Checking Presence Reachability Properties on Parameterized Shared-Memory Systems

Nicolas Waldburger
Nathalie Bertrand, Nicolas Markey, Ocan Sankur
SUMO team, Rennes
Univ Rennes, Inria, CNRS, IRISA, France

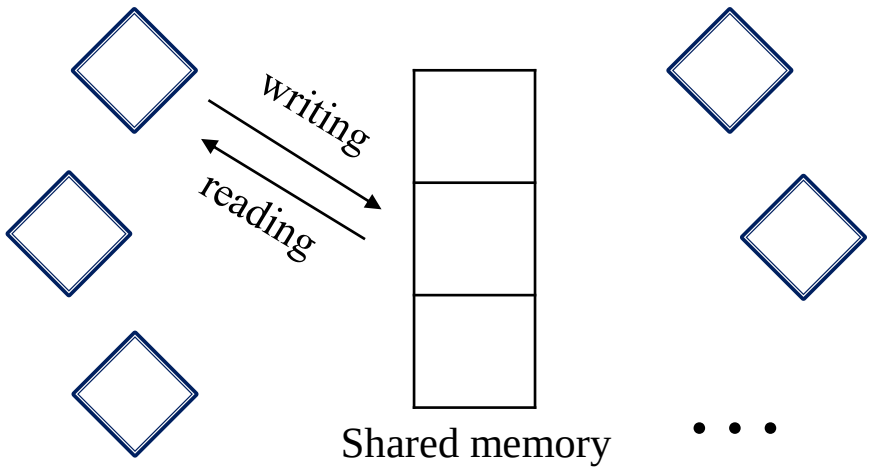
Shared-memory systems

Aim: automated verification of distributed algorithms.

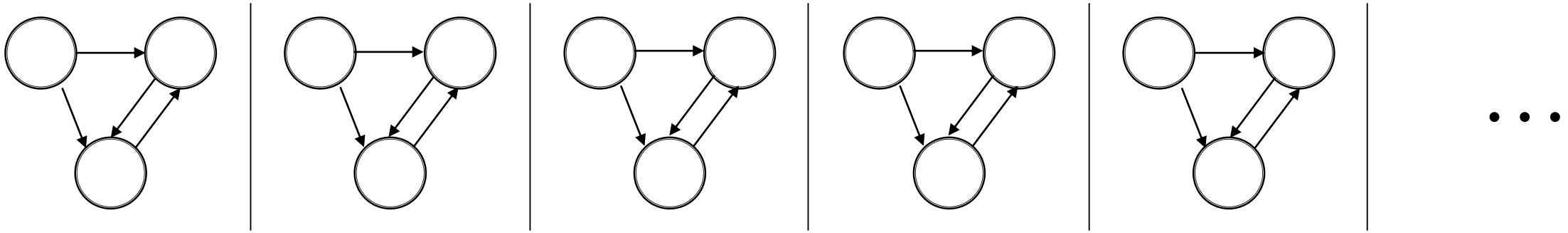
Parameterized verification: arbitrarily large systems.

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]

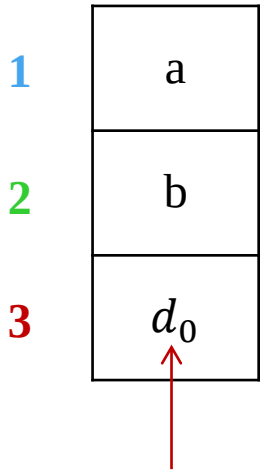


- *Arbitrary* number of processes
- Processes are *identical* agents
- No identifiers: processes are *anonymous*
- Modelled by a single, common *finite automaton*

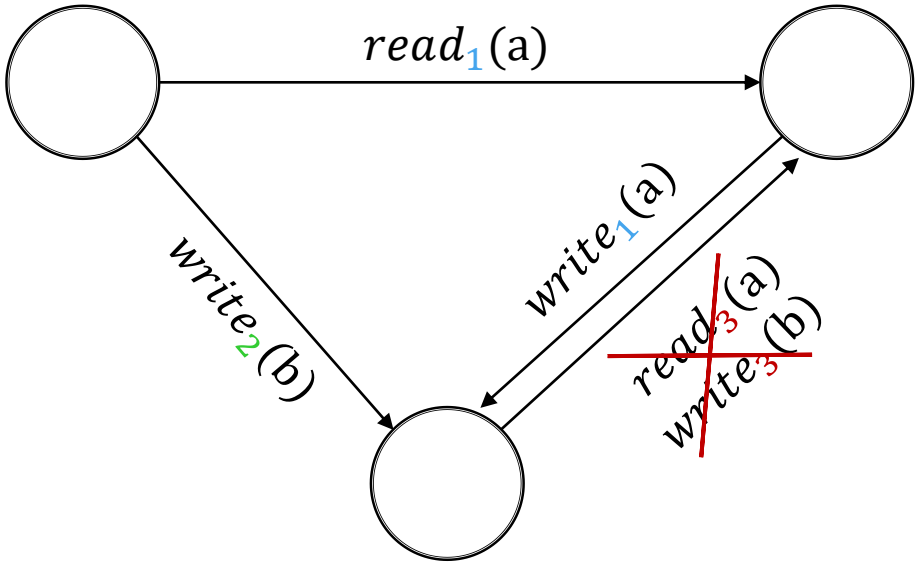
1. Model inspired from: Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems*. Journal of the ACM, 2016

A model for shared-memory systems[1]

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



Registers are *initialized* to value d_0

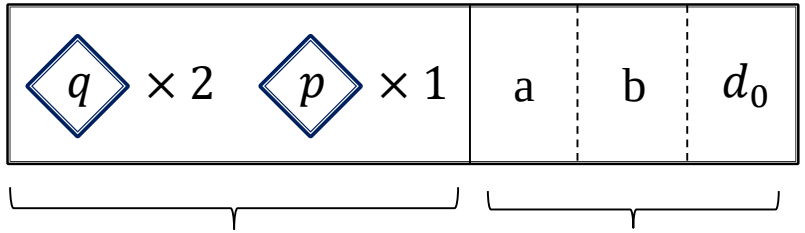


No atomic read/write combinations

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

Semantics

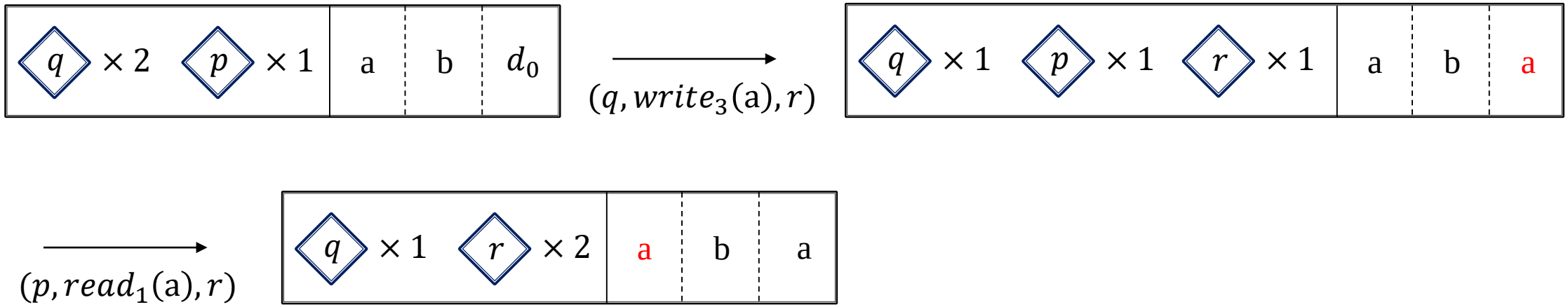
A configuration:



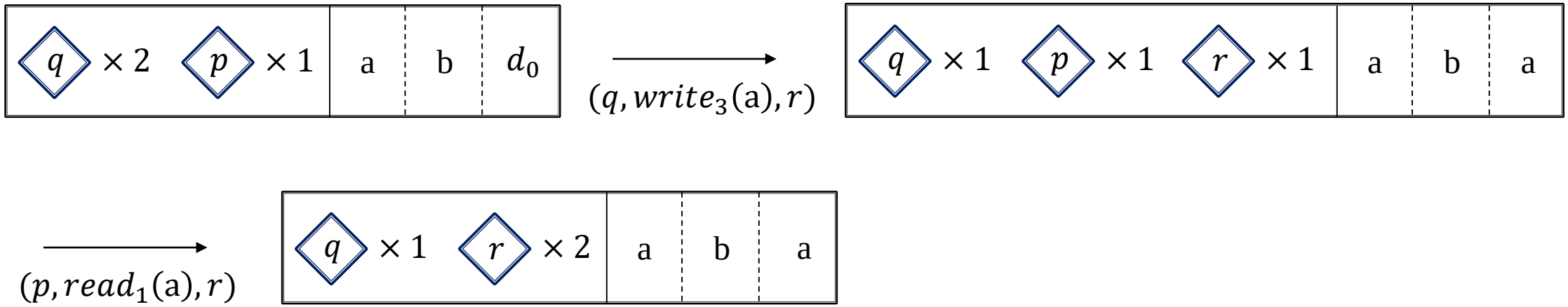
How many process are on each state

Content of the registers

Semantics


q × 2
p × 1
a
b
d_0
$(q, write_3(\mathrm{a}), r)$
q × 1
p × 1
r × 1
a
b
a
$(p, read_1(\mathrm{a}), r)$
q × 1
r × 2
a
b
a

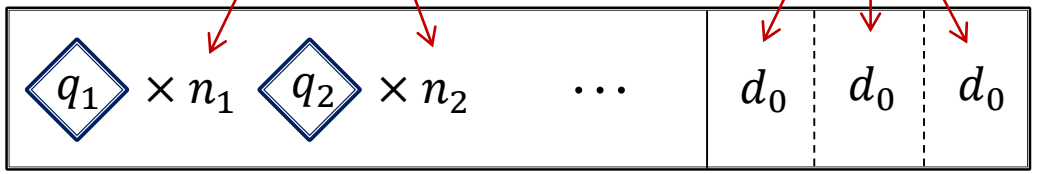
Semantics



Can be arbitrarily large

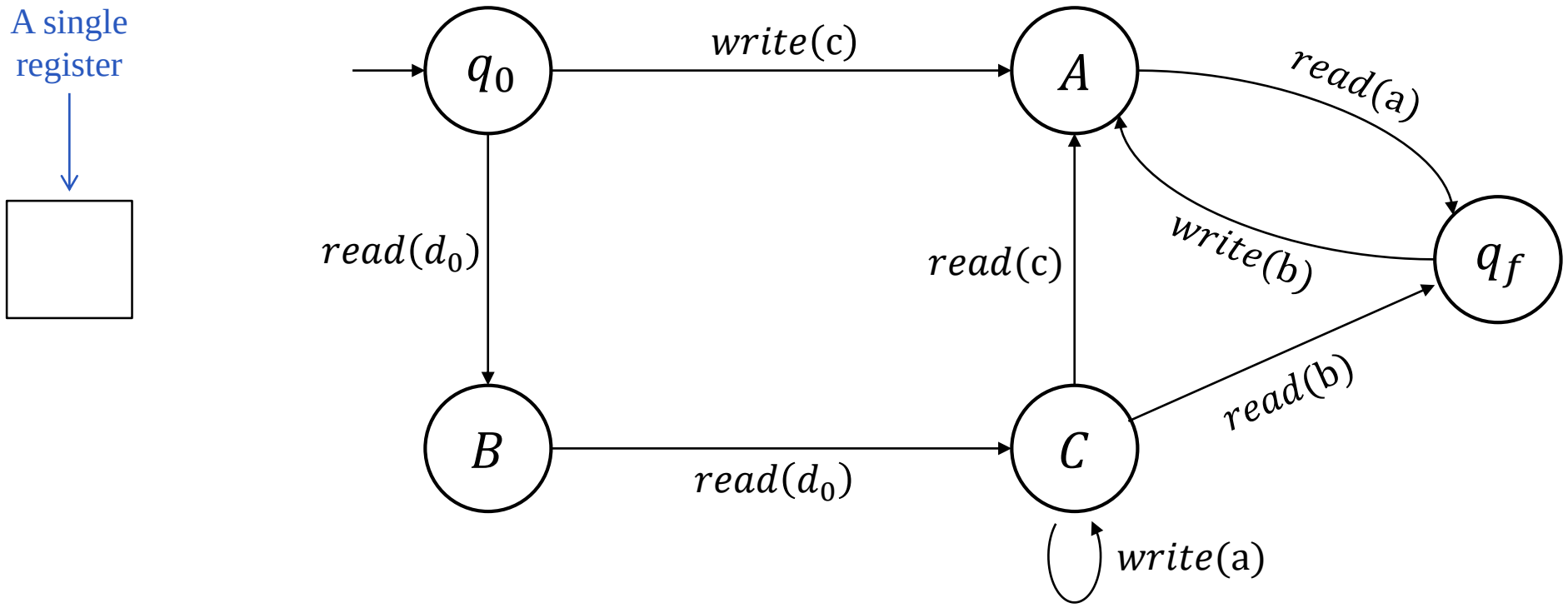
Registers are initialized to d_0

Initial configurations:

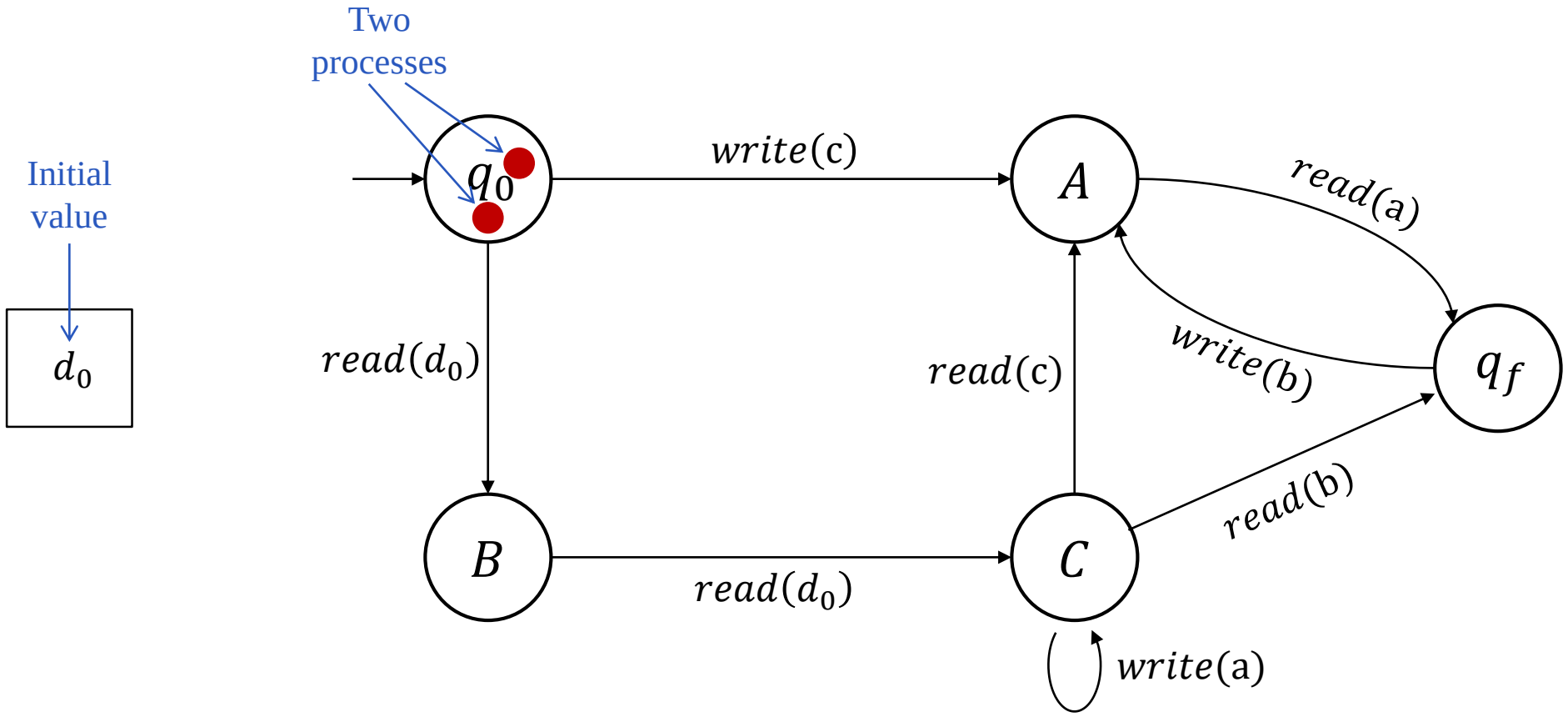


with $n_1, n_2, \ldots \geq 0$ and $q_1, q_2, \ldots$ initial states

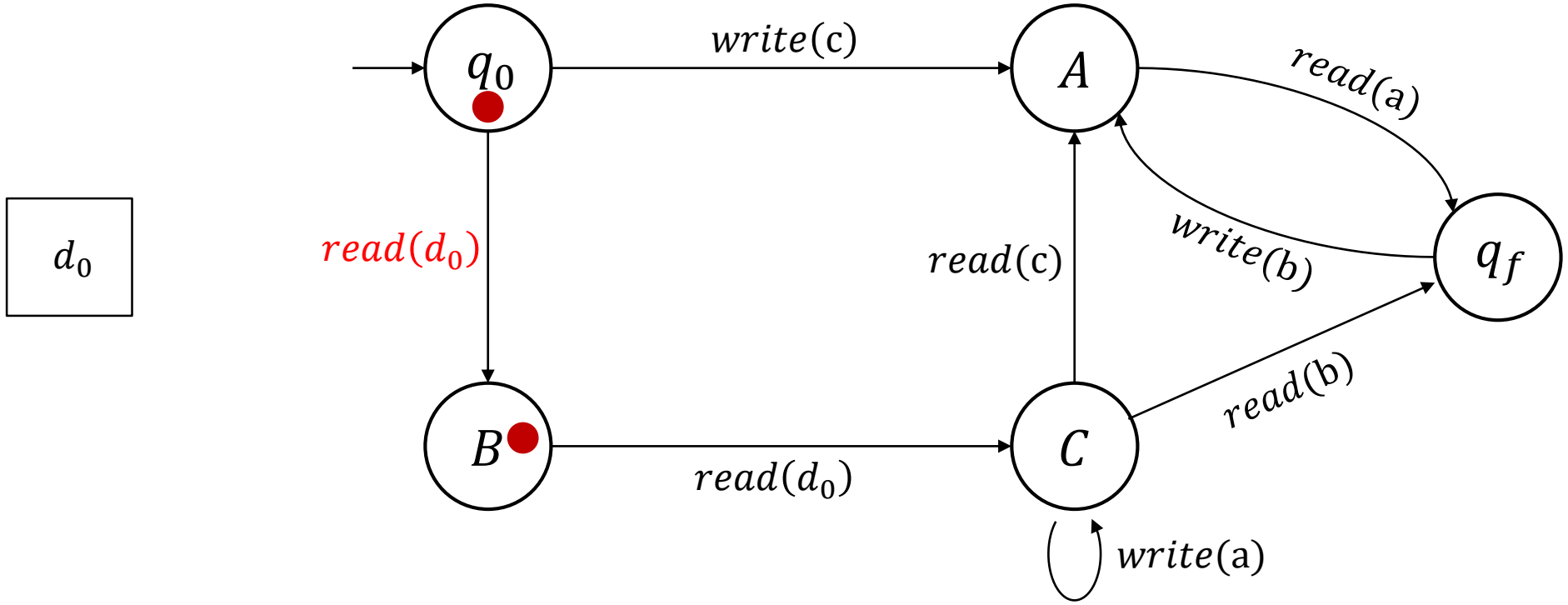
A small example



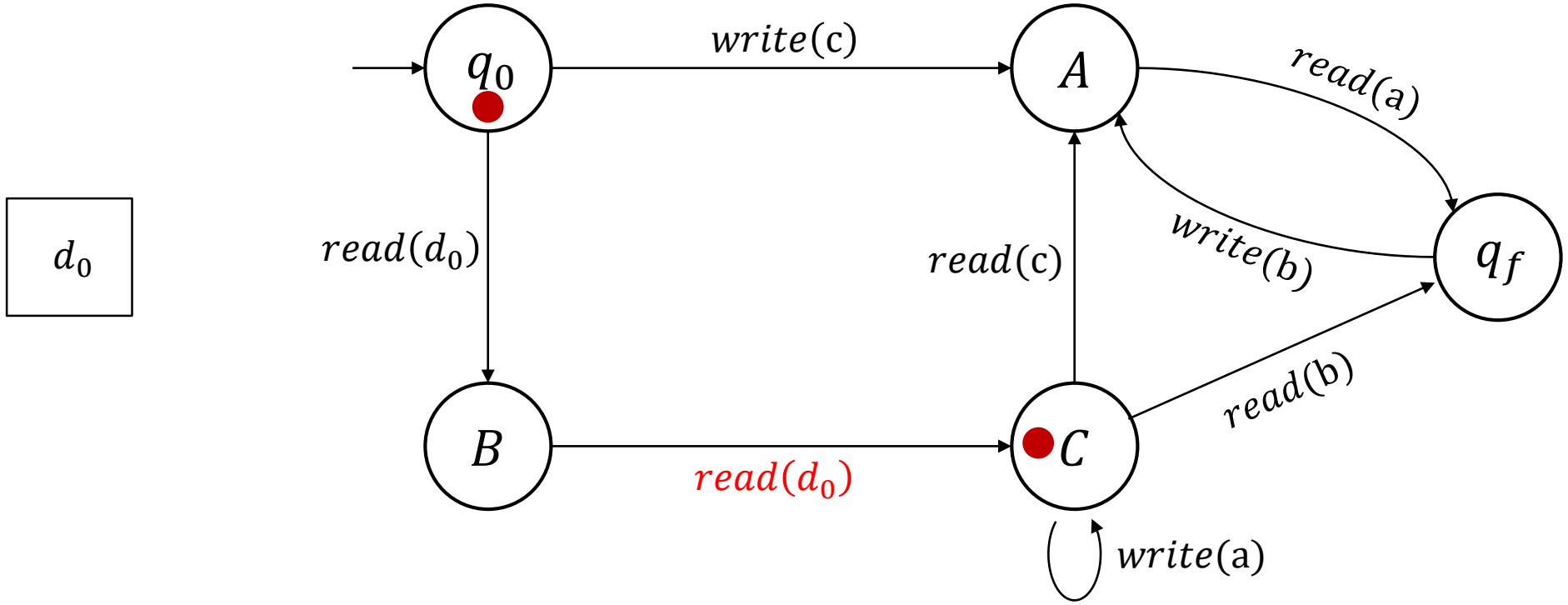
A small example



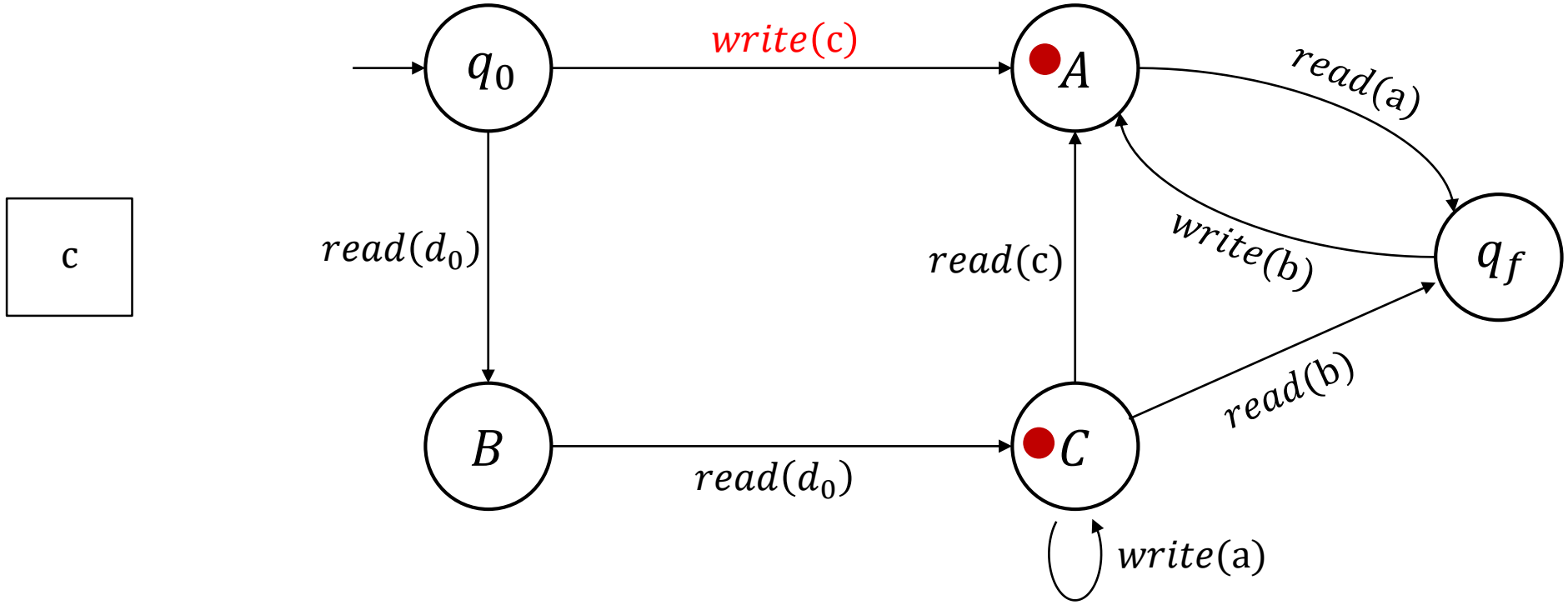
A small example



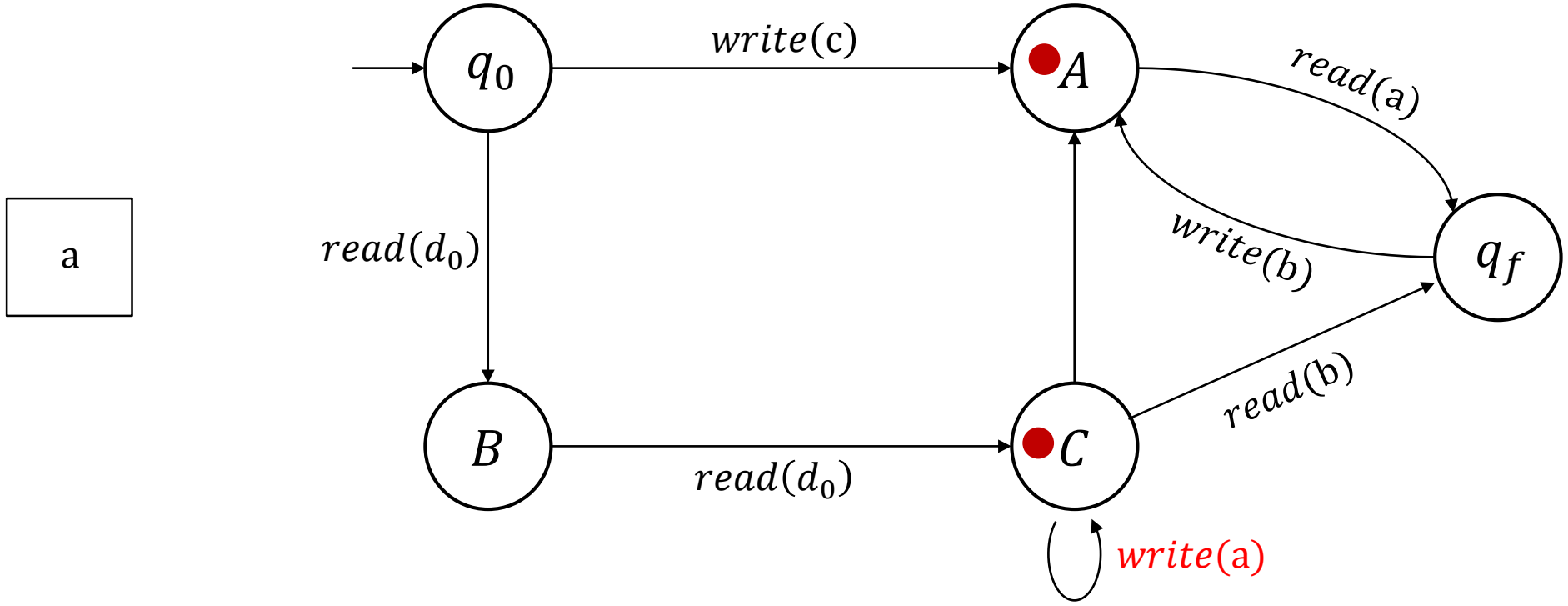
A small example



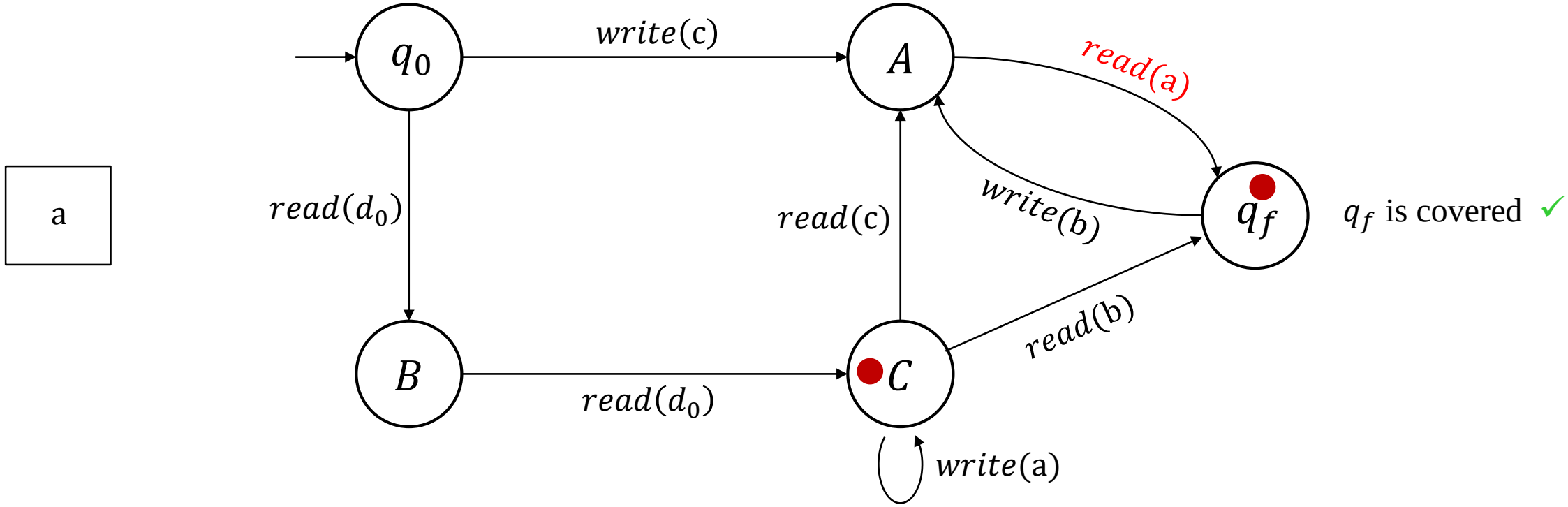
A small example



A small example



A small example



Reachability problems

An initial configuration

A least one process on q_f: "error state"

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

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

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

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

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

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

↑

Presence Reachability Problem

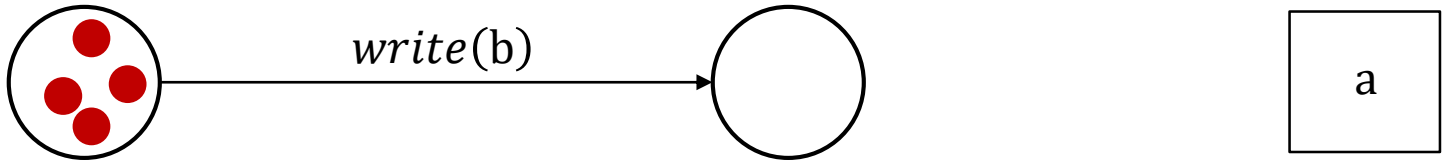
Examples: $\phi =$ ``$\#q_f > 0''$ (COVER), $\phi =$ ``$\bigvee_{q \neq q_f} \#q = 0\,''$ (TARGET)

