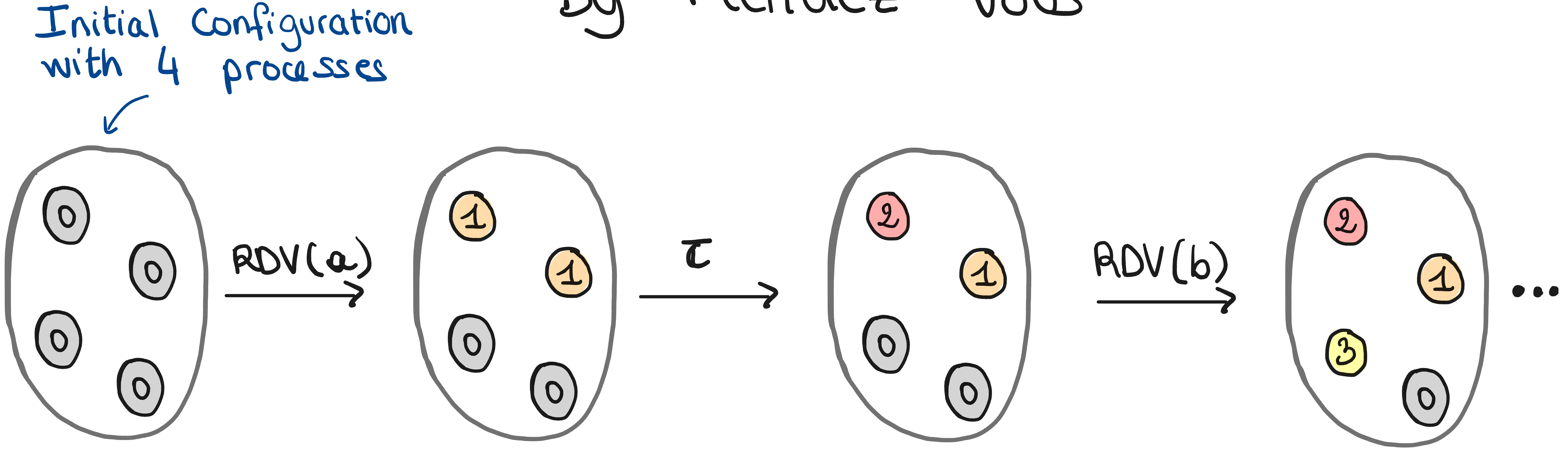
The Model

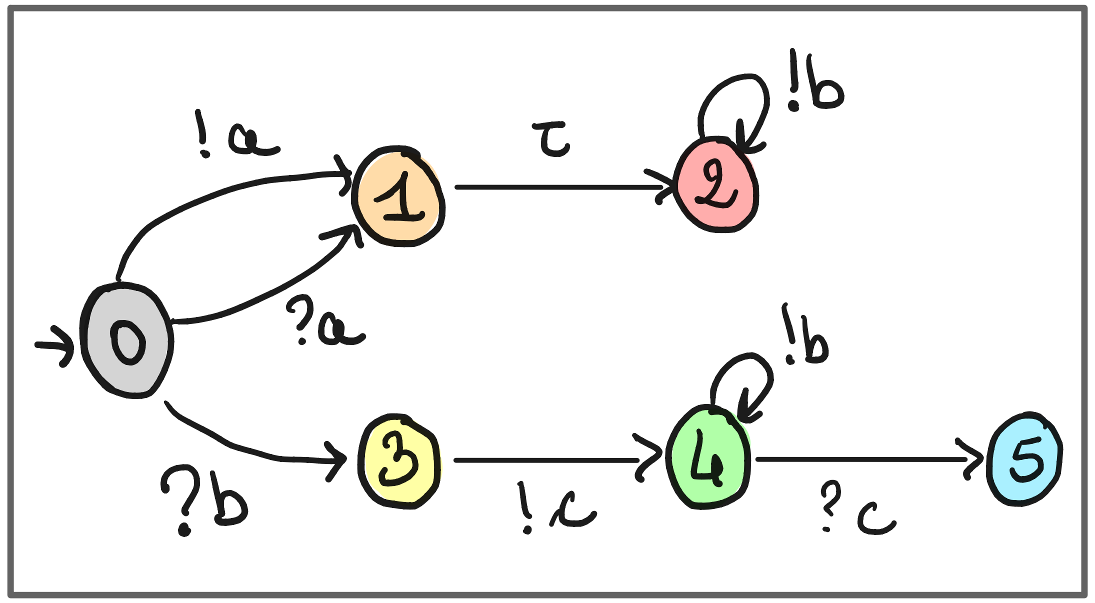
- All agents execute the same finite-state machine called <u>a Protocol</u>



