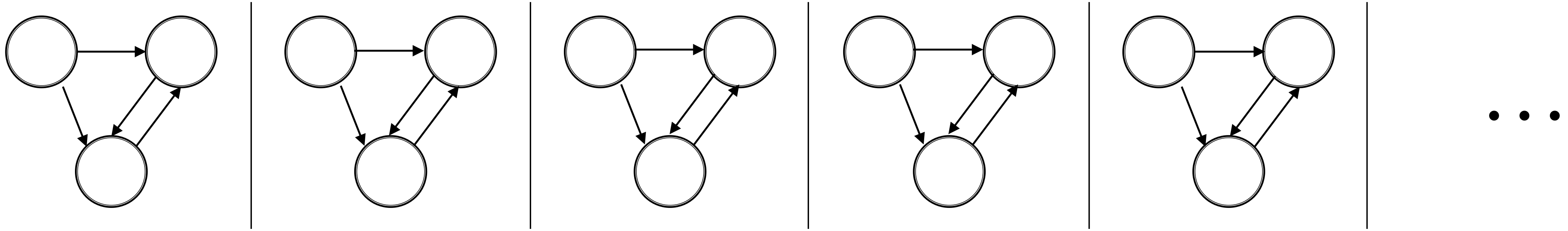
Checking Presence Reachability Properties on Parameterized Shared-Memory Systems

Nicolas Waldburger
PhD supervisors: Nathalie Bertrand, Nicolas Markey, Ocan Sankur
Séminaire MOVE, 29/07/2023

Parameterized verification



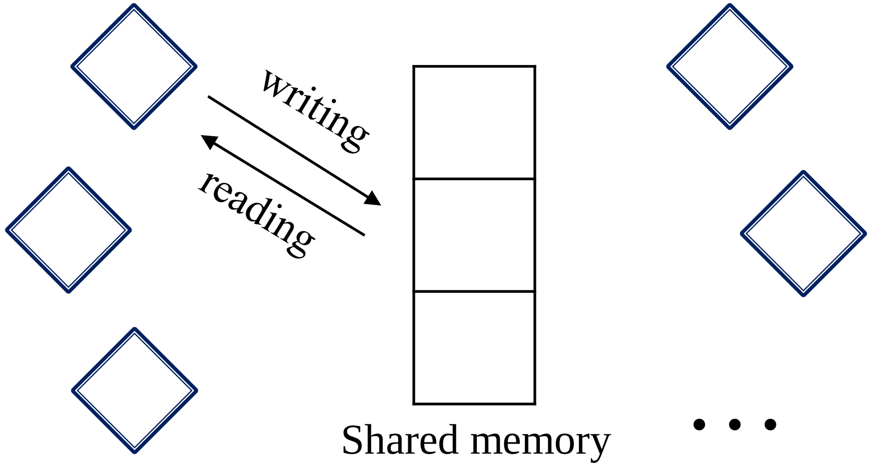
- *Arbitrary* number of processes
- Processes are *identical* agents
- No identifiers: processes are *anonymous*
- Modelled by a single, common *finite automaton*

1. Model inspired from: Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems*. Journal of the ACM, 2016

Shared-memory systems

Two models in this talk:

- Simple model: shared-memory systems with finite memory
- More complex model: round-based shared-memory systems

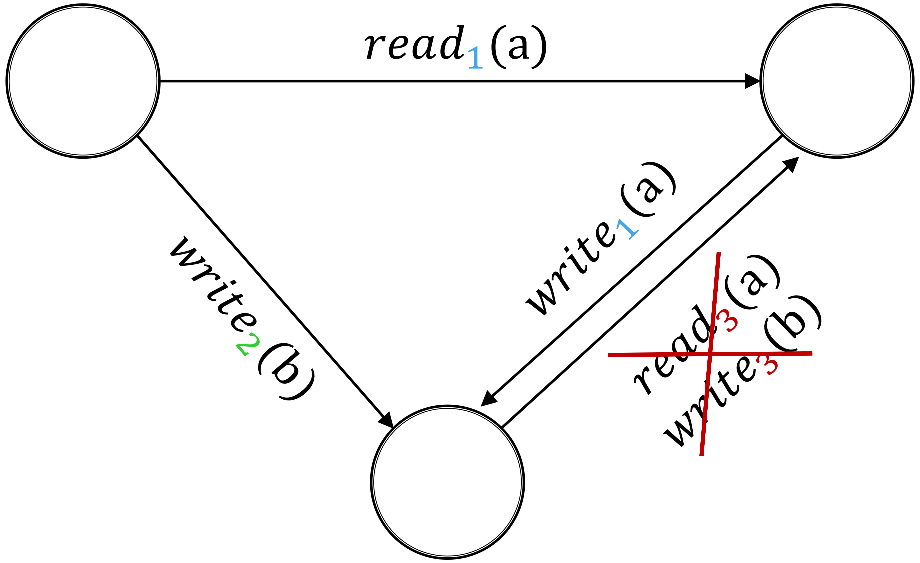


A model for shared-memory systems[1]

Finite number of shared registers, each register has a value from finite set of symbols Σ

1	a
2	b
3	d_0

Registers are *initialized* to value d_0

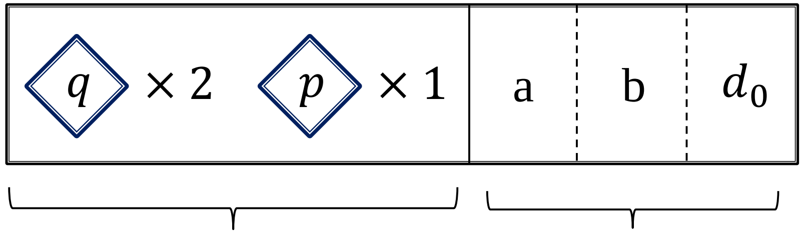


No atomic read/write combinations

1. Model inspired from: Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems.* Journal of the ACM, 2016

Semantics

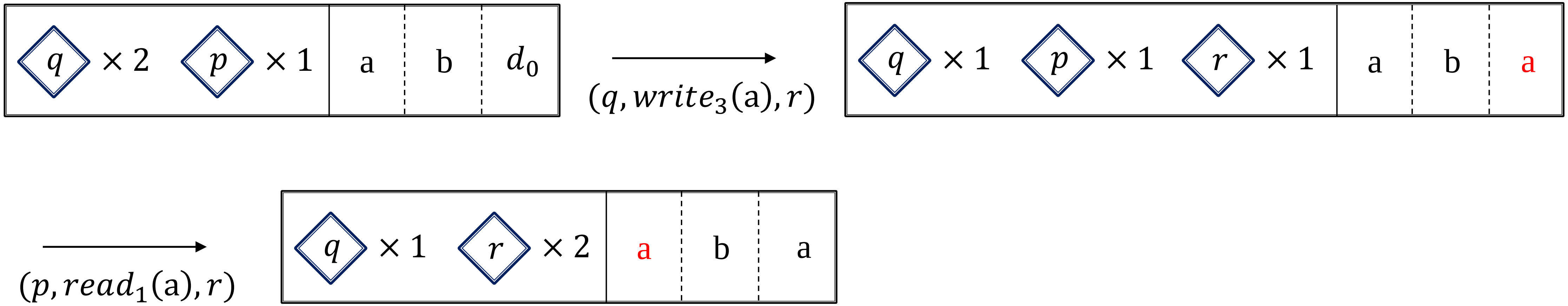
A configuration:



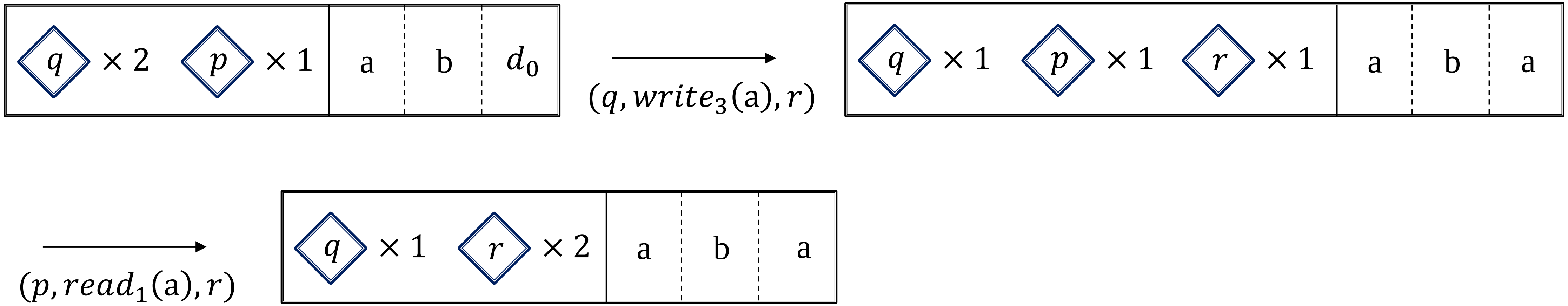
How many process are on each state

Content of the registers

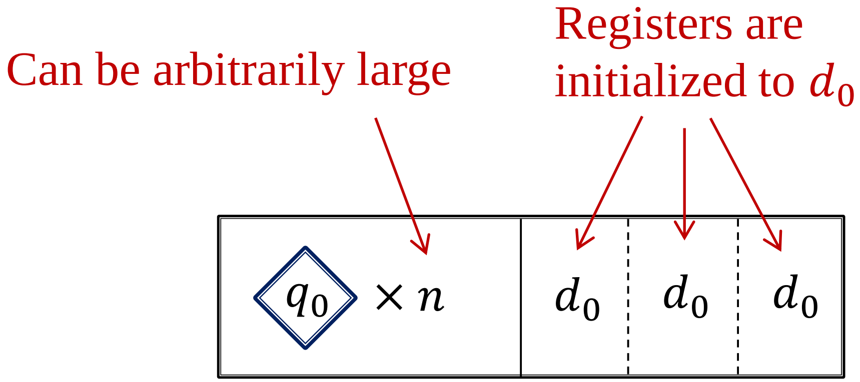
Semantics



Semantics

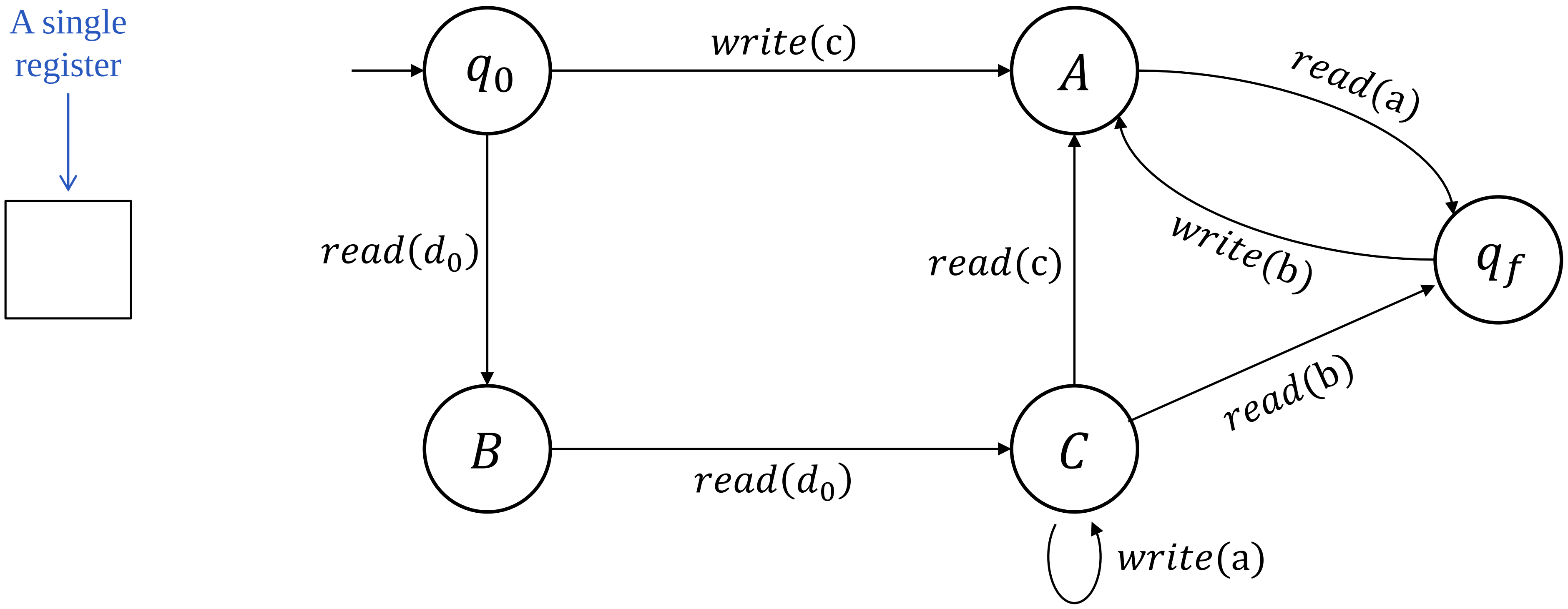


Initial configurations:

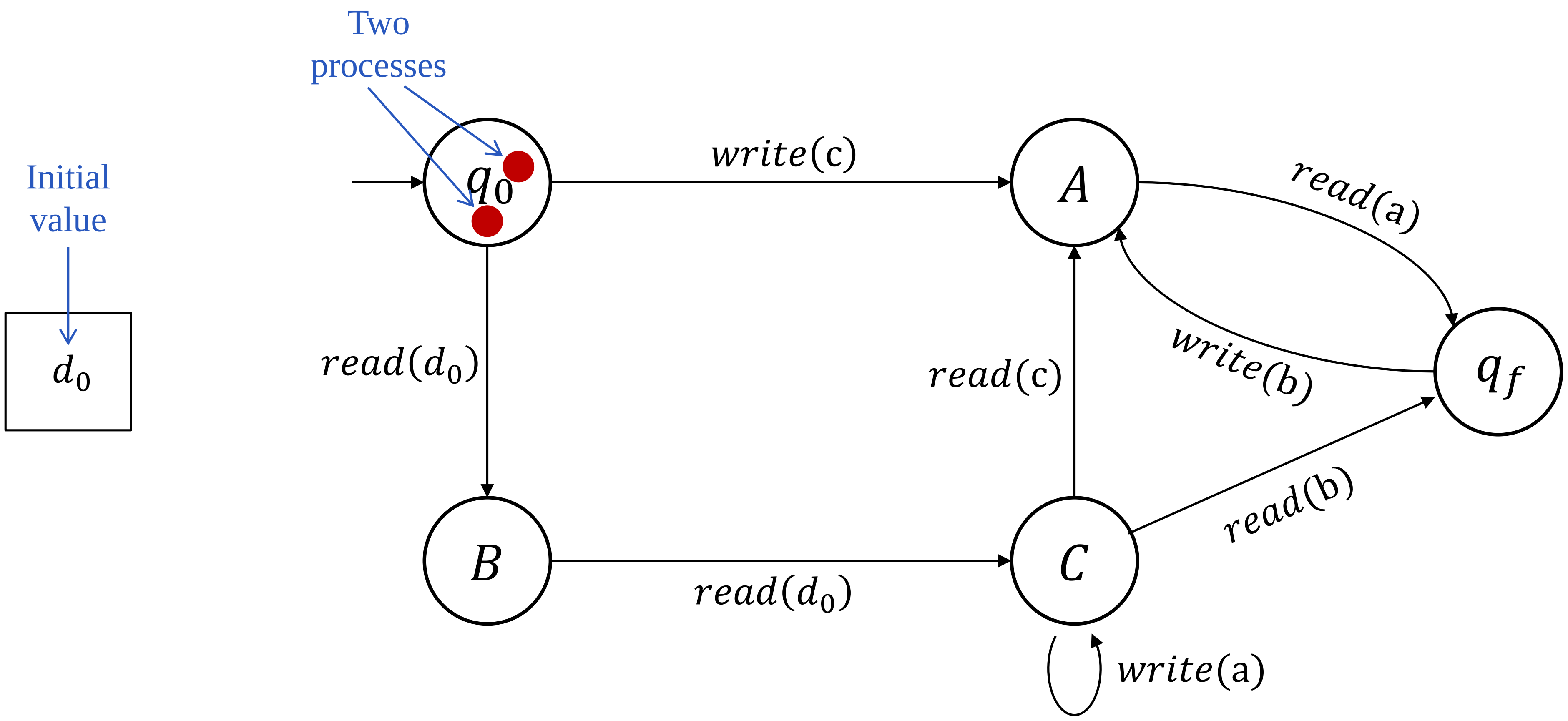


with $n \geq 1$ and q_0 the initial state

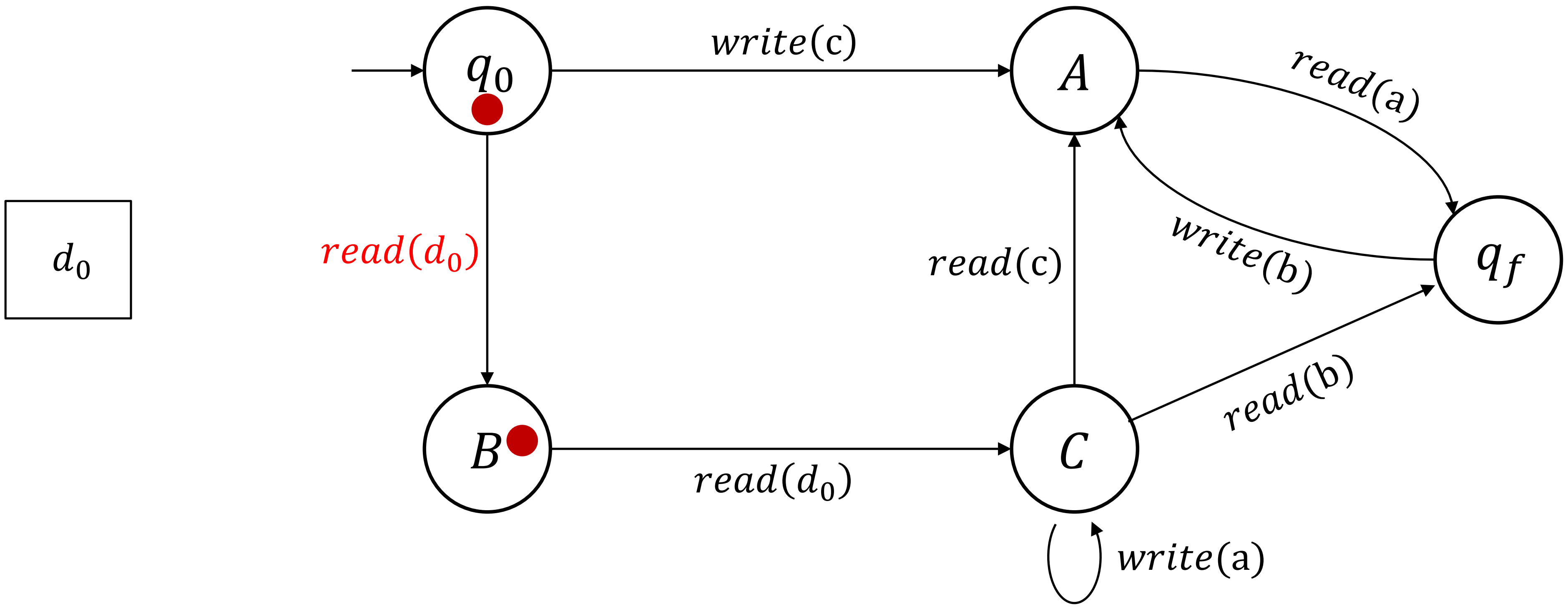
A small example



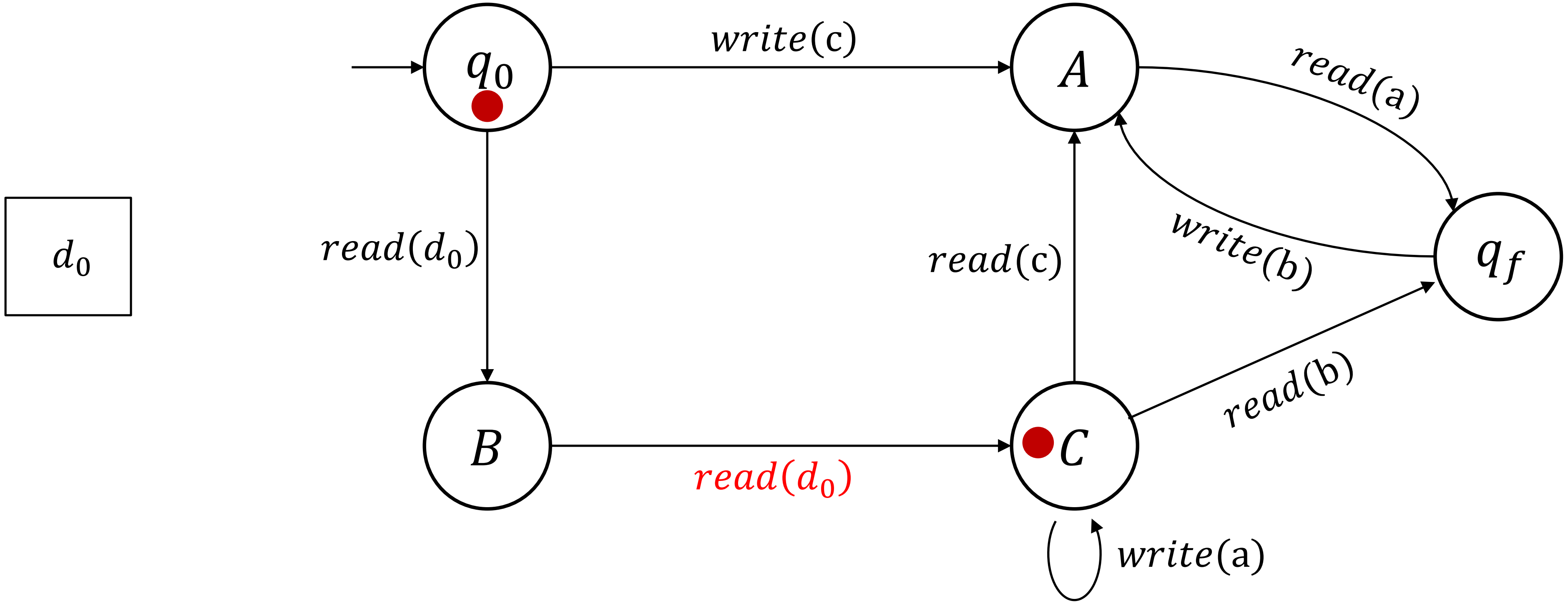
A small example



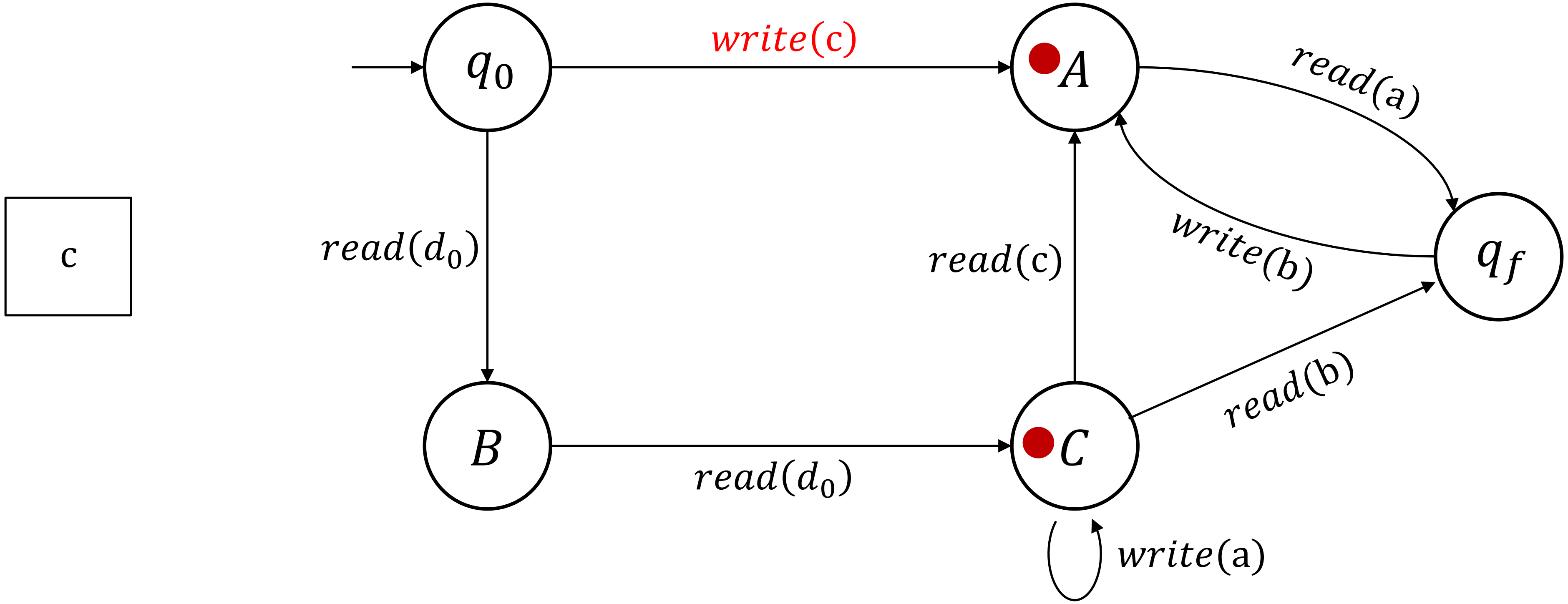
A small example



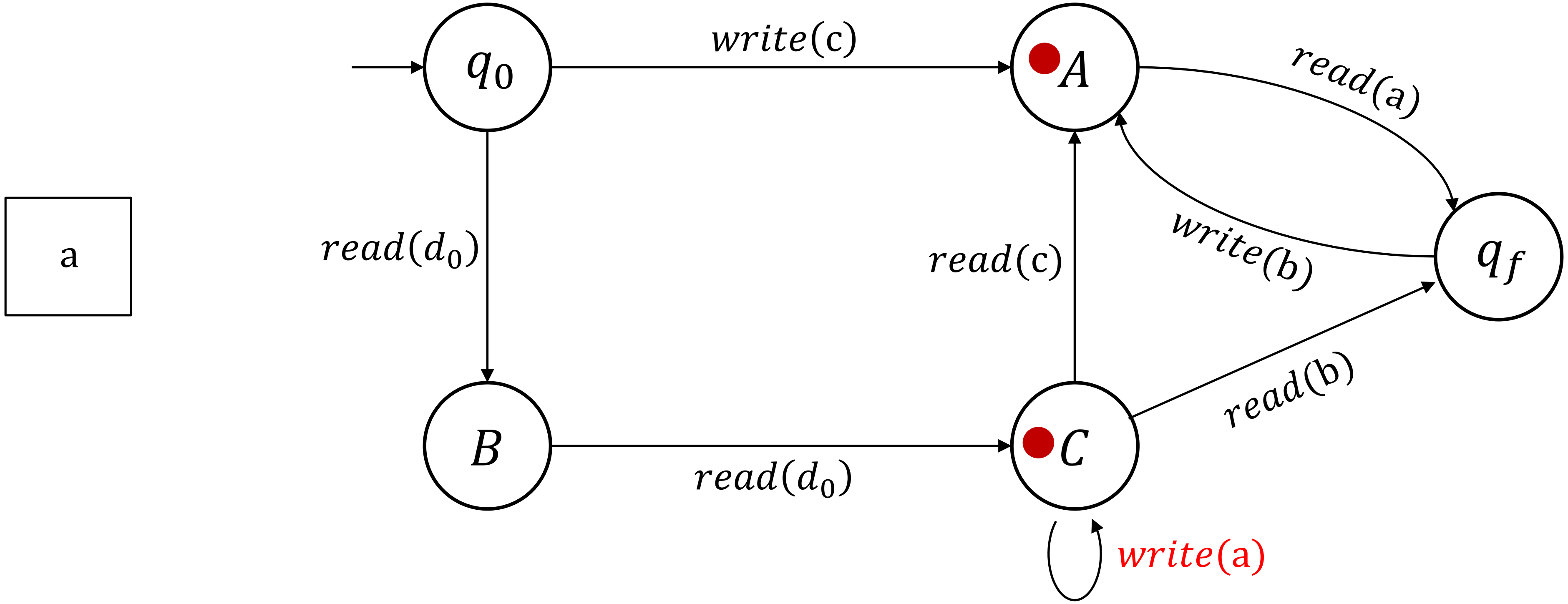
A small example



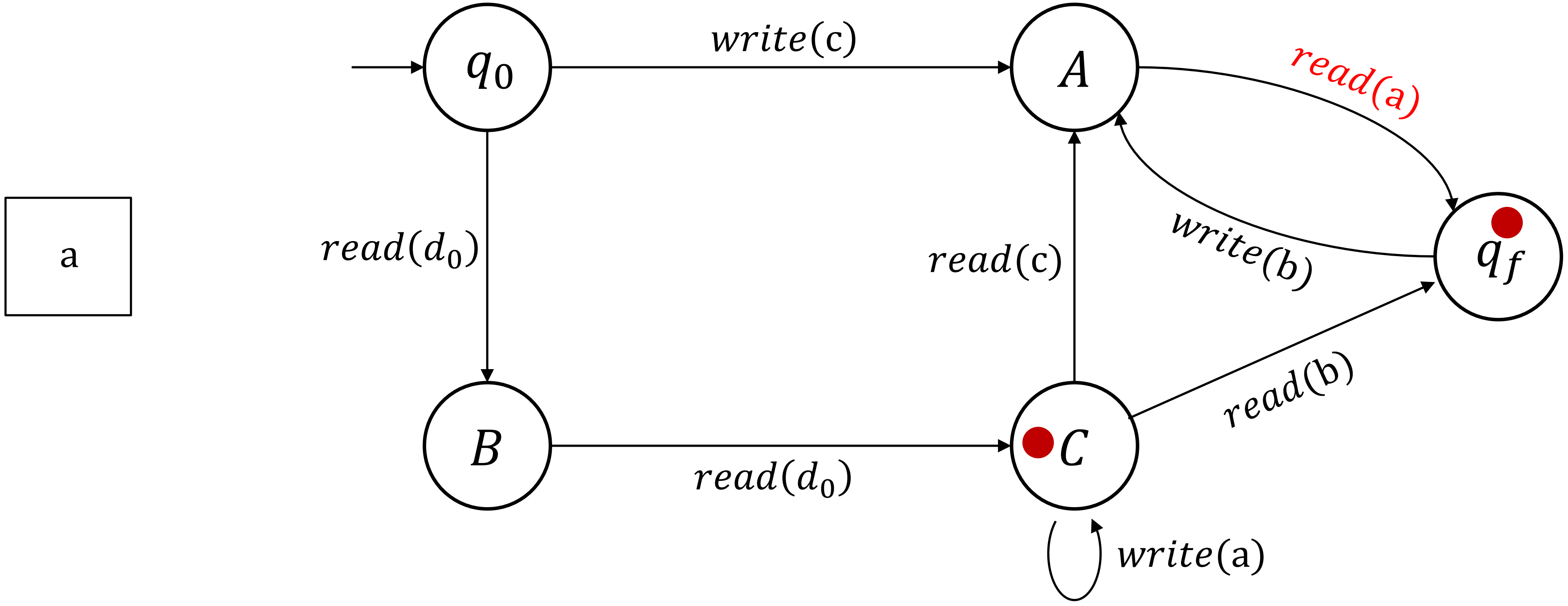
A small example



A small example



A small example



q_f is covered ✓

Reachability problems

An initial configuration

A least one process on q_f: “error state”

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \gamma(q_f) > 0$?