$\phi =$ ``$(\#q_1 > 0) \vee ([\#q_2 = 0] \wedge [\mathbf{reg}_1 = d_0])''$

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

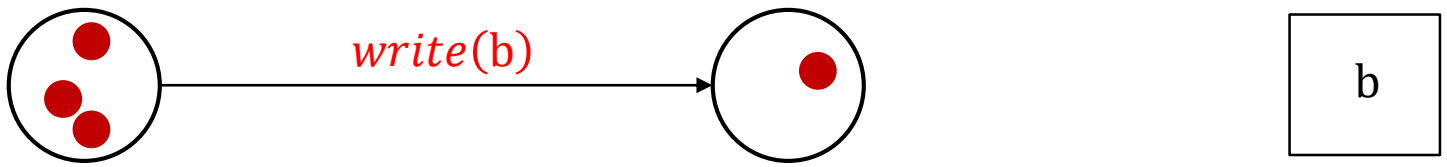
Monotonicity

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



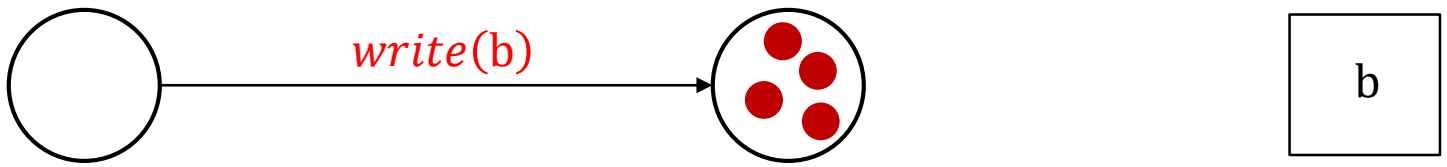
Monotonicity

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



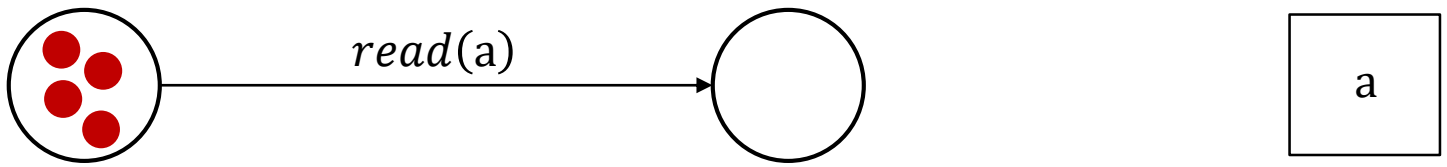
Monotonicity

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



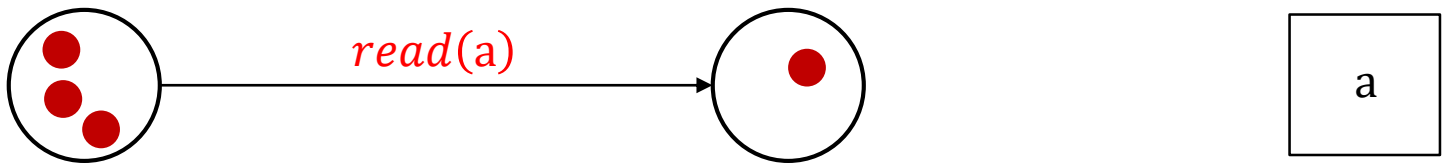
Monotonicity

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



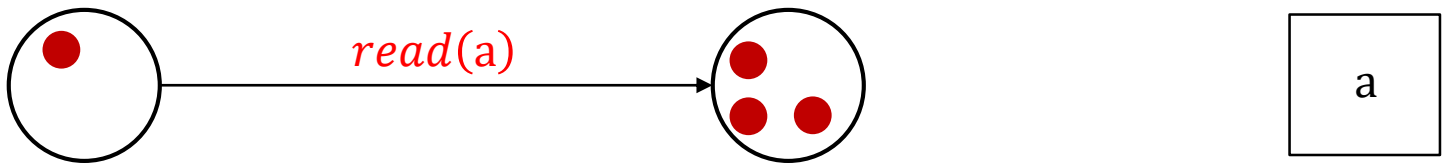
Monotonicity

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

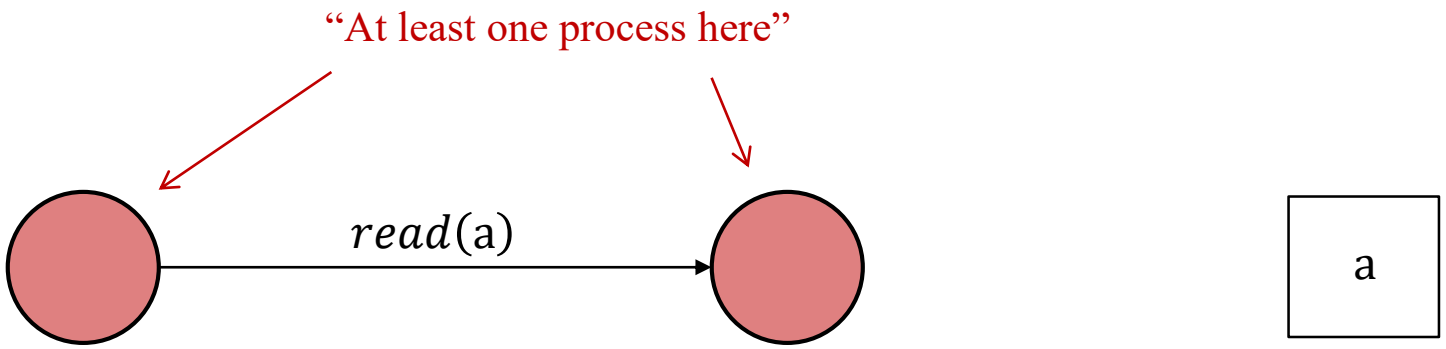


Monotonicity

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



Monotonicity

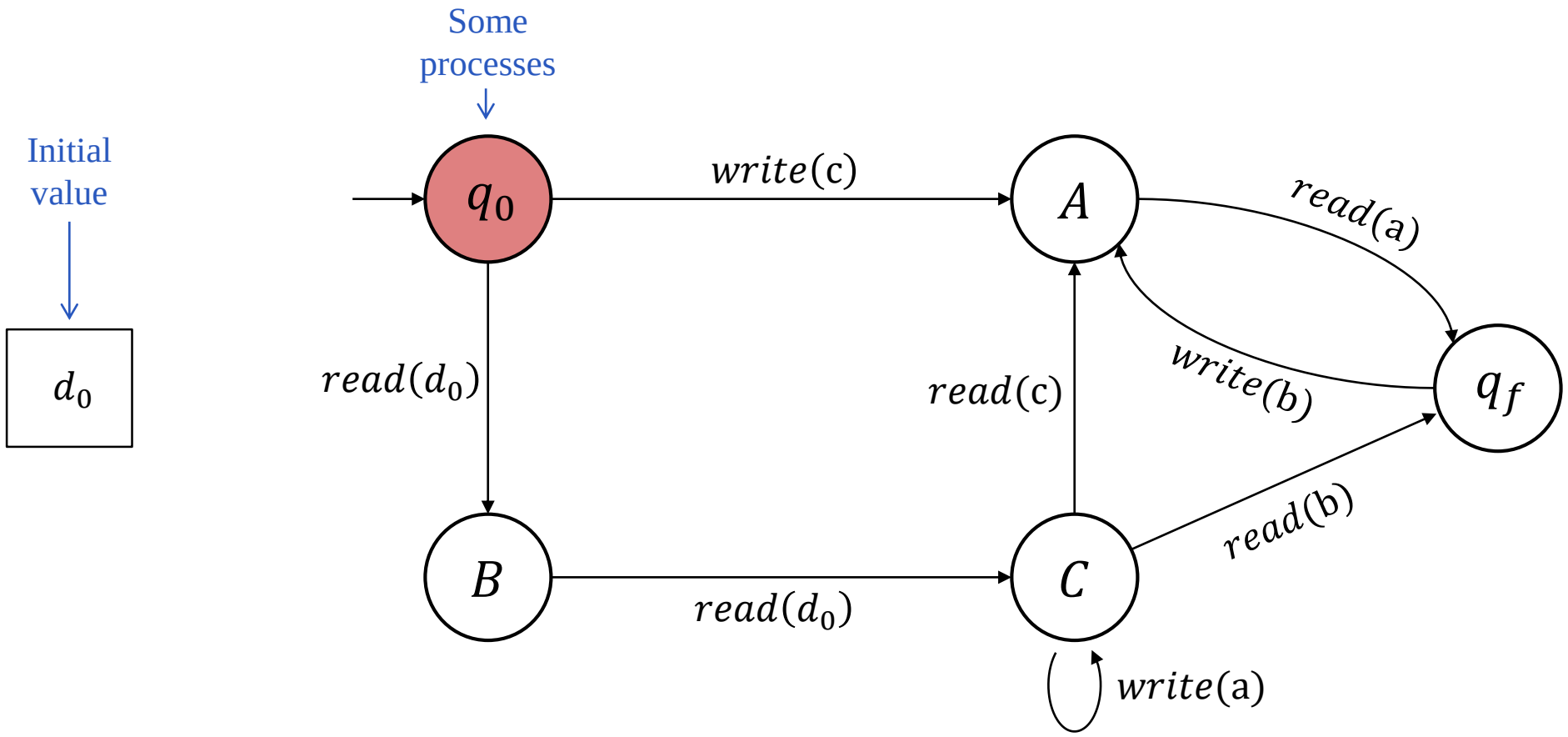


<u>Abstraction</u>: remember whether there is at least one process on a given state.

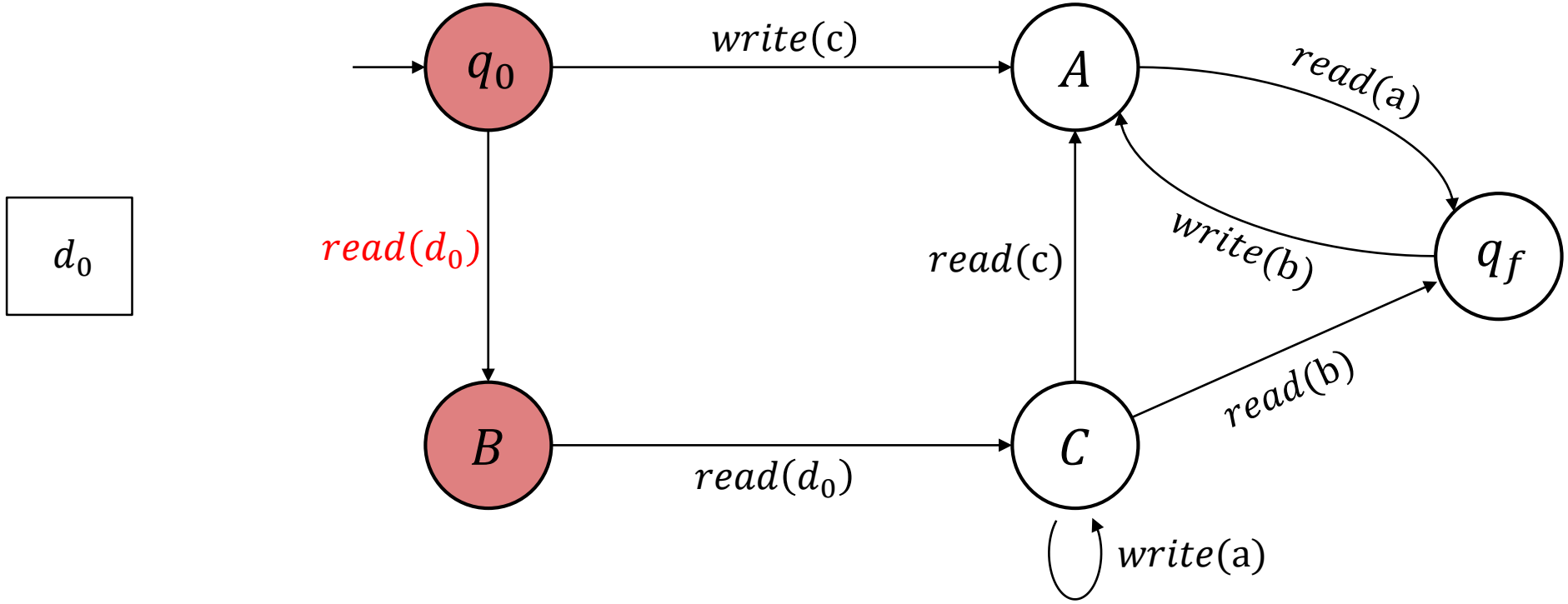
Sound and *complete* for PRP because:

- Monotonicity property (thanks to non-atomicity)
- Arbitrarily many properties in initial configurations
- Interested in parameterized properties (does there exists n such that…?)
- ϕ cannot count processes