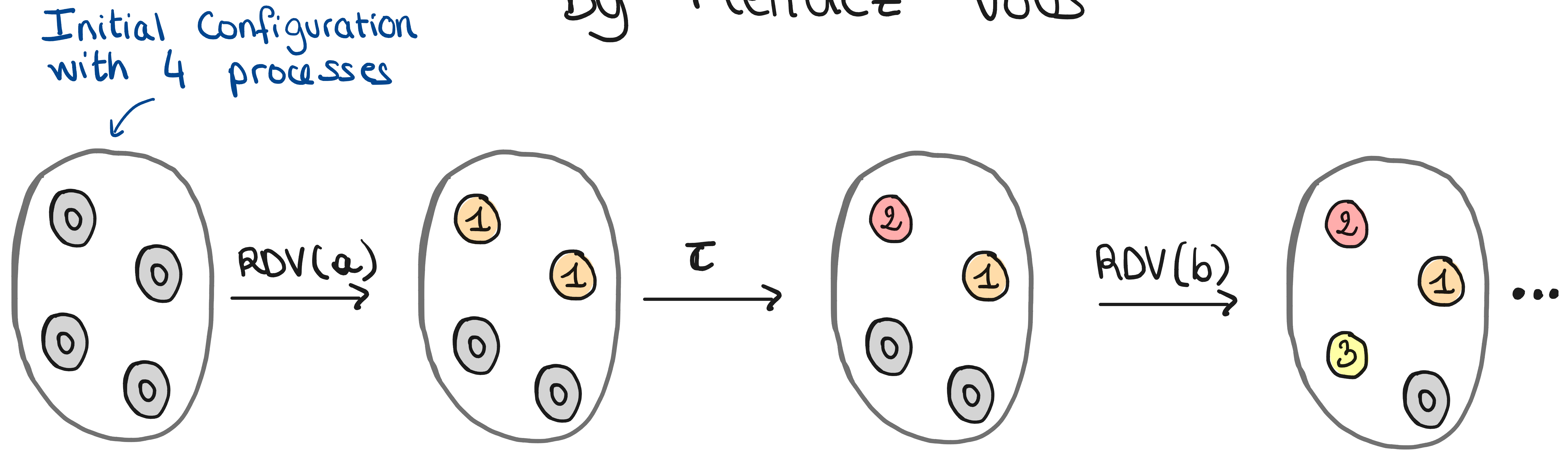
<u>A Protocol</u>

Communication by Rendez-Vous

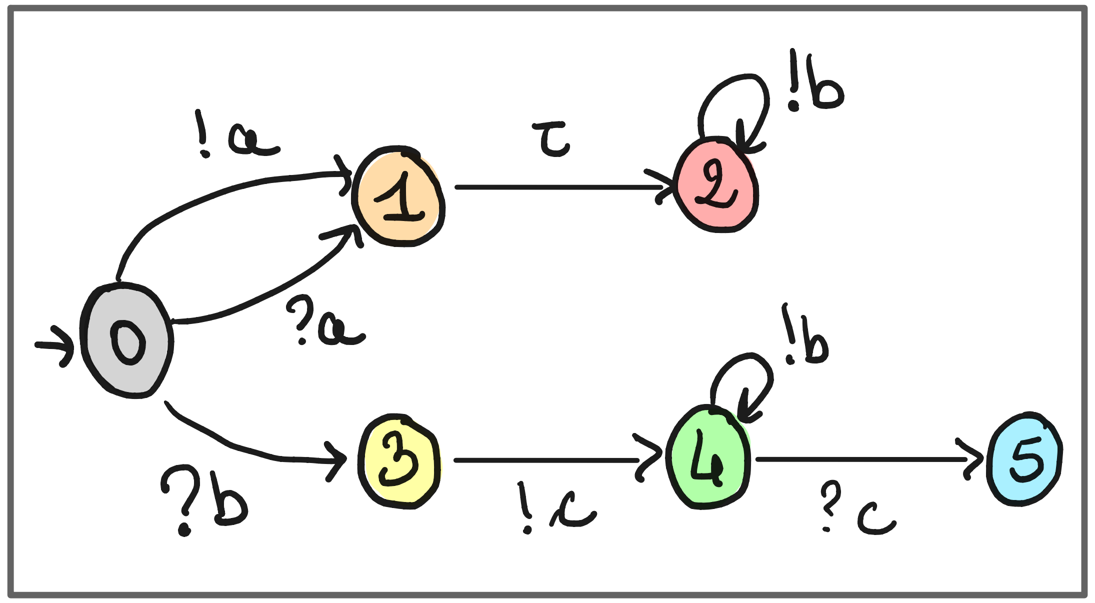




Communication by Rendez-Vous



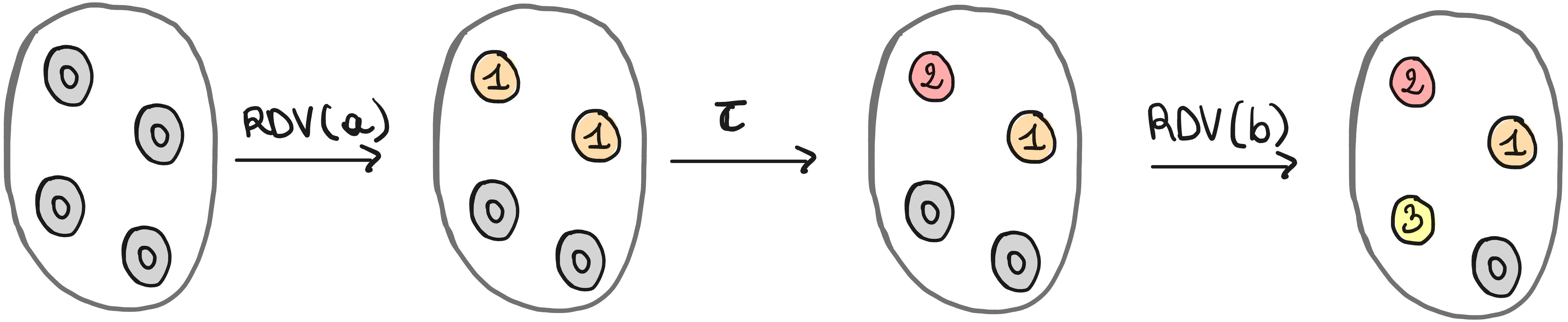
→ IMPOSSIBLE TO REACH STATES 4 AND 5.

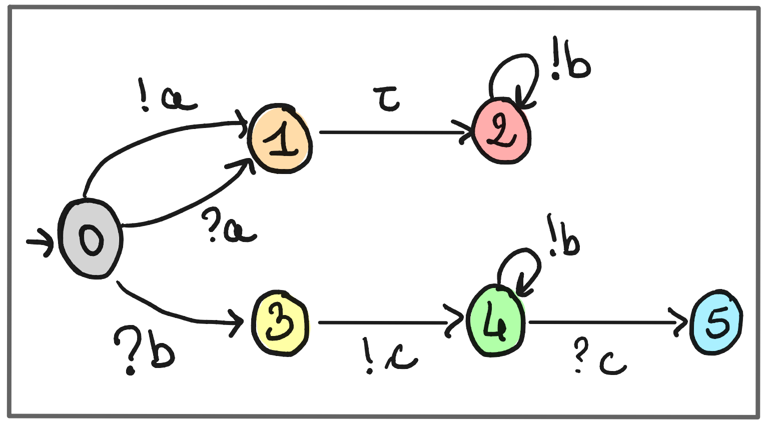


Communication by Non-Blocking Rendez-Vous

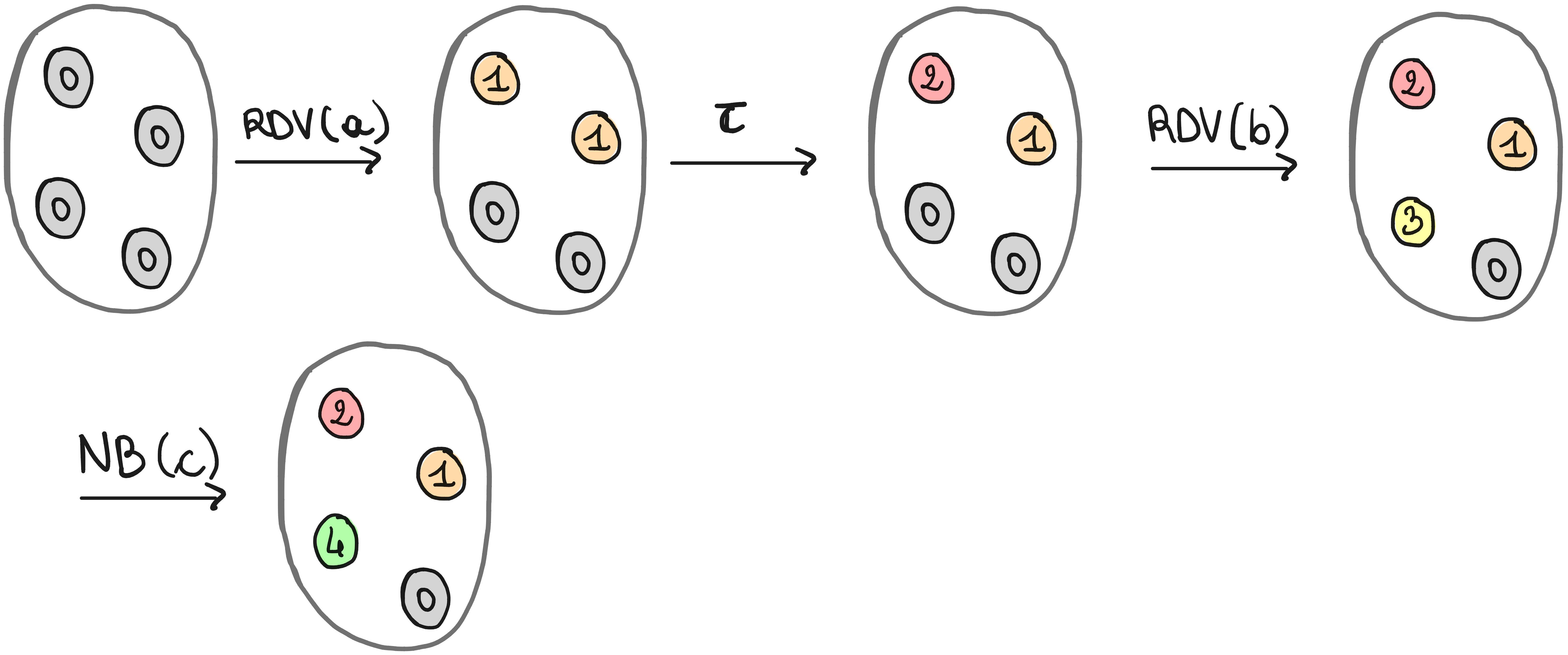
- Ex: Java Parallel Multithreads Programming
  Wait / Notify
- Rendez-Vous is no longer symmetric
- More behaviors than in the rendez-vous semantics.