Parameterized: arbitrarily many processes

Execution

Reachability problems

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma(q_f) > 0$?

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

All processes
"synchronize" on q_f

Reachability problems

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma(q_f) > 0$?

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

PRP[2]: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma \vDash \phi$?

with $\phi \in \mathcal{B}(\{\#q = 0, \#q > 0\}, \{\mathbf{reg}_i = d, \mathbf{reg}_i \neq d\})$

Presence Reachability Problem

$\#q$ = number of processes on q

2. Inspired from CRP in: Delzanno, G., Sangnier, A., Traverso, R., Zavattaro, G.: *On the Complexity of Parameterized Reachability in Reconfigurable Broadcast Networks*, Tech. Rep., 2012

Reachability problems

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma(q_f) > 0$?

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

PRP[2]: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma \vDash \phi$?

↑
Presence Reachability Problem

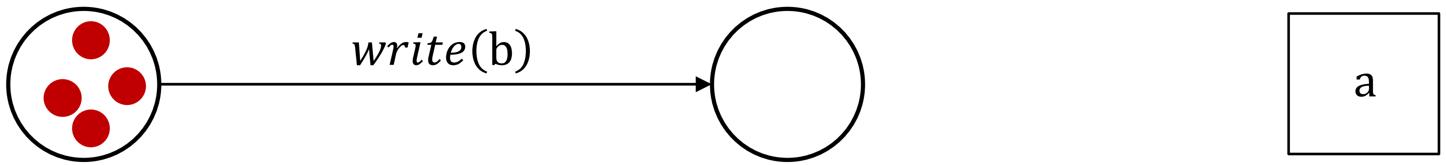
with $\phi \in \mathcal{B}(\{\#q = 0, \#q > 0\}, \{\mathbf{reg}_i = d, \mathbf{reg}_i \neq d\})$

Examples: $\phi =$ ``$\#q_f > 0''$ (COVER),
$\phi =$ ``$\bigwedge_{q \neq q_f} \#q = 0''$ (TARGET)
$\phi =$ ``$(\#q_1 > 0)\ \vee\ ([\#q_2 = 0]\ \wedge\ [\mathbf{reg}_1 = d_0])''$

2. Inspired from CRP in: Delzanno, G., Sangnier, A., Traverso, R., Zavattaro, G.: *On the Complexity of Parameterized Reachability in Reconfigurable Broadcast Networks*, Tech. Rep., 2012

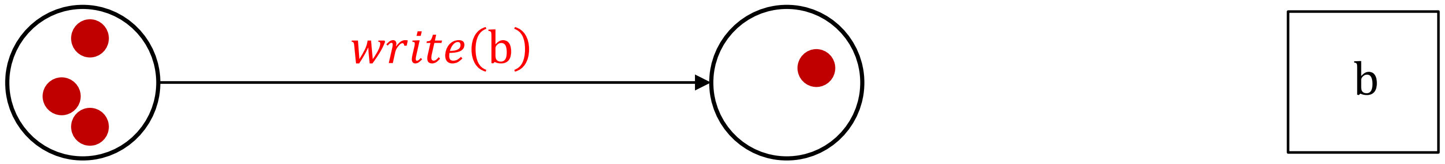
Monotonicity

A process may "copy" the behavior of another process on the same state.



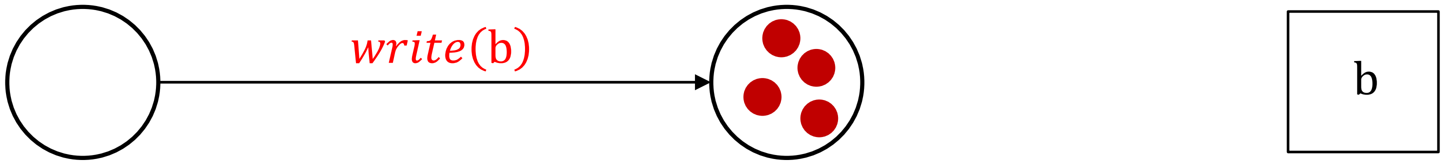
Monotonicity

A process may "copy" the behavior of another process on the same state.



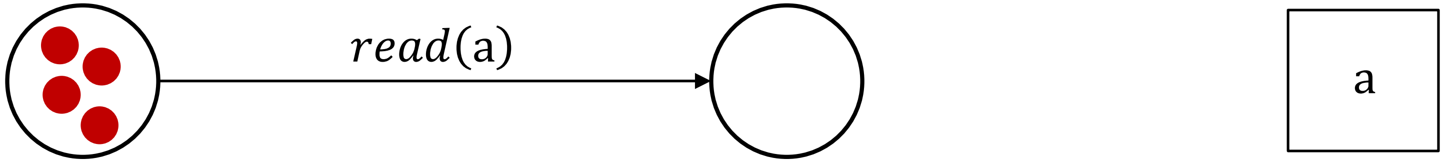
Monotonicity

A process may “copy” the behavior of another process on the same state.



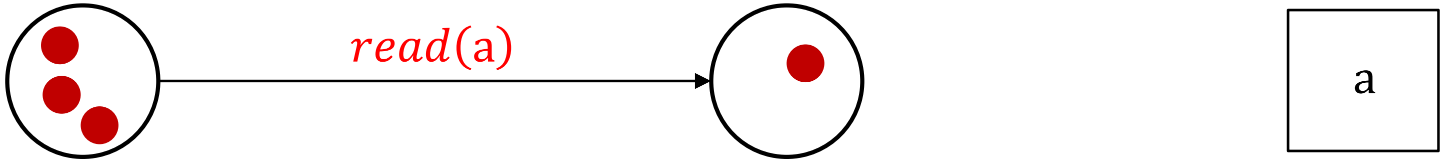
Monotonicity

A process may "copy" the behavior of another process on the same state.



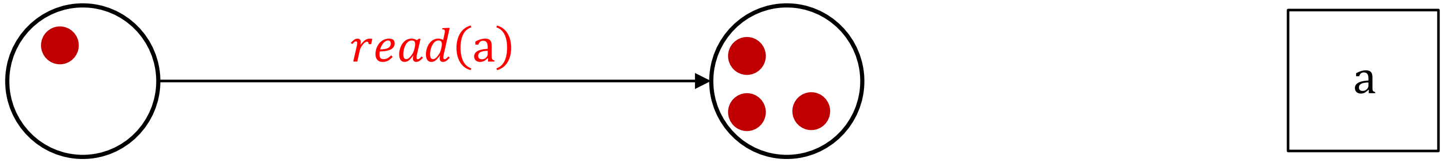
Monotonicity

A process may "copy" the behavior of another process on the same state.

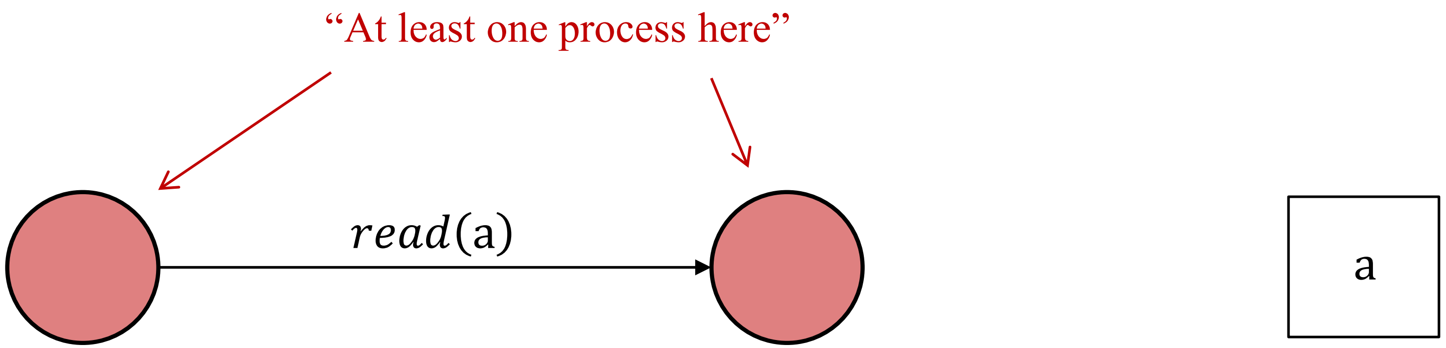


Monotonicity

A process may "copy" the behavior of another process on the same state.



Monotonicity



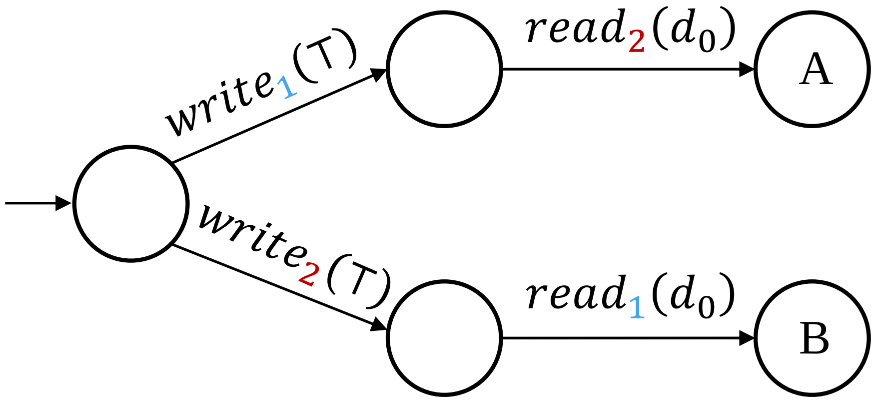
Abstraction: remember whether there is at least one process on a given state.

Sound and *Complete* for PRP because of monotonicity property

NP-completeness of COVER

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \gamma(q_f) > 0$?

1	d_0
2	d_0



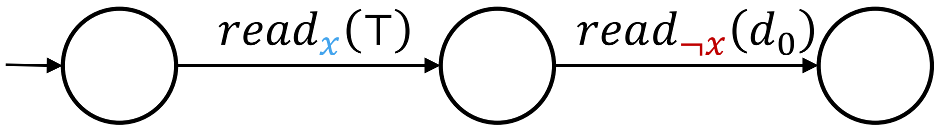
NP-completeness of COVER

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma(q_f) > 0$?

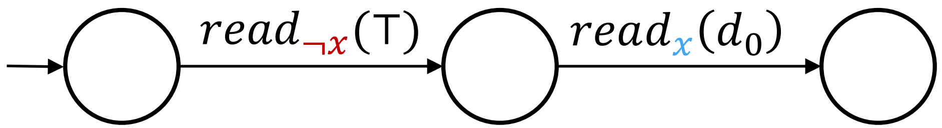
Reduction from 3-SAT:

x	d_0
$\neg x$	d_0

Check x:



Check $\neg x$:



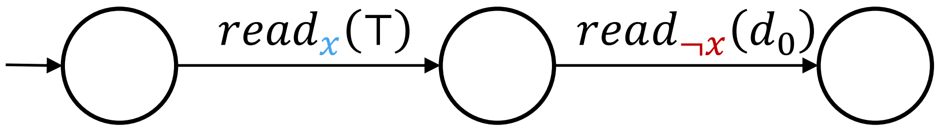
NP-completeness of COVER

COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma(q_f) > 0$?

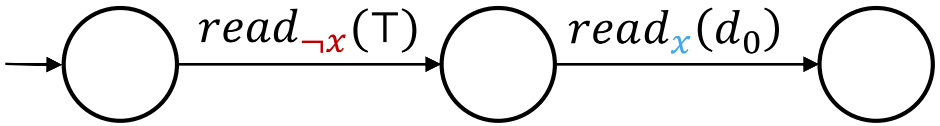
Reduction from 3-SAT:

x	d_0
$\neg x$	d_0

Check x:



Check $\neg x$:



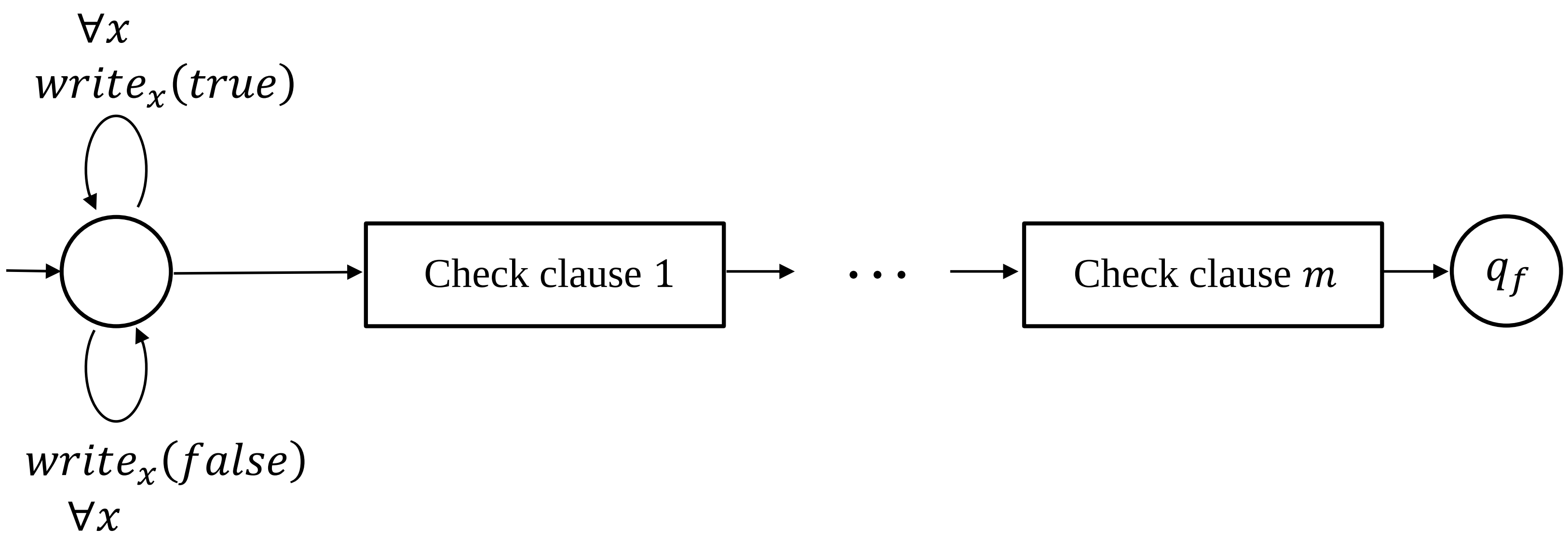
Directly relies on initialization of registers!

COVER drops down to PTIME when the registers are not initialized (applying a simple saturation technique).

TARGET when registers are not initialized

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

TARGET is still NP-complete when registers are not initialized. Reduction from 3-SAT:



TARGET with a single register

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

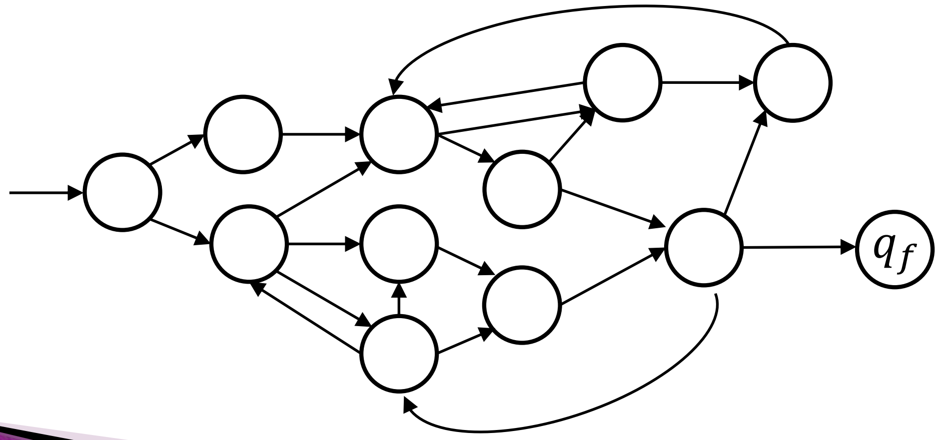
4. Fournier, P.: *Parameterized verification of networks of many identical processes*. PhD thesis, 2015

TARGET with a single register

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (q_f may be reached from some configuration containing the state).



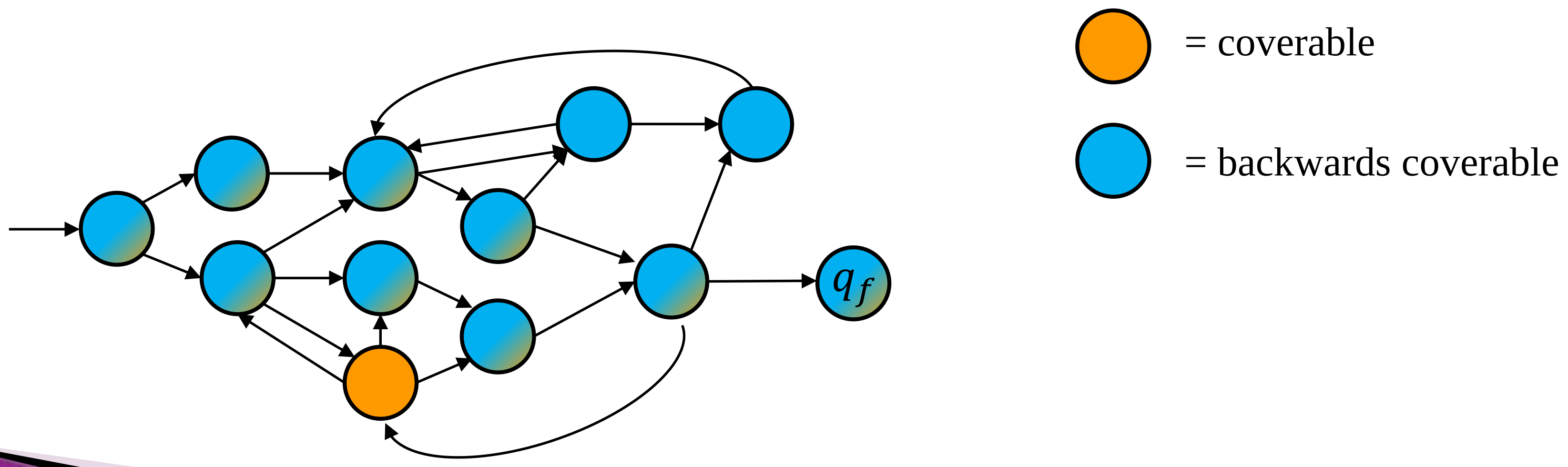
4. Fournier, P.: *Parameterized verification of networks of many identical processes*. PhD thesis, 2015

TARGET with a single register

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (q_f may be reached from some configuration containing the state).



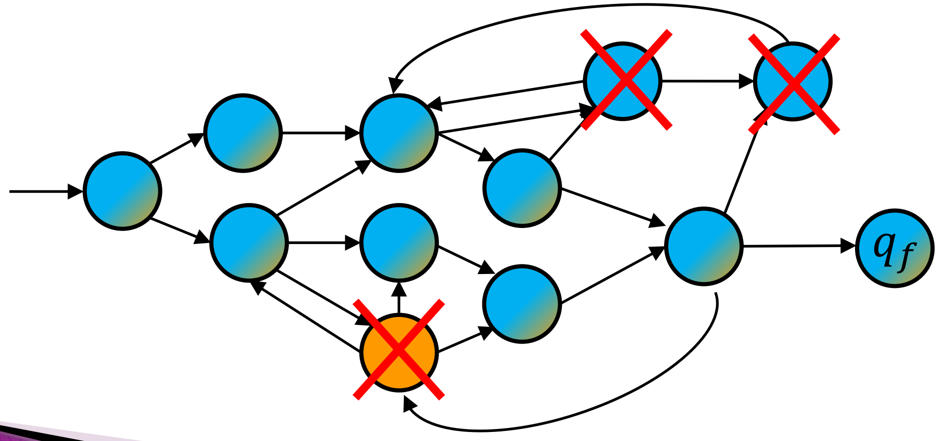
4. Fournier, P.: *Parameterized verification of networks of many identical processes.* PhD thesis, 2015

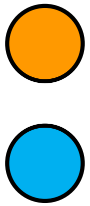
TARGET with a single register

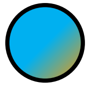
TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (q_f may be reached from some configuration containing the state).





Iteratively remove all states that are not 

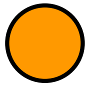
4. Fournier, P.: *Parameterized verification of networks of many identical processes*. PhD thesis, 2015

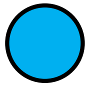
TARGET with a single register

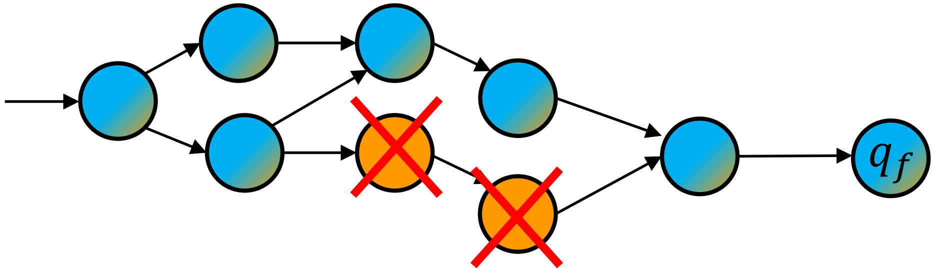
TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

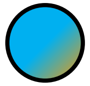
TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (q_f may be reached from some configuration containing the state).

 = coverable

 = backwards coverable



Iteratively remove all states that are not 

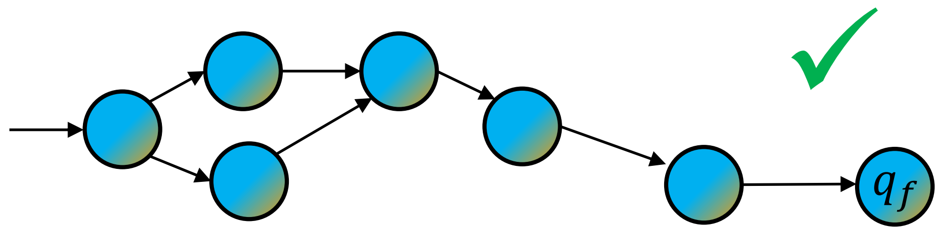
4. Fournier, P.: *Parameterized verification of networks of many identical processes.* PhD thesis, 2015

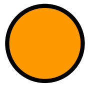
TARGET with a single register

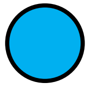
TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

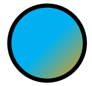
TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (q_f may be reached from some configuration containing the state).




= coverable


= backwards coverable

Iteratively remove all states that are not 

4. Fournier, P.: *Parameterized verification of networks of many identical processes.* PhD thesis, 2015

TARGET with a single register

TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \forall q \neq q_f,\ \gamma(q) = 0$?

TARGET is PTIME when only one register.
One can reduce the problem to the case when the register is not initialized.
Algorithm inspired from broadcast protocols[4].

Compute *coverable states* (the state can be covered from initial configurations)
and *backwards coverable states* (q_f may be reached from some configuration containing the state).

The algorithm is generalizable to PRP when the formula is in Disjunctive Normal Form (DNF).

DNF-PRP: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \to^* \gamma\ ,\ \ \gamma \vDash \phi,$

ϕ in DNF: $\phi = \bigvee_i \left(t_{i,1} \wedge t_{i,2} \wedge \cdots \wedge t_{i,m_i}\right),$

$t_{i,j} \in \{\#q = 0, \#q > 0\} \cup \{\mathbf{reg}_i = d, \mathbf{reg}_i \neq d\}$

4. Fournier, P.: *Parameterized verification of networks of many identical processes.* PhD thesis, 2015

Summary of complexity results[5]

	COVER	TARGET	DNF-PRP	PRP
General case	NP-complete	NP-complete	NP-complete	NP-complete
Not initialized	PTIME-complete	NP-complete	NP-complete	NP-complete
One register	PTIME-complete	PTIME-complete	PTIME-complete	NP-complete

5. W: *Checking Presence Reachability Properties on Parameterized Shared-Memory Systems*, MFCS23

Round-based shared-memory systems

A motivating example

Binary consensus problem:
Make all processes agree on a common value, each process starting an initial preference p.
Validity: If a process decided value p, some process started with value p
Agreement: Two processes that decide decide of the same value
Termination: All processes eventually decide of a value

Aspnes' consensus algorithm:

int $k := 0$, bool $p \in \{0,1\}$, $(\mathrm{rg}_b[r])_{b\in\{0,1\}, r\in\mathbb{N}}$ all initialized to no;
while true **do**
  read from $\mathrm{rg}_0[k]$ and $\mathrm{rg}_1[k]$;
  if $\mathrm{rg}_0[k]$ = yes and $\mathrm{rg}_1[k]$ = no **then** $p := 0$;
  else if $\mathrm{rg}_0[k]$ = no and $\mathrm{rg}_1[k]$ = yes **then** $p := 1$;
  write yes to $\mathrm{rg}_p[k]$;
  if $k > 0$ **then**
    read from $\mathrm{rg}_{1-p}[k-1]$;
    if $\mathrm{rg}_{1-p}[k-1]$ = no **then** return p;
  $k := k+1$;

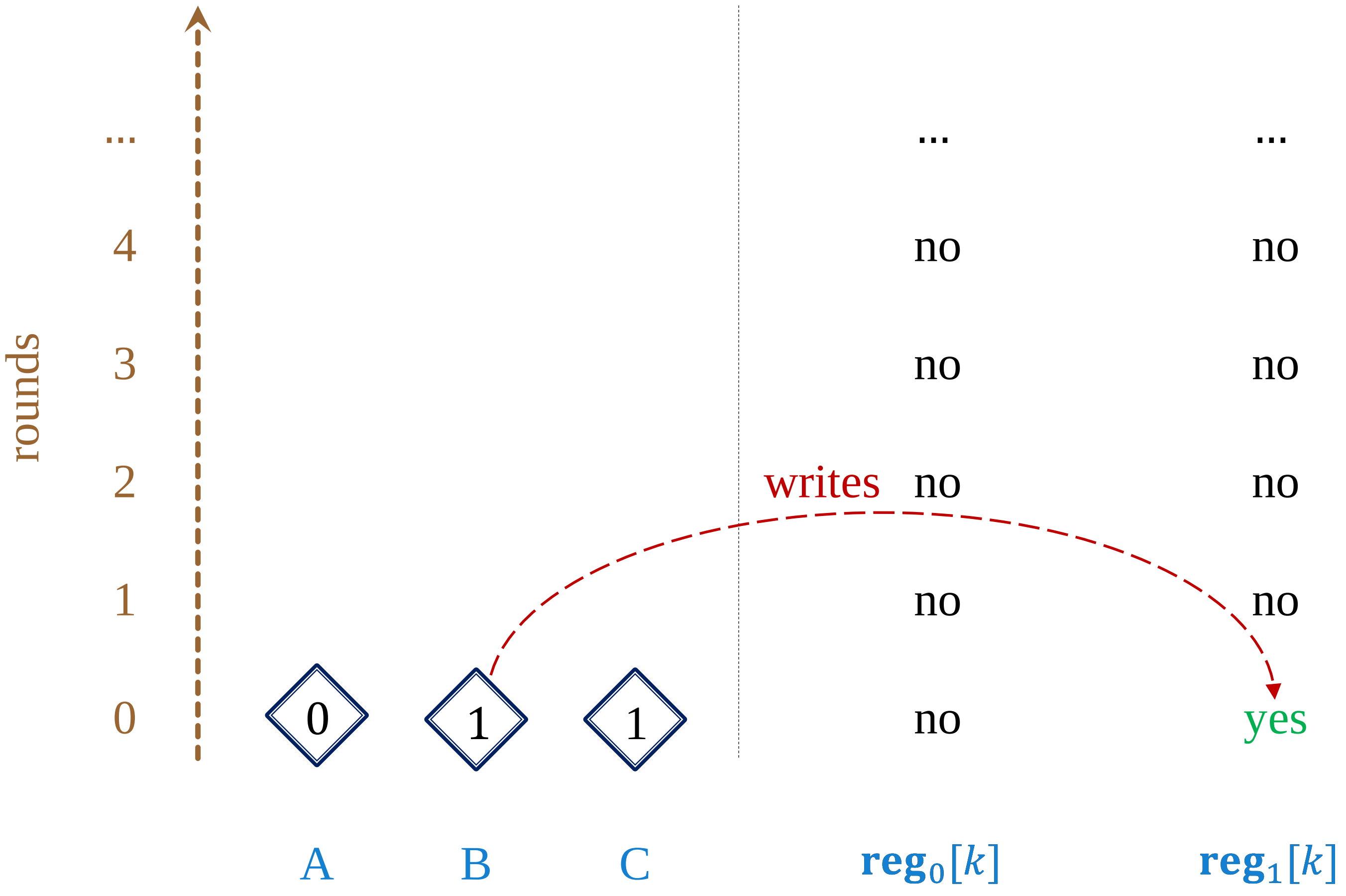
read from registers of rounds k and $k-1$

write to registers of round k

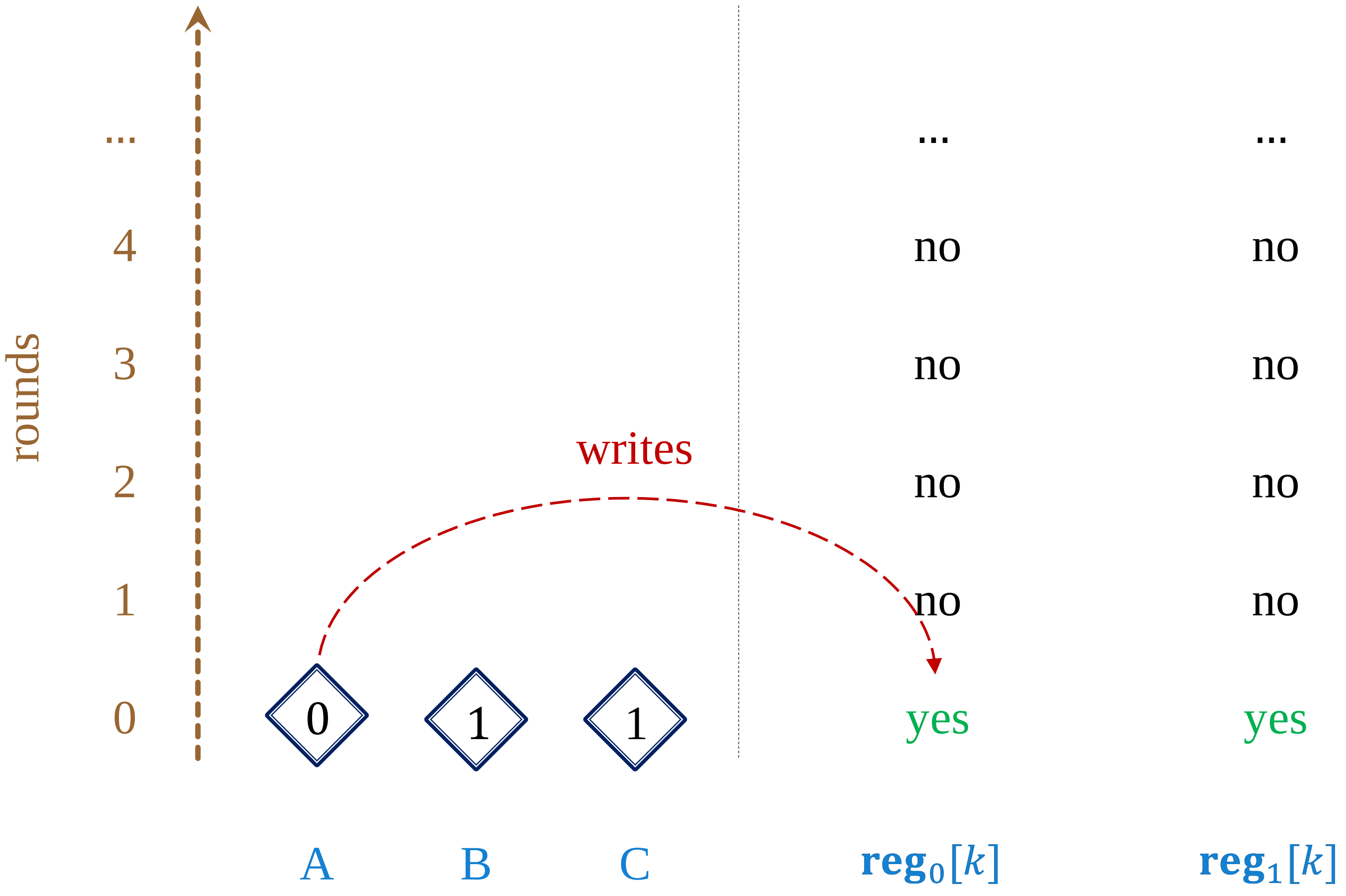
Example of execution of the algorithm

rounds	A	B	C	$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
…				…	…
4				no	no
3				no	no
2				no	no
1				no	no
0	0	1	1	no	no