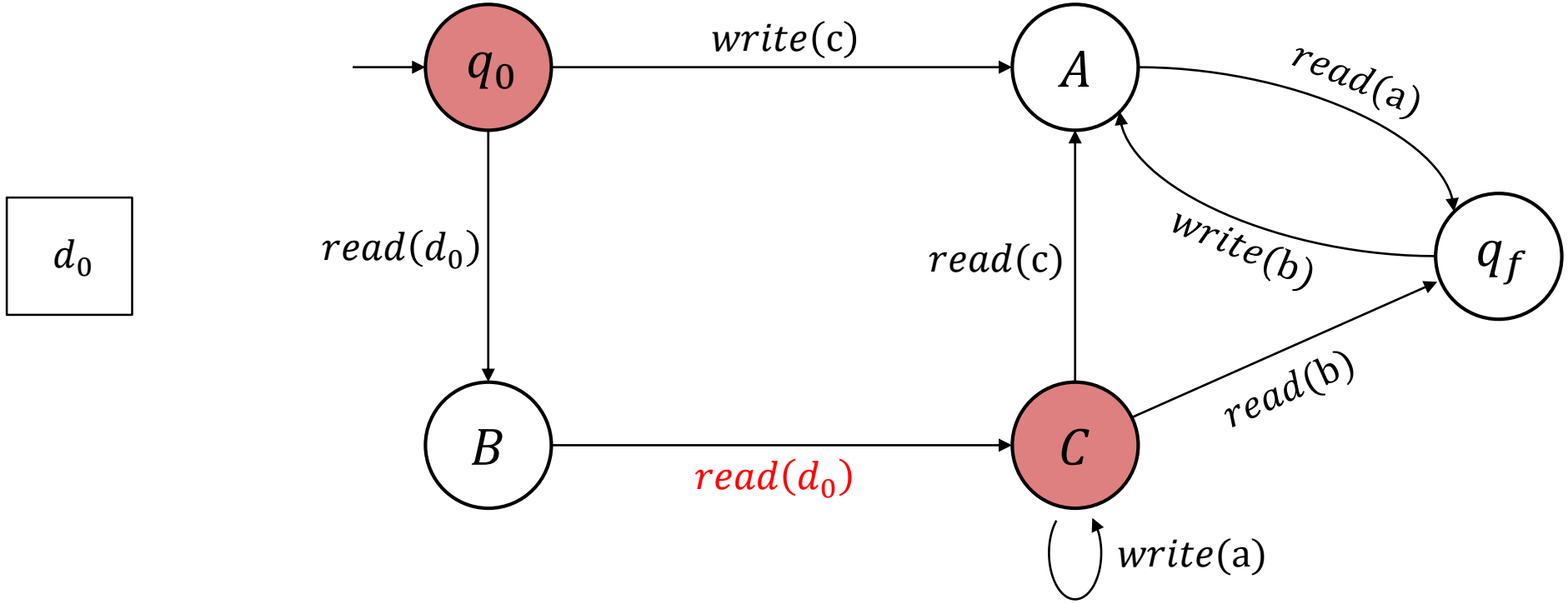
Back to the example



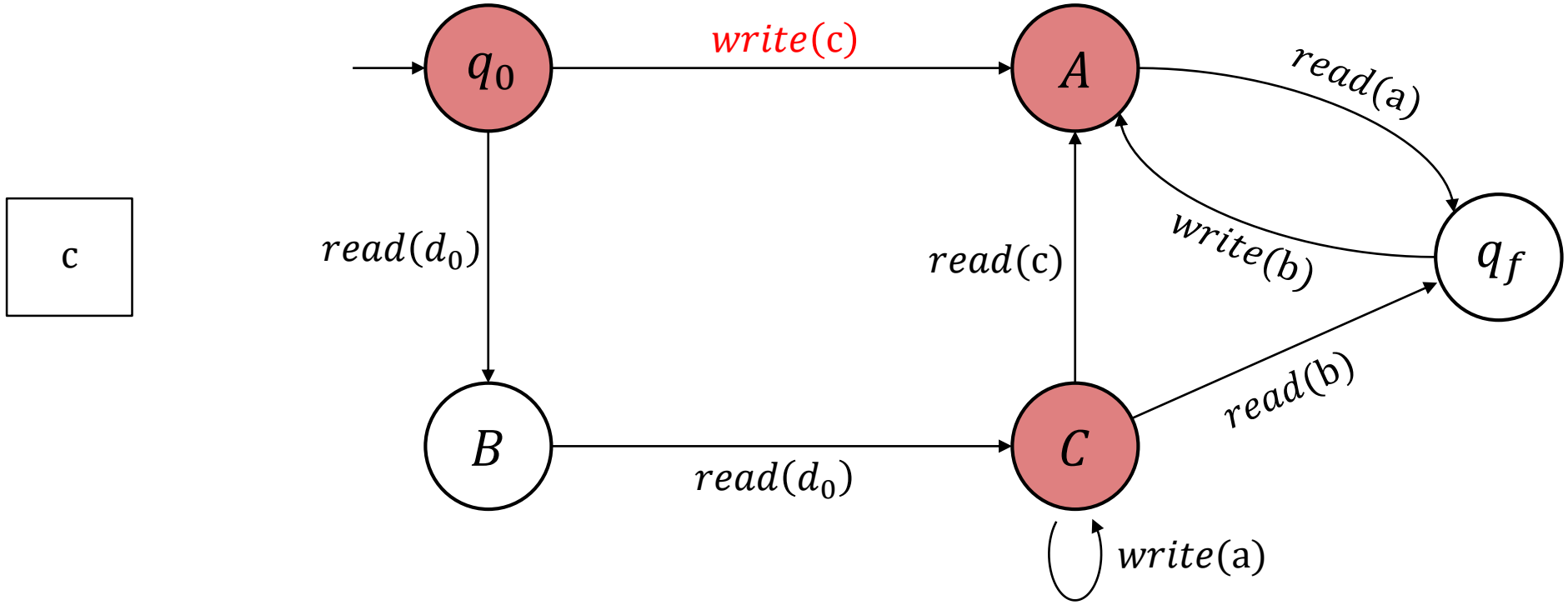
Back to the example



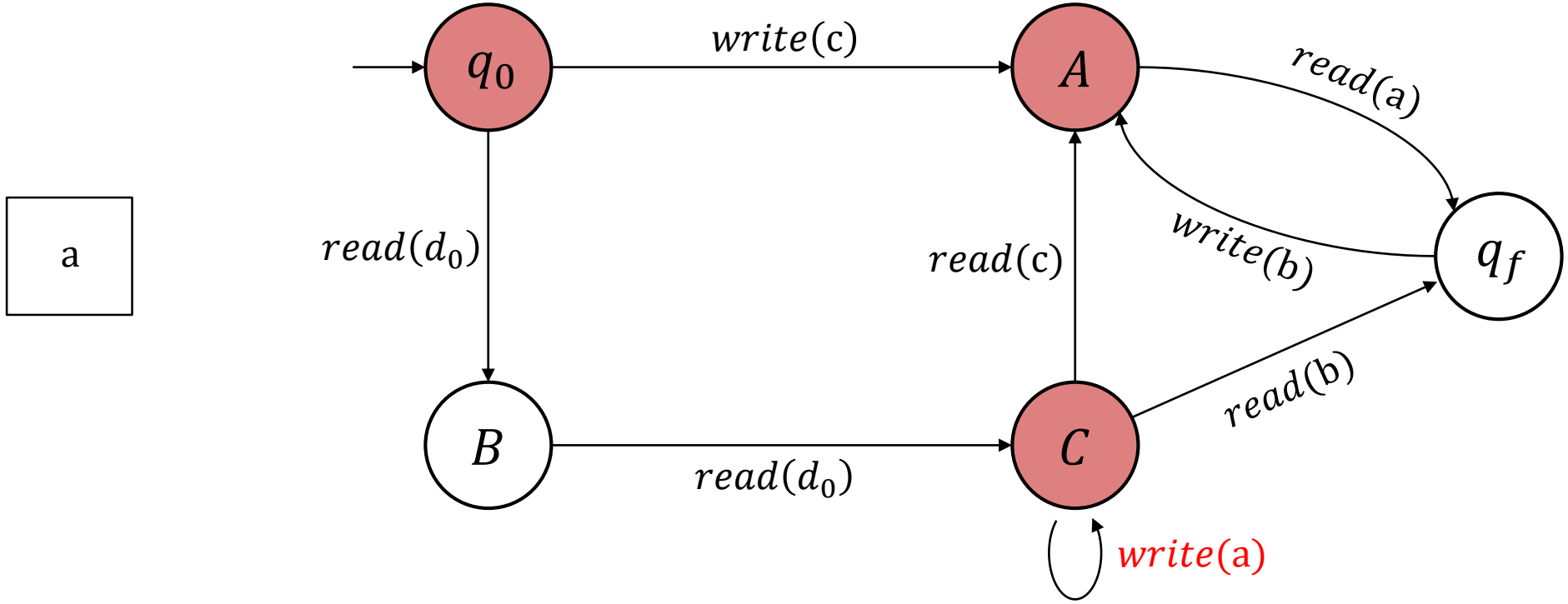
Back to the example



Back to the example

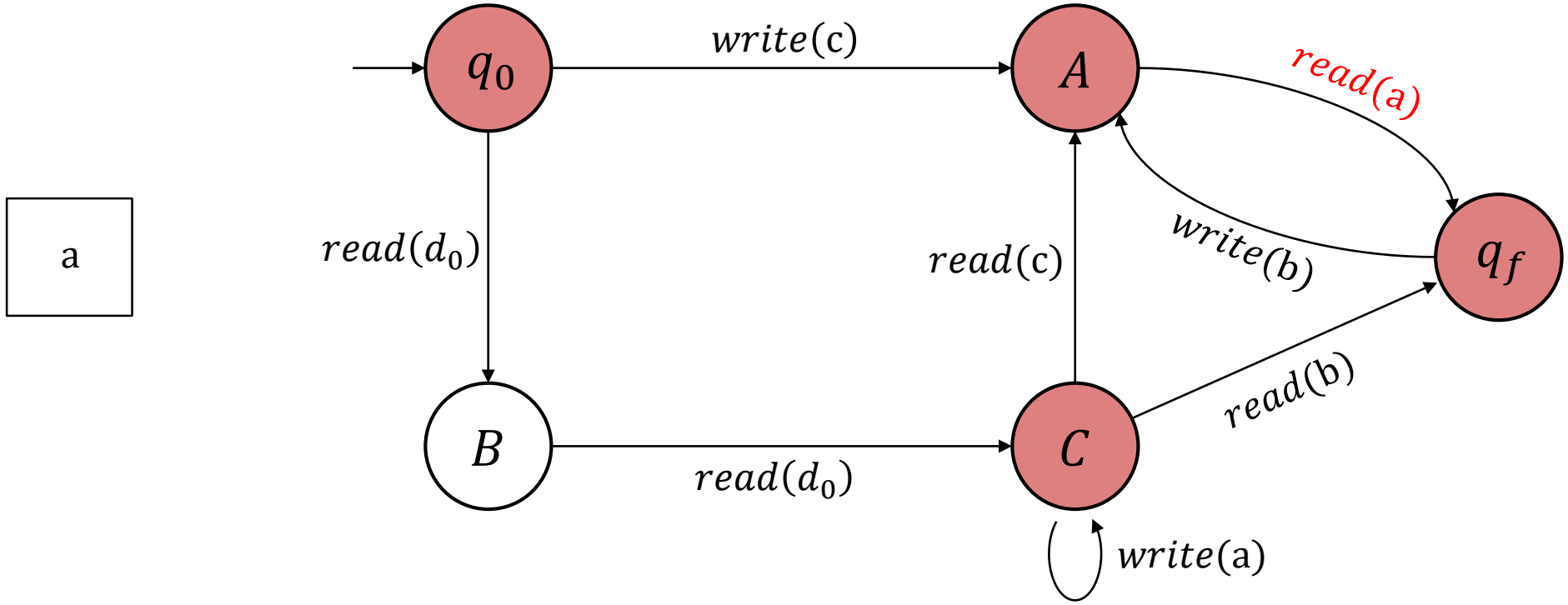


Back to the example



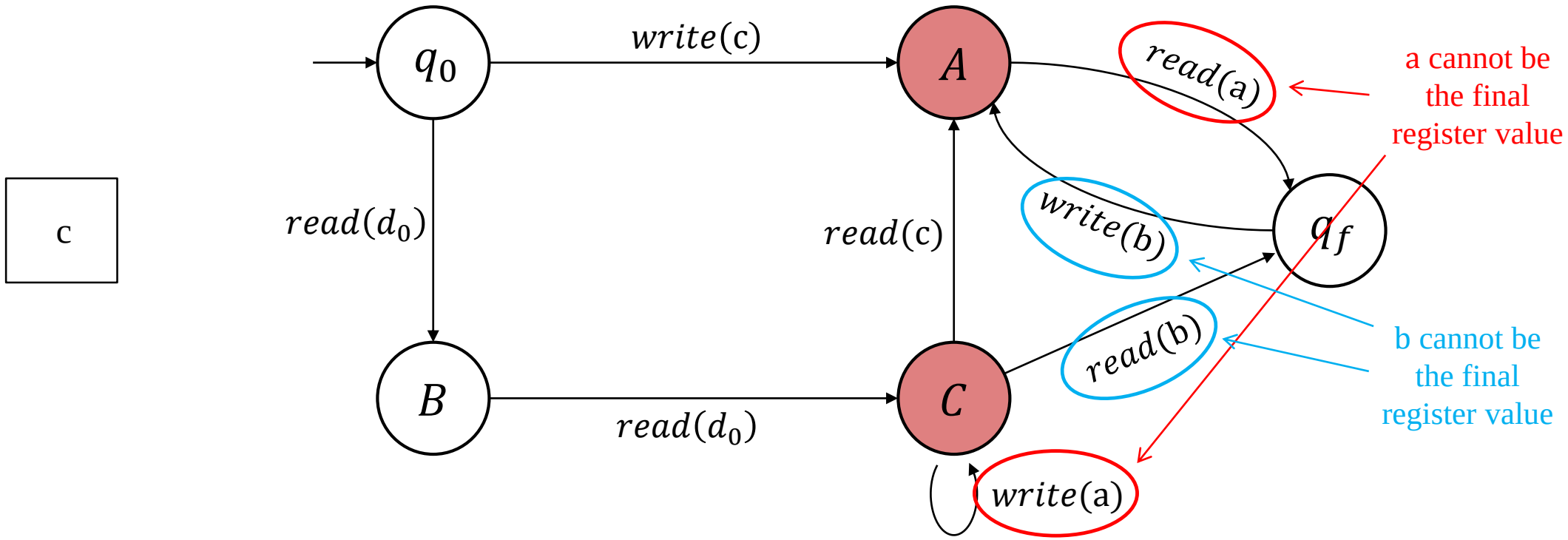
Back to the example

Positive instance of COVER ✓



Back to the example

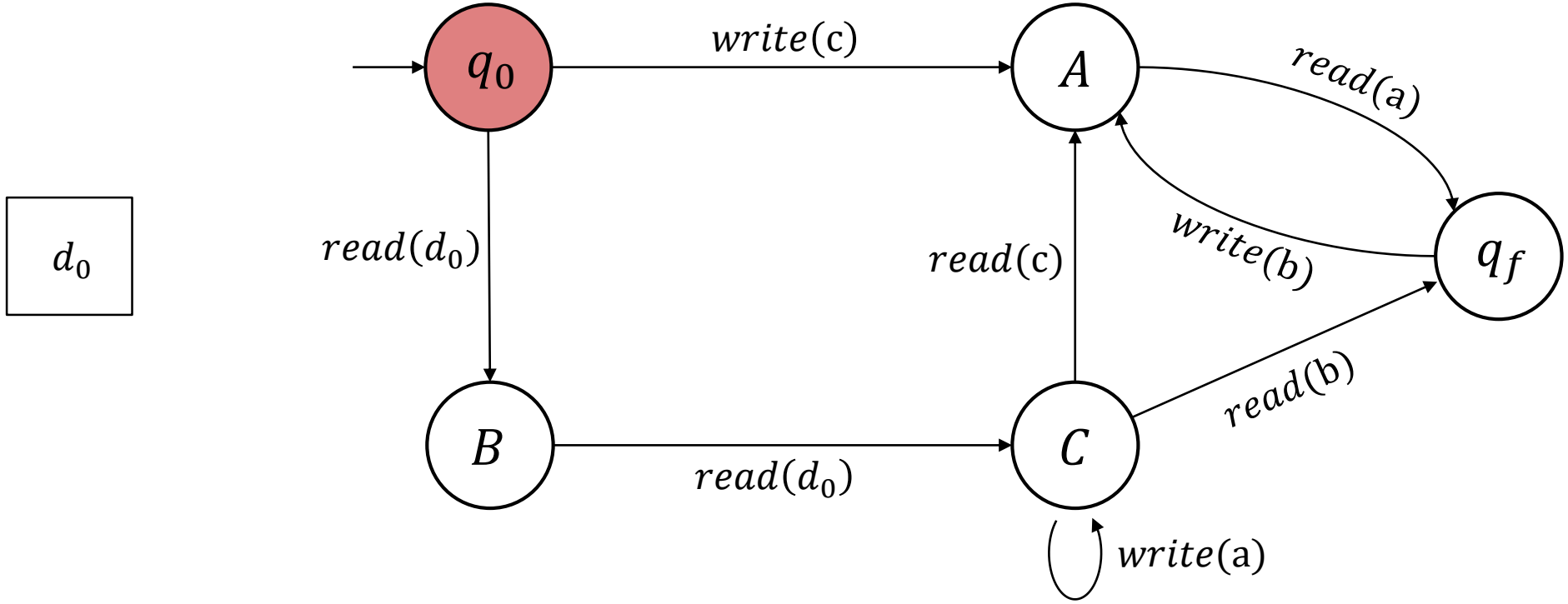
Negative instance of TARGET ✗



Back to the example

Positive instance of PRP with

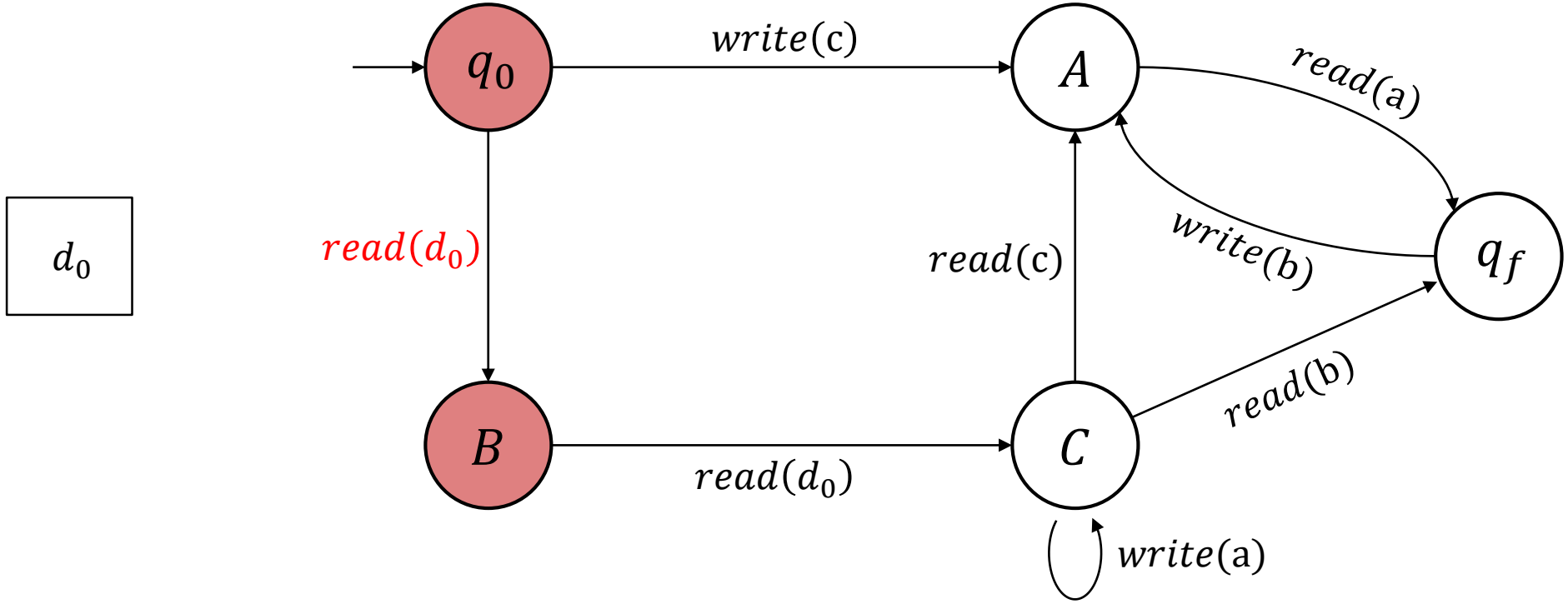
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