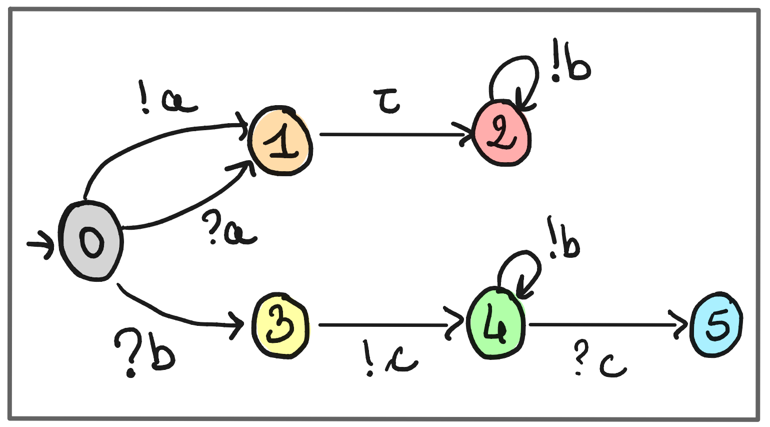
Communication by Non-Blocking Rendez-Vous



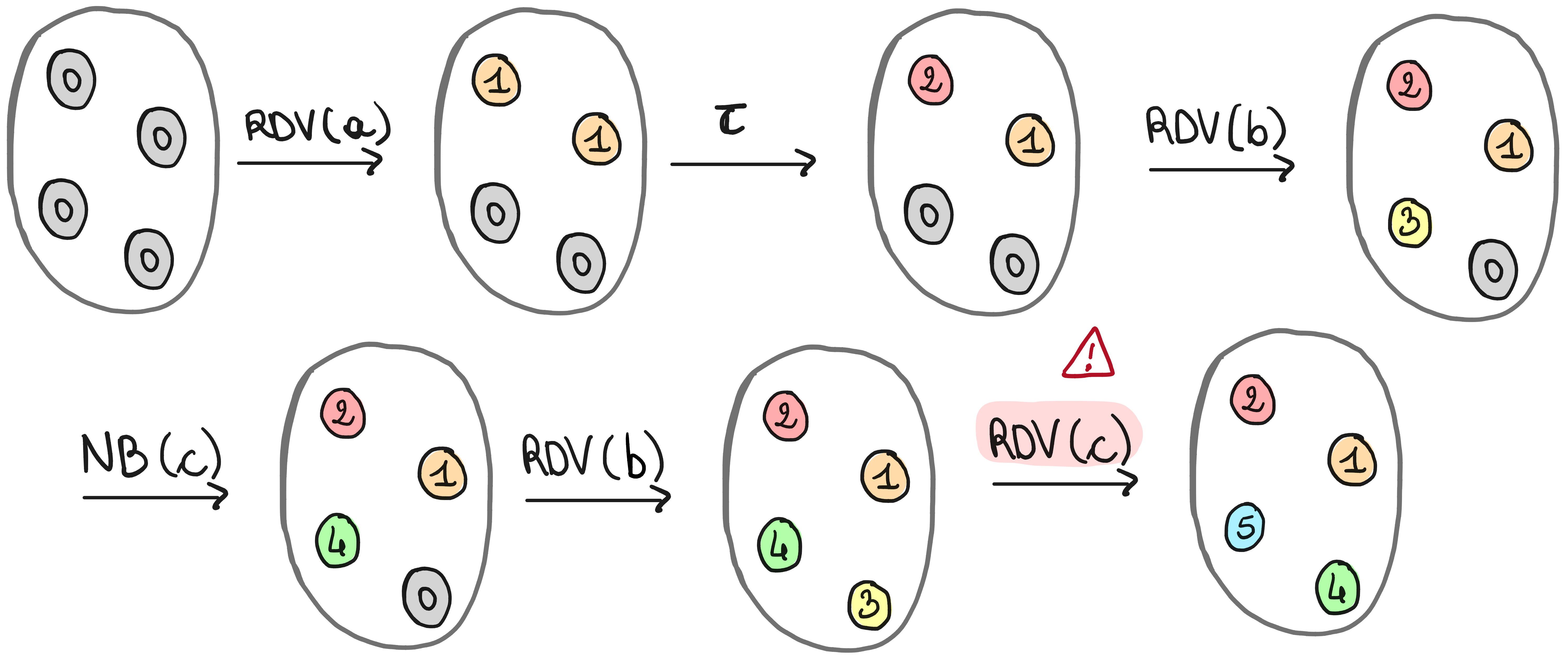


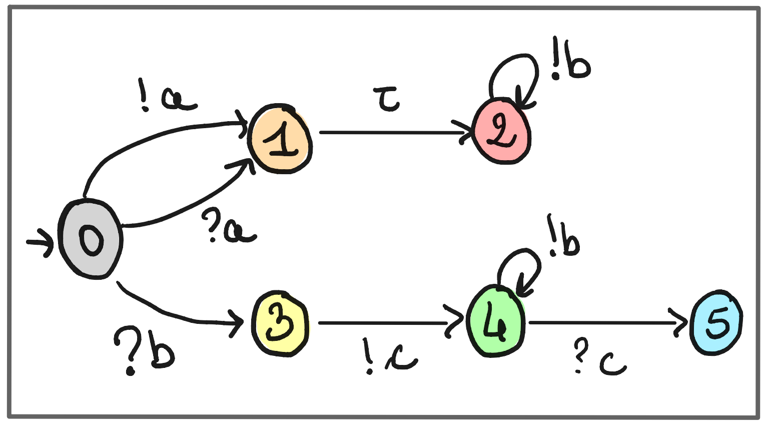
Communication by Non-Blocking Rendez-Vous