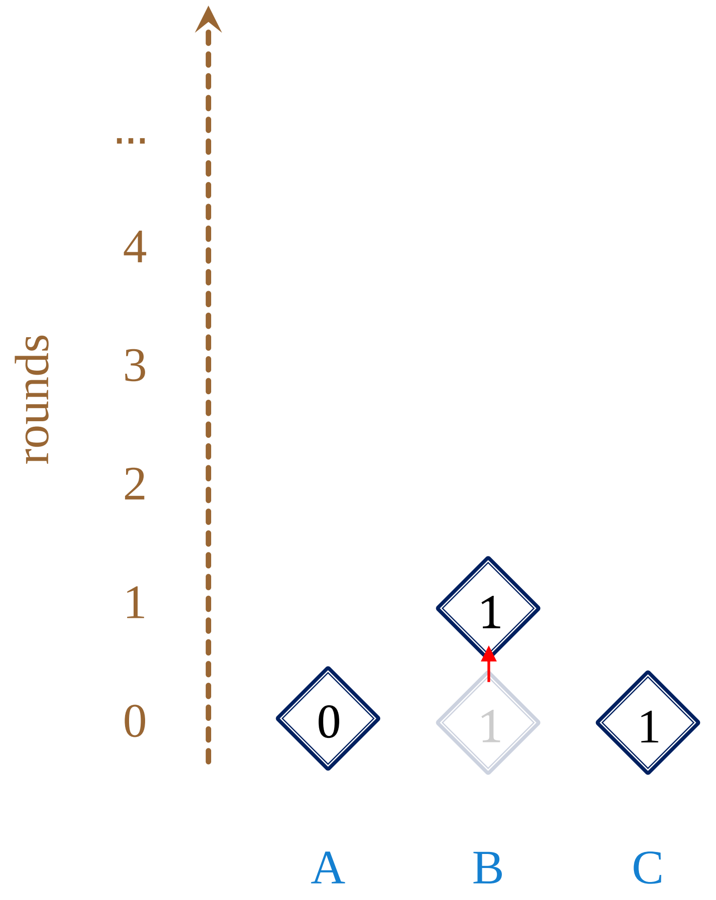
Example of execution of the algorithm



Example of execution of the algorithm

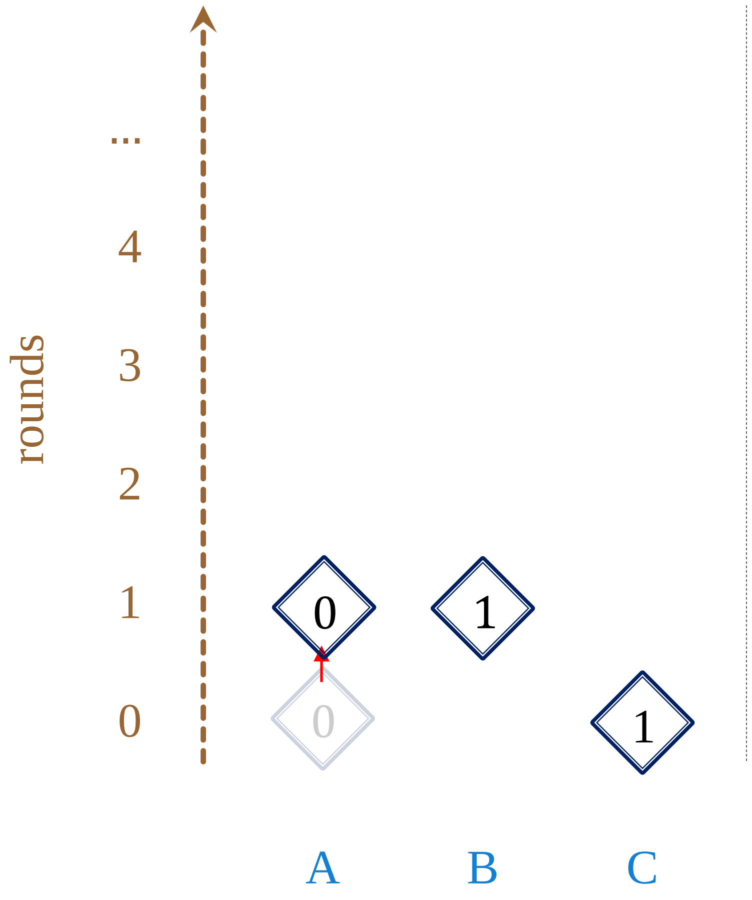


Example of execution of the algorithm



$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...
no	no
no	no
no	no
no	no
yes	yes

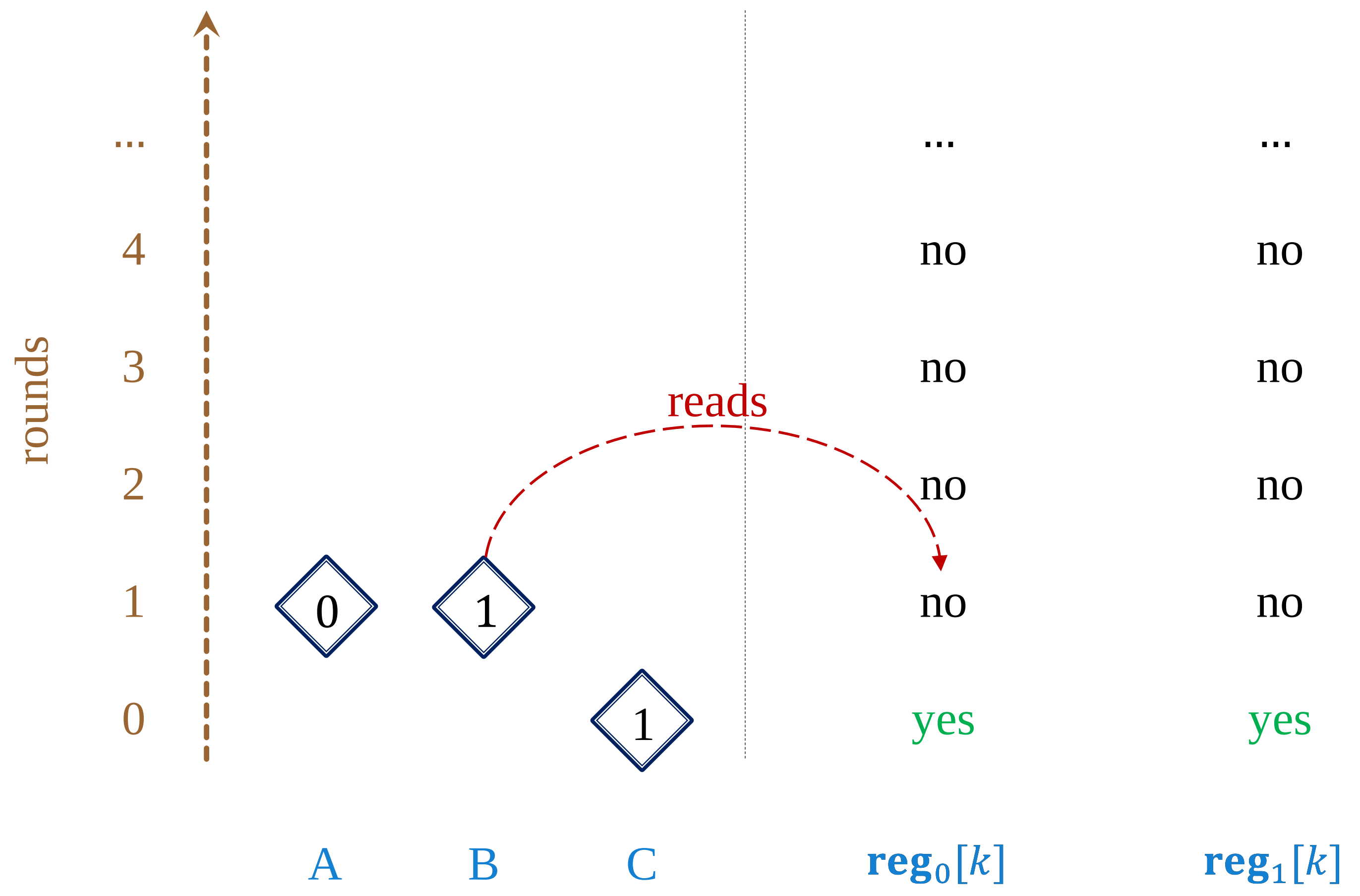
Example of execution of the algorithm



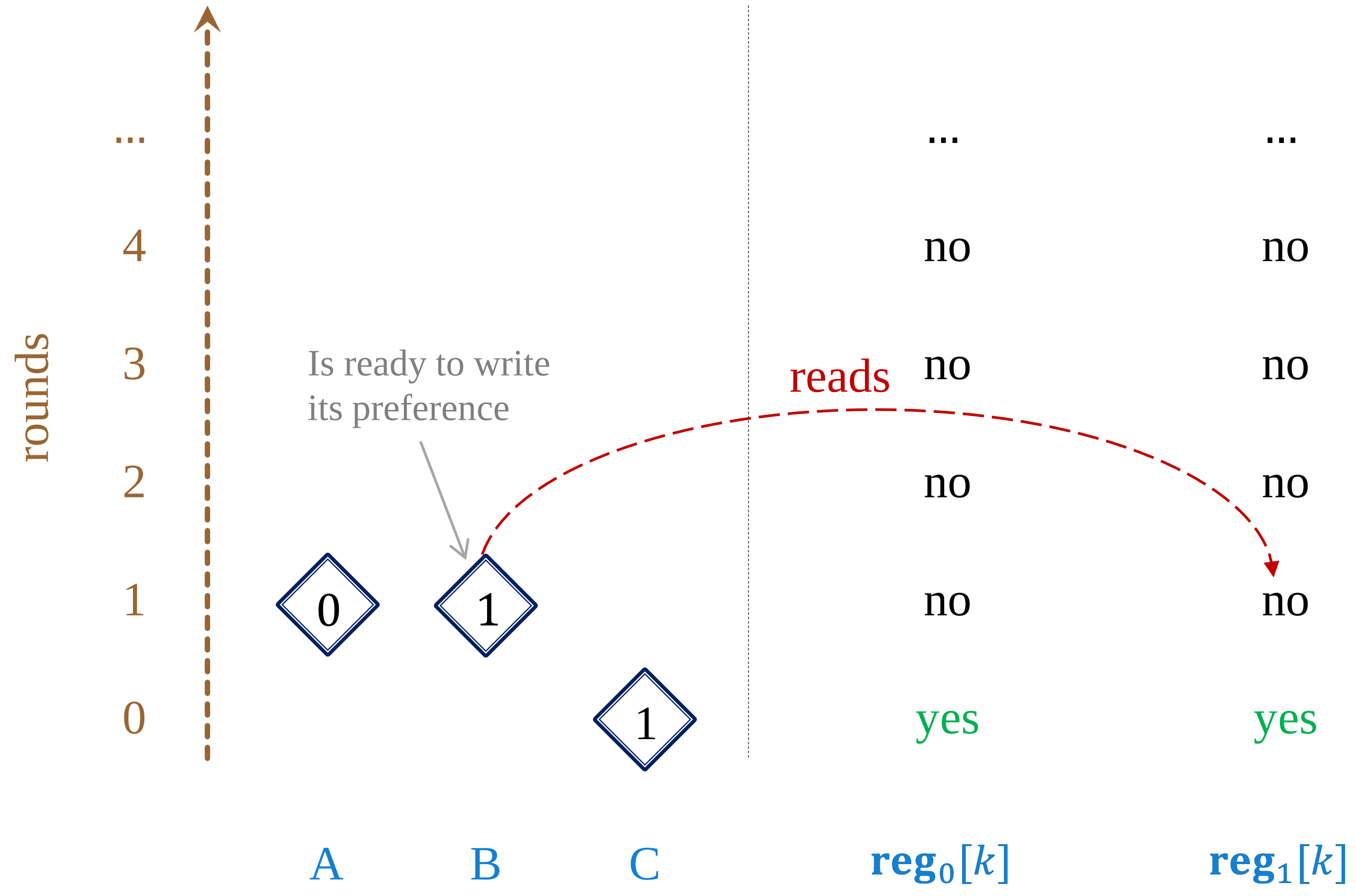
$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...
no	no
no	no
no	no
no	no
yes	yes

 Nicolas Waldburger

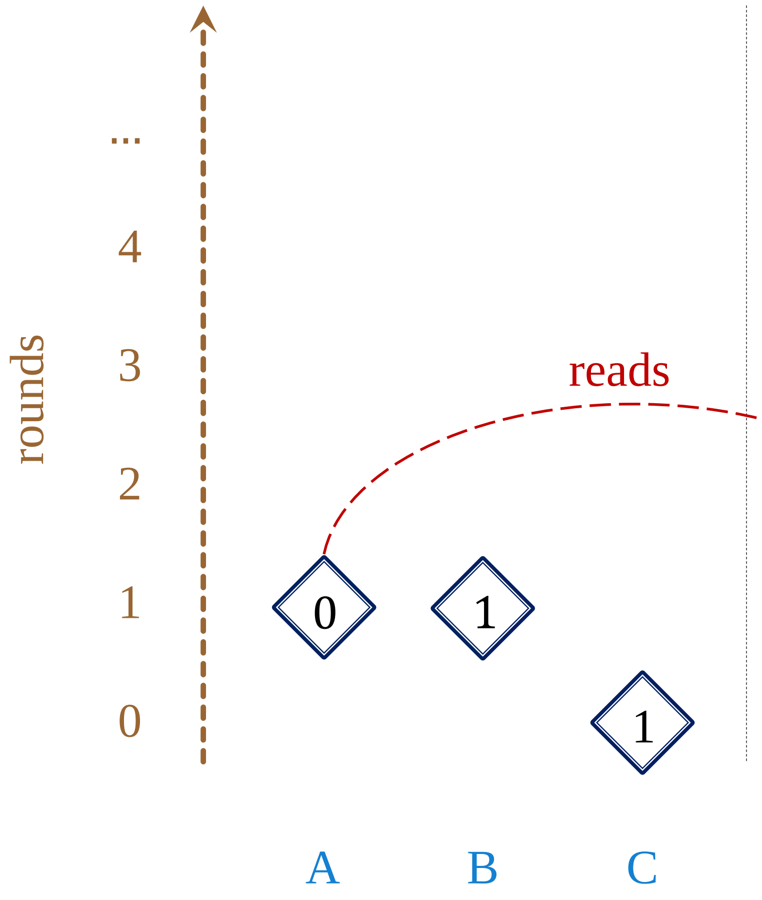
Example of execution of the algorithm



Example of execution of the algorithm

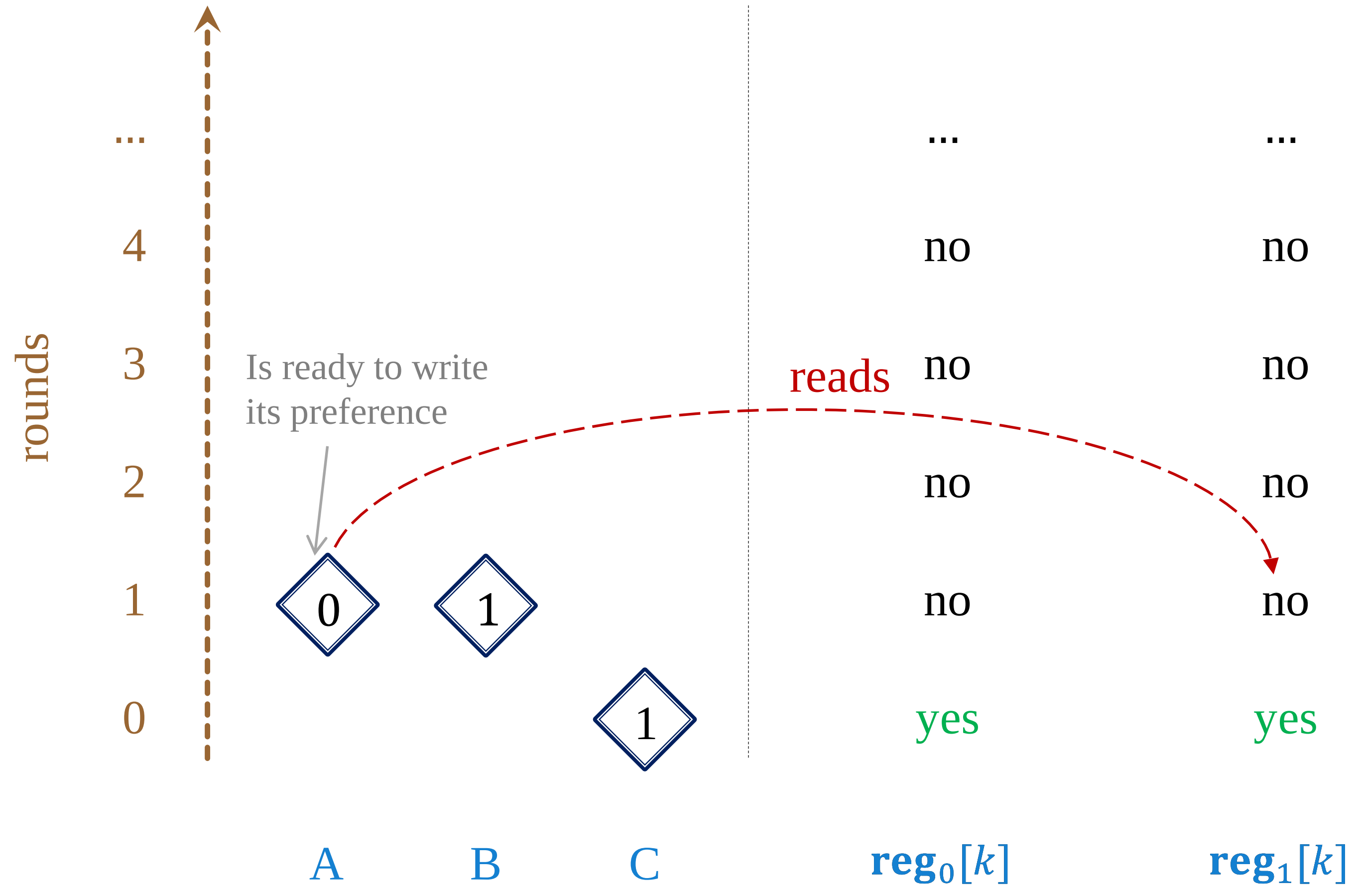


Example of execution of the algorithm

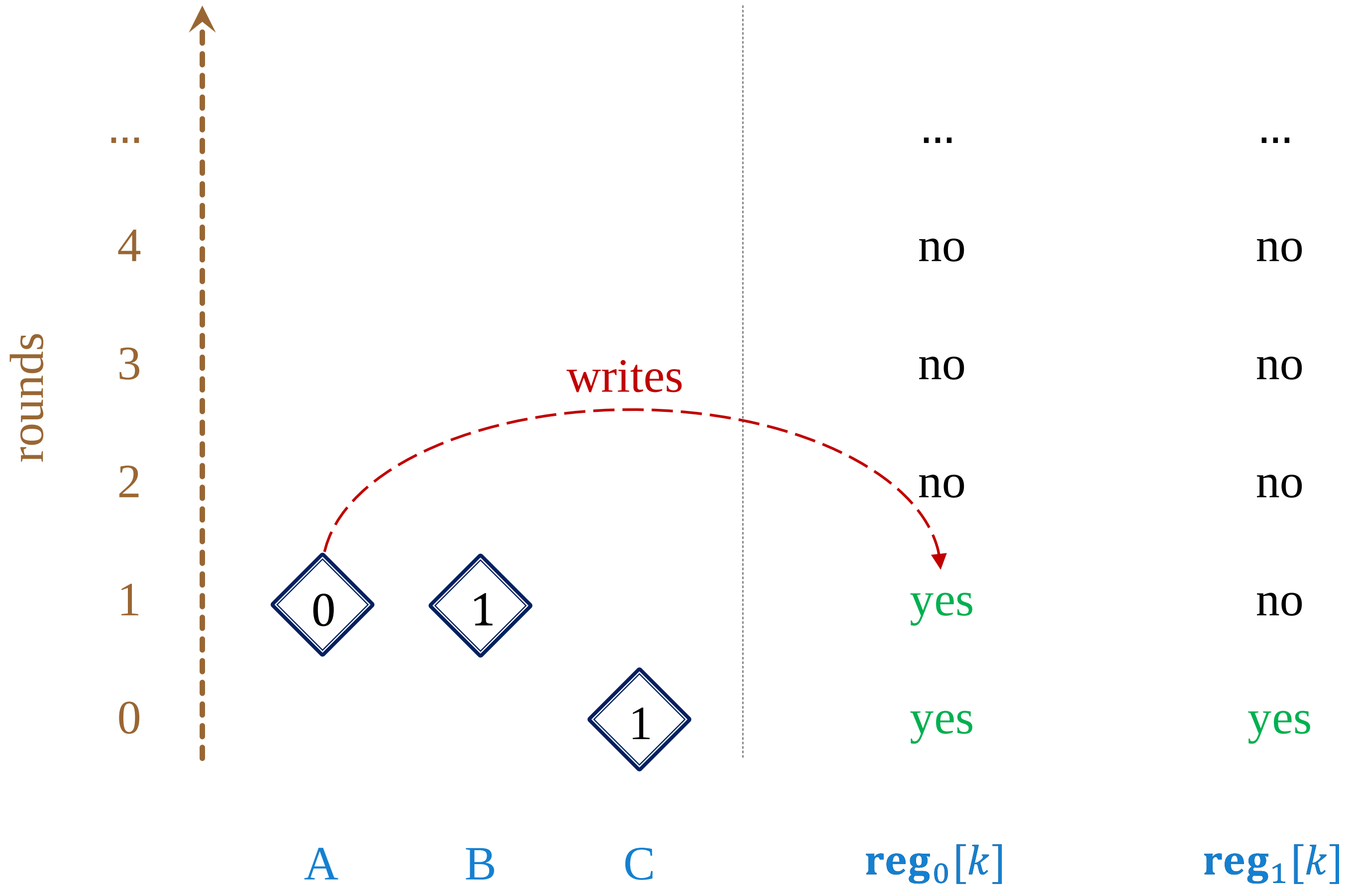


$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...
no	no
no	no
no	no
no	no
yes	yes