Positive instance of PRP with

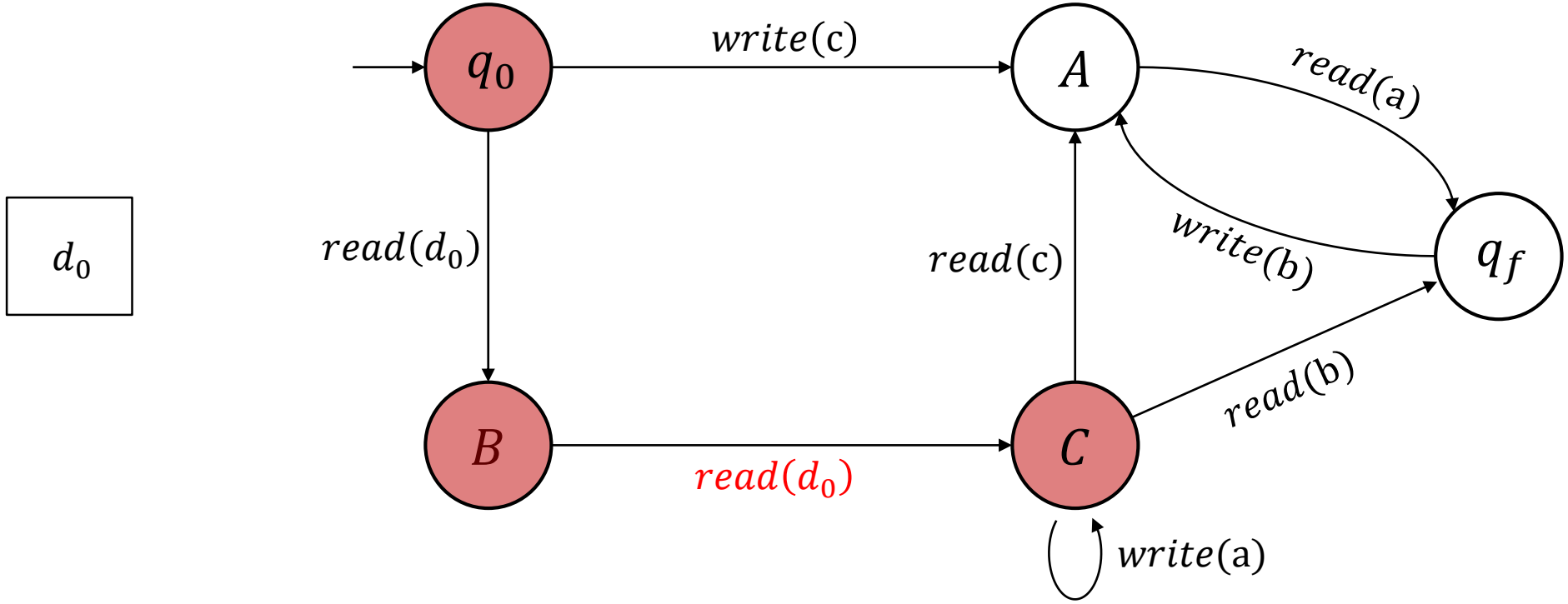
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

Positive instance of PRP with

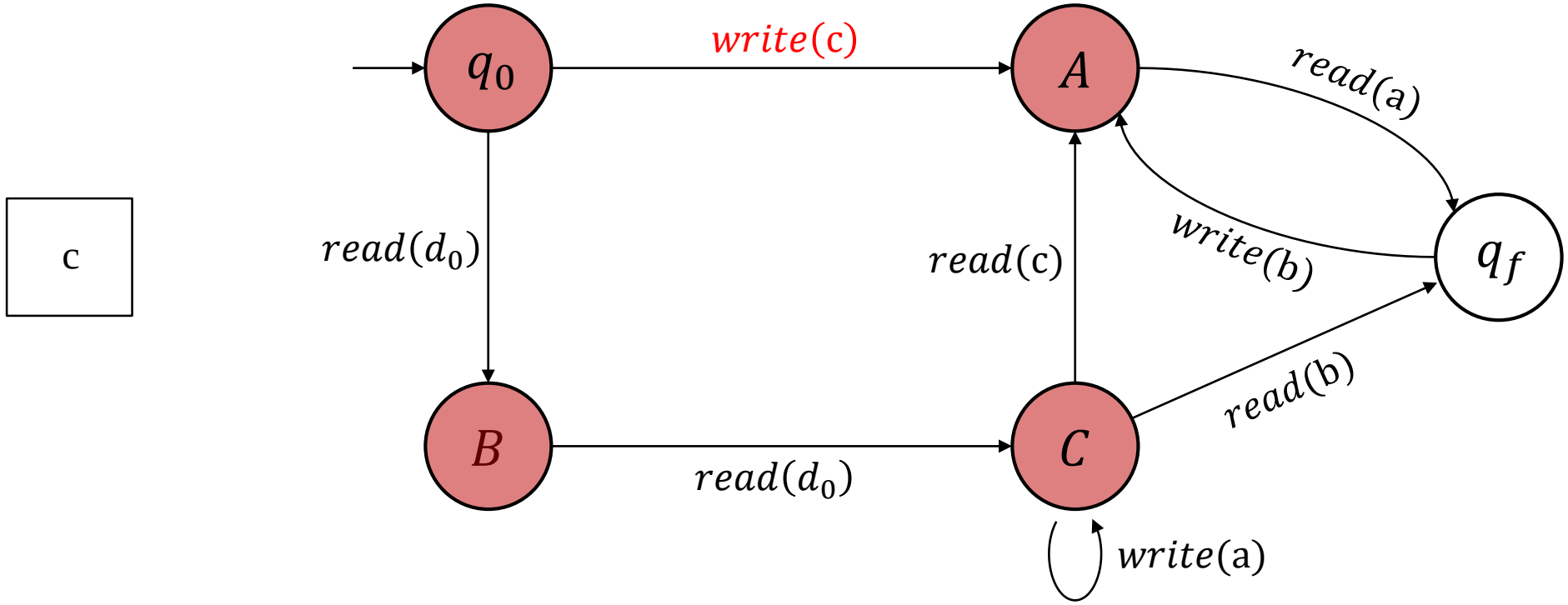
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

Positive instance of PRP with

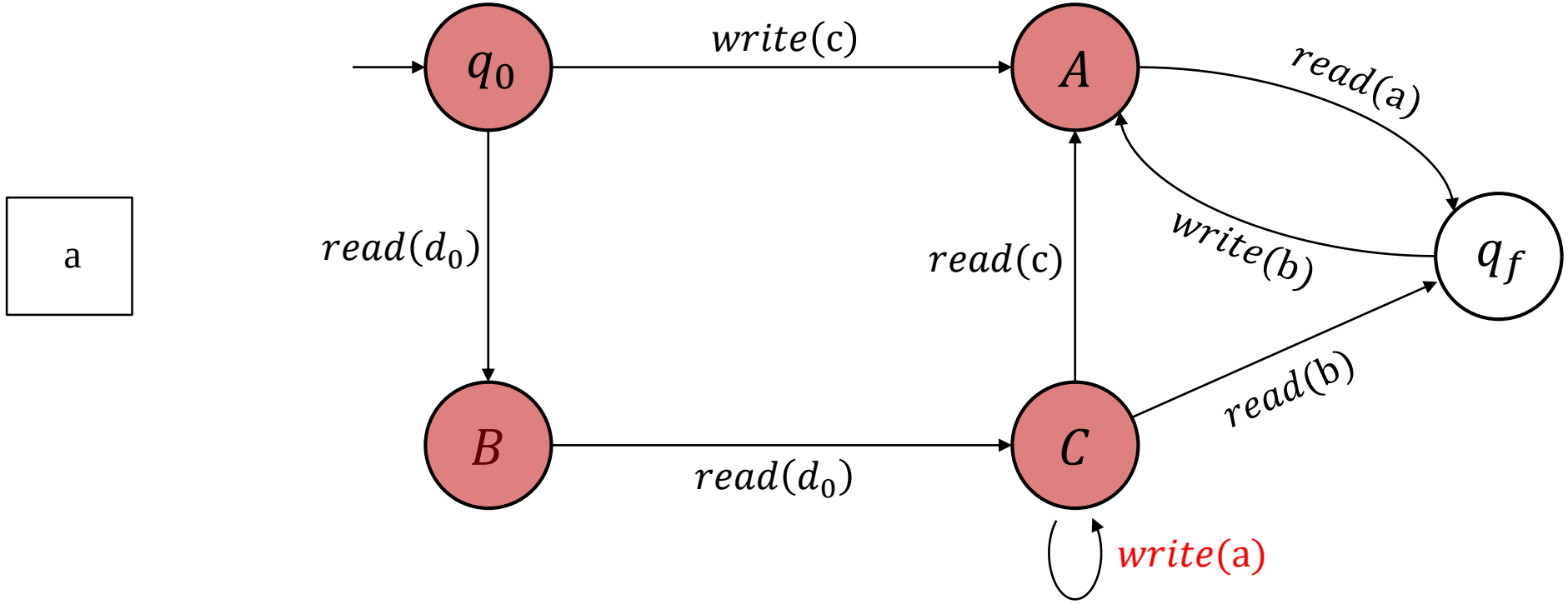
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

Positive instance of PRP with

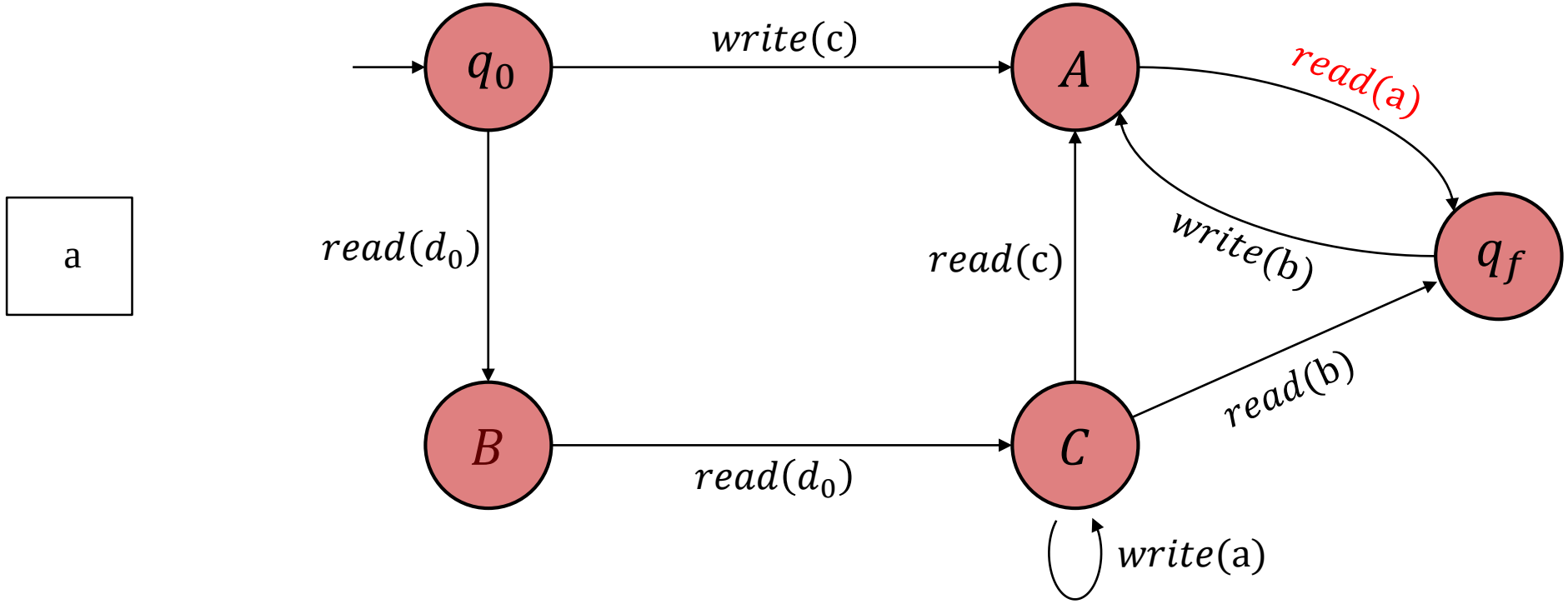
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left([\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0] \vee [\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0]\right)$$



Back to the example

Positive instance of PRP with

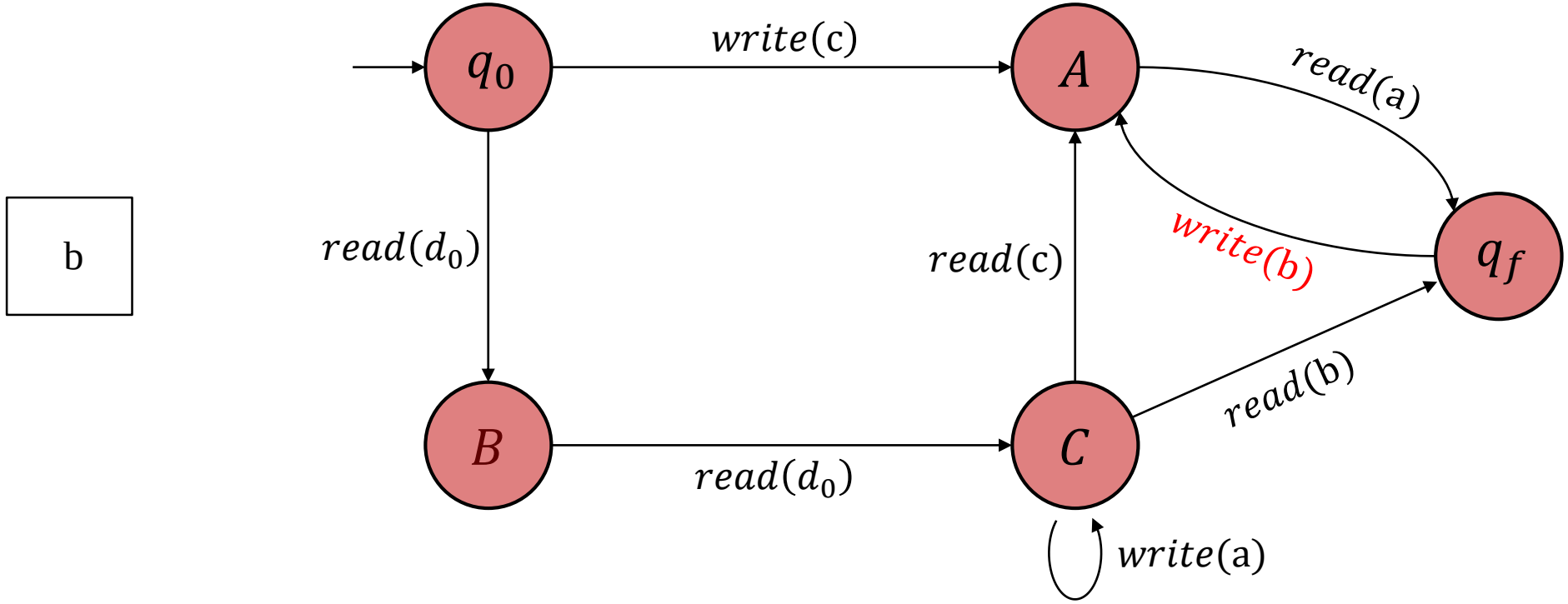
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