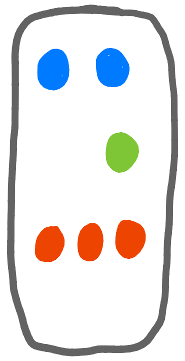
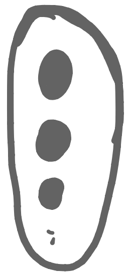
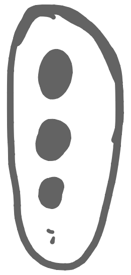
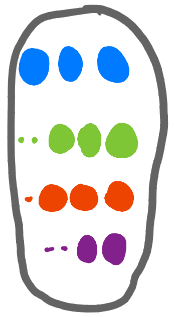
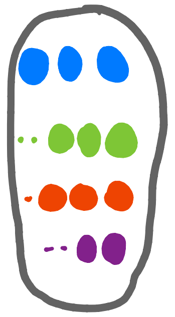
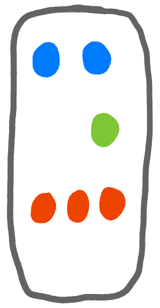


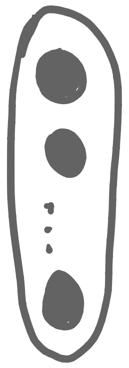
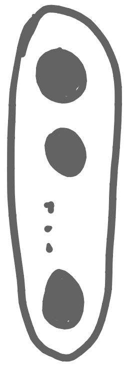
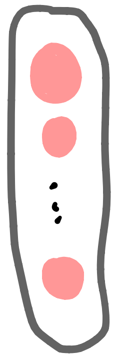
Communication by Non-Blocking Rendez-Vous





Verification Problems.

Conf-Cover: Given a protocol and a configuration  , is there  such that  $\xrightarrow{*}$  and  $\geq$  ?

Synchro: Given a protocol and a state ● , is there  such that  $\xrightarrow{*}$  ?

Results

- Rendez-Vous :
  - CONF - COVER : $\in$ Ptime [GS 92]
  - SYNCHRO : $\in$ Ptime [HS2020] [BER2021]
- Non-Blocking Rendez-Vous :
  - CONF - COVER : EXPSPACE - complete [CONCUR'23]
  - SYNCHRO : Undecidable [CONCUR'23]

Results

- Non-Blocking Rendez-Vous :
  - CONF - COVER : EXPSPACE - complete
  - SYNCHRO : Undecidable

◆ EXPSPACE - membership:

Rackoff, EXPSPACE-membership of Coverability for Vector Addition Systems with States (VASS).

▲ EXPSPACE - hardness:

Lipton, EXPSPACE-hardness of coverability for VASS.

⚠ No trivial translation with VASS

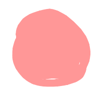
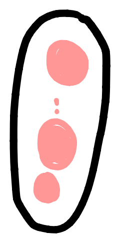
■ Undecidability :

Simulation of a 2-counters machine with tests to 0.

Why such a complexity gap?

In Rendez-Vous semantics, we have a nice property:

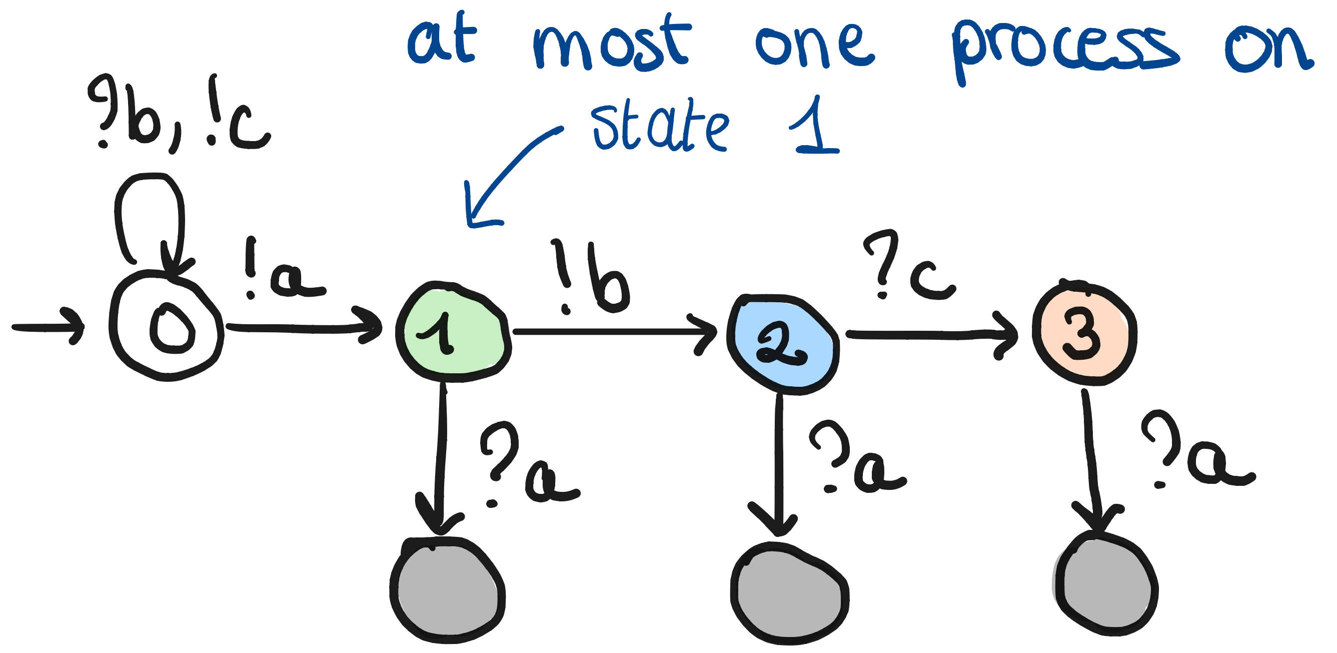
Copycat Lemma:

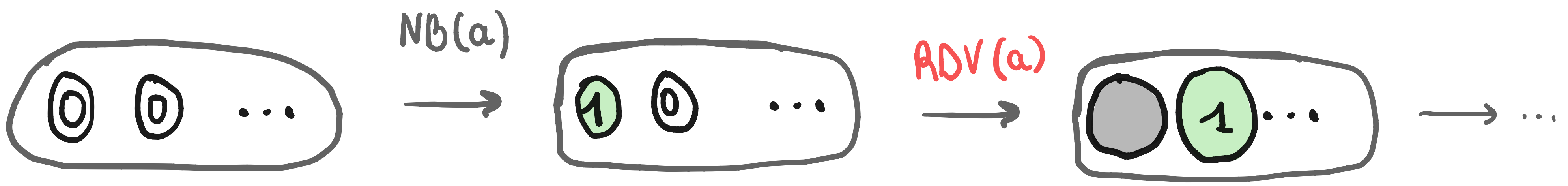
If a state  is coverable, then any configuration  is coverable

$\Rightarrow$ Conf-COVER and SYNCHRO in Ptime

Why such a complexity gap?