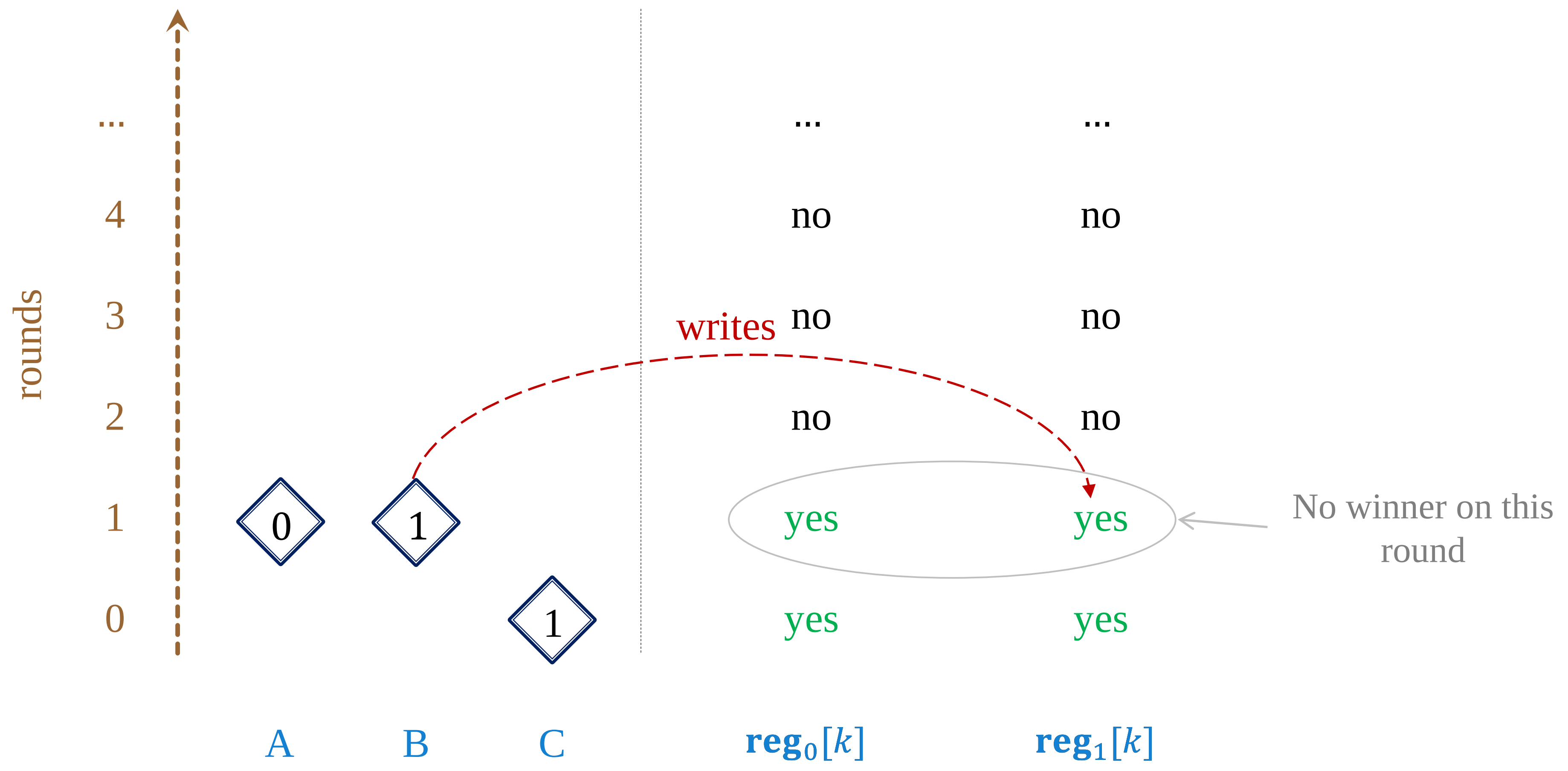
Example of execution of the algorithm



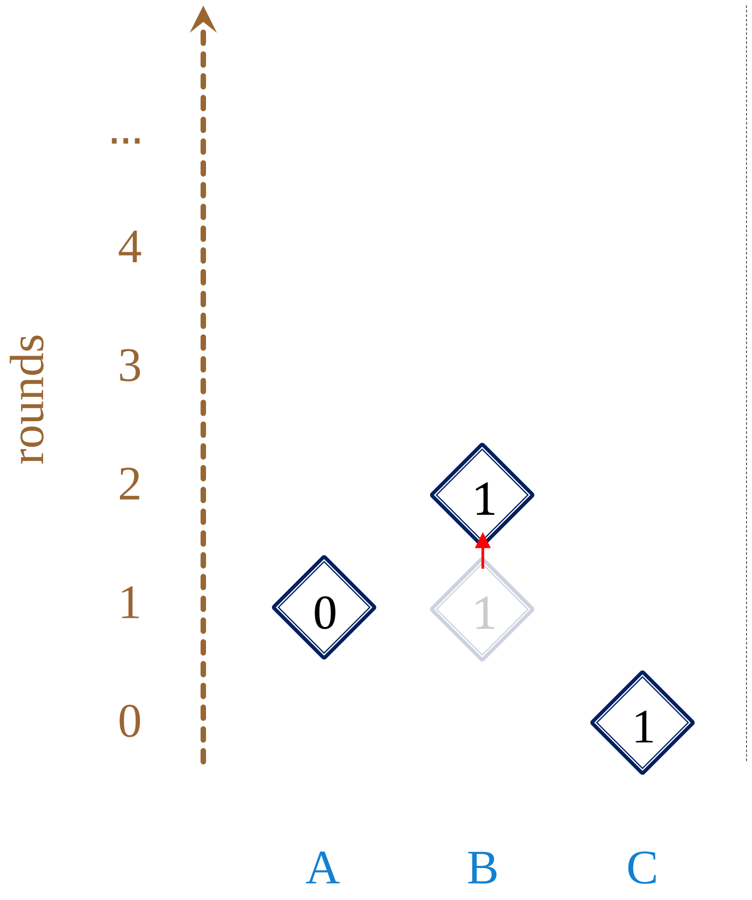
Example of execution of the algorithm



Example of execution of the algorithm

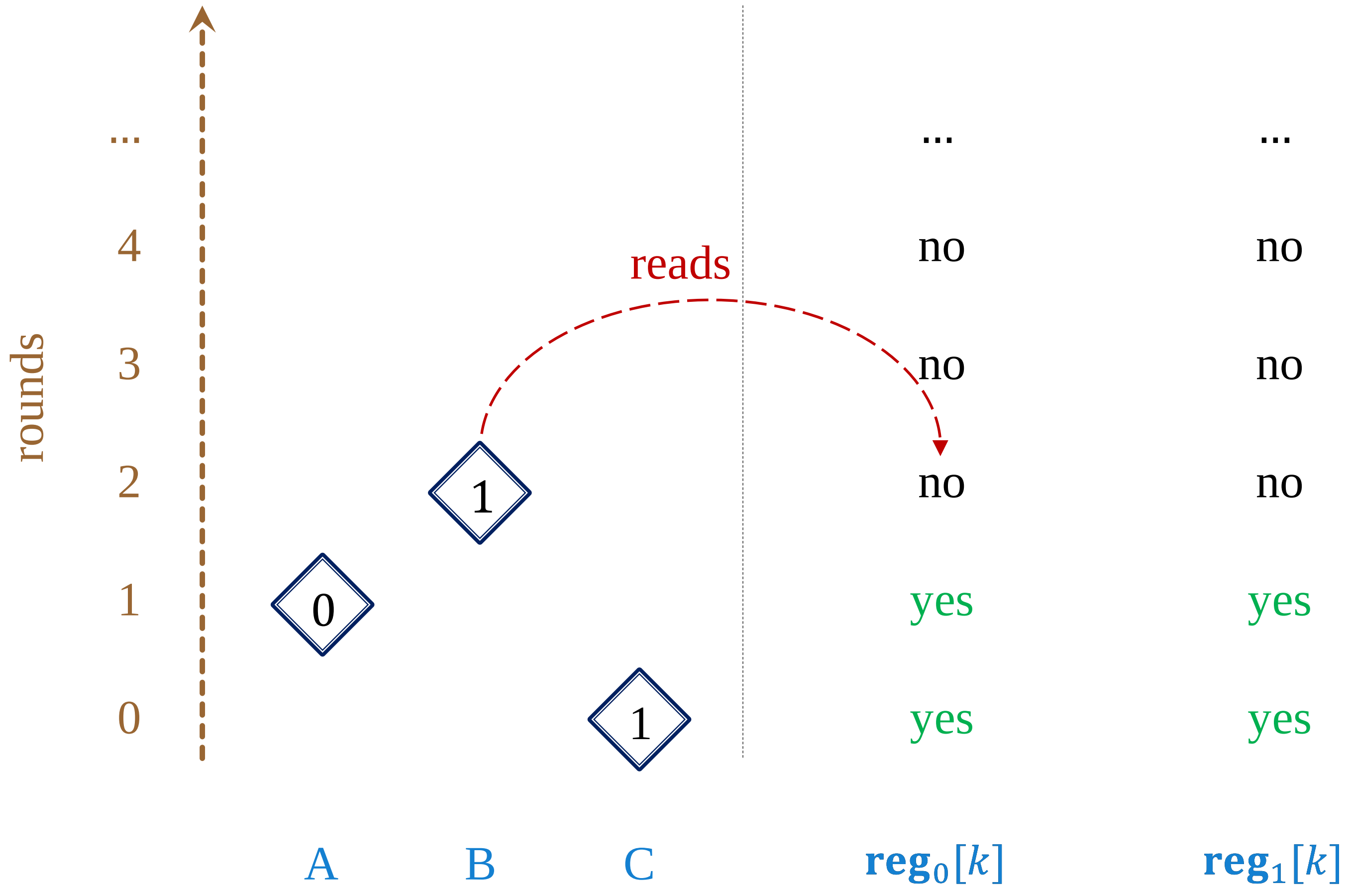


Example of execution of the algorithm

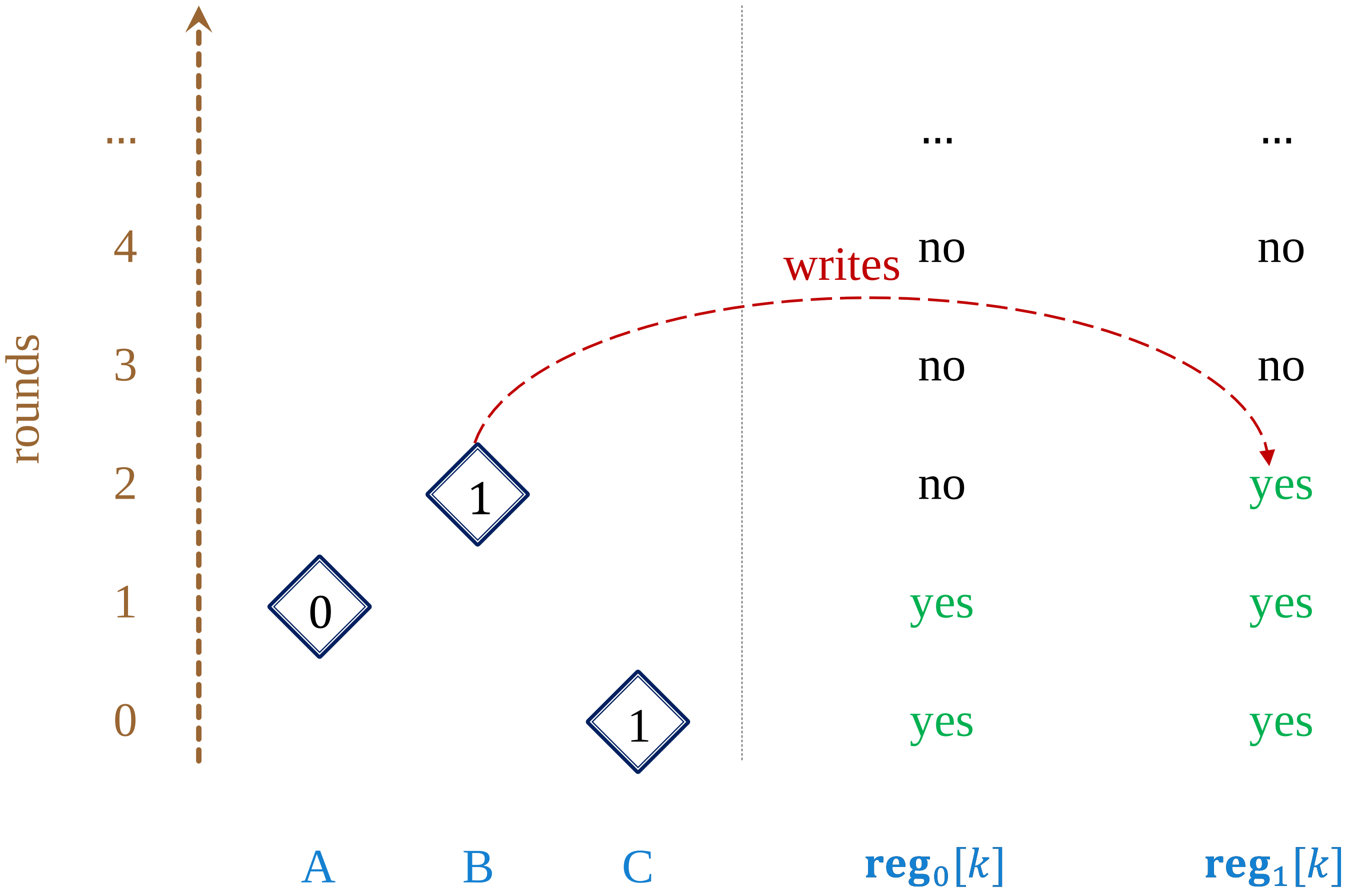


$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...
no	no
no	no
no	no
yes	yes
yes	yes

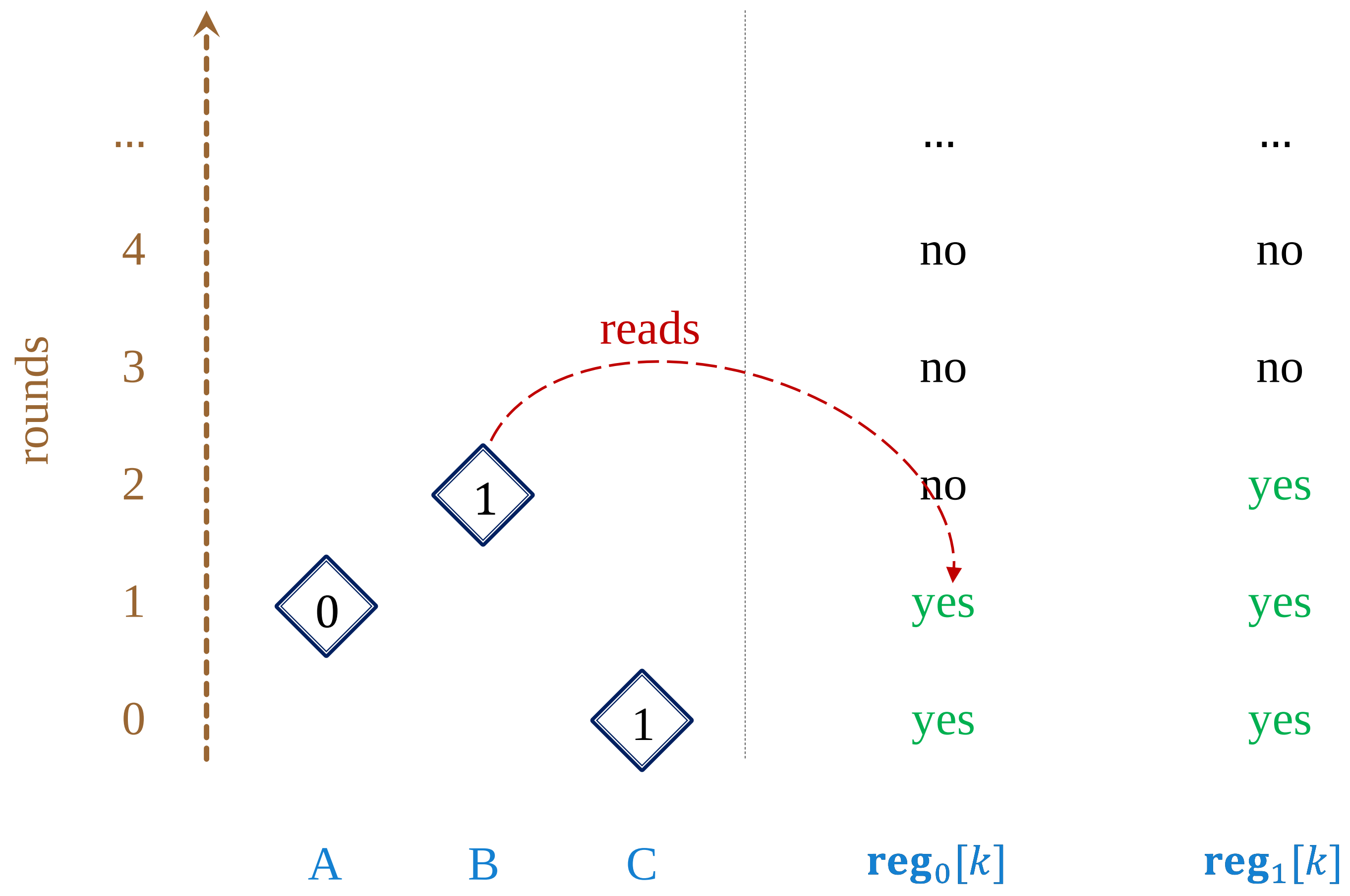
Example of execution of the algorithm



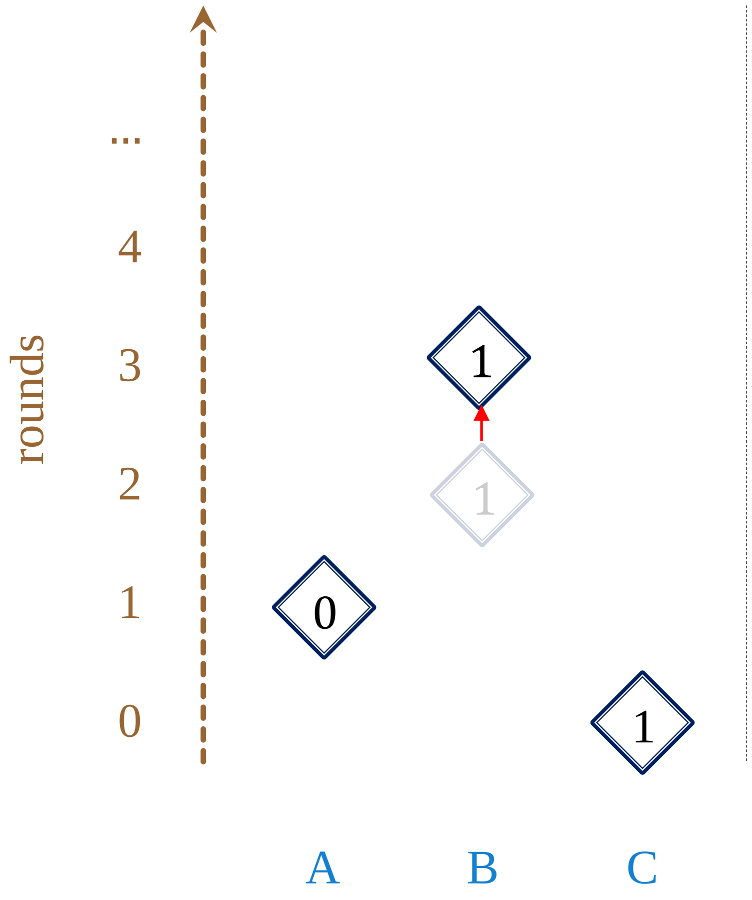
Example of execution of the algorithm



Example of execution of the algorithm

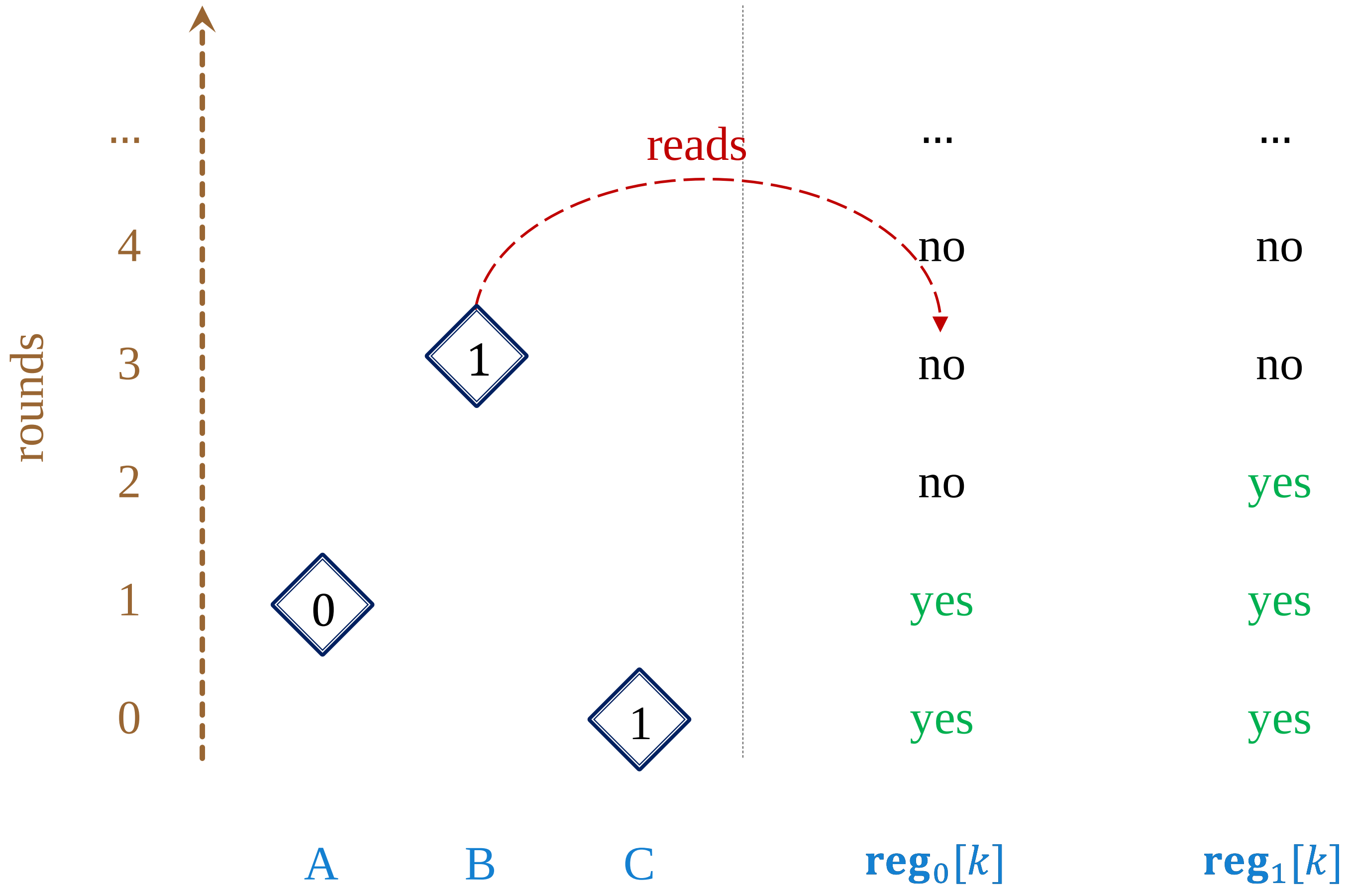


Example of execution of the algorithm

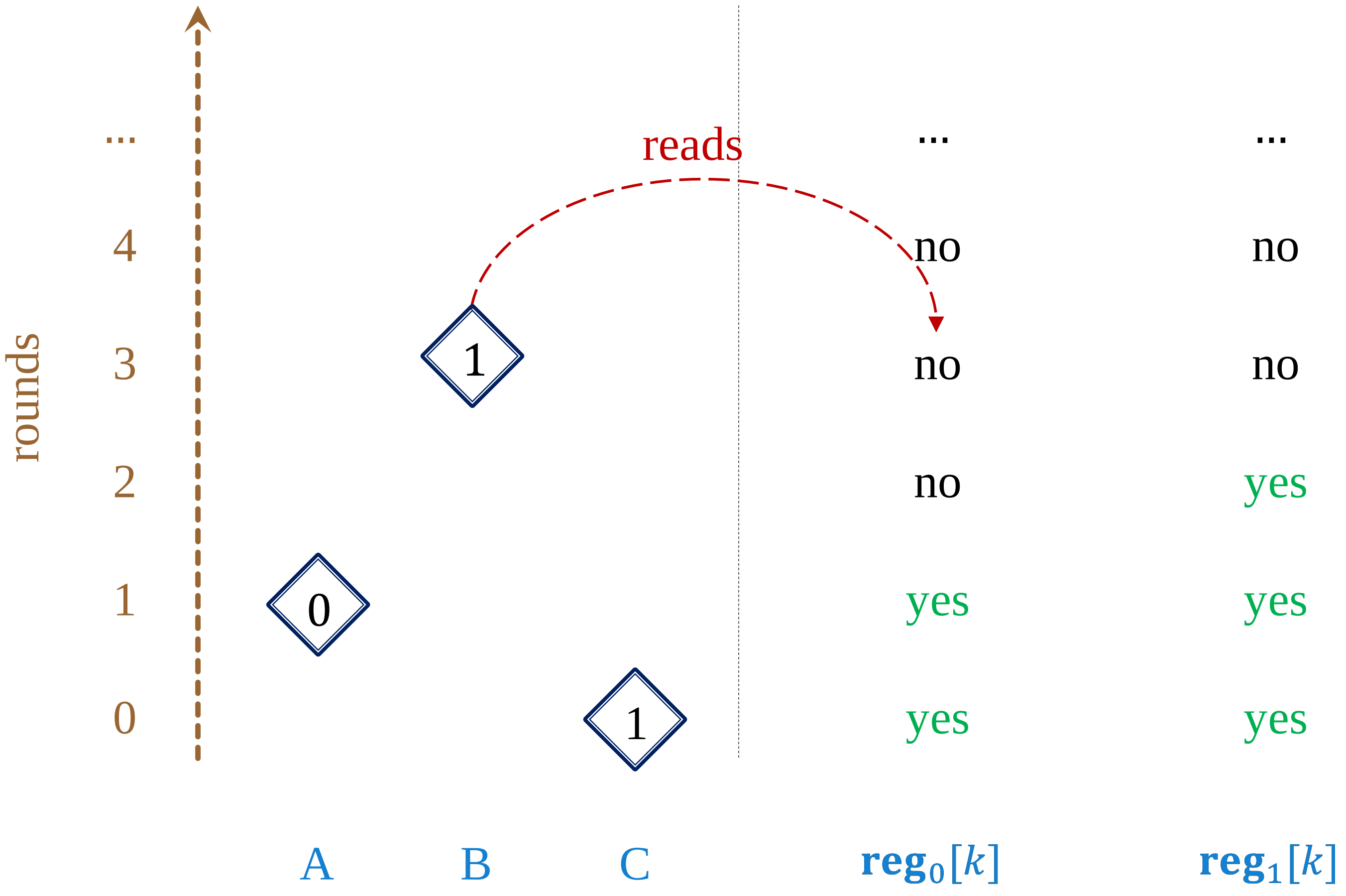


round	$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...	...
4	no	no
3	no	no
2	no	yes
1	yes	yes
0	yes	yes

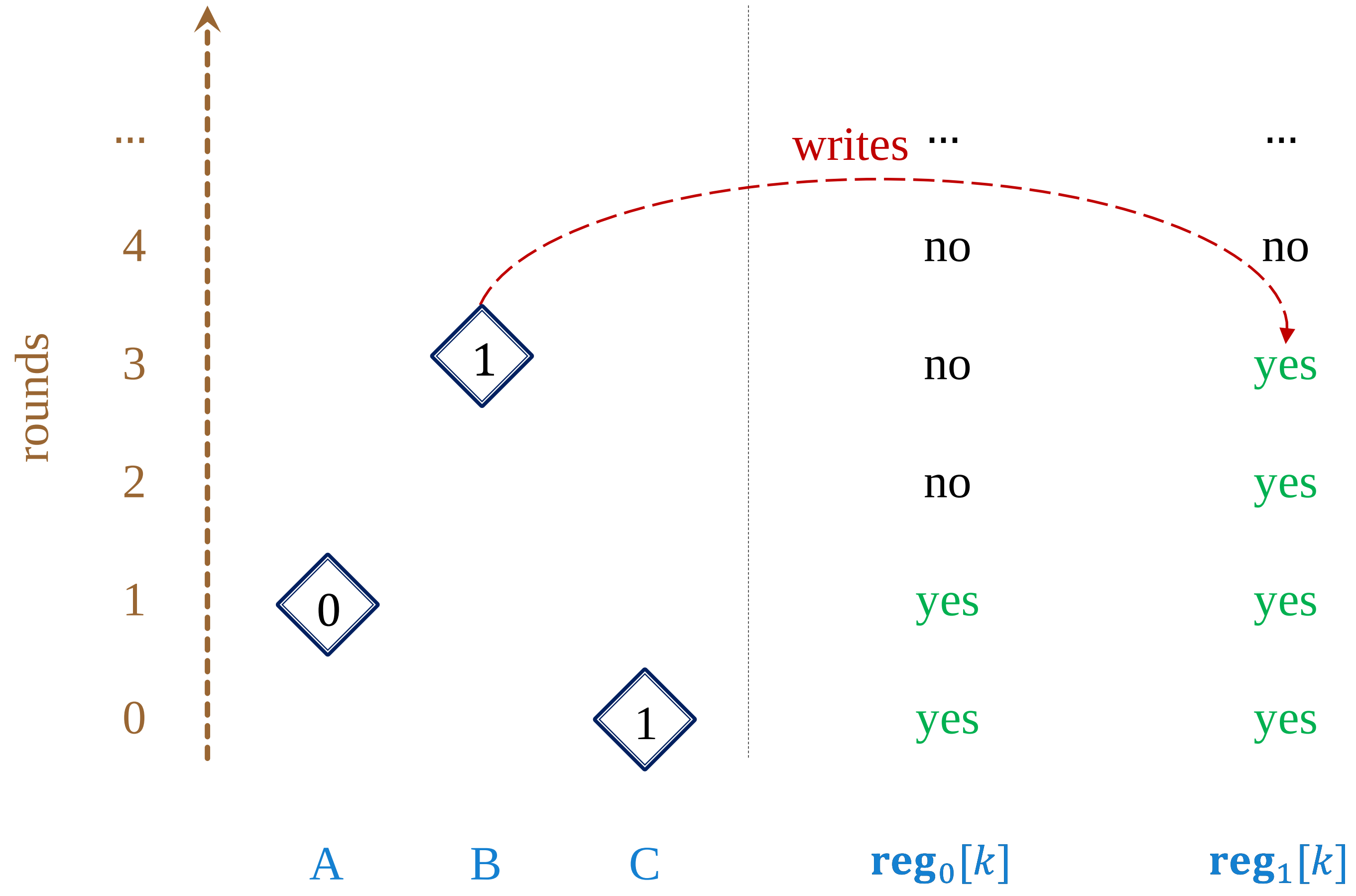
Example of execution of the algorithm



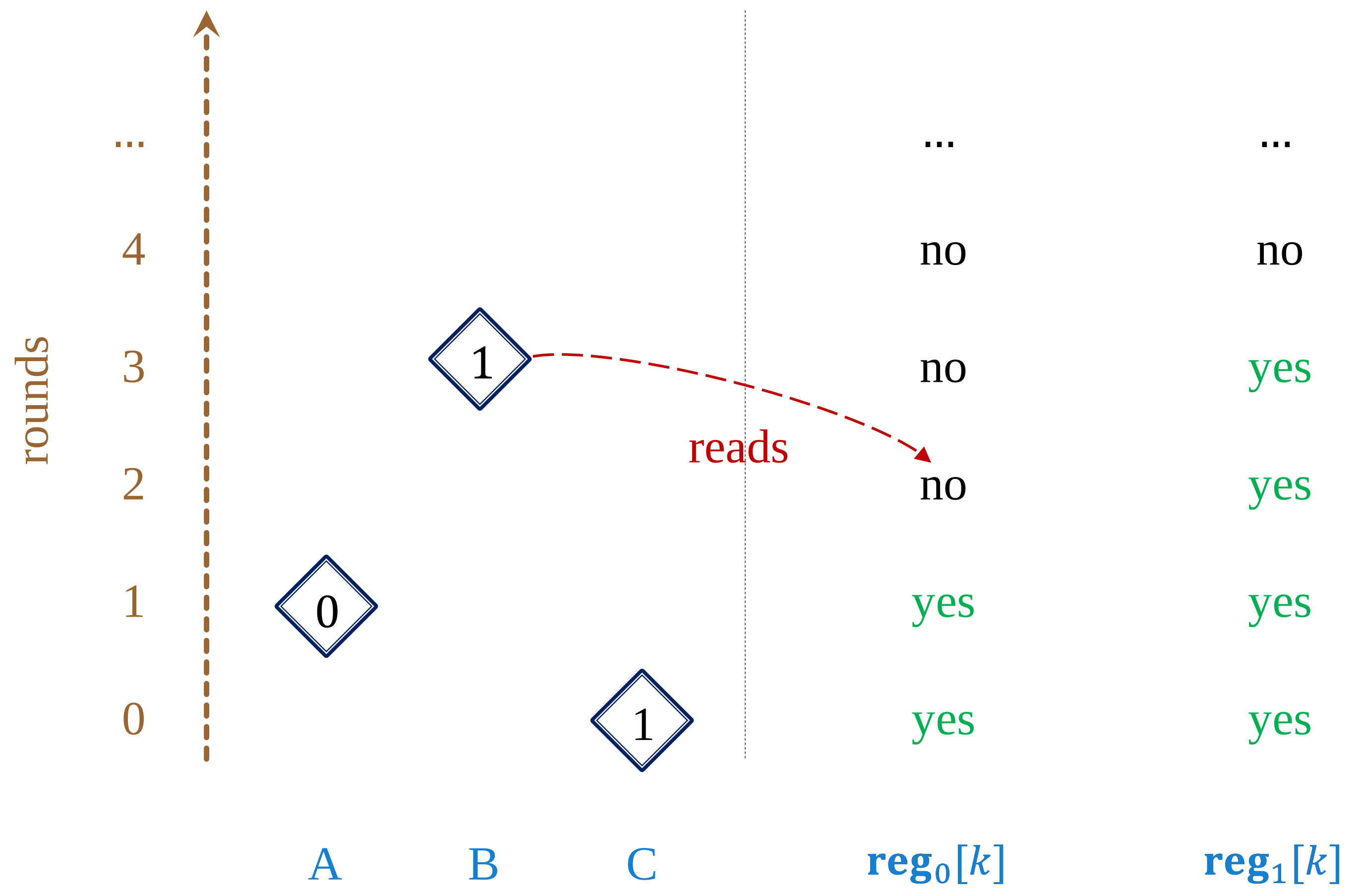
Example of execution of the algorithm



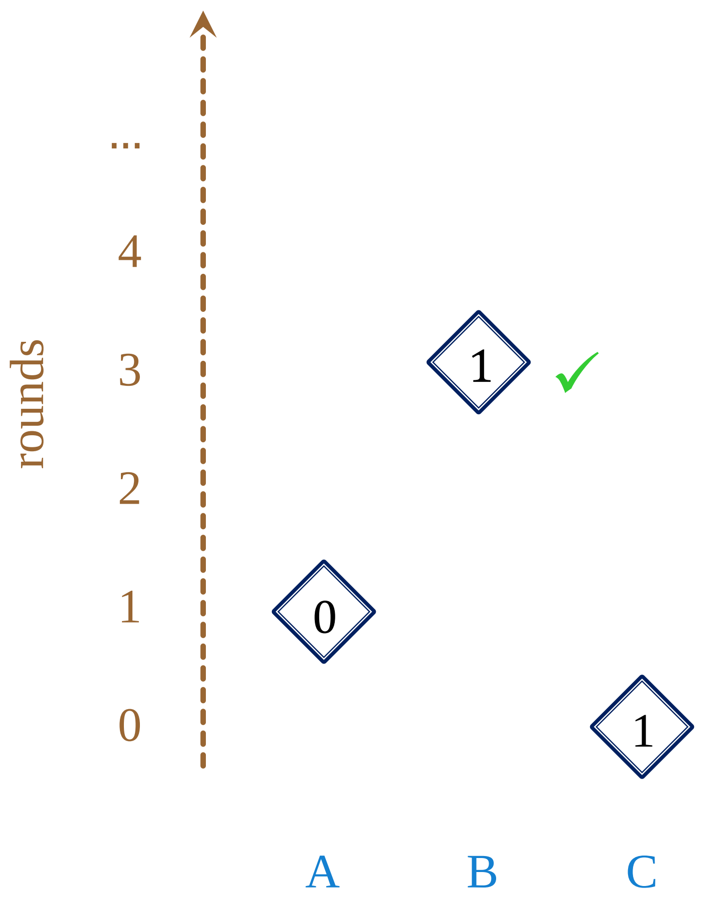
Example of execution of the algorithm



Example of execution of the algorithm



Example of execution of the algorithm



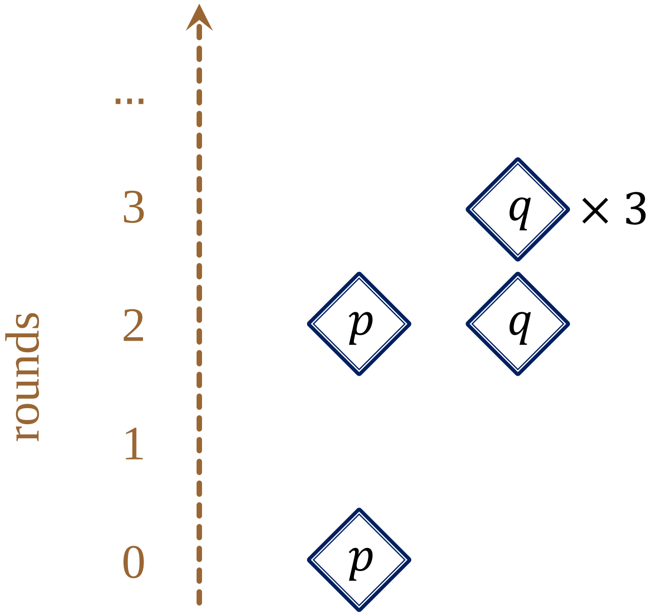
$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...
no	no
no	yes
no	yes
yes	yes
yes	yes

A process getting to this round will convert to preference 1

Round-based shared-memory systems

Model inspired by round-based algorithms from the literature[6,7,8].