Positive instance of PRP with

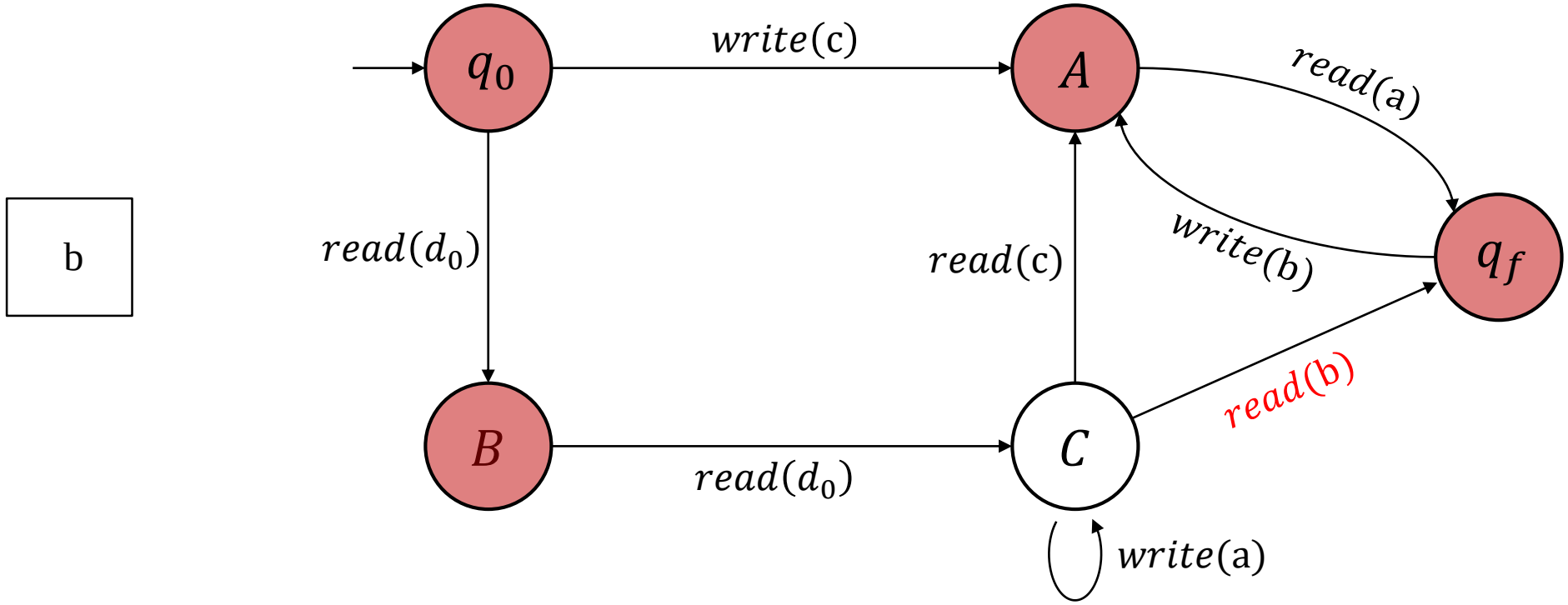
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

Positive instance of PRP with

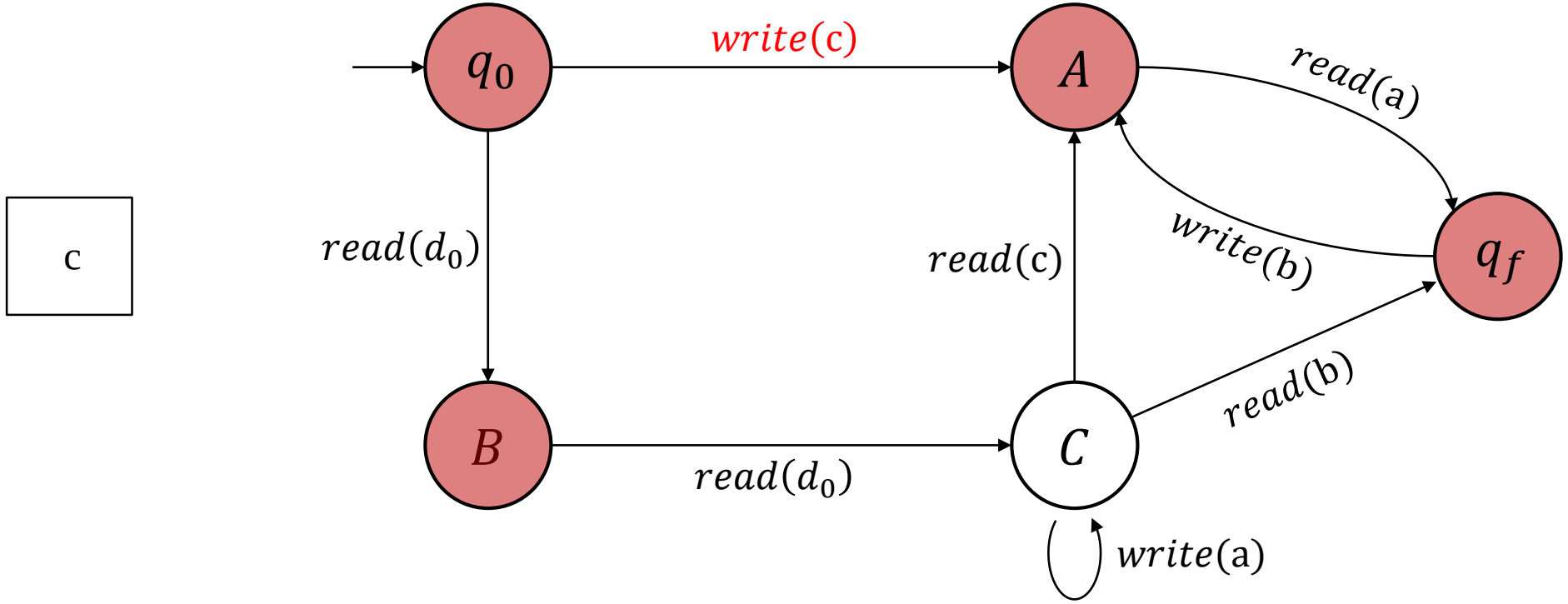
$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left(\left[\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0\right] \vee \left[\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0\right]\right)$$



Back to the example

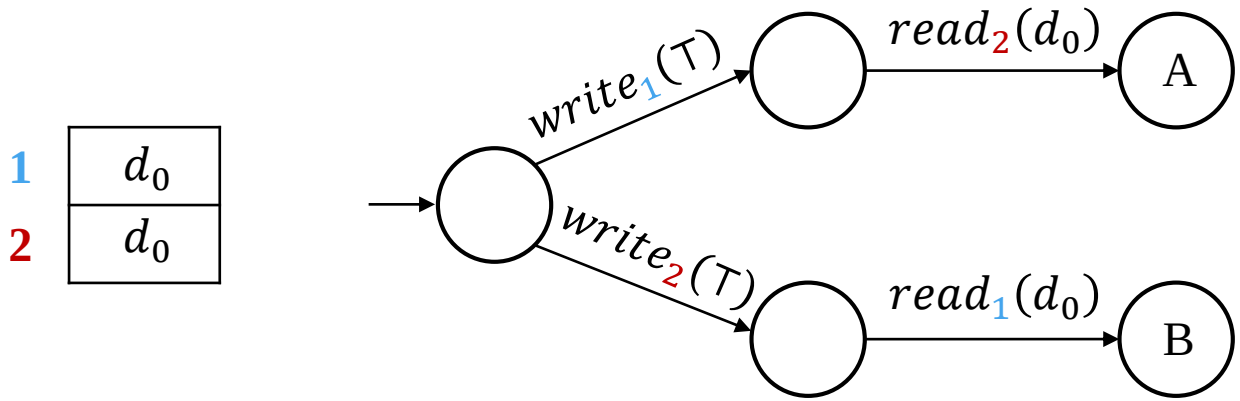
Positive instance of PRP with

$$\phi = (\mathbf{reg} = \mathrm{c} \vee \mathbf{reg} = d_0) \wedge \left([\#A > 0 \wedge \#q_f > 0 \wedge \#C = 0] \vee [\#A = 0 \wedge \#q_f > 0 \wedge \#C = 0]\right)$$



NP-completeness of COVER

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



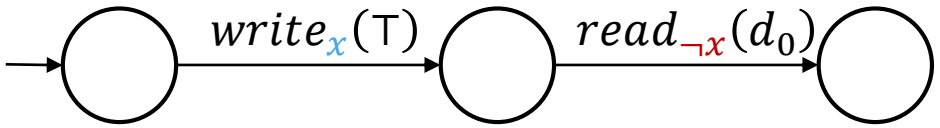
NP-completeness of COVER

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

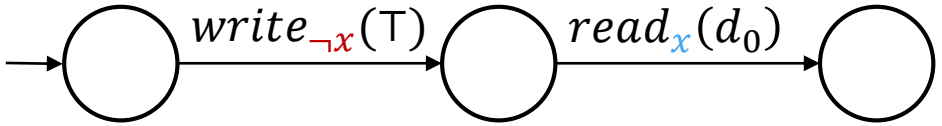
Reduction from 3-SAT:

x	d_0
$\neg x$	d_0

Check x:



Check $\neg x$:



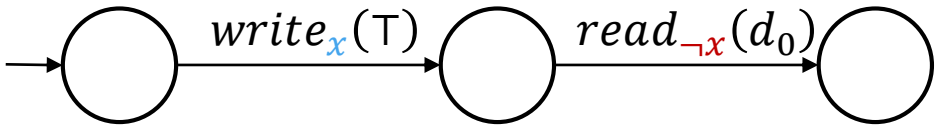
NP-completeness of COVER

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

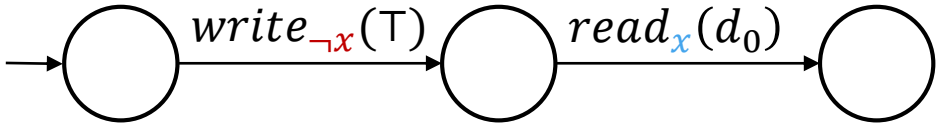
Reduction from 3-SAT:

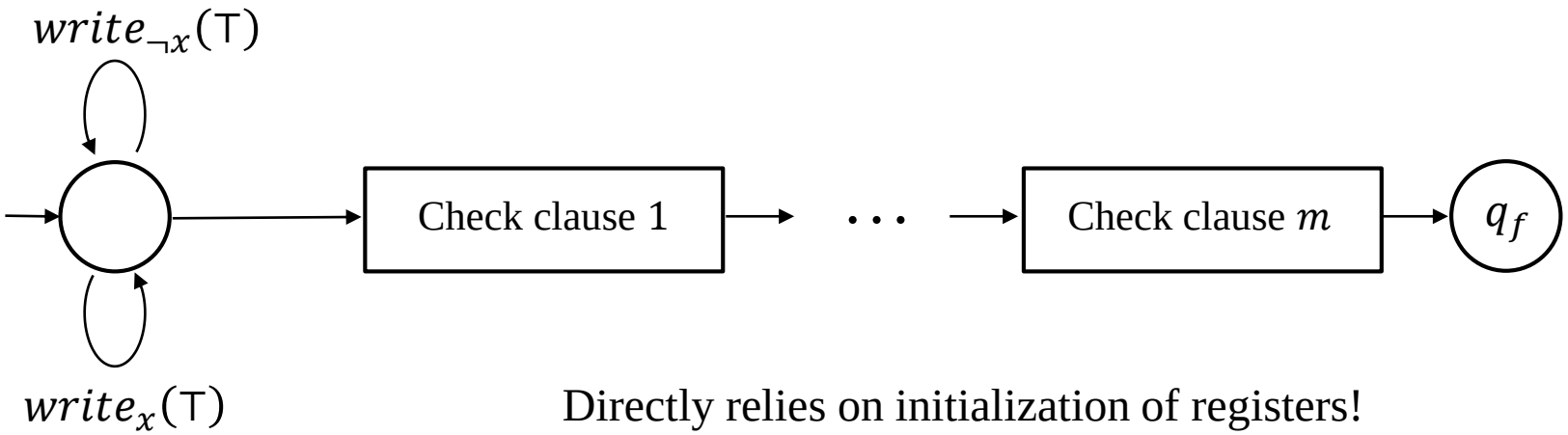
x	d_0
$\neg x$	d_0

Check x:



Check $\neg x$:





Directly relies on initialization of registers!

COVER when registers are uninitialized

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

COVER drops down to PTIME when registers are uninitialized[3].

Saturation algorithm:

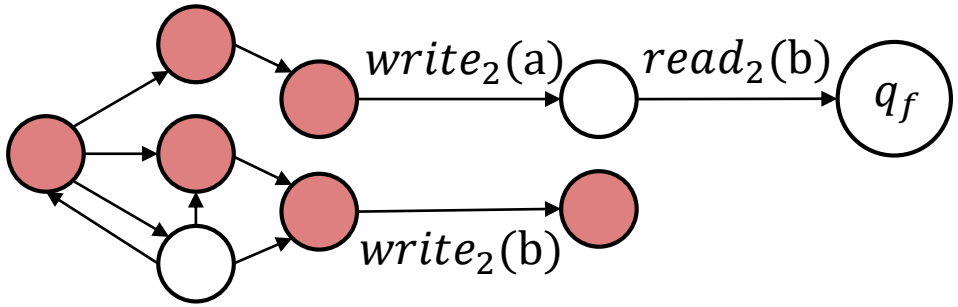
3. Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems.* Journal of the ACM, 2016

COVER when registers are uninitialized

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

COVER drops down to PTIME when registers are uninitialized[3].

Saturation algorithm:



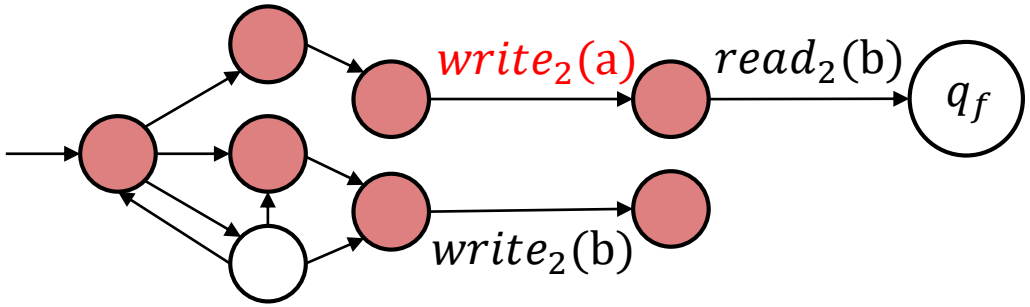
3. Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems.* Journal of the ACM, 2016

COVER when registers are uninitialized

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

COVER drops down to PTIME when registers are uninitialized[3].

Saturation algorithm:



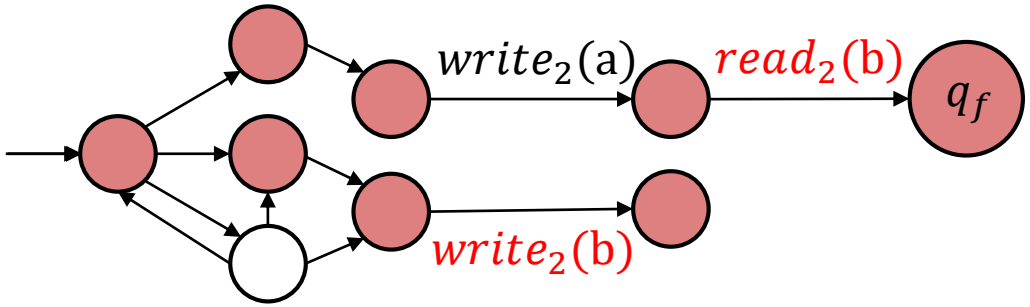
3. Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems.* Journal of the ACM, 2016

COVER when registers are uninitialized

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

COVER drops down to PTIME when registers are uninitialized[3].