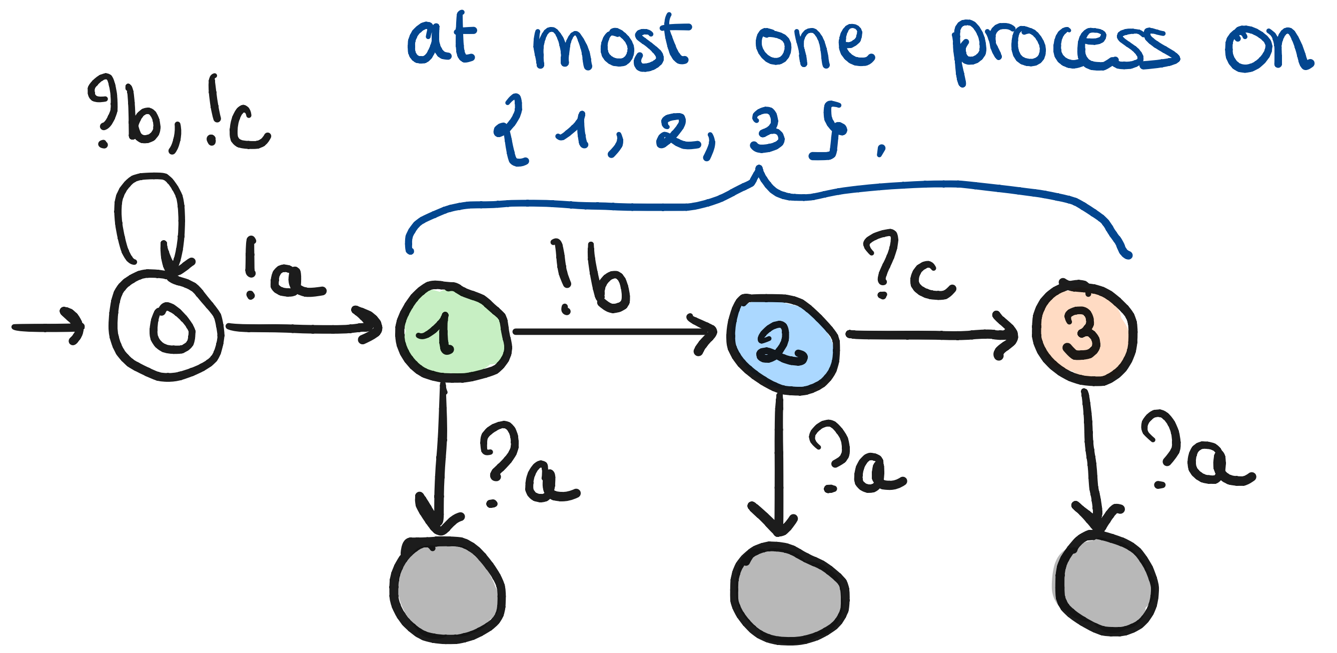
Main ingredient : with non-blocking rendez-vous, we can isolate some processes

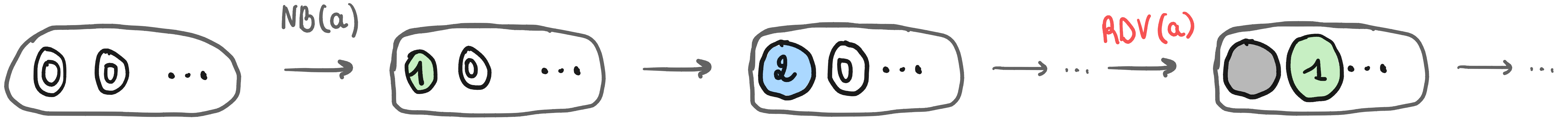




Why such a complexity gap?

Main ingredient : with non-blocking rendez-vous, we can isolate some processes





$\Rightarrow$ Conf - COVER EXPSPACE - hard

SYNCHRO undecidable

Results

- Non-Blocking Rendez-Vous :
  - CONF-COVER : EXPSPACE-complete
  - SYNCHRO : Undecidable

♦ EXPSPACE-membership :

Rackoff, EXPSPACE-membership of Coverability for Vector Addition Systems with States (VASS).

▲ EXPSPACE-hardness :

Lipton, EXPSPACE-hardness of coverability for VASS.

↳ State-COVER (covering one state)

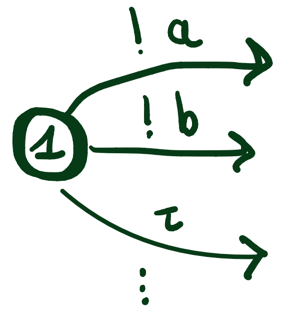
⚠ No trivial translation with VASS

■ Undecidability :

Simulation of a 2-counters machine with tests to 0.

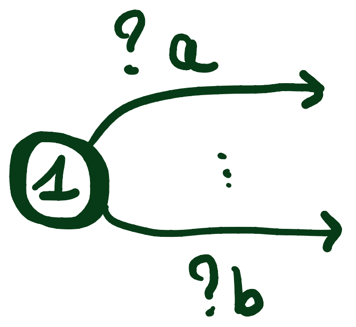
A Restriction: Wait-Only Protocol

- Protocols where each state is either:

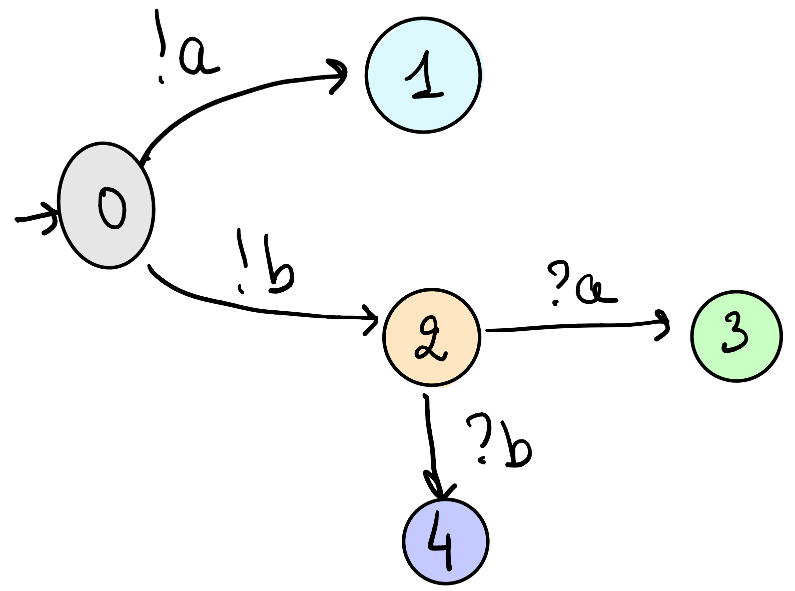


an action state

or



a waiting state

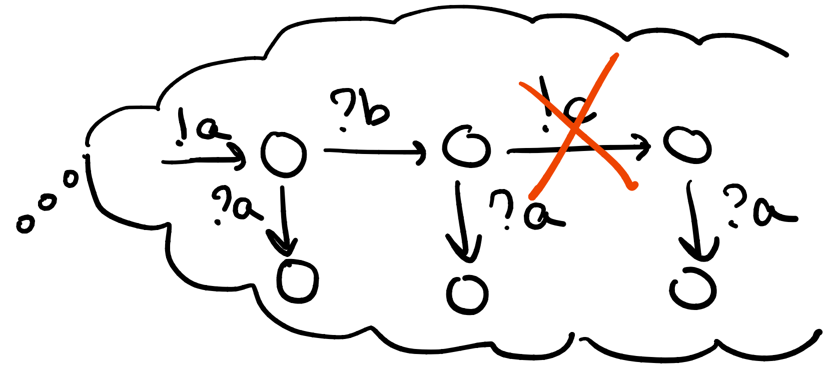


A Wait-Only Protocol

Wait - Only Protocols

* Non-Blocking Rendez-Vous : with Wait-Only Protocols
  - Conf - Cover : ~~EXPSPACE complete~~ in Ptime [CONCUR 23]
  - Synchro : Undecidable

- Why ? Isolation mecanism is less powerful



- How ? Abstraction on Configurations.
  Inductive computation until saturation

Abstract Configurations.