Process progress in asynchronous rounds, each round having its own finite set of registers.



...	...	...	...
a	d_0	d_0	d_0
b	c	a	b
a	a	c	b
a	a	b	d_0
$\mathbf{reg}_1[k]$	$\mathbf{reg}_2[k]$	$\mathbf{reg}_3[k]$	$\mathbf{reg}_4[k]$

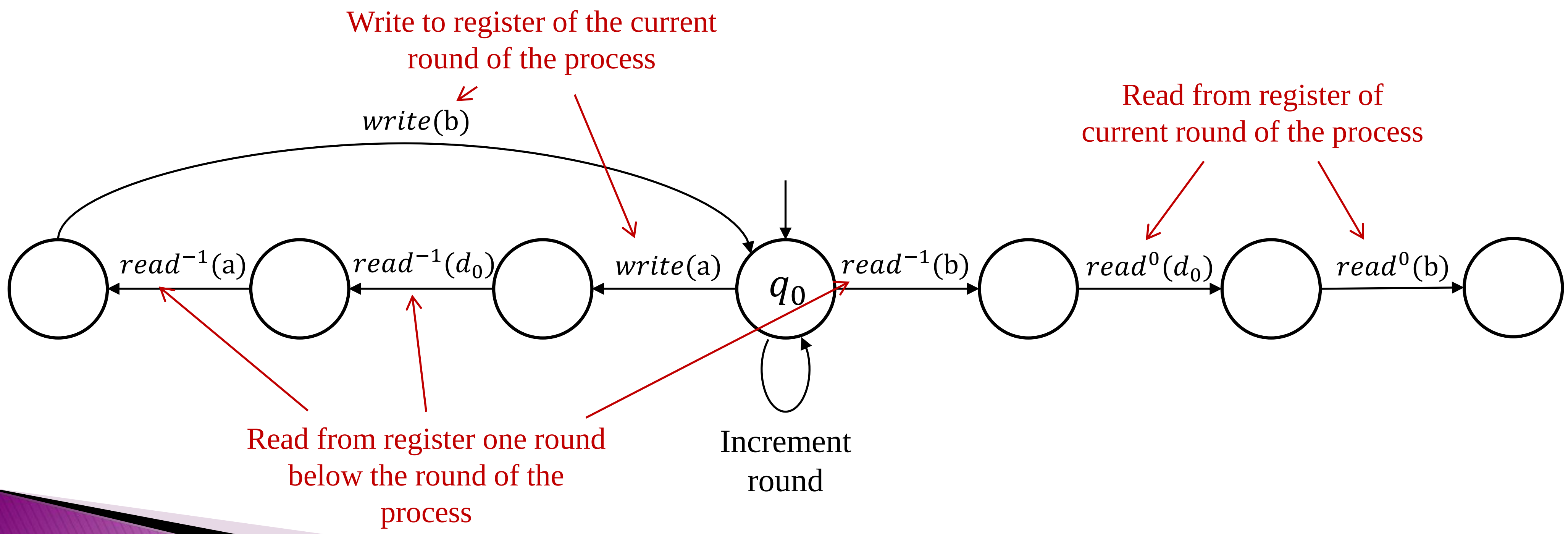
Initial value

6. Aspnes, J.: *Fast deterministic consensus in a noisy environment.* Journal of Algorithms, 2002
7. Guerraoui, R., Ruppert, E.: *Anonymous and fault-tolerant shared-memory computing*. Distrib. Comput., 2007
8. Raynal, M., Stainer, J.: *A Simple Asynchronous Shared Memory Consensus Algorithm Based on Omega and Closing Sets. CISIS, 2012*

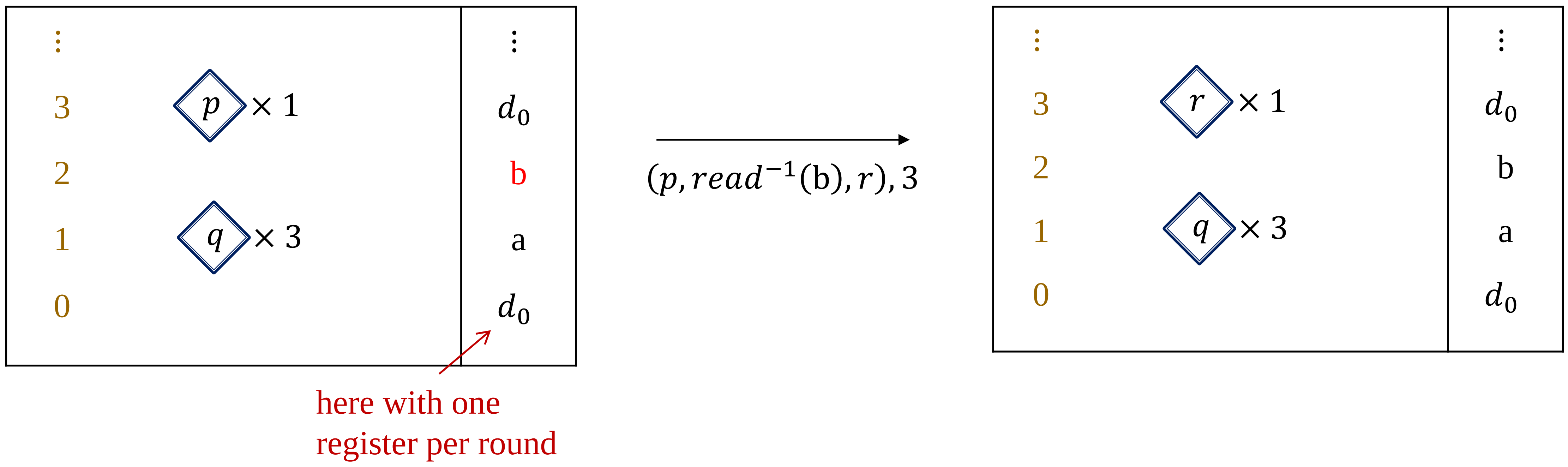
The round-based model

- Read transitions now mention from which round they are reading, relatively to the current round of the process
- A new type of transitions: *round increments*, which send the process to the next round

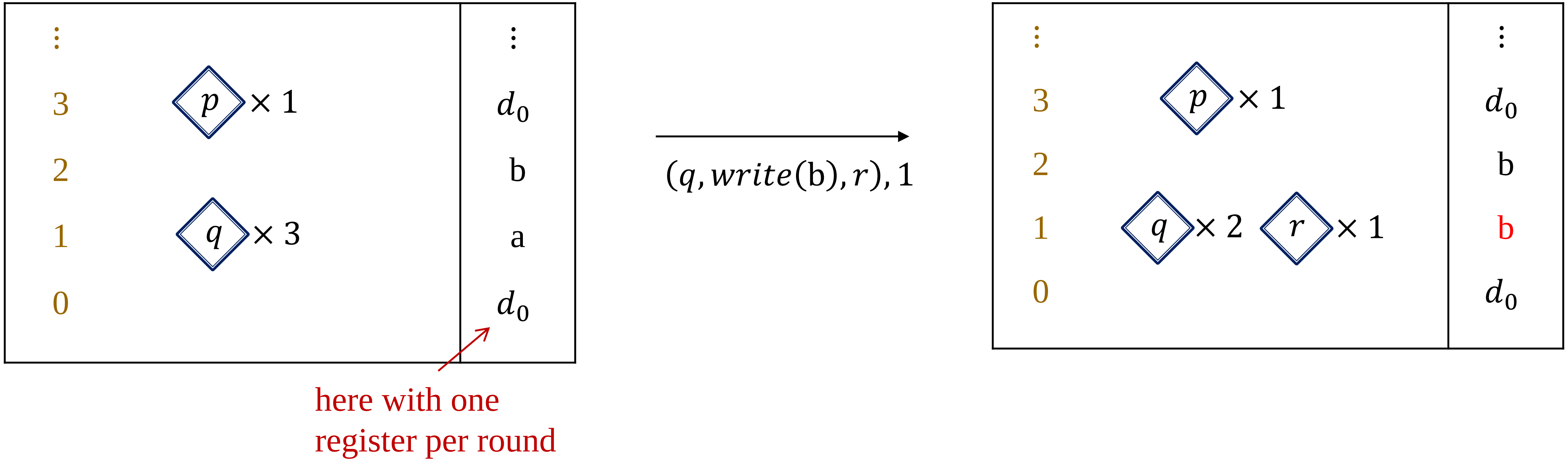
Example with one register per round:



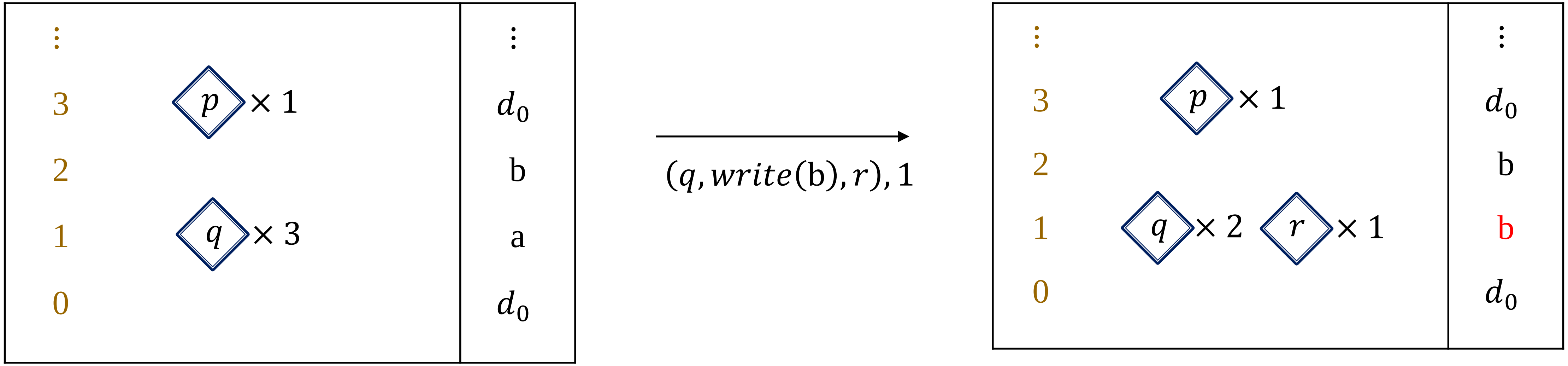
Semantics


⋮
3
2
1
0
p × 1
q × 3
⋮
d_0
b
a
d_0
here with one
register per round
$(p, read^{-1}(\mathrm{b}), r), 3$
⋮
3
2
1
0
r × 1
q × 3
⋮
d_0
b
a
d_0

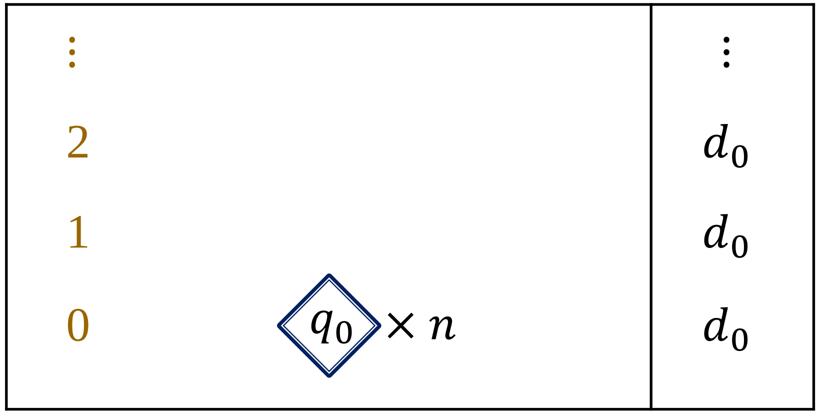
Semantics


⋮
3
2
1
0
p × 1
q × 3
⋮
d_0
b
a
d_0
here with one
register per round
$(q, write(\mathrm{b}), r), 1$
⋮
3
2
1
0
p × 1
q × 2
r × 1
⋮
d_0
b
b
d_0

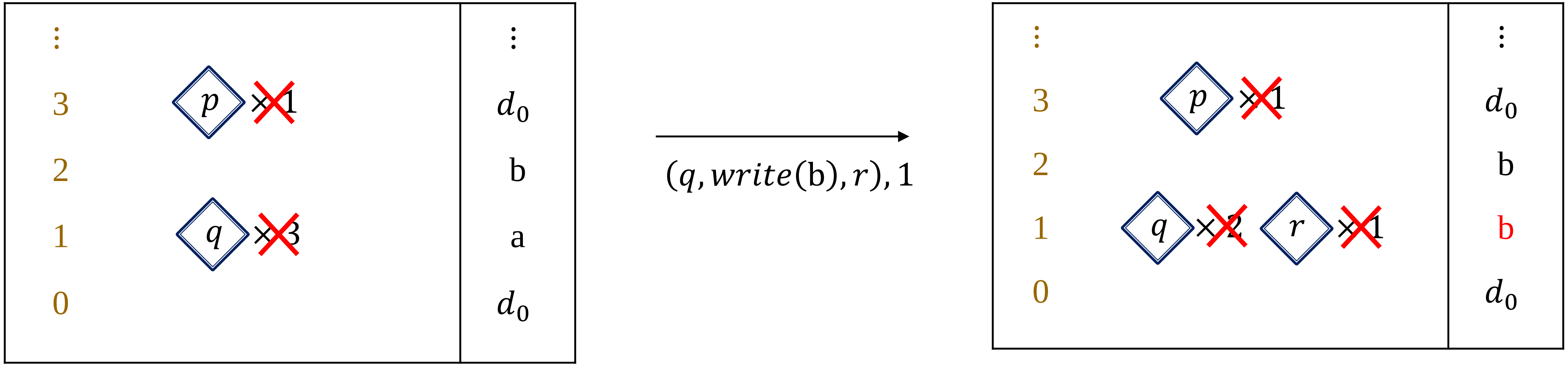
Semantics



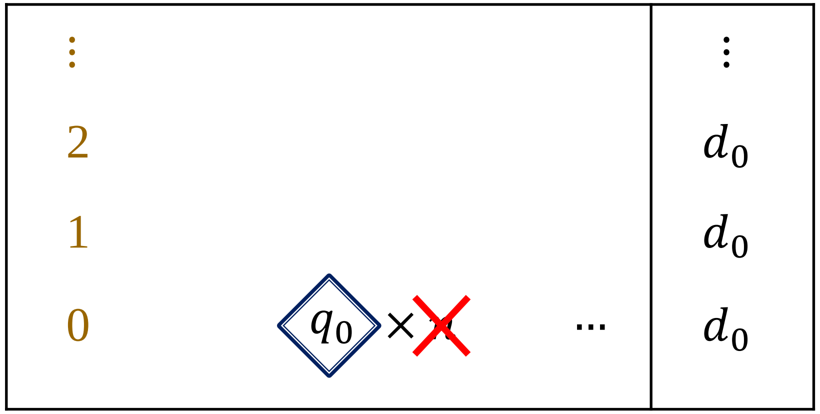
Initial configurations:



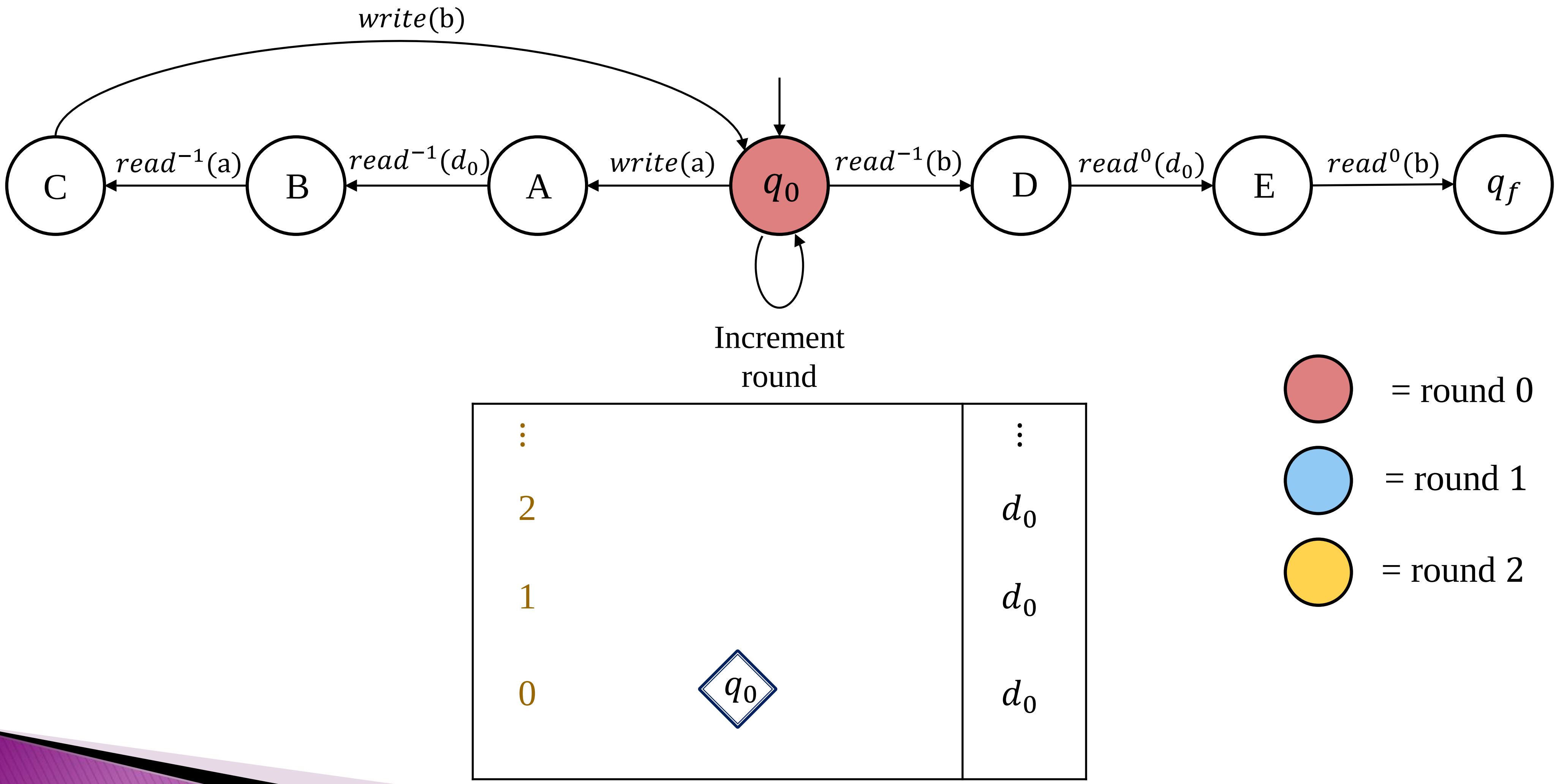
Abstraction



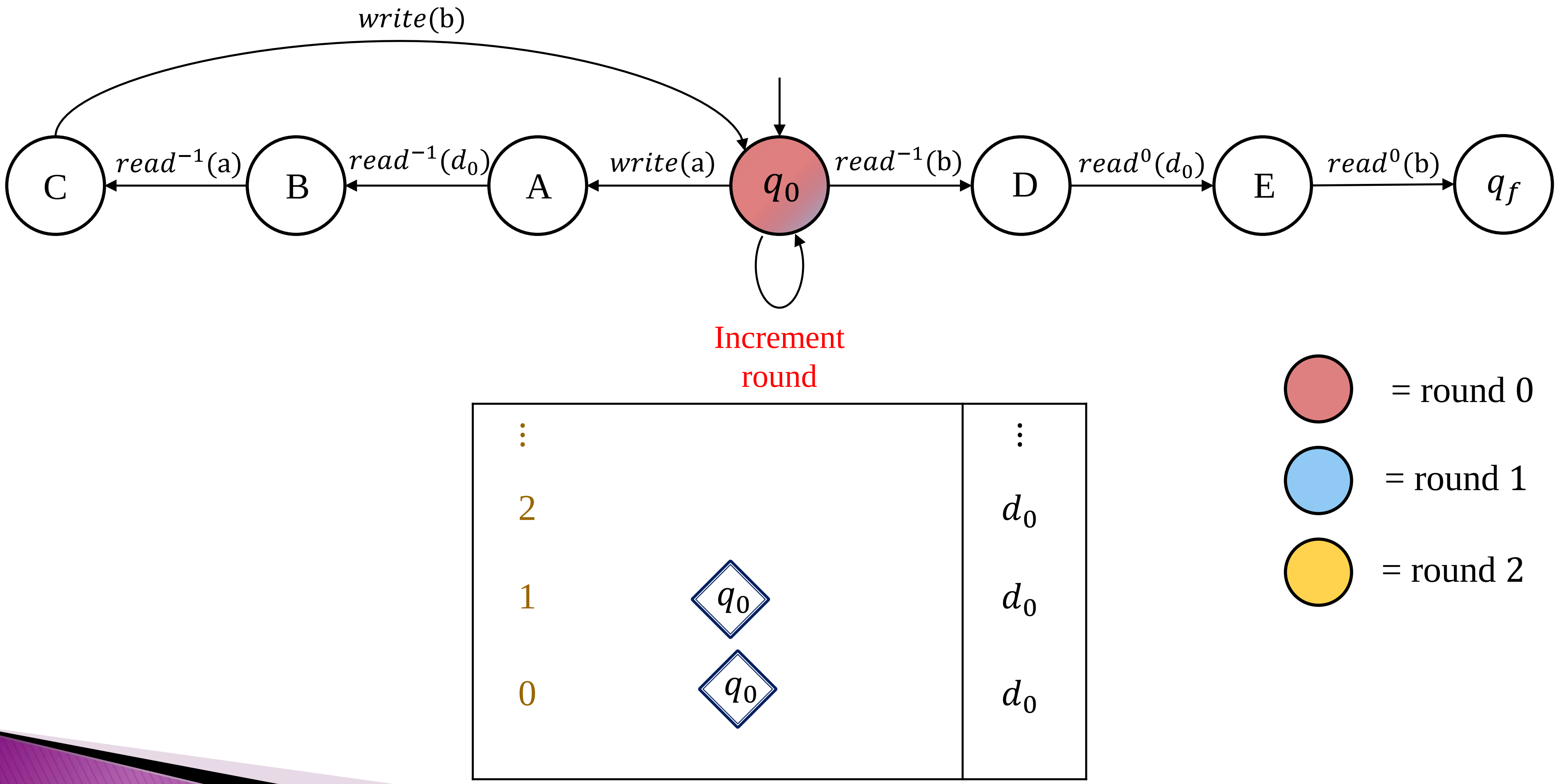
Initial configurations:



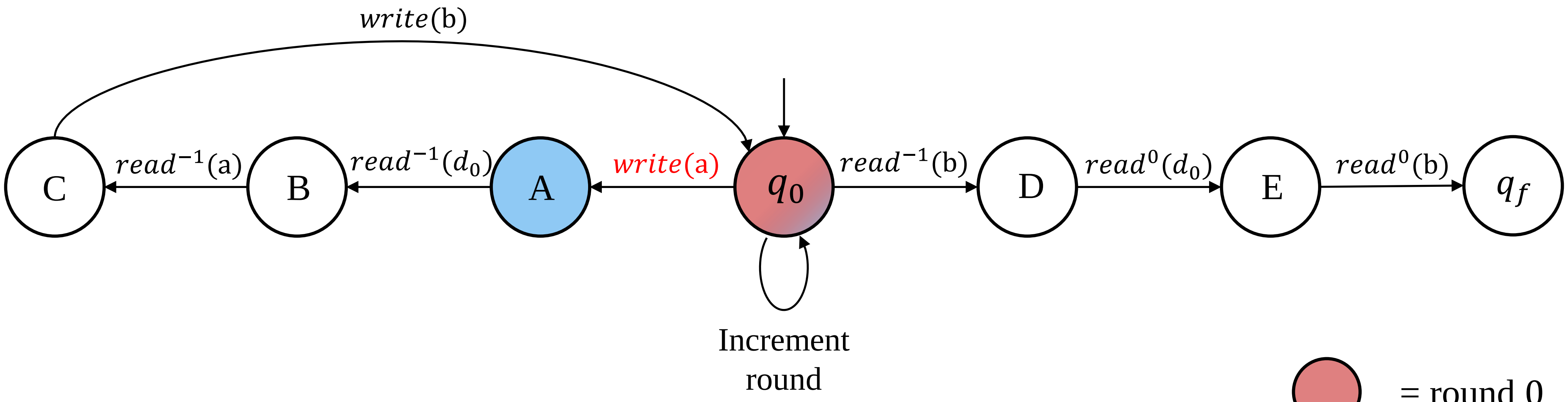
An example of round-based register protocol

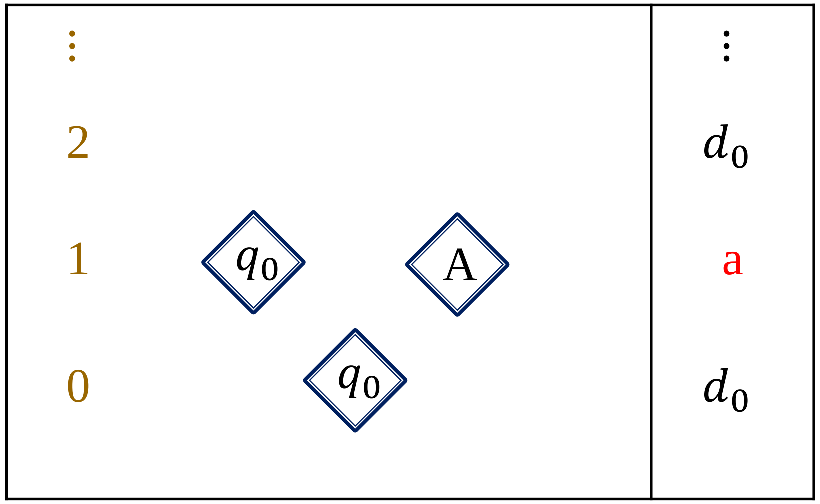


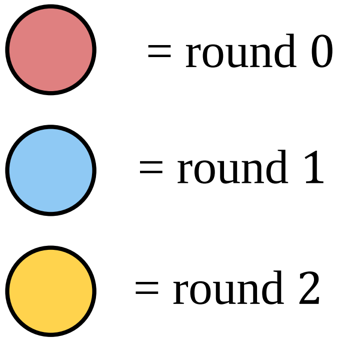
An example of round-based register protocol



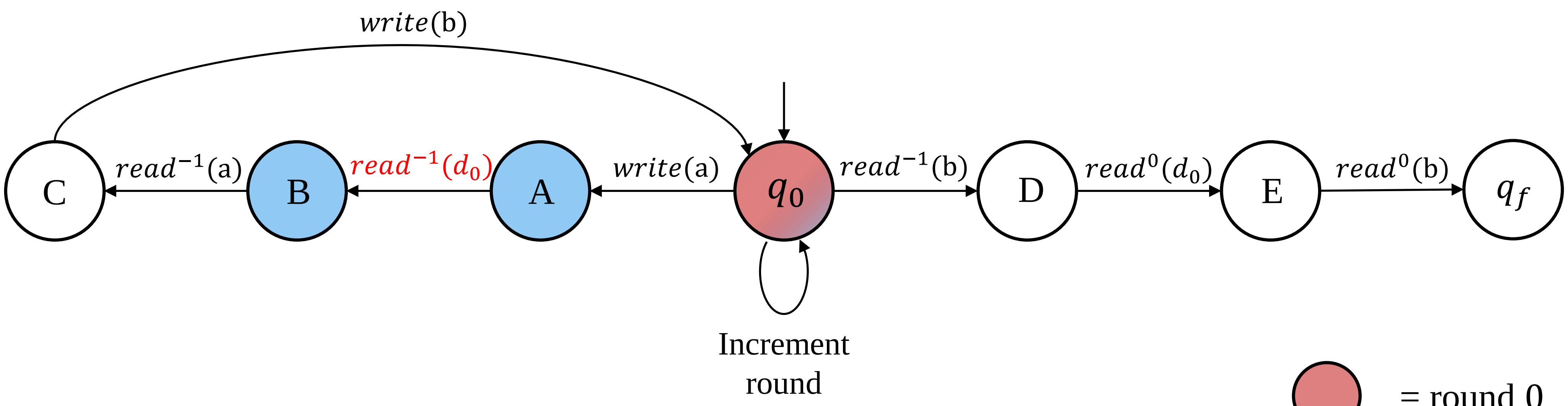
An example of round-based register protocol