Saturation algorithm:

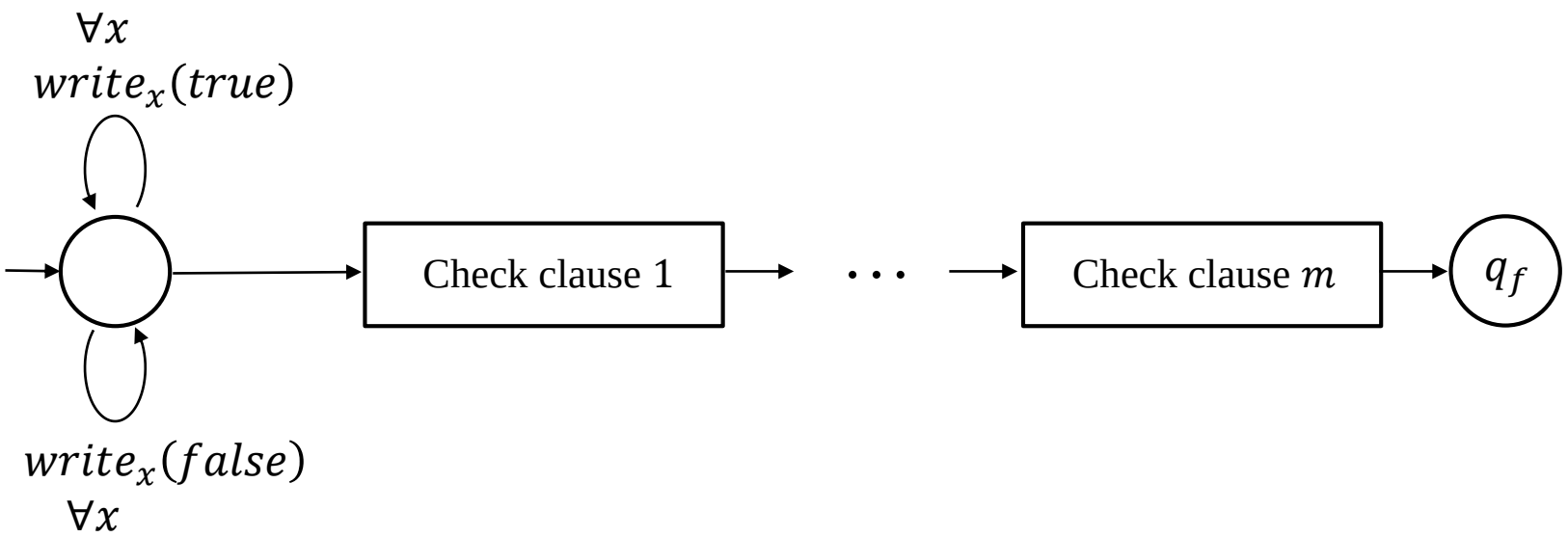


3. Esparza, J., Ganty, P., Majumdar, R.: *Parameterized verification of asynchronous shared-memory systems.* Journal of the ACM, 2016

TARGET when registers are uninitialized

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 uninitialized. 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.
For simplicity: the register is uninitialized. Algorithm inspired from broadcast protocols[4].

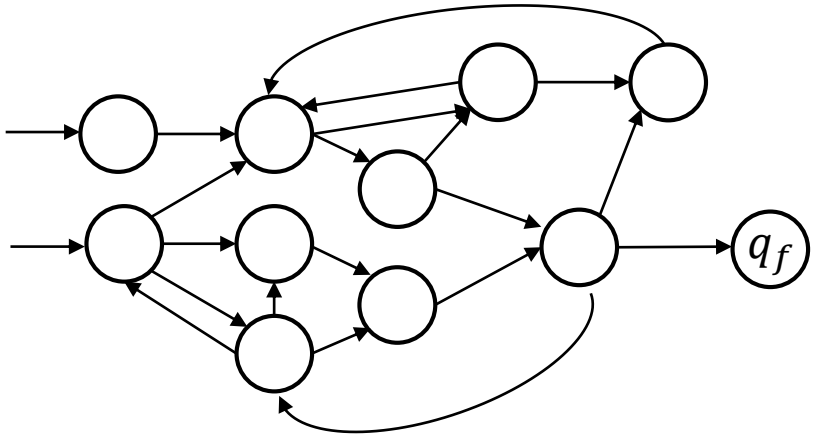
4. Fournier, P.:
Parameterized verification of networks of many identical processes. PhD thesis, 2015

TARGET with a single register

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

TARGET is PTIME when only one register.
For simplicity: the register is uninitialized. 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).



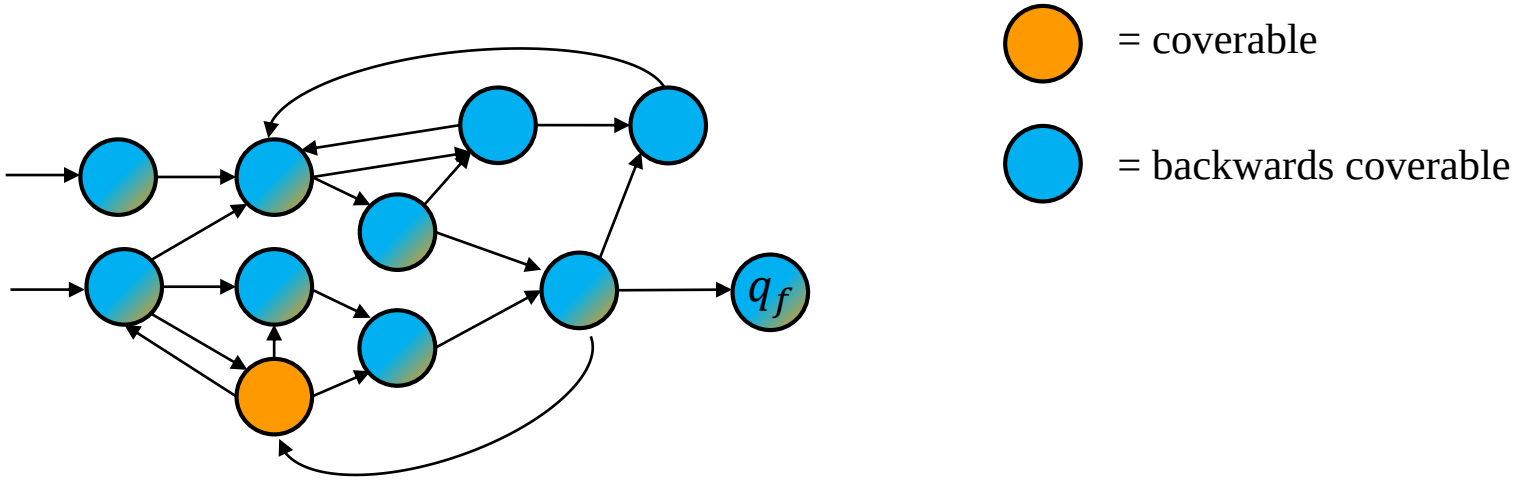
4. Fournier, P.:
Parameterized verification of networks of many identical processes. PhD thesis, 2015

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.
For simplicity: the register is uninitialized. 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).



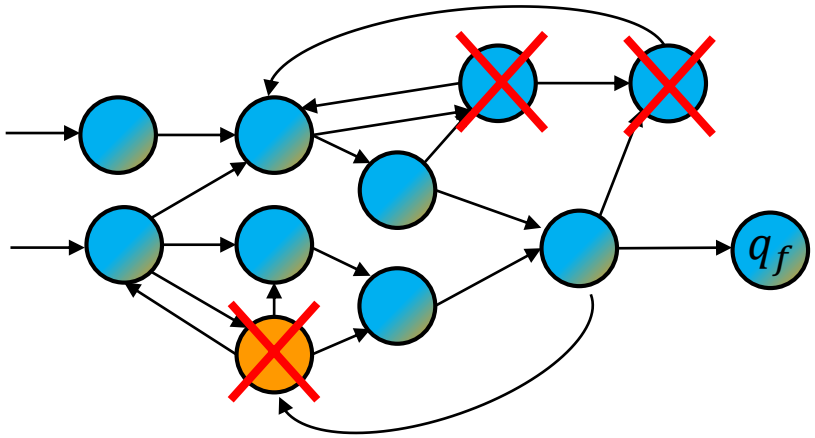
4. Fournier, P.:
Parameterized verification of networks of many identical processes. PhD thesis, 2015

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.
For simplicity: the register is uninitialized. 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 

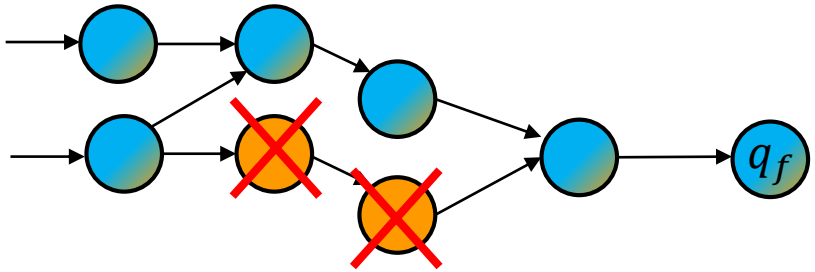
4. Fournier, P.:
Parameterized verification of networks of many identical processes. PhD thesis, 2015

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.
For simplicity: the register is uninitialized. 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 

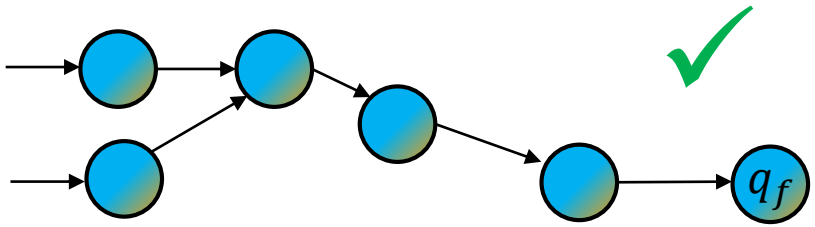
4. Fournier, P.:
Parameterized verification of networks of many identical processes. PhD thesis, 2015

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.
For simplicity: the register is uninitialized. 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.
For simplicity: the register is uninitialized. 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
Uninitialized	PTIME-complete	NP-complete	NP-complete	NP-complete
One register	PTIME-complete	PTIME-complete	PTIME-complete	NP-complete

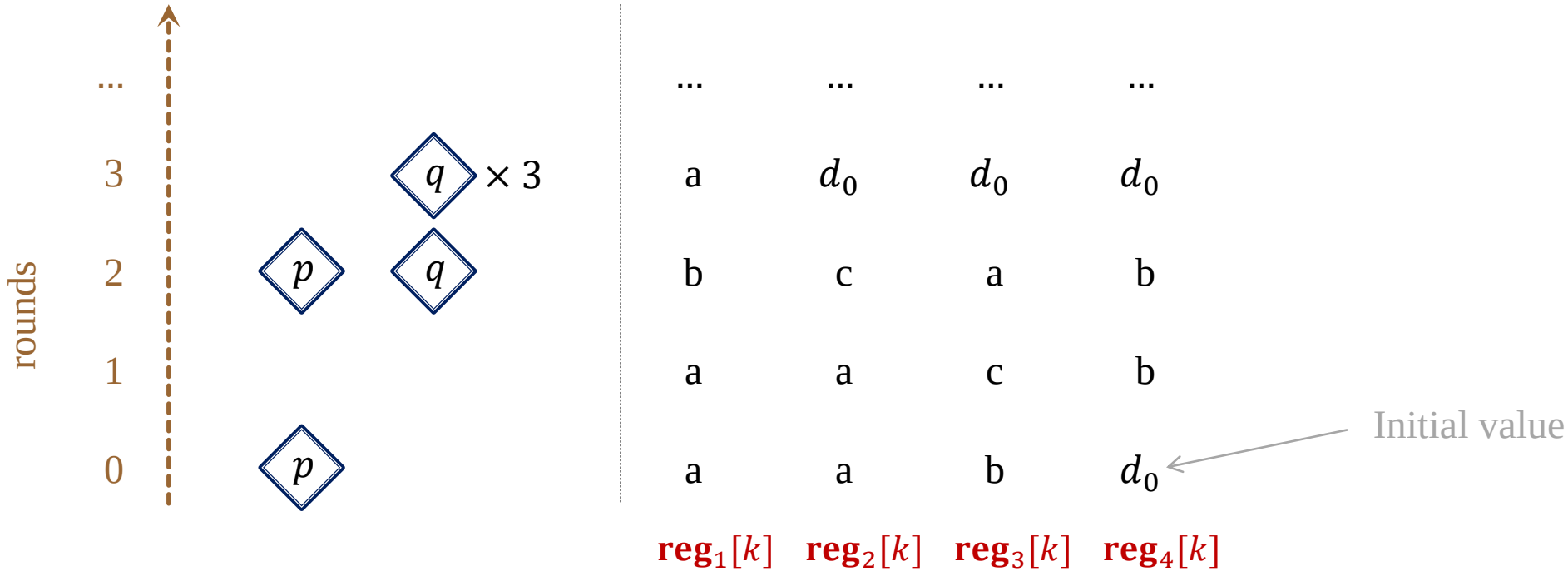
5. W: *Checking Presence Reachability Properties on Parameterized Shared-Memory Systems*, submitted

Round-based shared-memory systems

Round-based shared-memory systems

Model inspired by round-based algorithms from the literature[678].

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

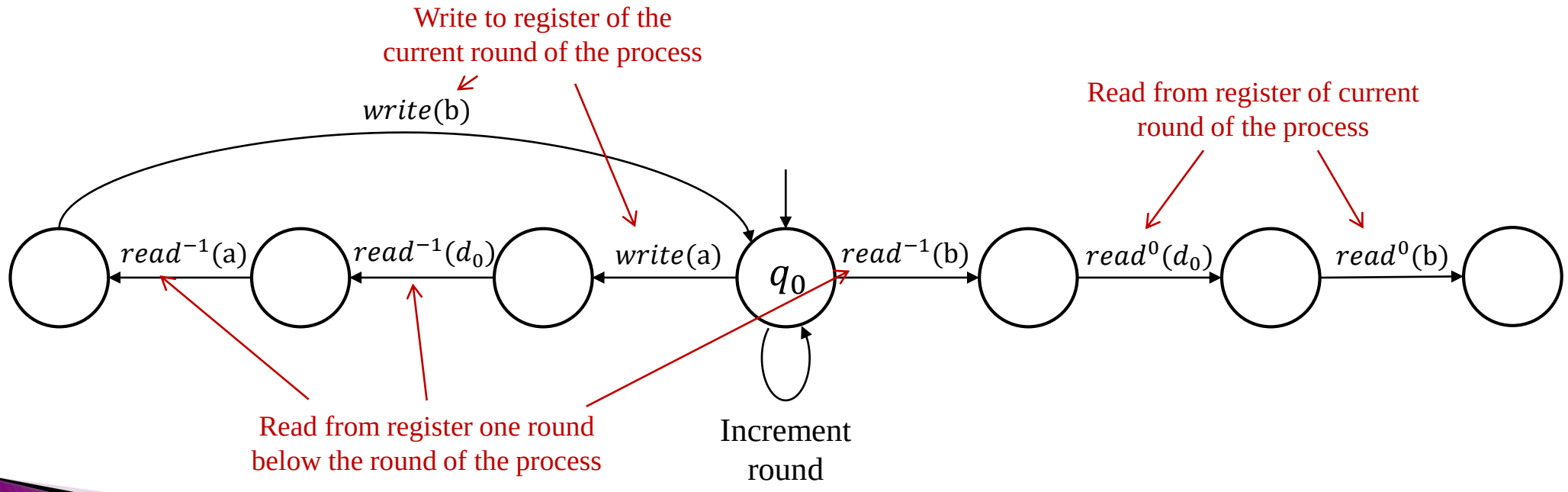


6. Aspnes, J.: *Fast deterministic consensus in a noisy environment.* Journal of Algorithms, 2002
7. Guerraoui, R., Ruppert, E.: *Anonymous and fault-tolerant shared-memory computing.* Distrib. Comput., 2007
8. Raynal, M., Stainer, J.: *A Simple Asynchronous Shared Memory Consensus Algorithm Based on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