S , Toks

coverable states with any number of agents → S

↓

Reachable all together

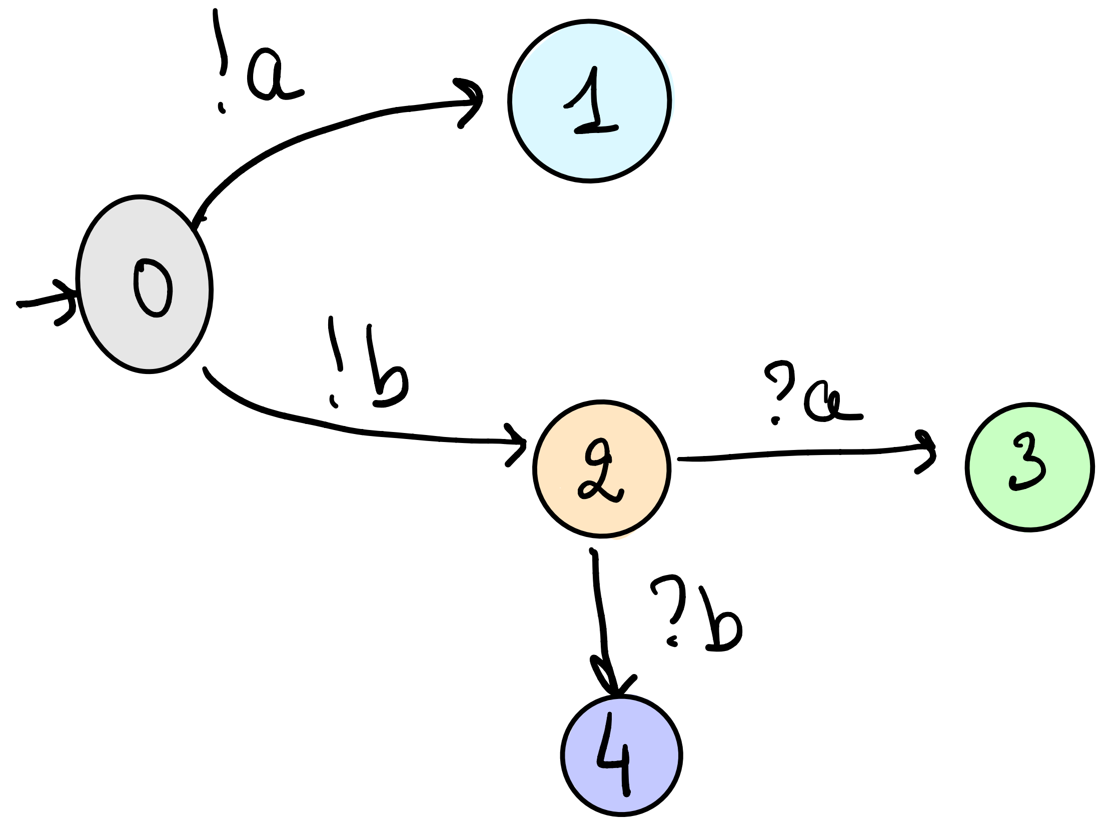
<state, message> (a token). Each state coverable by 1 agent. → Toks

↓

Not necessarily reachable all together (message)

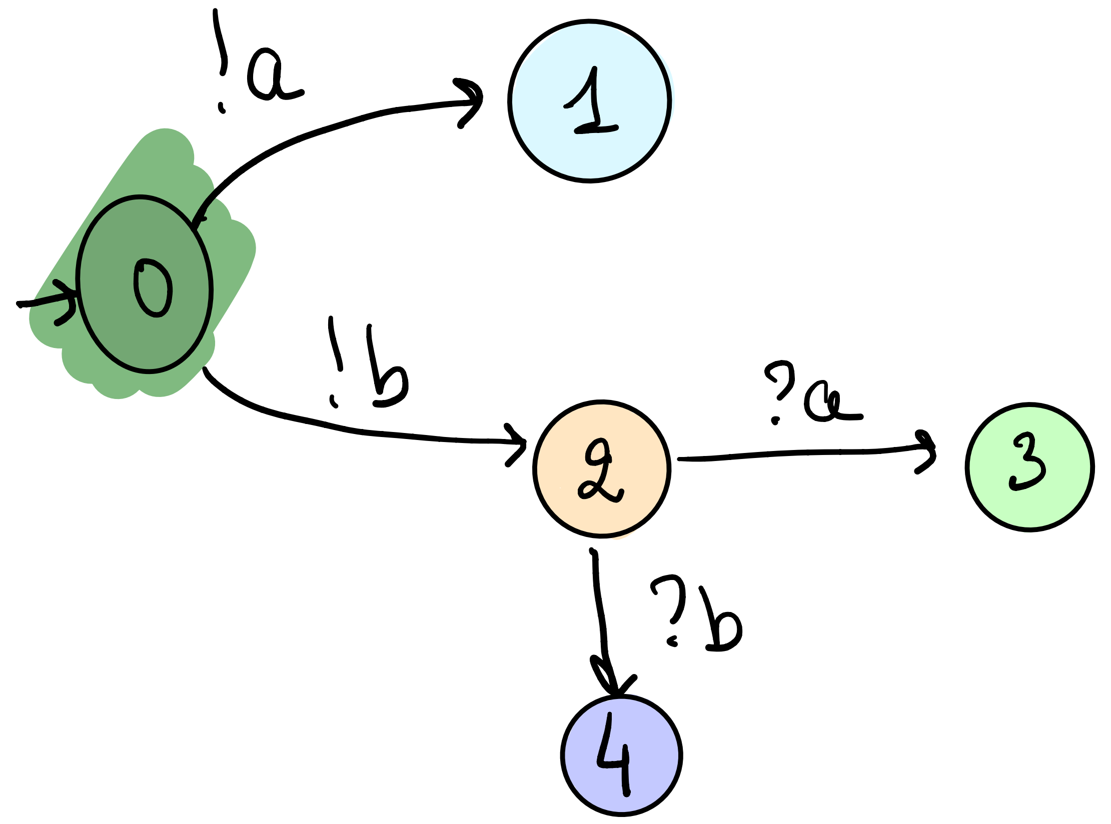
Gives us all the coverable configurations.