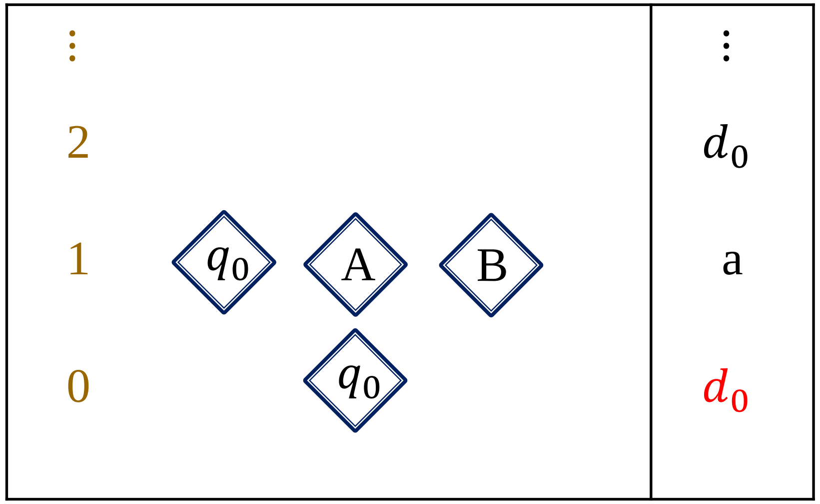


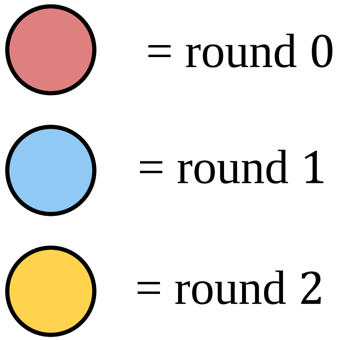




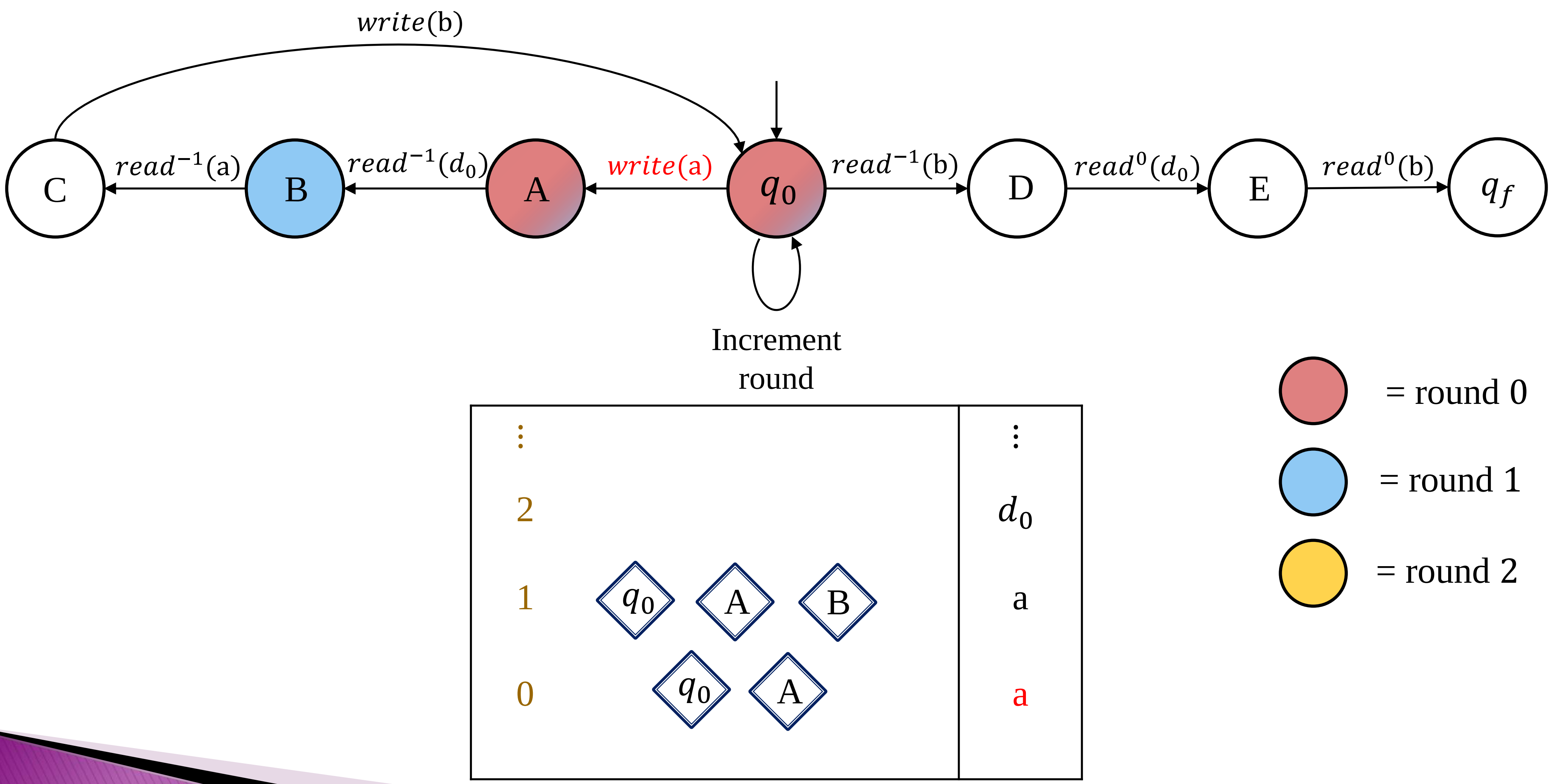
An example of round-based register protocol



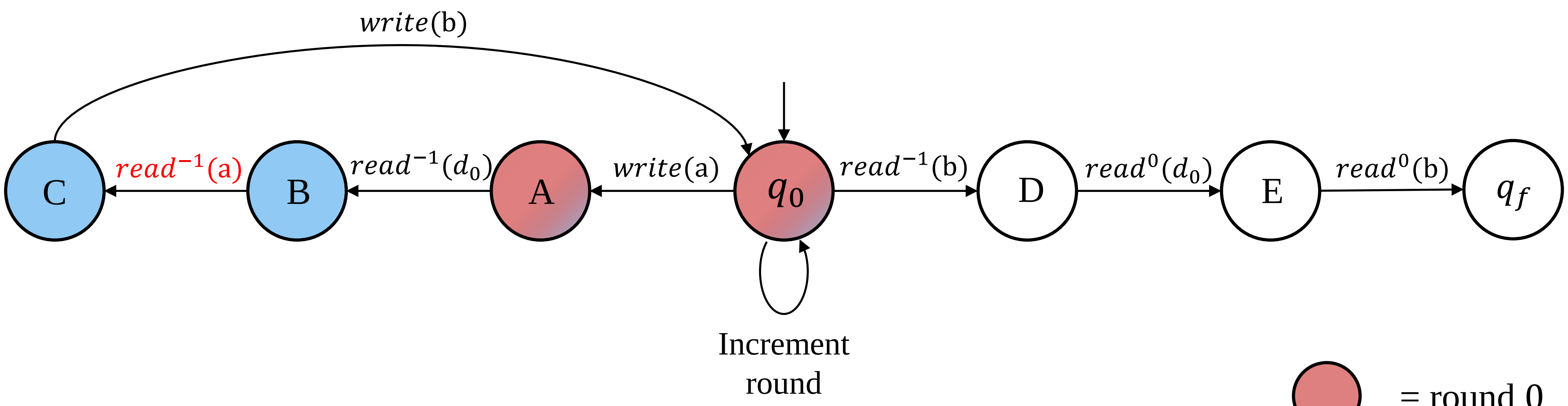


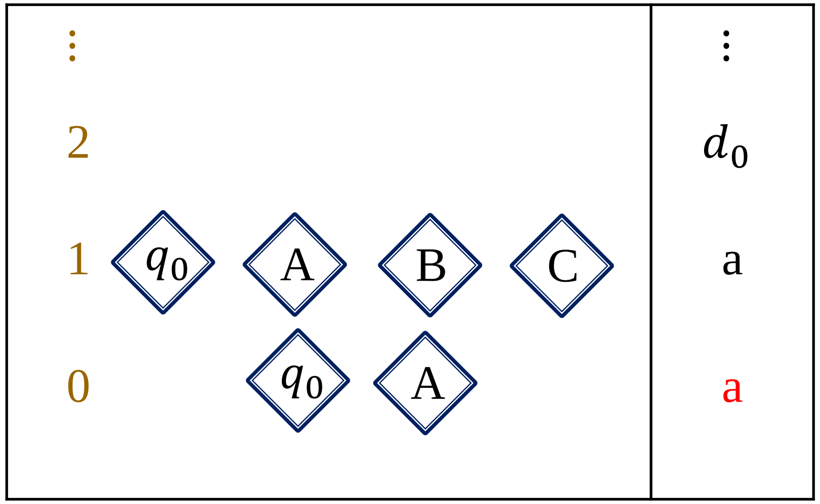


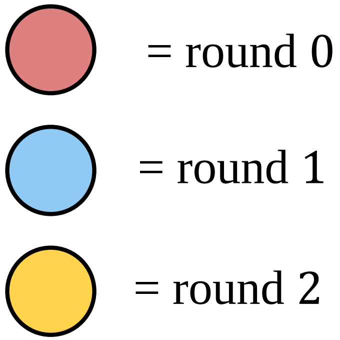
An example of round-based register protocol



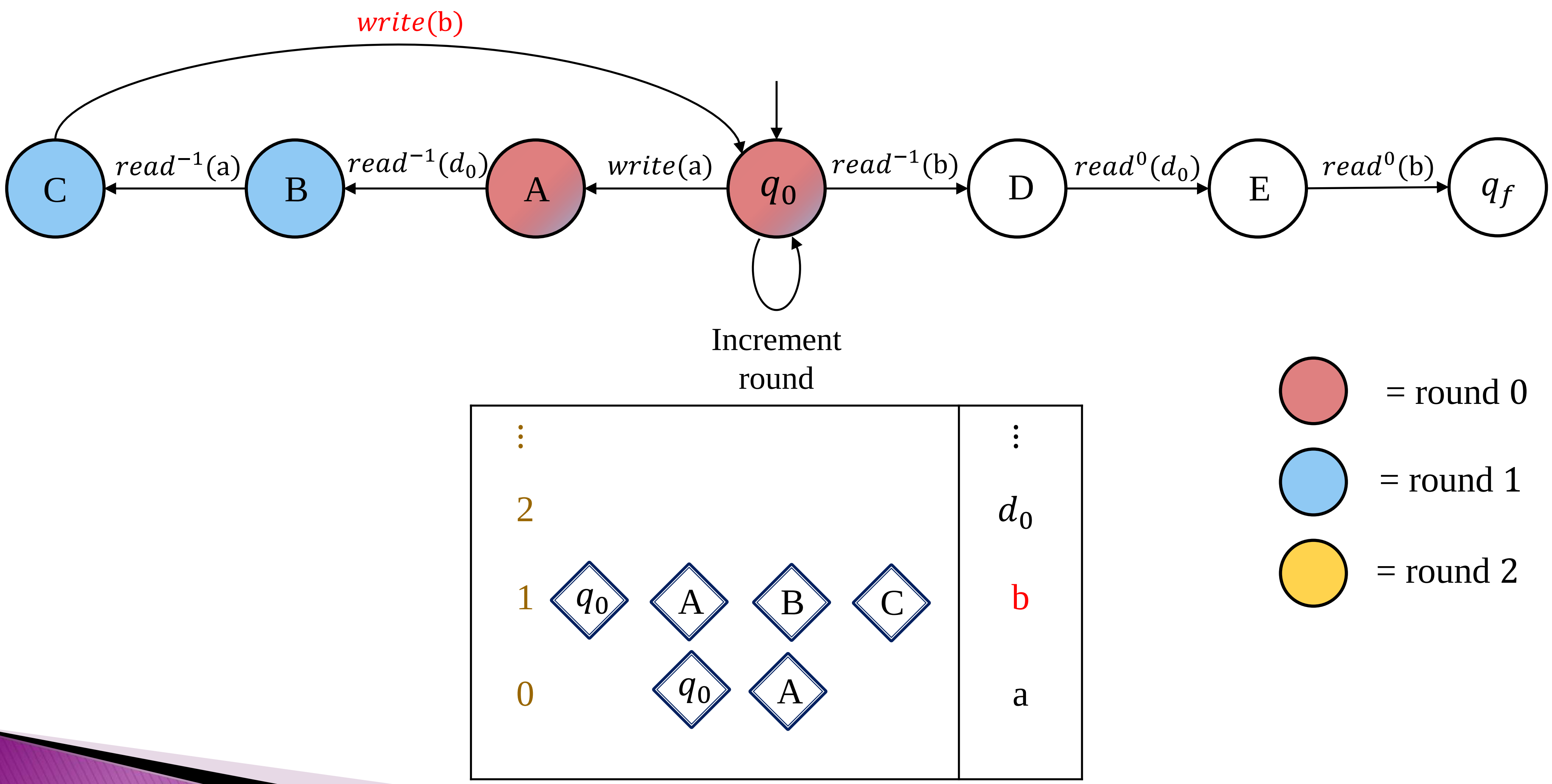
An example of round-based register protocol



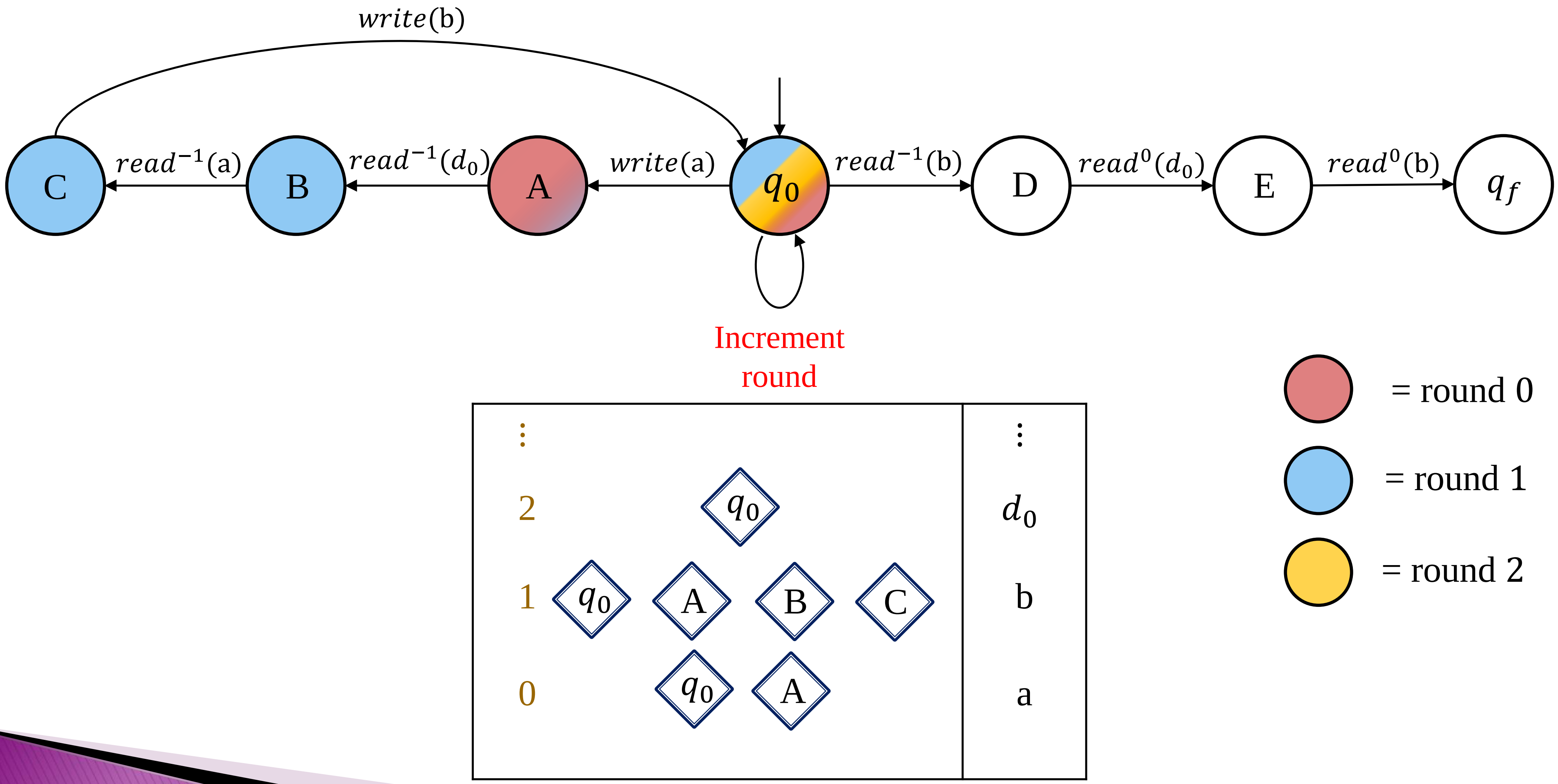




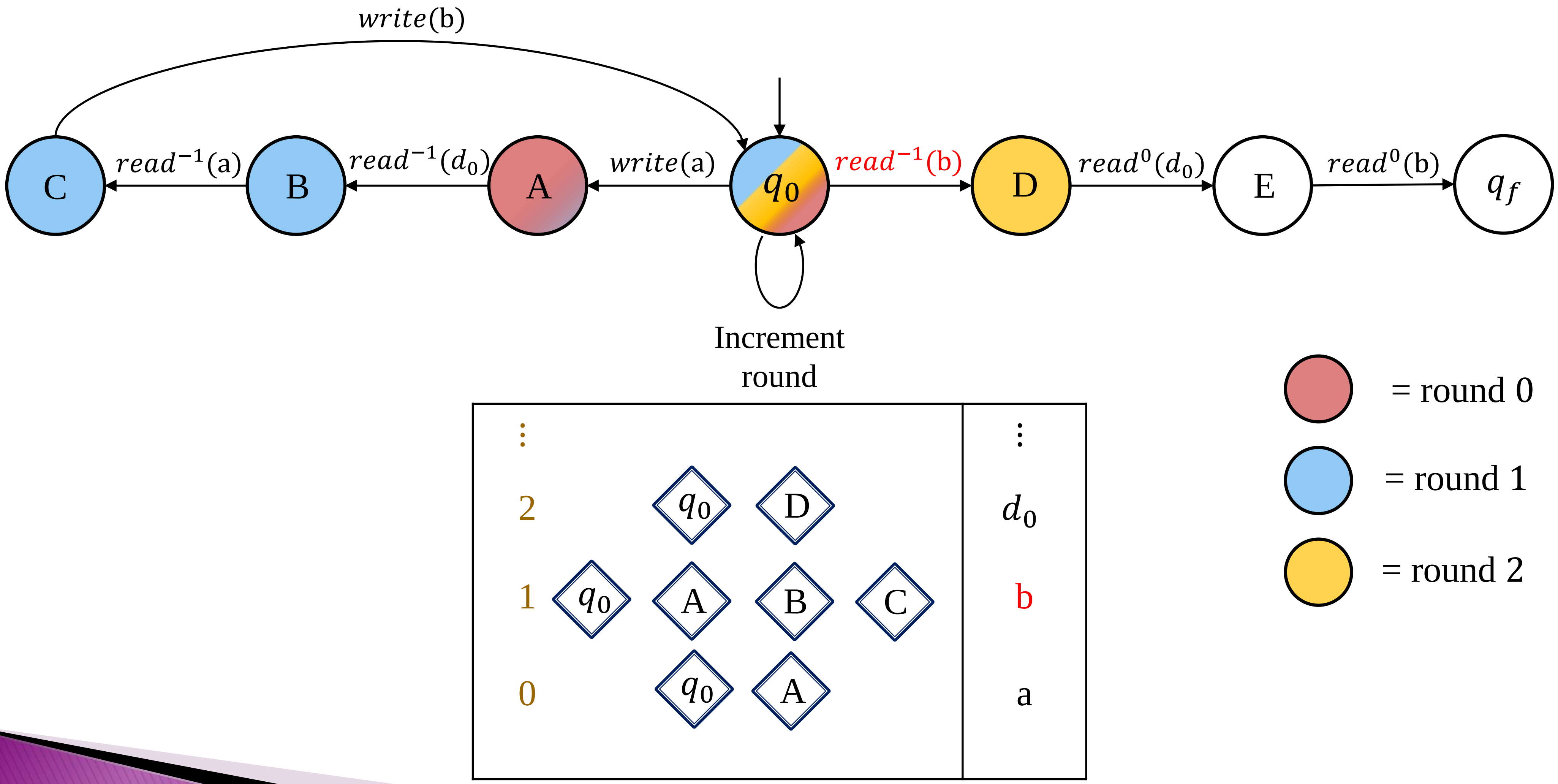
An example of round-based register protocol



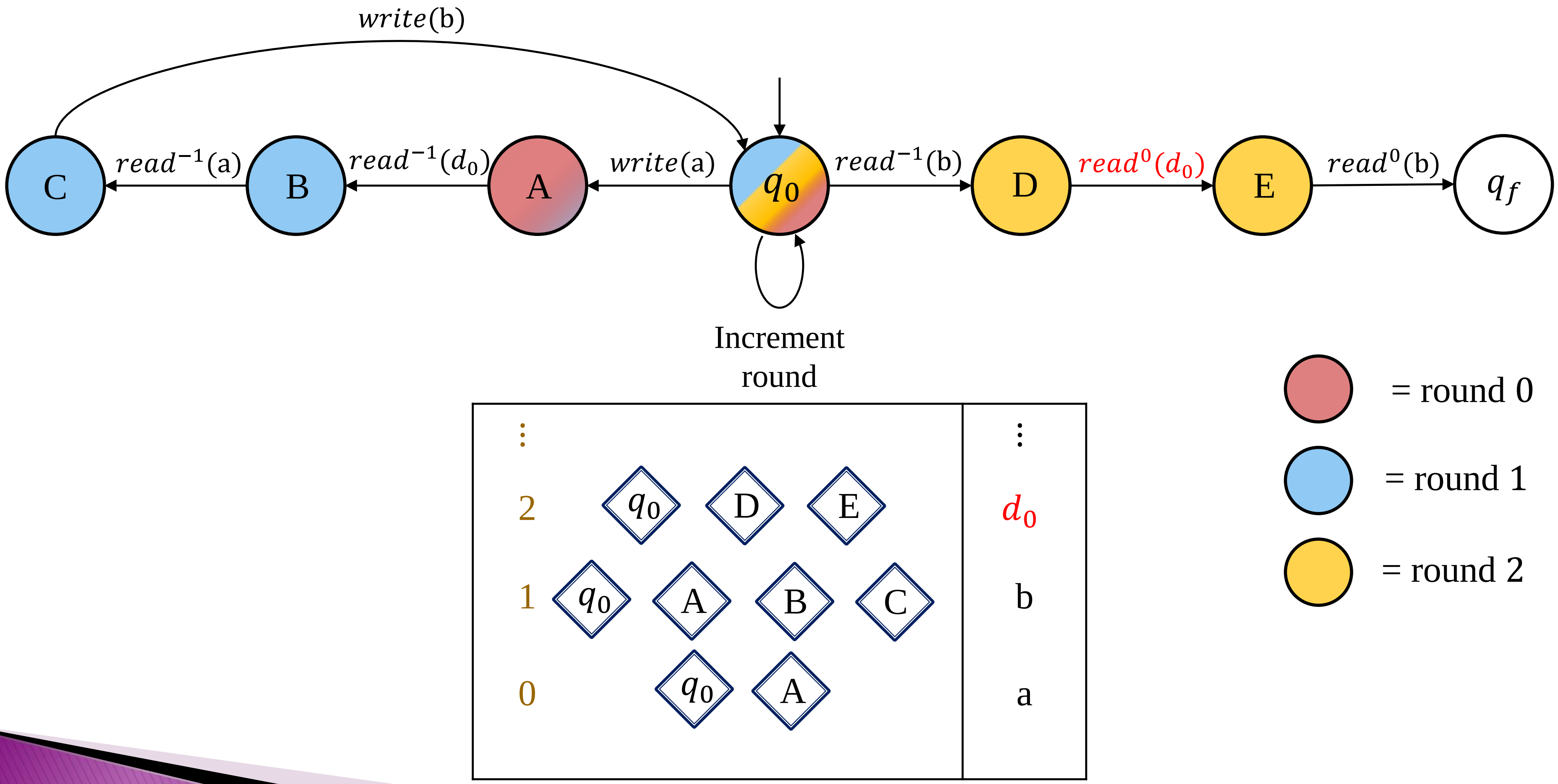
An example of round-based register protocol



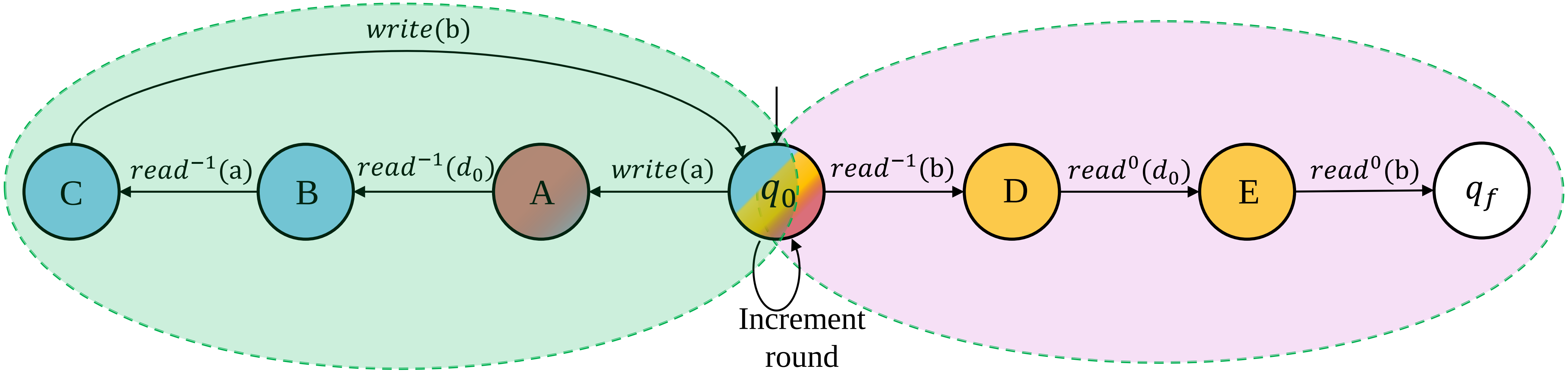
An example of round-based register protocol



An example of round-based register protocol



An example of round-based register protocol



To write b to $\mathbf{reg}[k]$, one must write to $\mathbf{reg}[k]$ while $\mathbf{reg}[k-1]$ still has value d_0

To cover q_f at round k, one must have written b to $\mathbf{reg}[k-1]$ while $\mathbf{reg}[k]$ still has value d_0

q_f cannot be covered !

Reachability problems in round-based setting

Round-based COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \exists k\ \ \gamma(q_f, k) > 0$?

There exists a round k such that some process is at round k and on state q_f

Reachability problems in round-based setting

Round-based COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \exists k\ \ \gamma(q_f, k) > 0$?

Round-based TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \forall k, \forall q \neq q_f,\ \gamma(q, k) = 0$?

Every process is on state q_f regardless of its round

Reachability problems in round-based setting

Round-based COVER: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \exists k\ \ \gamma\left(q_f, k\right) > 0$?

Round-based TARGET: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \forall k, \forall q \neq q_f,\ \gamma(q,k) = 0$?