Computation



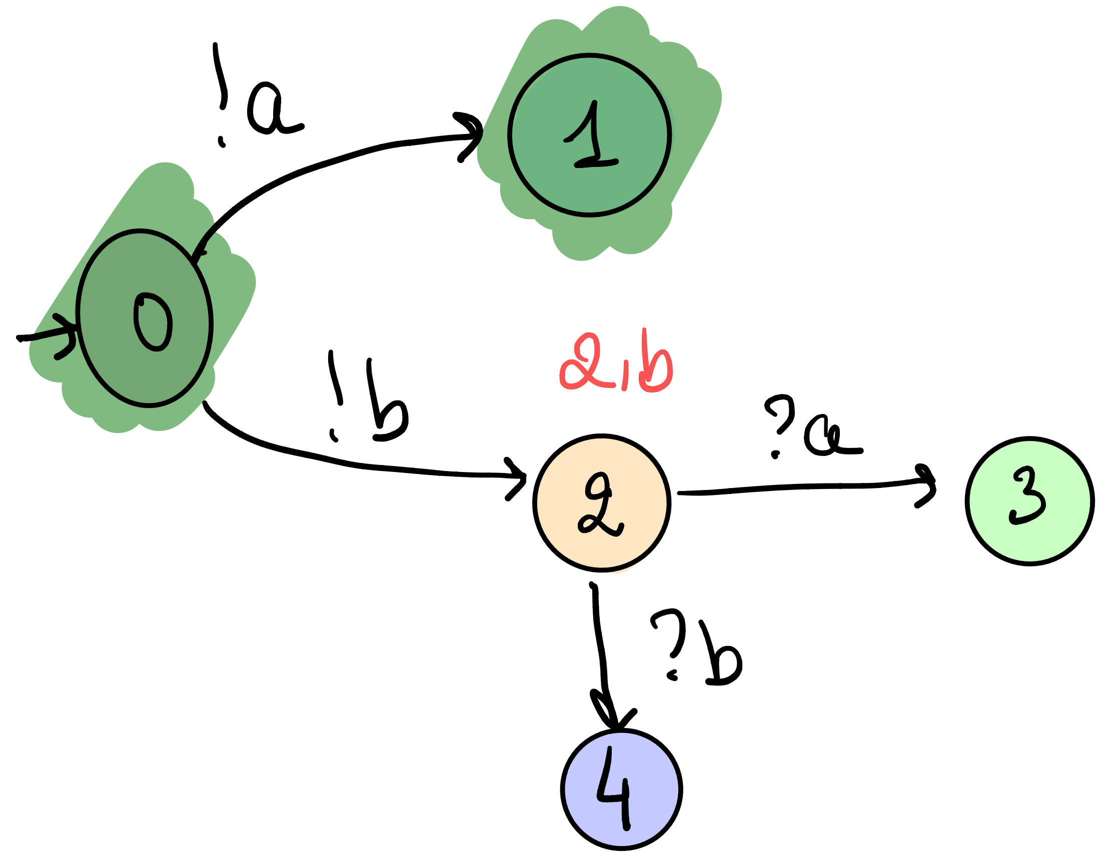
S	Toks

Computation



S	Toks
0	

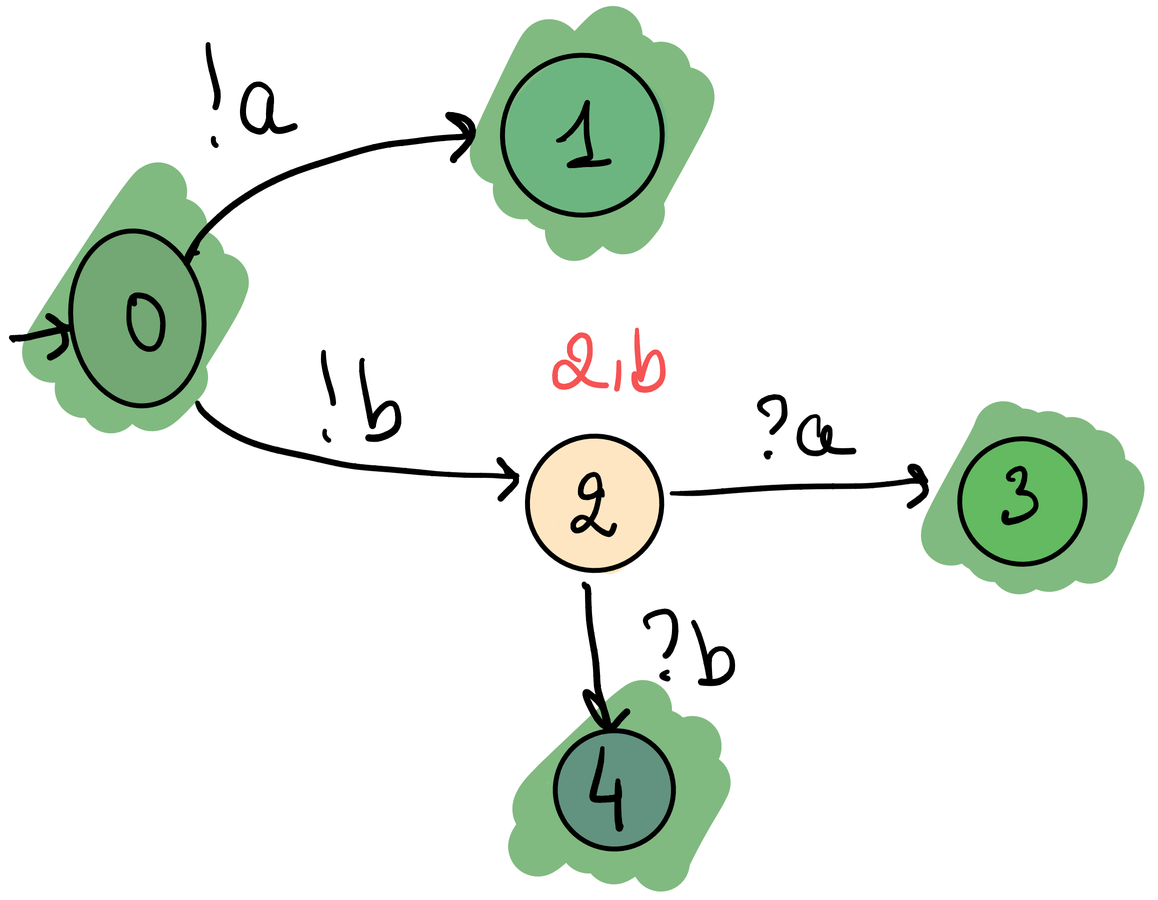
Computation



S	Toks
0	
0, 1	2, b

At most one agent on state 2, reachable through a non-blocking sending of b.

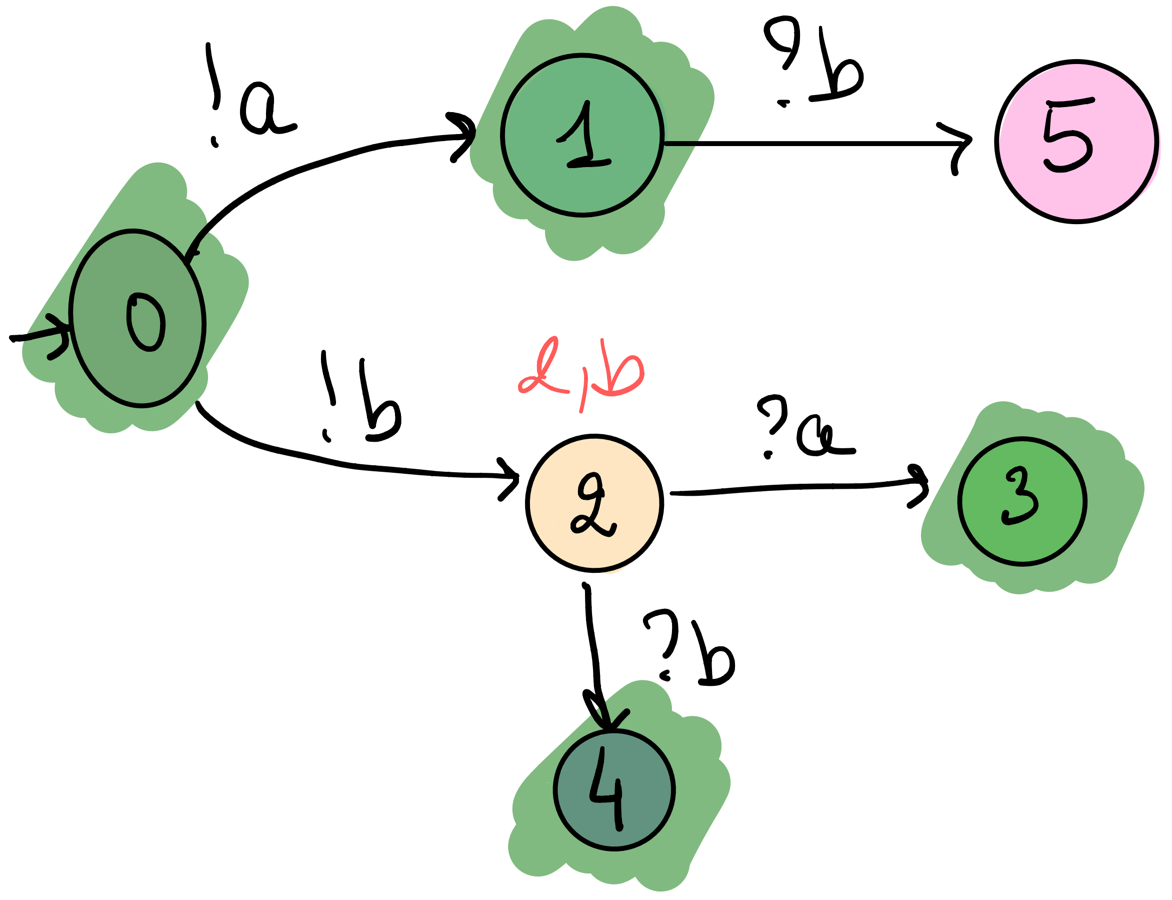
Computation



S	Toks
0	
0, 1	2, b
0, 1, 3, 4	2, b

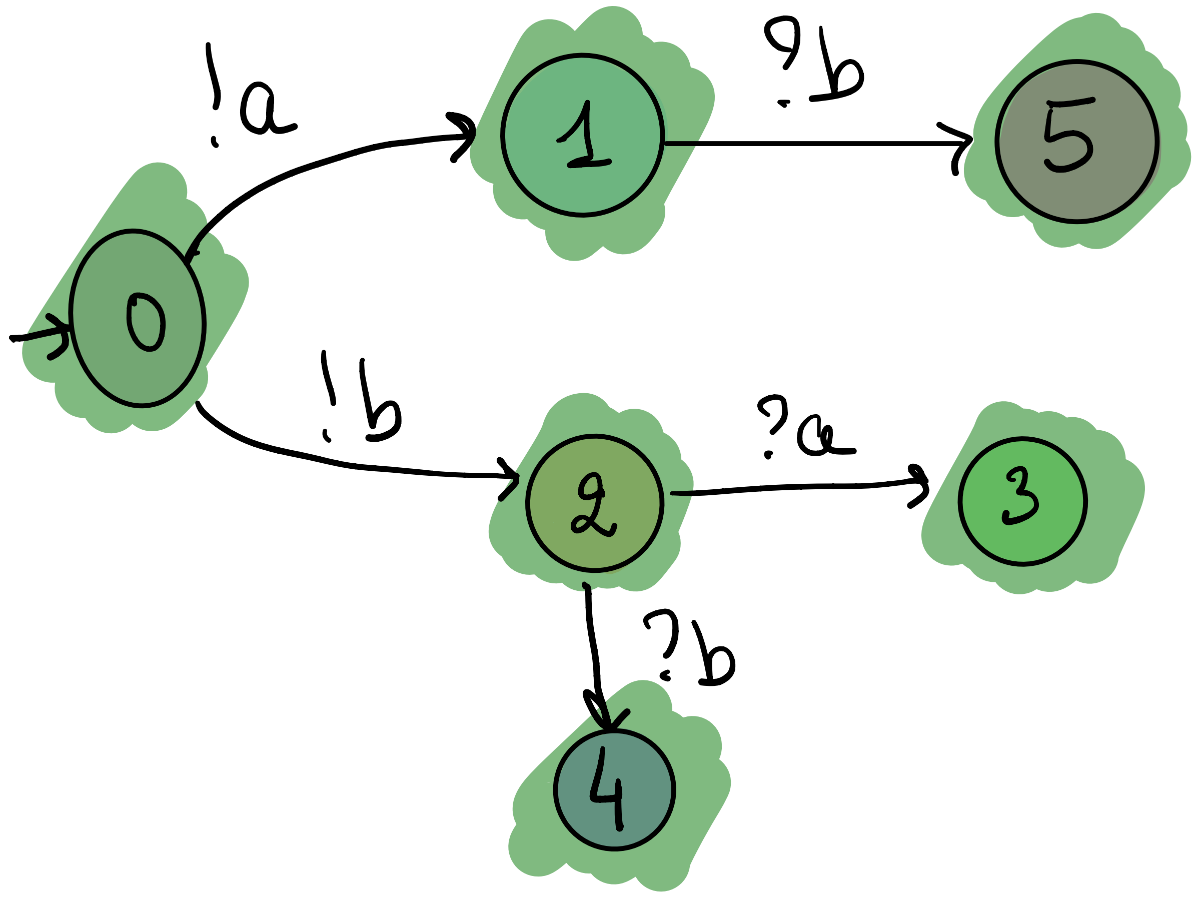
At most one agent on state 2, reachable through a non-blocking sending of b.

Computation



S	Toks
0	
0, 1, 3, 4	2, b

Computation



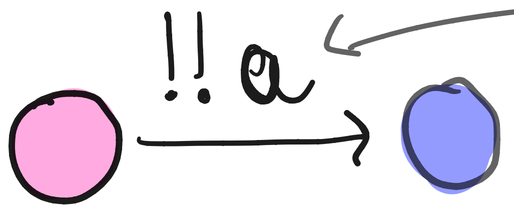
S	Toks
0	
0, 1,3,4	2, b
0, 1, 3, 4, 5, 2	

$NB(a)^K$ $RDV(b)^K$

One step closer to Java Threads Programming...

When we add **Broadcast...**

- new type of transitions in the protocol



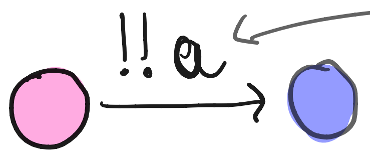
broadcast of message "a".

- when a process broadcasts "a", all the processes ready to receive the message, do so.

One step closer to Java Threads Programming...

When we add Broadcast...

- new type of transitions in the protocol



broadcast of message "a".

- when a process broadcasts "a", all the processes ready to receive the message, do so.

Without restrictions, things get worse...

Conf-COVER decidable but Ackermann-hard
[EK03] [SS13] and [ARZ15]

Wait-Only with Broadcasts

Conf-COVER is PSPACE-complete

State-COVER is P-complete

Conf-COVER with a configuration equals to one single state.

Conclusion

- New semantics leading to an important complexity gap compared to the rendez-vous semantics.
- Restriction allowing to regain a Ptime algorithm for the conf-COVER problem
- New restriction allowing to regain decidability for the SYNCHRO problem ?

Thank you everyone for your attention