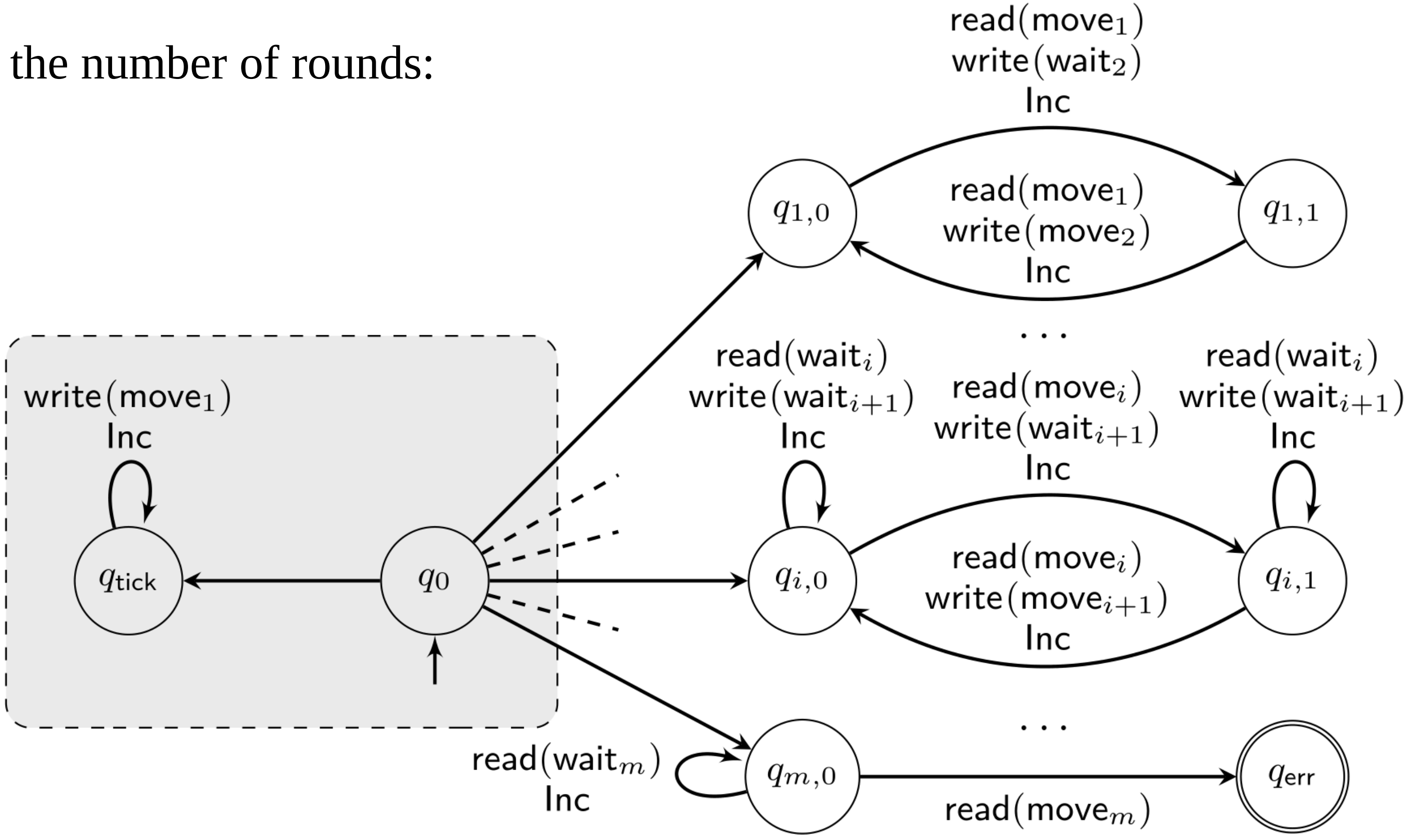
Round-based PRP: $\exists n,\ \exists \gamma_0,\ \exists \rho\colon \gamma_0 \rightarrow^* \gamma\ ,\ \ \gamma \vDash \psi$?

with ψ a first-order formula on rounds with no nested quantifiers

Examples: $\psi =$ ``$\underbrace{\exists k\ (\#(q_1, k+1) > 0 \wedge \mathbf{reg}_i[k] = d)}_{\text{At some round, there is a process on state } q_1 \text{ while register } i \text{ of previous round has value } d}\ \vee\ \underbrace{\forall k\ \#(q_0, k) = 0}_{\text{no process is on } q_0}$''

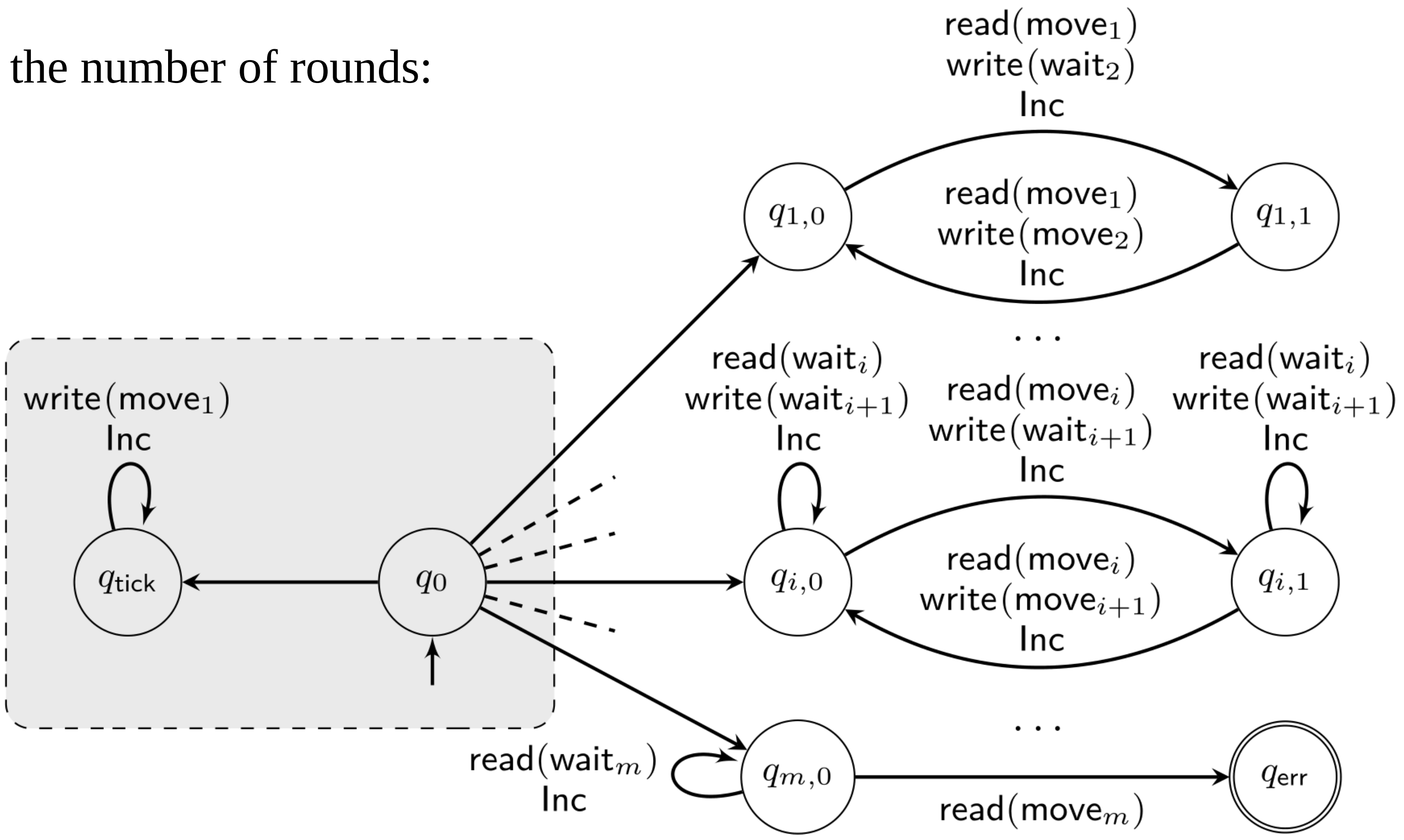
A challenge: exponential lower bounds

Exponential lower bounds on the number of rounds:



A challenge: exponential lower bounds

Exponential lower bounds on the number of rounds:



Similar lower bounds for the number of processes and of *active* rounds

Complexity results

Theorem[9]*:* Round-based COVER is PSPACE-hard.

9. Bertrand, N., Markey, N., Sankur, O., Waldburger, N.: *Parameterized safety verification of round-based shared-memory systems.* ICALP, 2022

Complexity results

Theorem[9]*:* Round-based COVER is PSPACE-hard.

Theorem[9][10]*:* Round-based PRP is PSPACE-complete.

9. Bertrand, N., Markey, N., Sankur, O., Waldburger, N.:
Parameterized safety verification of round-based shared-memory systems. ICALP, 2022
10. W: *Checking Presence Reachability Properties on Parameterized Shared-Memory Systems*, submitted

Complexity results

Theorem[9]*:* Round-based COVER is PSPACE-hard.

Theorem[9][10]*:* Round-based PRP is PSPACE-complete.

Challenge: the number of rounds relevant at the same time may need to be exponential.

9. Bertrand, N., Markey, N., Sankur, O., Waldburger, N.: *Parameterized safety verification of round-based shared-memory systems.* ICALP, 2022
10. W: *Checking Presence Reachability Properties on Parameterized Shared-Memory Systems*, submitted

A non-deterministic polynomial-space algorithm

Witness execution: $\sigma_0 \xrightarrow{\theta_0} \sigma_1 \xrightarrow{\theta_1} \sigma_2 \xrightarrow{\theta_2} \sigma_3 \xrightarrow{\theta_3} \sigma_4 \xrightarrow{\theta_4} \sigma_5 \xrightarrow{\theta_5} \sigma_6 \xrightarrow{\theta_6} \sigma_7 \models \psi$

A non-deterministic polynomial-space algorithm

Witness execution: $\sigma_0 \xrightarrow{\theta_0} \sigma_1 \xrightarrow{\theta_1} \sigma_2 \xrightarrow{\theta_2} \sigma_3 \xrightarrow{\theta_3} \sigma_4 \xrightarrow{\theta_4} \sigma_5 \xrightarrow{\theta_5} \sigma_6 \xrightarrow{\theta_6} \sigma_7 \vDash \psi$

$$\Downarrow$$

Actions:	$\boxed{\theta_0}$	$\boxed{\theta_1}$	$\boxed{\theta_2}$	$\boxed{\theta_3}$	$\boxed{\theta_4}$	$\boxed{\theta_5}$	$\boxed{\theta_6}$
Rounds:	1	4	3	2	0	1	4

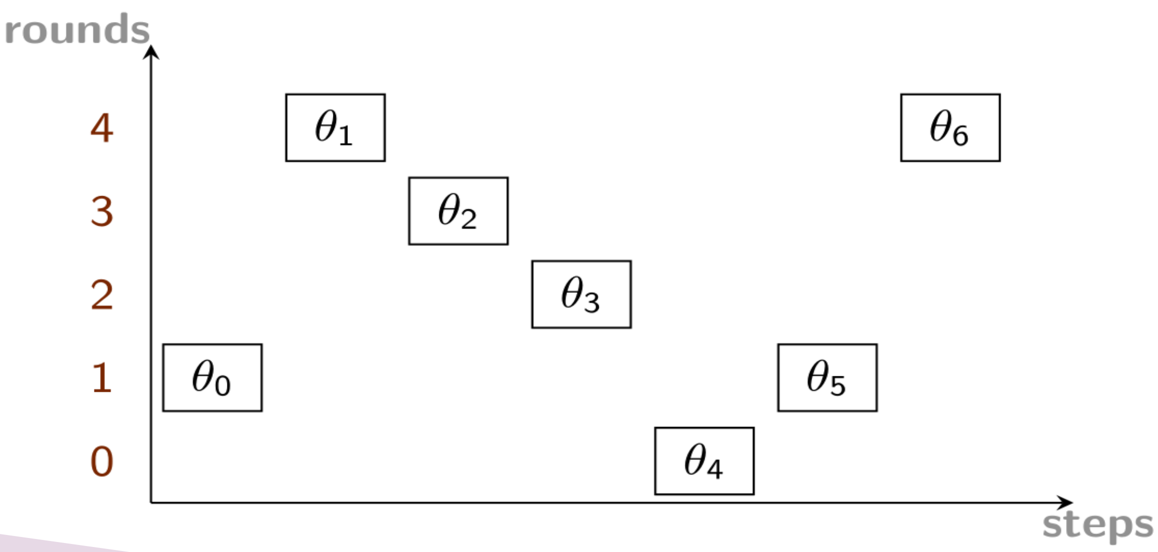
A non-deterministic polynomial-space algorithm

Witness execution: $\sigma_0 \xrightarrow{\theta_0} \sigma_1 \xrightarrow{\theta_1} \sigma_2 \xrightarrow{\theta_2} \sigma_3 \xrightarrow{\theta_3} \sigma_4 \xrightarrow{\theta_4} \sigma_5 \xrightarrow{\theta_5} \sigma_6 \xrightarrow{\theta_6} \sigma_7 \models \psi$

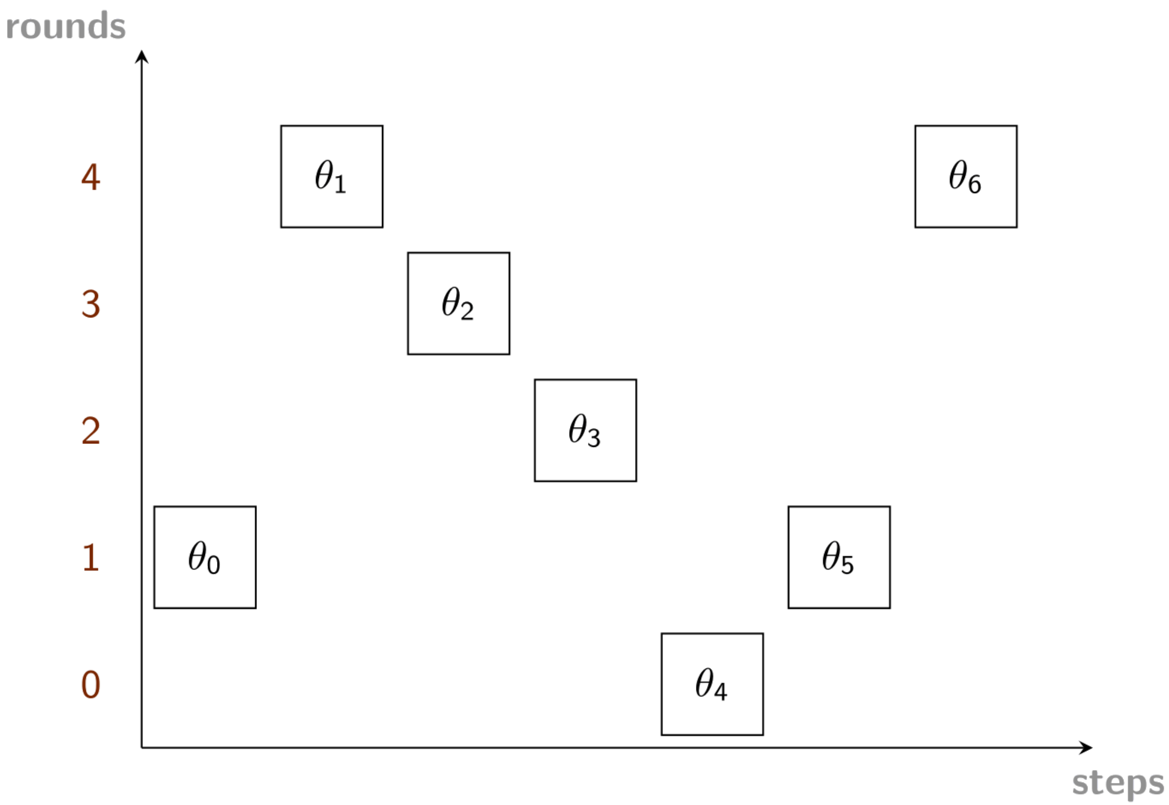
$\Downarrow$

Actions:	$\boxed{\theta_0}$	$\boxed{\theta_1}$	$\boxed{\theta_2}$	$\boxed{\theta_3}$	$\boxed{\theta_4}$	$\boxed{\theta_5}$	$\boxed{\theta_6}$
Rounds:	1	4	3	2	0	1	4

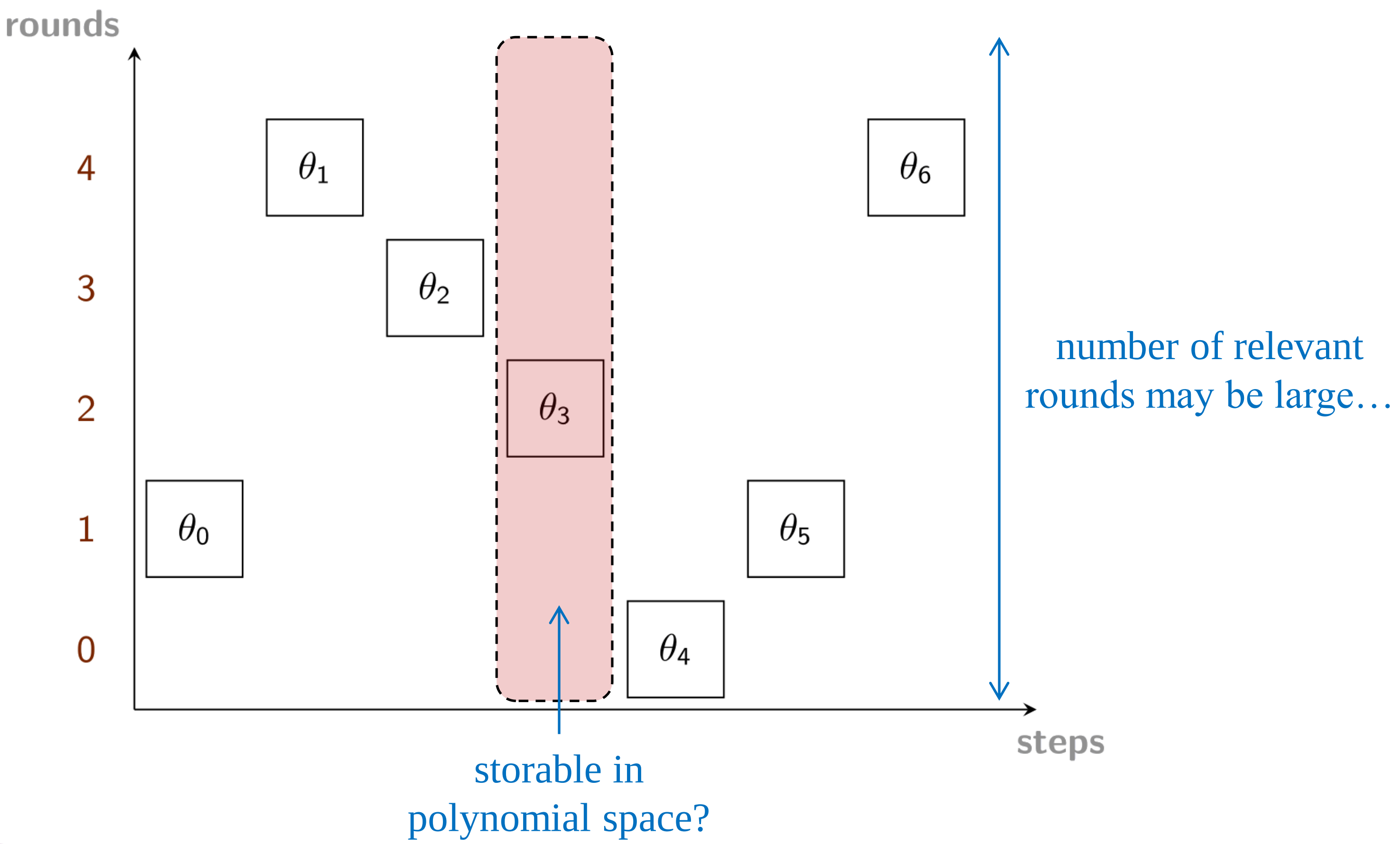
$\Downarrow$



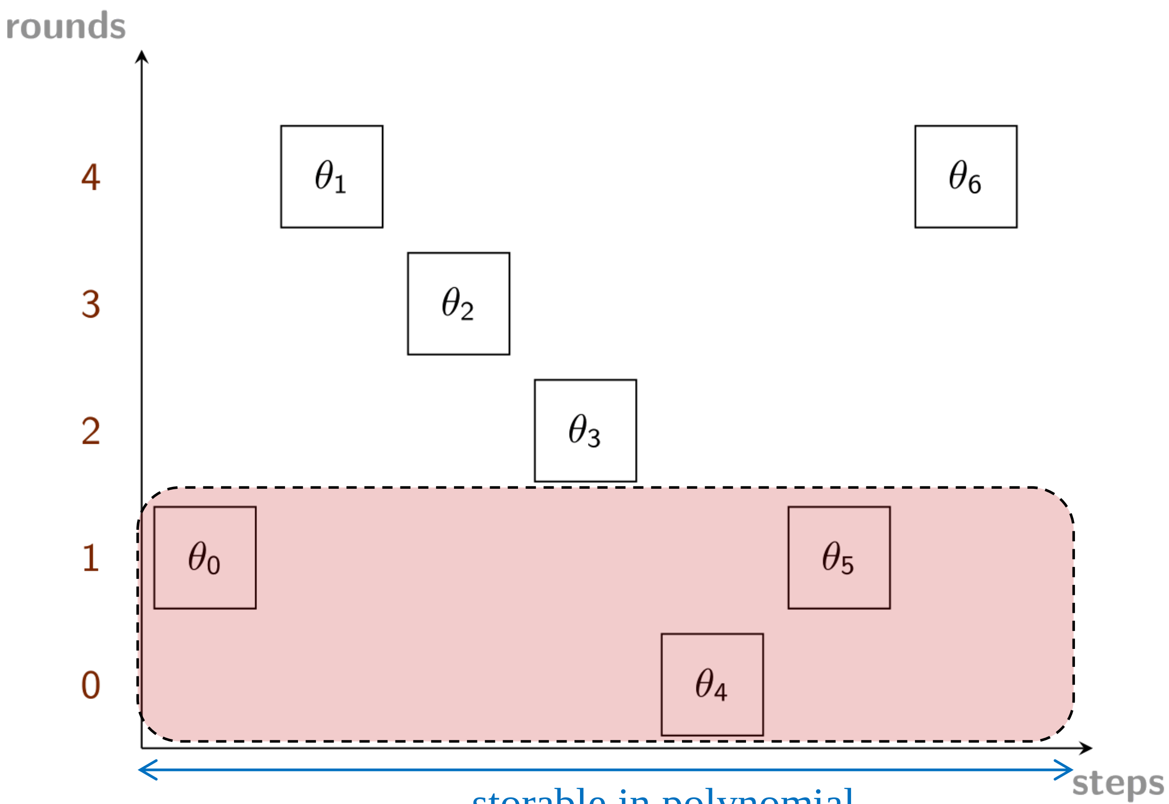
A non-deterministic polynomial-space algorithm



A non-deterministic polynomial-space algorithm



A non-deterministic polynomial-space algorithm

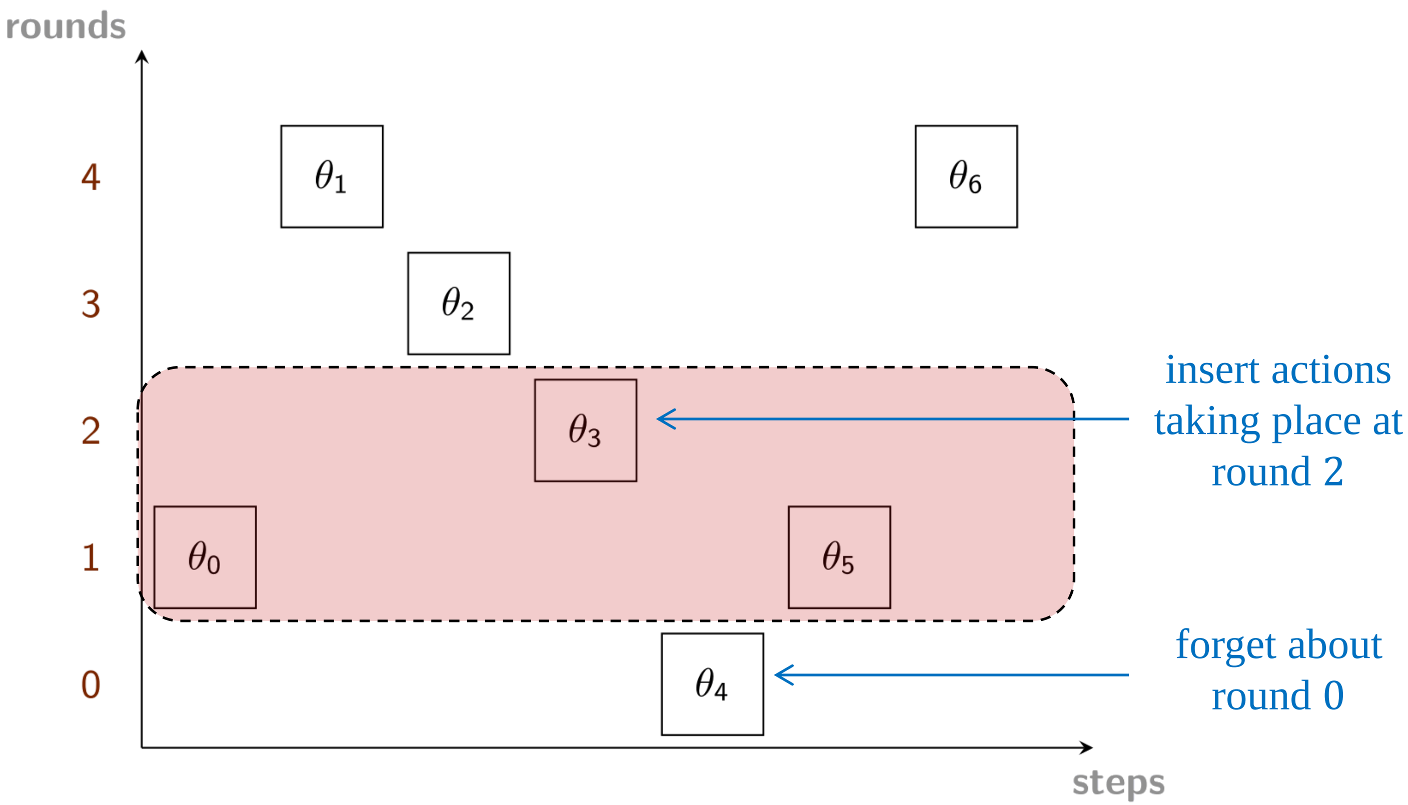


storable in polynomial space using abstract representation

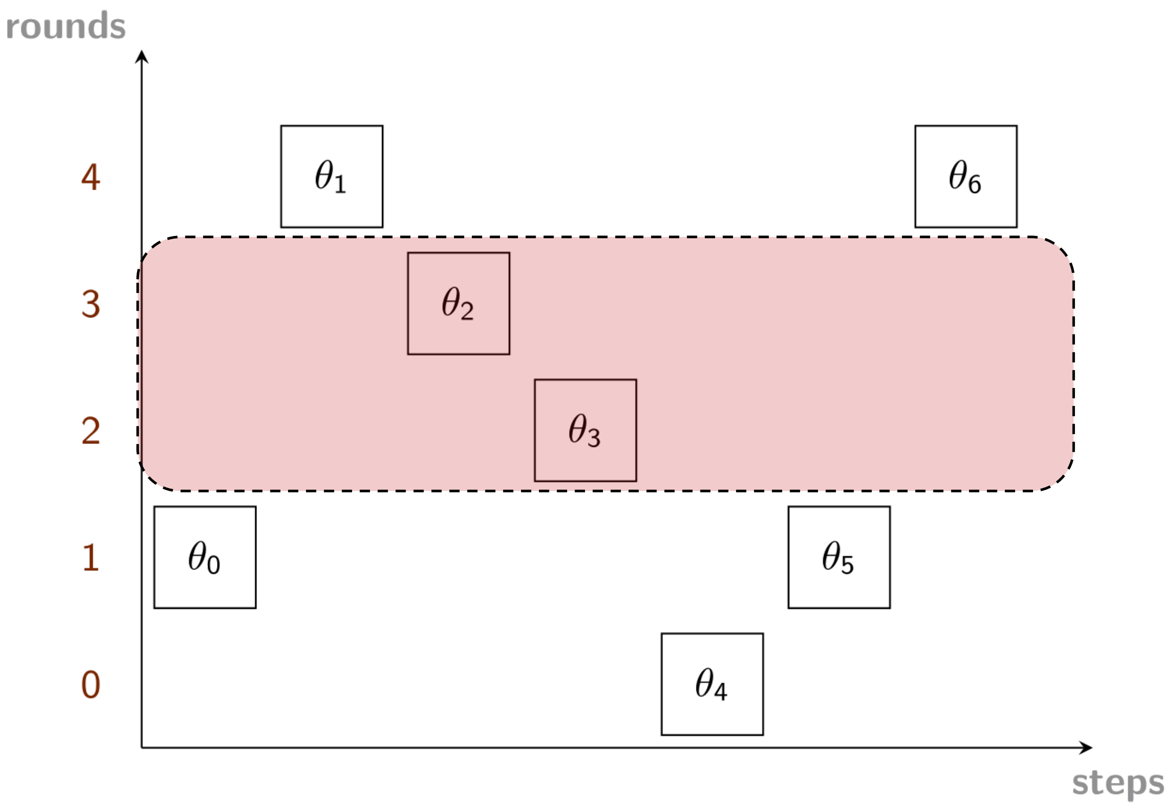
sliding window on $v + 1$ rounds where v is the highest i such that some $read^{-i}(x)$ appears in the protocol

v is assumed to be given in unary (here $v = 1$)

A non-deterministic polynomial-space algorithm

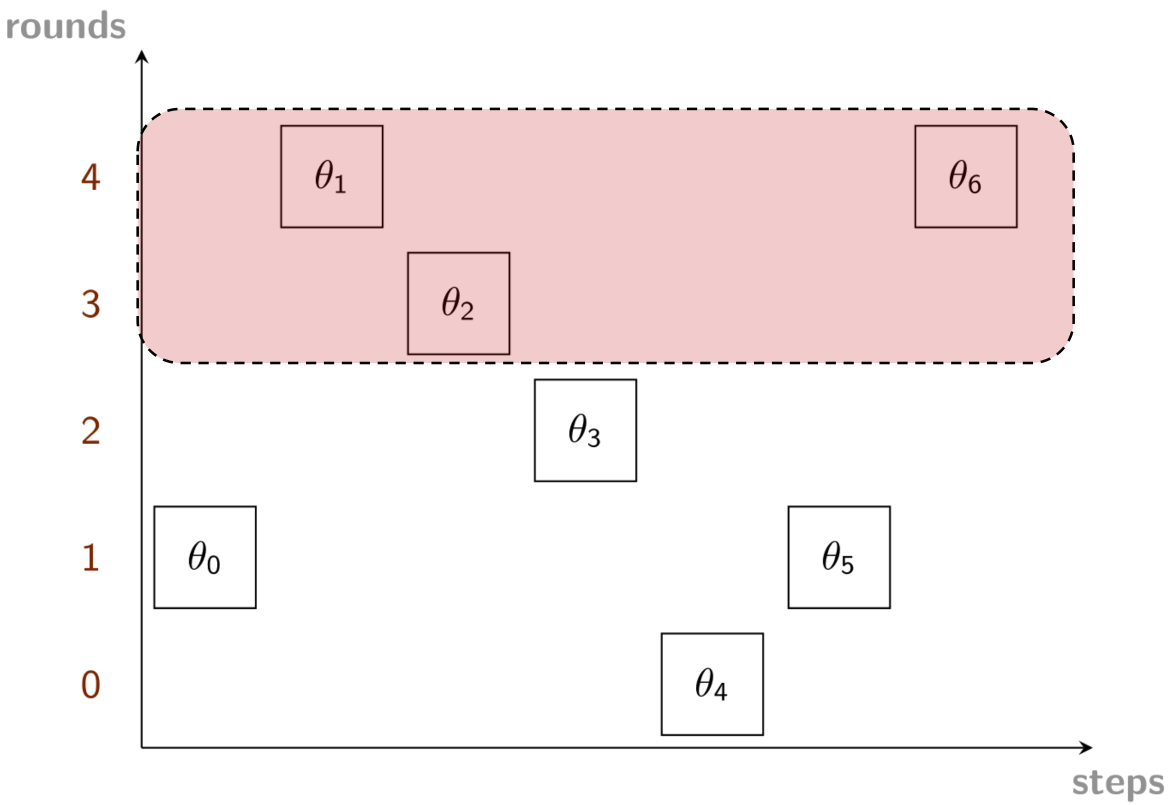


A non-deterministic polynomial-space algorithm



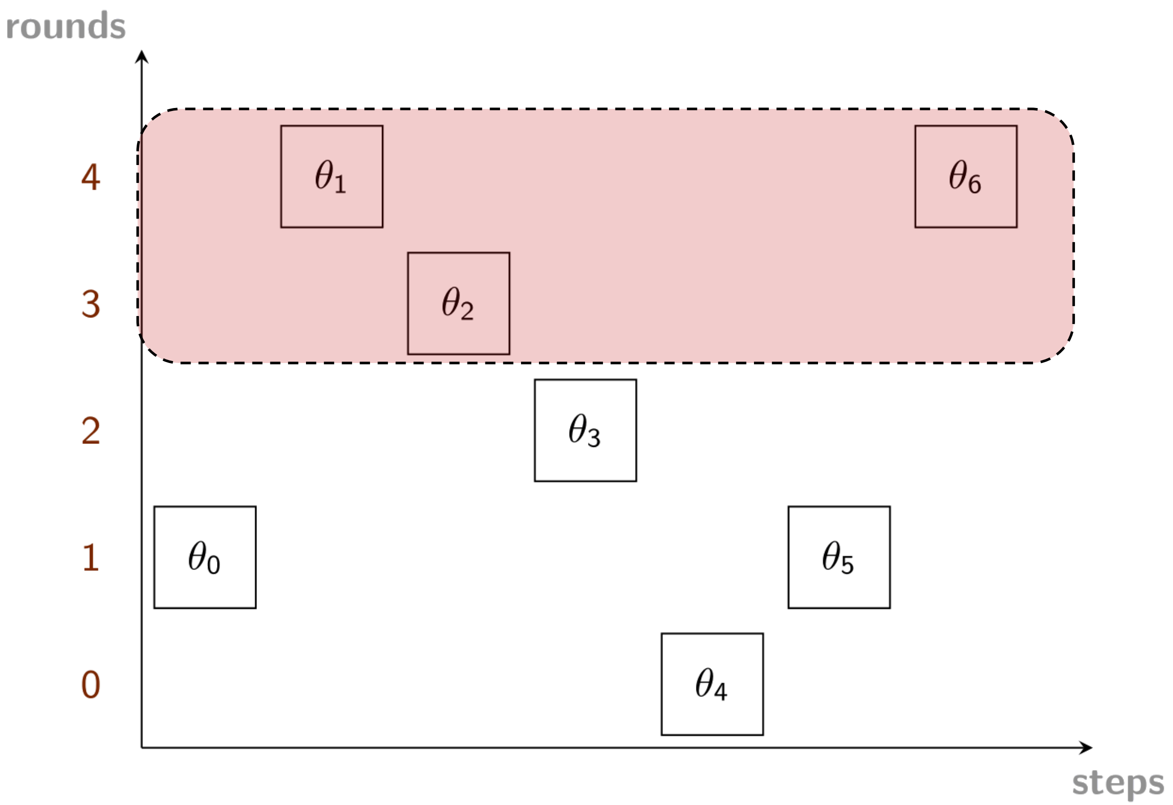
A non-deterministic polynomial-space algorithm

As the execution is guessed, we progressively guess why the configuration reached will satisfy ψ.



A non-deterministic polynomial-space algorithm

As the execution is guessed, we progressively guess why the configuration reached will satisfy ψ.



From this algorithm, we obtain exponential upper bounds on the number of processes and rounds needed.

Round-based shared-memory systems with stochastic schedulers

Fairness for round-based systems

Many consensus algorithms rely on good luck for termination.

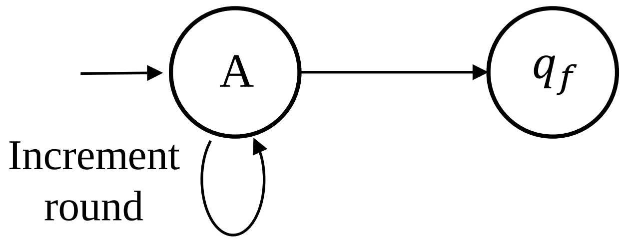
First idea: considering *fair* executions.

Fairness for round-based systems

Many consensus algorithms rely on good luck for termination.

First idea: considering *fair* executions.

Definition 1: For a given k, any transition that is available infinitely often at round k is taken infinitely often.



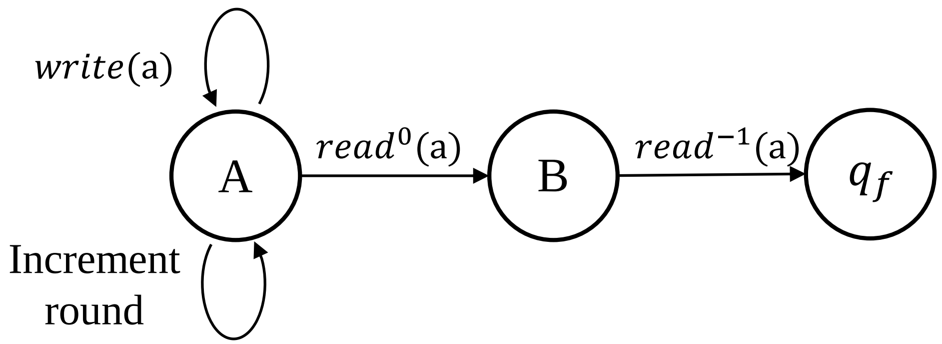
Fairness for round-based systems

Many consensus algorithms rely on good luck for termination.

First idea: considering *fair* executions.

Definition 1: For a given k, any transition that is available infinitely often at round k is taken infinitely often.

Definition 2: Any transition that is available infinitely often overall is taken infinitely often.



Fairness for round-based systems

Many consensus algorithms rely on good luck for termination.

First idea: considering *fair* executions.

Definition 1: For a given k, any transition that is available infinitely often at round k is taken infinitely often.

Definition 2: Any transition that is available infinitely often overall is taken infinitely often.

Definition 3: Processes tend to perform similar number of steps.

For Aspnes' algorithm, a process must win the race !

Fairness for round-based systems

Many consensus algorithms rely on good luck for termination.

First idea: considering *fair* executions.

Definition 1: For a given k, any transition that is available infinitely often at round k is taken infinitely often.

Definition 2: Any transition that is available infinitely often overall is taken infinitely often.

Definition 3: Processes tend to perform similar number of steps.

For Aspnes' algorithm, a process must win the race !

Definition 4: For every m, every process eventually performs m steps in a row.

For Aspnes' algorithm, a process that is far behind could perform many steps in a row and not decide…

→ We need stochastic schedulers !

Stochastic schedulers

At every step:

- the next process to move is picked uniformly at random among all processes,
- its action is picked uniformly at random among all its available actions.

Almost-sure coverability*: Is it the case that, for n large enough,* $\mathbb{P}_n(\mathit{eventually\ somebody\ on}\ q_f) = 1$ *?*

Almost-sure target*: Is it the case that, for n large enough,* $\mathbb{P}_n(\mathit{eventually\ everybody\ on}\ q_f) = 1$ *?*

Stochastic schedulers

At every step:
- the next process to move is picked uniformly at random among all processes,
- its action is picked uniformly at random among all its available actions.

Almost-sure coverability: *Is it the case that, for n large enough,* $\mathbb{P}_n(\textit{eventually somebody on } q_f) = 1$ *?*

Almost-sure target: *Is it the case that, for n large enough,* $\mathbb{P}_n(\textit{eventually everybody on } q_f) = 1$ *?*

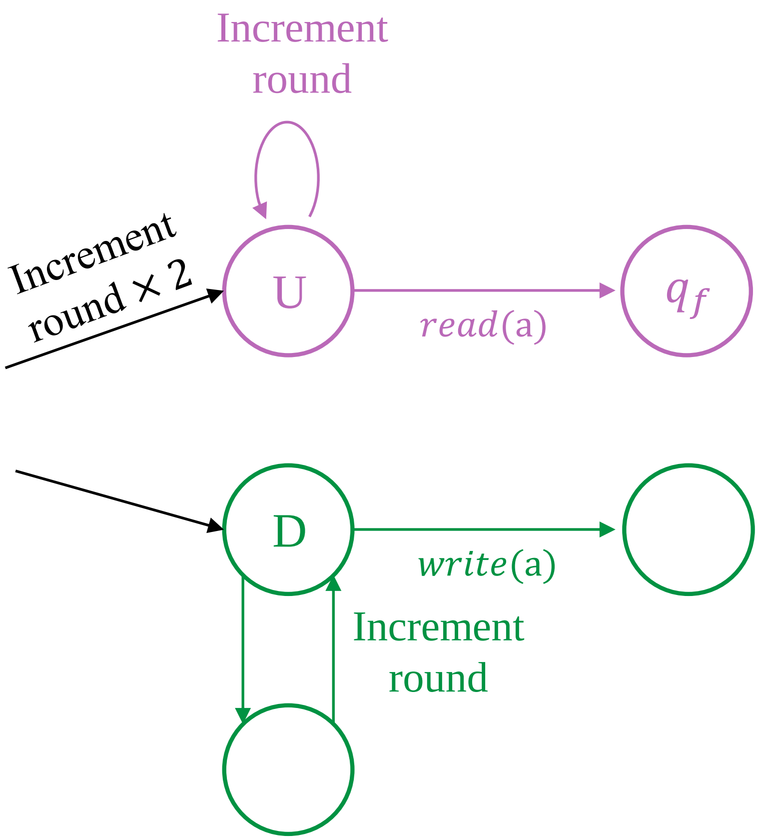
In the roundless case, almost-sure coverability can be stated as a deterministic property:
q_f is covered with probability 1 iff, from every reachable configuration, some process can cover q_f.

Not true for round-based systems…

An annoying example

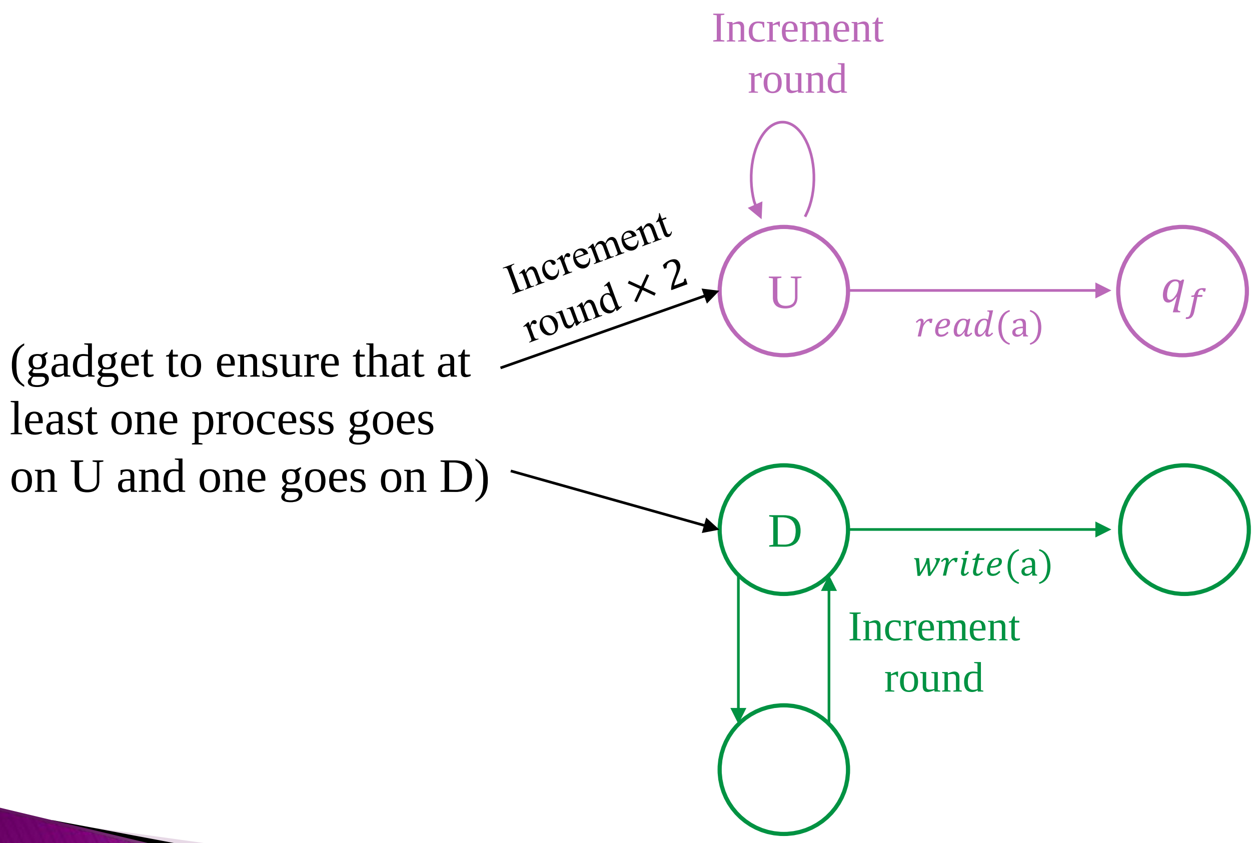
An example where, from any reachable configuration, q_f can still be covered, but q_f is not covered with probability 1.

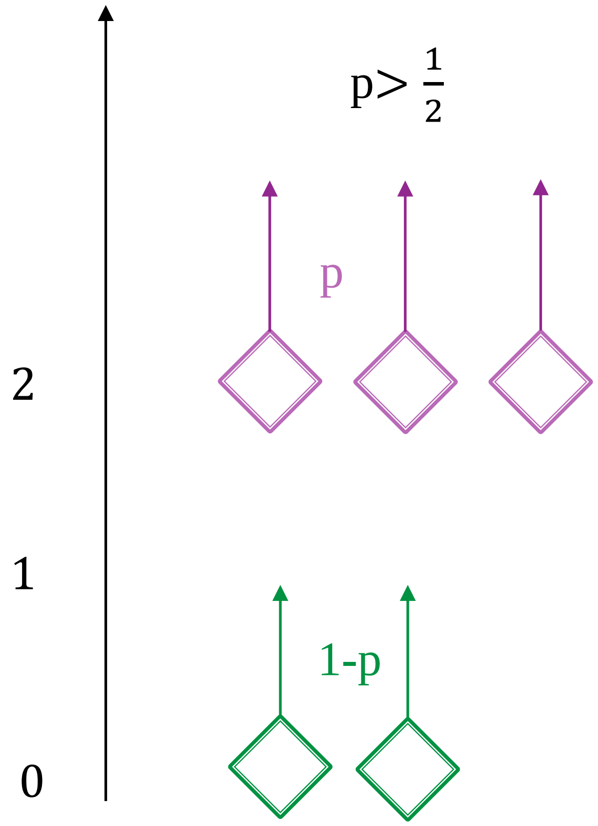
(gadget to ensure that at least one process goes on U and one goes on D)



An annoying example

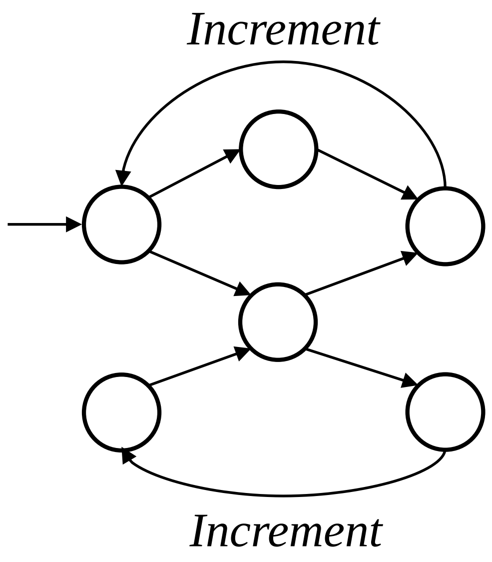
An example where, from any reachable configuration, q_f can still be covered, but q_f is not covered with probability 1.





Preventing random walk behaviors

First idea: Forbid processes to move up at different rates.
Balanced condition: there exists m s.t., on every path of length m of the automaton, there is exactly one increment.



Preventing random walk behaviors

First idea: Forbid processes to move up at different rates.
Balanced condition: there exists m s.t., on every path of length m of the automaton, there is exactly one increment.

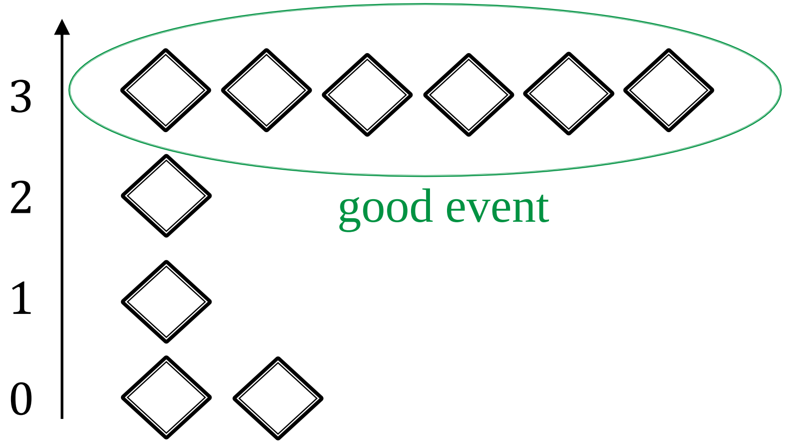
Not enough!
« m processes synchronize on some round » $\sim$ return to zero of a balanced $(m-1)$ - dimensional random walk
If m is large, non-zero probability of never occurring after some point (proven for $m \geq 6$, conjectured for $m \geq 4$)

Preventing random walk behaviors

First idea: Forbid processes to move up at different rates.
Balanced condition: there exists m s.t., on every path of length m of the automaton, there is exactly one increment.

Not enough!
« m processes synchronize on some round » $\sim$ return to zero of a balanced $(m-1)$ - dimensional random walk
If m is large, non-zero probability of never occurring after some point (proved for $m \geq 6$, conjectured for $m \geq 4$)

We can build a protocol where:
- the balanced condition is met
- q_f can be reached from all reachable configurations (for n large enough)
- $\mathbb{P}(q_f \text{ covered}) < 1$ for every n



A stronger restriction

Almost-sure obstruction freedom (ASOF): from any reachable configuration, any process left to play in isolation (all other processes are left idle) reaches q_f with probability 1.

A stronger restriction

Almost-sure obstruction freedom (ASOF): from any reachable configuration, any process left to play in isolation (all other processes are left idle) reaches q_f with probability 1.

For example, it is the case in Aspnes' algorithm: any process acting in isolation will reach blank rounds.

A stronger restriction

Almost-sure obstruction freedom (ASOF): from any reachable configuration, any process left to play in isolation (all other processes are left idle) reaches q_f with probability 1.

For example, it is the case in Aspnes' algorithm: any process acting in isolation will reach blank rounds.

Proposition: Deciding whether a given protocol is ASOF is a PSPACE-complete problem.

A stronger restriction

Almost-sure obstruction freedom (ASOF): from any reachable configuration, any process left to play in isolation (all other processes are left idle) reaches q_f with probability 1.

For example, it is the case in Aspnes' algorithm: any process acting in isolation will reach blank rounds.

Proposition: Deciding whether a given protocol is ASOF is a PSPACE-complete problem.

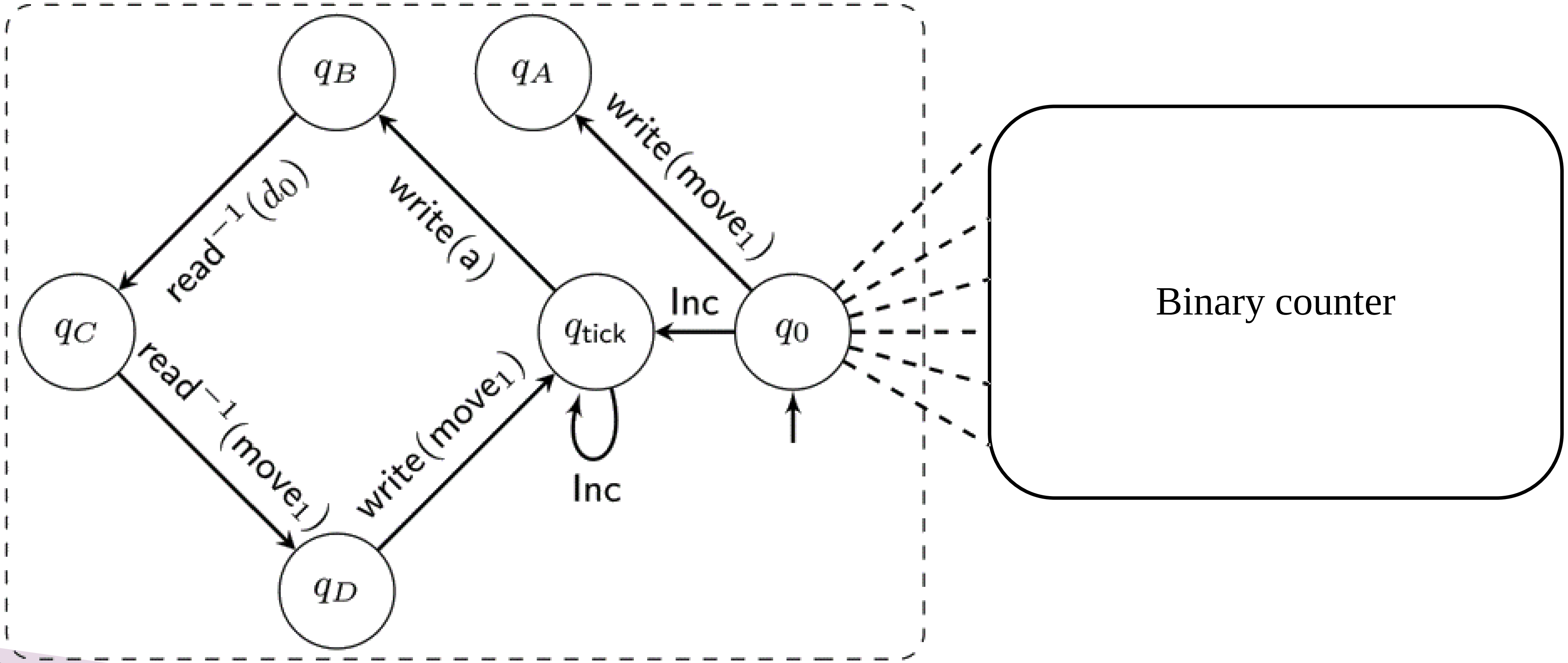
Proposition: If a protocol is ASOF, then for every n, all agents end up in q_f with probability 1 (almost-sure TARGET).

q_f is a deadlocked state

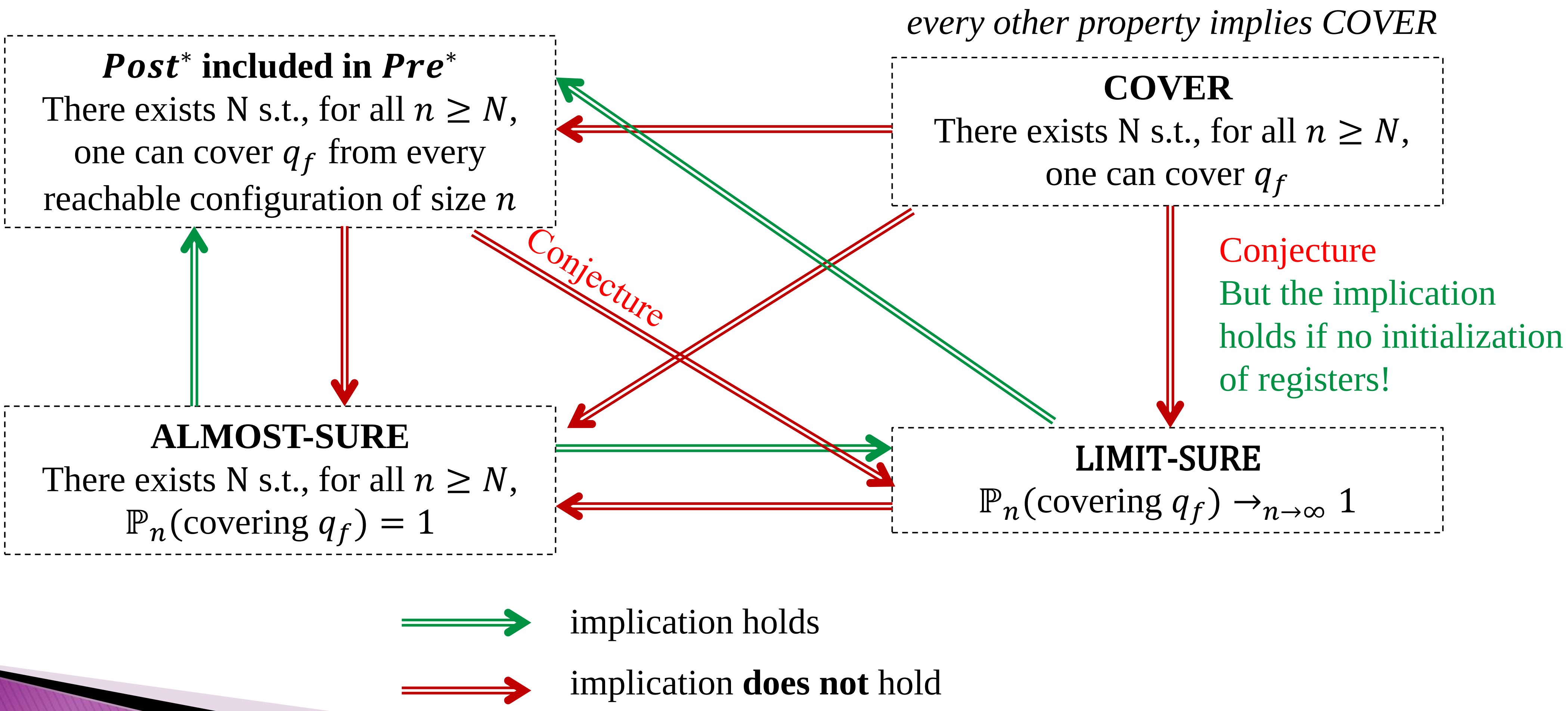
Thanks for your attention !
Any questions?

A challenge: exponential lower bounds

Exponential lower bounds on the number of *active* rounds:

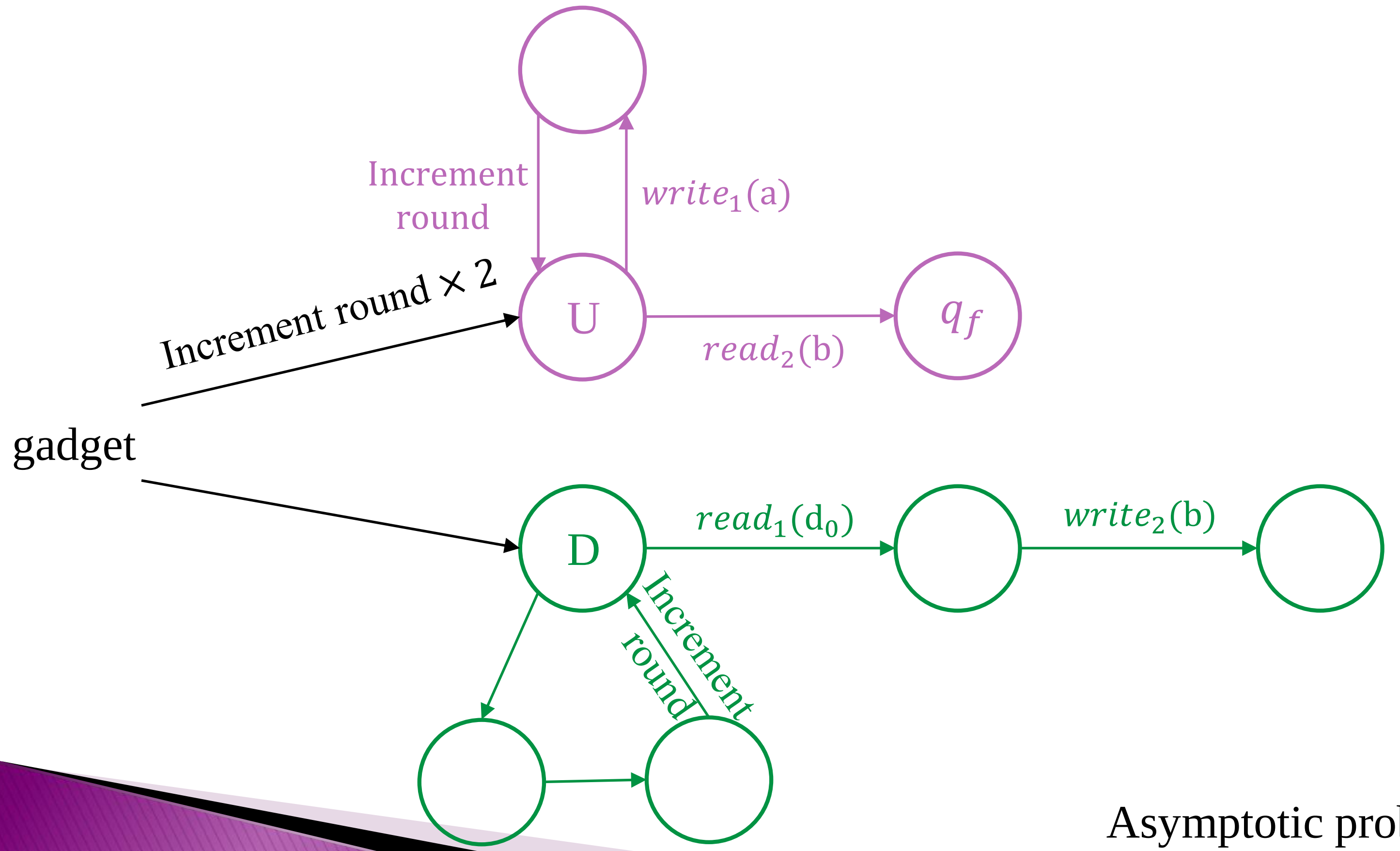


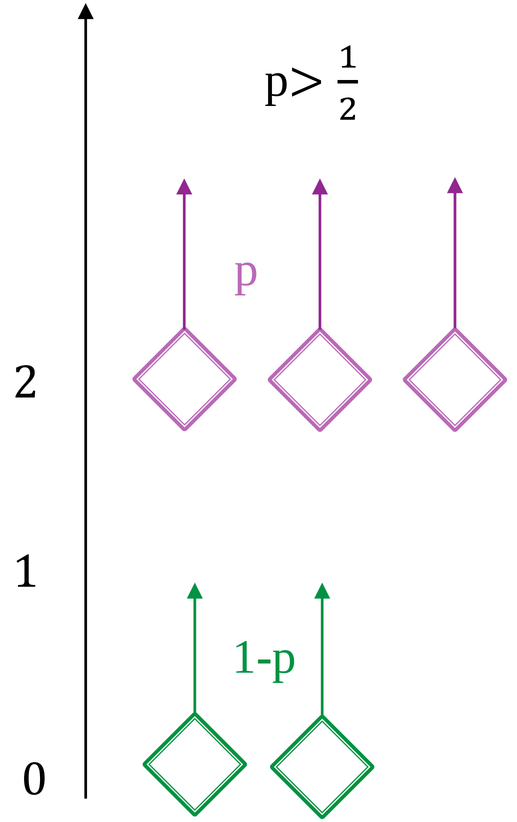
Several negative results



The conjecture

Conjecture: In the following example, $\mathbb{P}_n(\text{covering } q_f) \not\to_{n\to\infty} 1$.





Asymptotic probability that a process in the D region catches up with the highest process in the U region?

Simulations