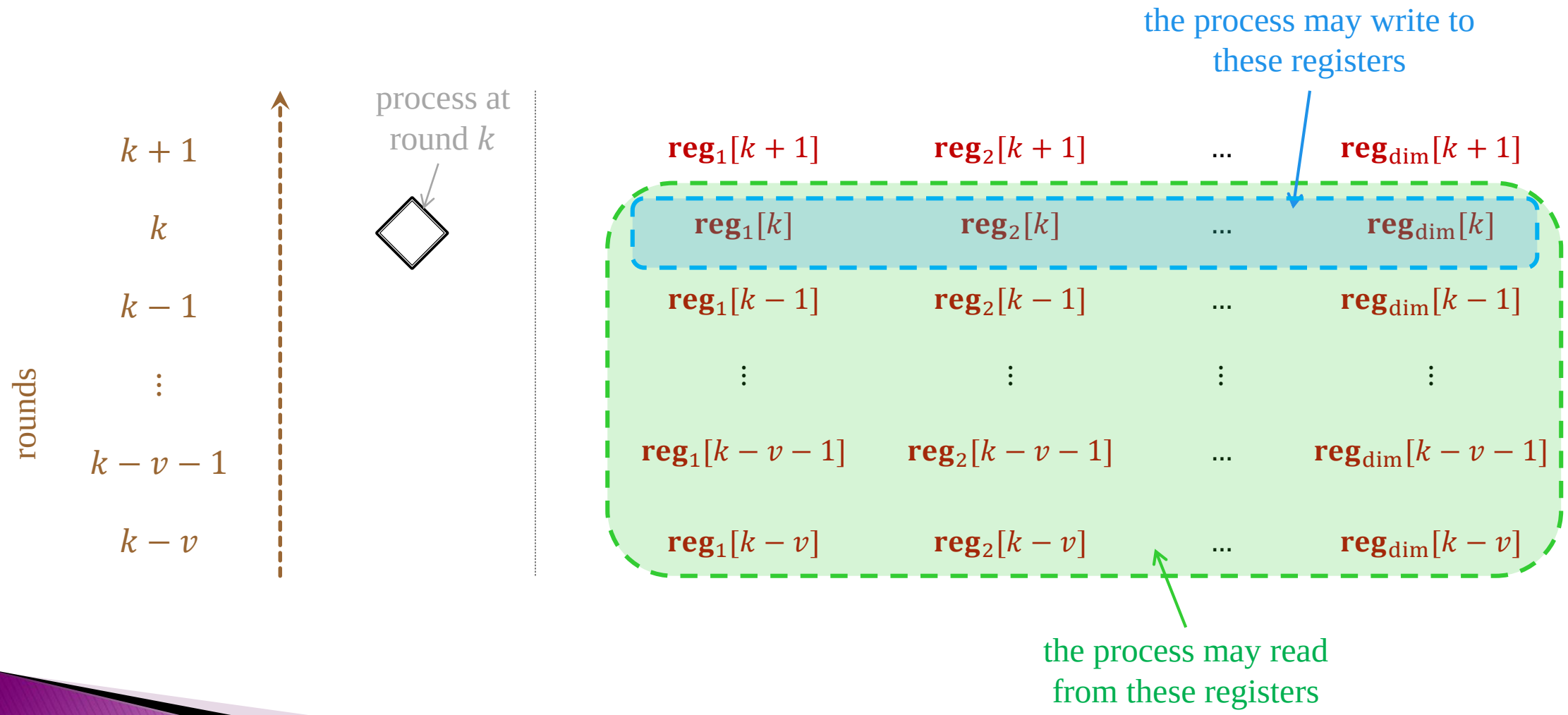
The round-based model

- 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

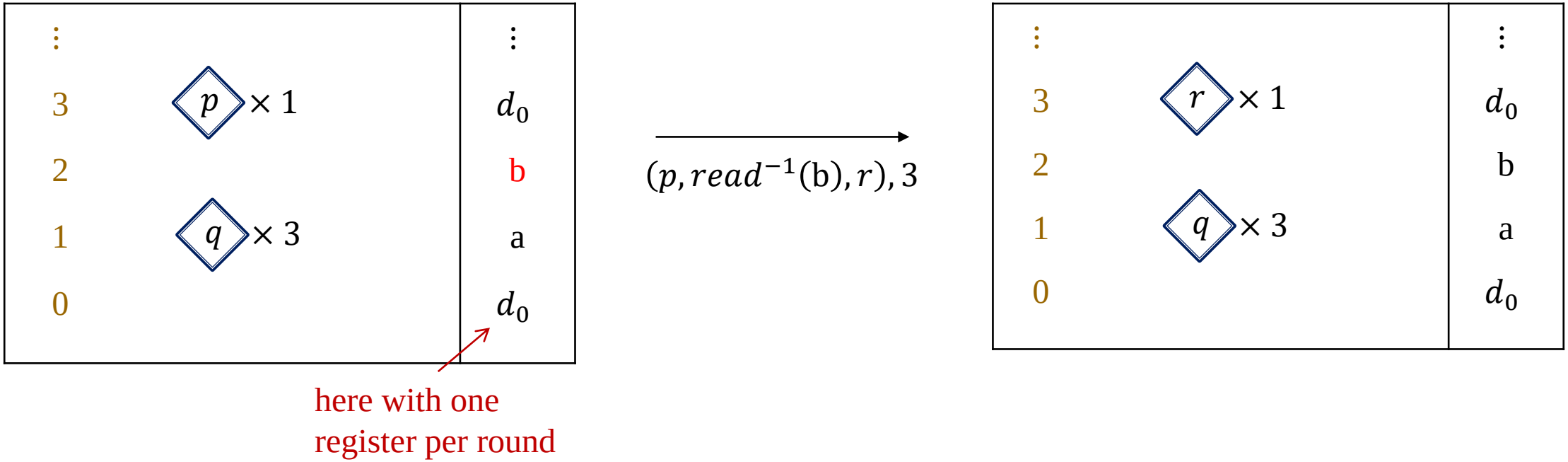
Example with one register per round:



A limited visibility range

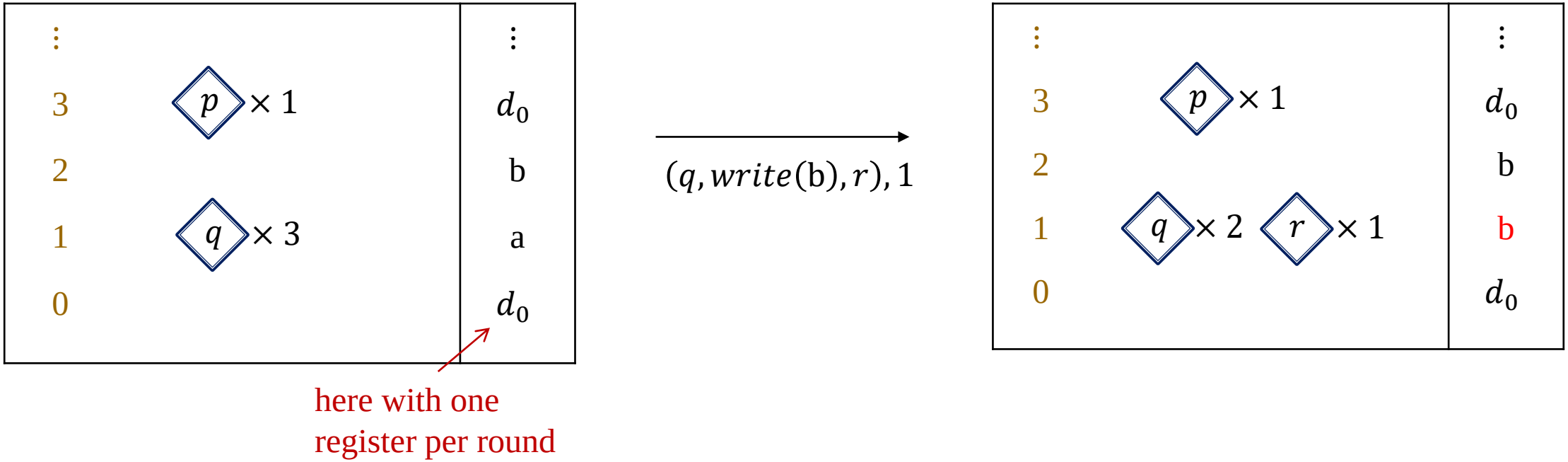


Semantics



here with one register per round

Semantics



here with one register per round

Semantics

⋮		⋮
3	$p \times 1$	d_0
2		b
1	$q \times 3$	a
0		d_0

$\xrightarrow{(q, write(\mathrm{b}), r), 1}$

⋮		⋮
3	$p \times 1$	d_0
2		b
1	$q \times 2$ $r \times 1$	b
0		d_0

Initial configurations:

⋮		⋮
2		d_0
1		d_0
0	$q_1 \times n_1$ $q_2 \times n_2$ ...	d_0

with $n_1, n_2, \ldots \geq 0$ and $q_1, q_2, \ldots$ initial states

Abstraction

⋮		⋮
3	p ~~×1~~	d_0
2		b
1	q ~~×3~~	a
0		d_0

$\xrightarrow{(q, write(\mathrm{b}), r), 1}$

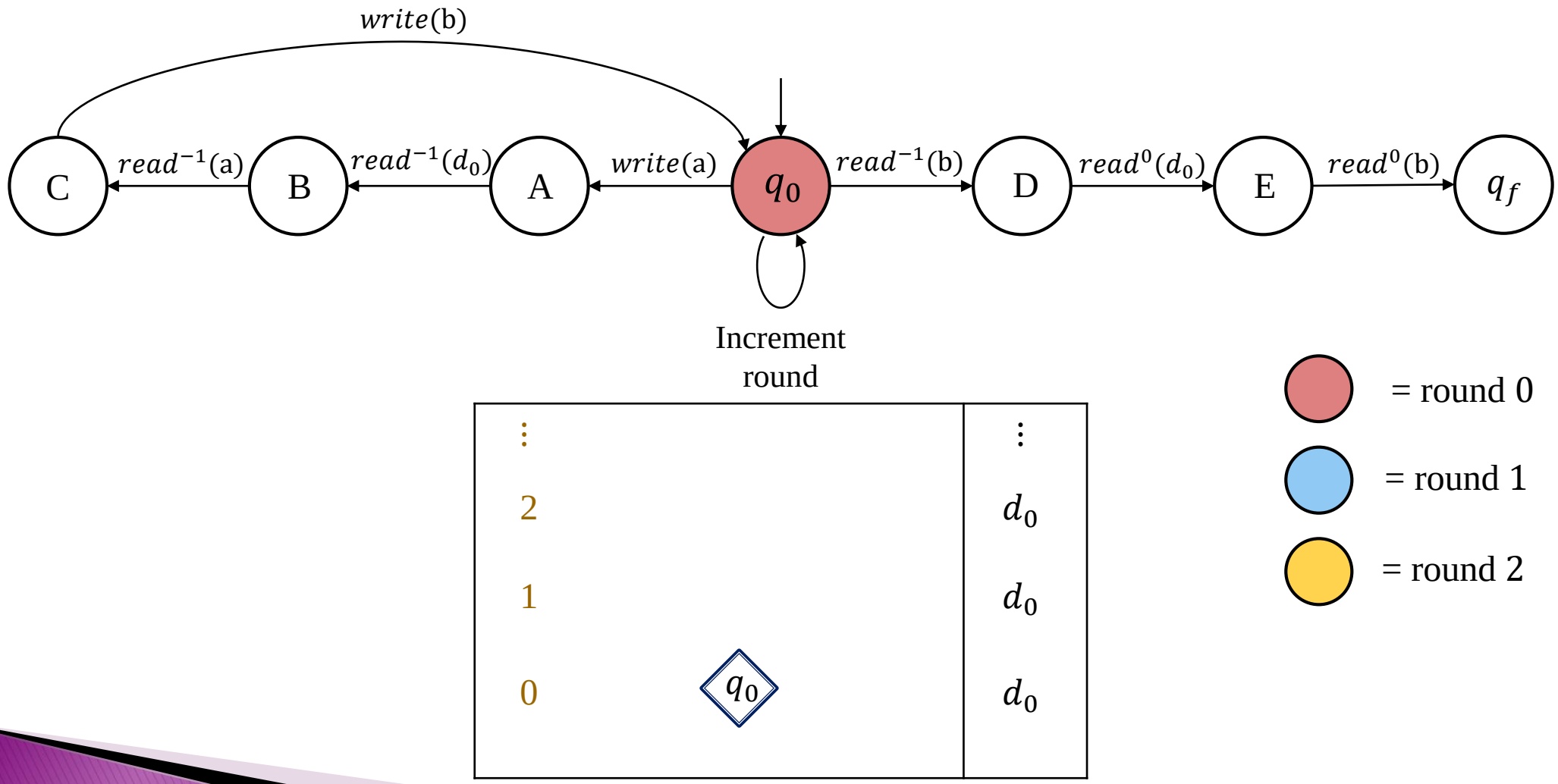
⋮		⋮
3	p ~~×1~~	d_0
2		b
1	q ~~×2~~ r ~~×1~~	b
0		d_0

Initial configurations:

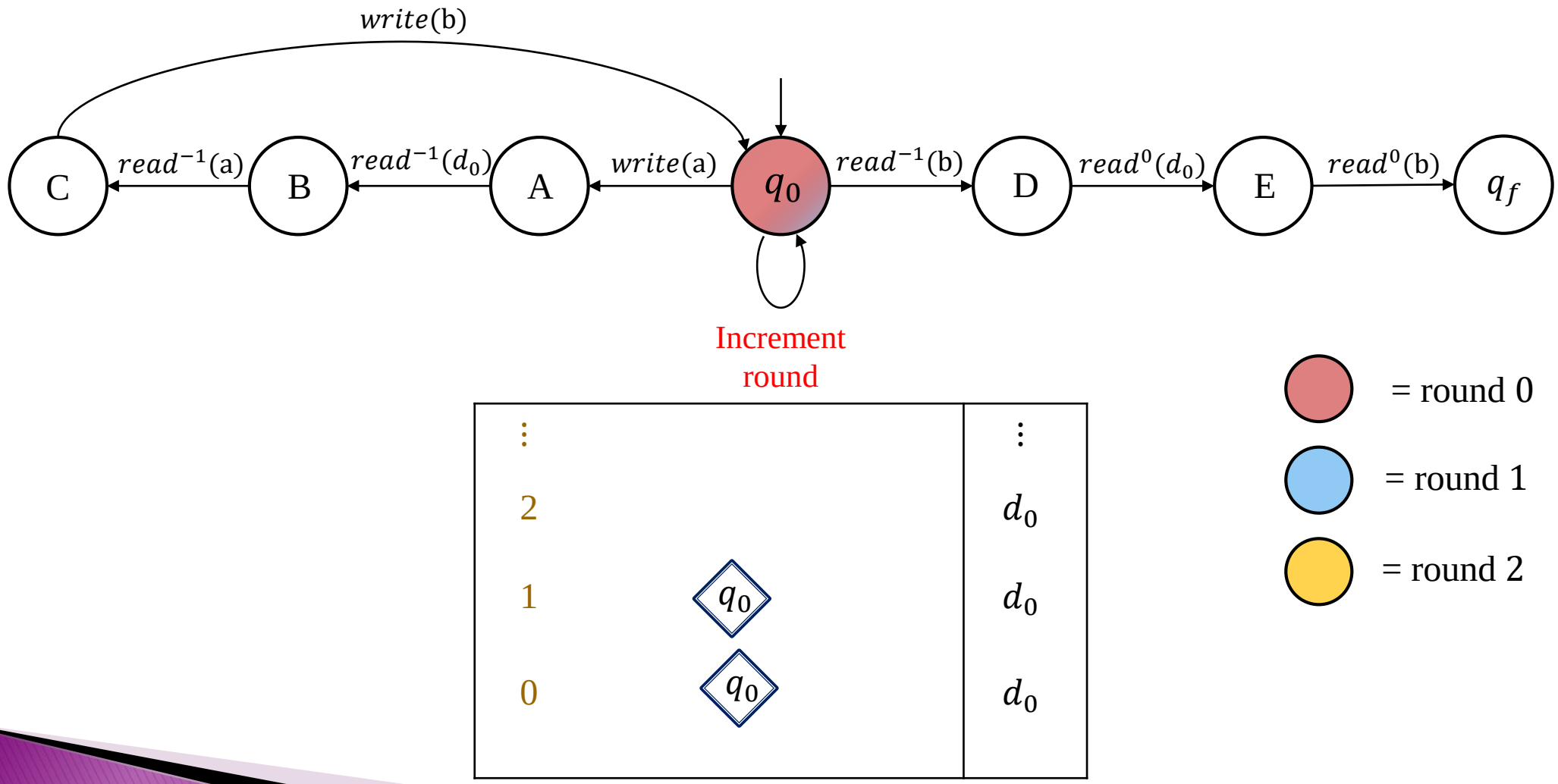
⋮		⋮
2		d_0
1		d_0
0	q_1 ~~×n_1~~ q_2 ~~×n_2~~ ...	d_0

with $n_1, n_2, \ldots \geq 0$ and $q_1, q_2, \ldots$ initial states

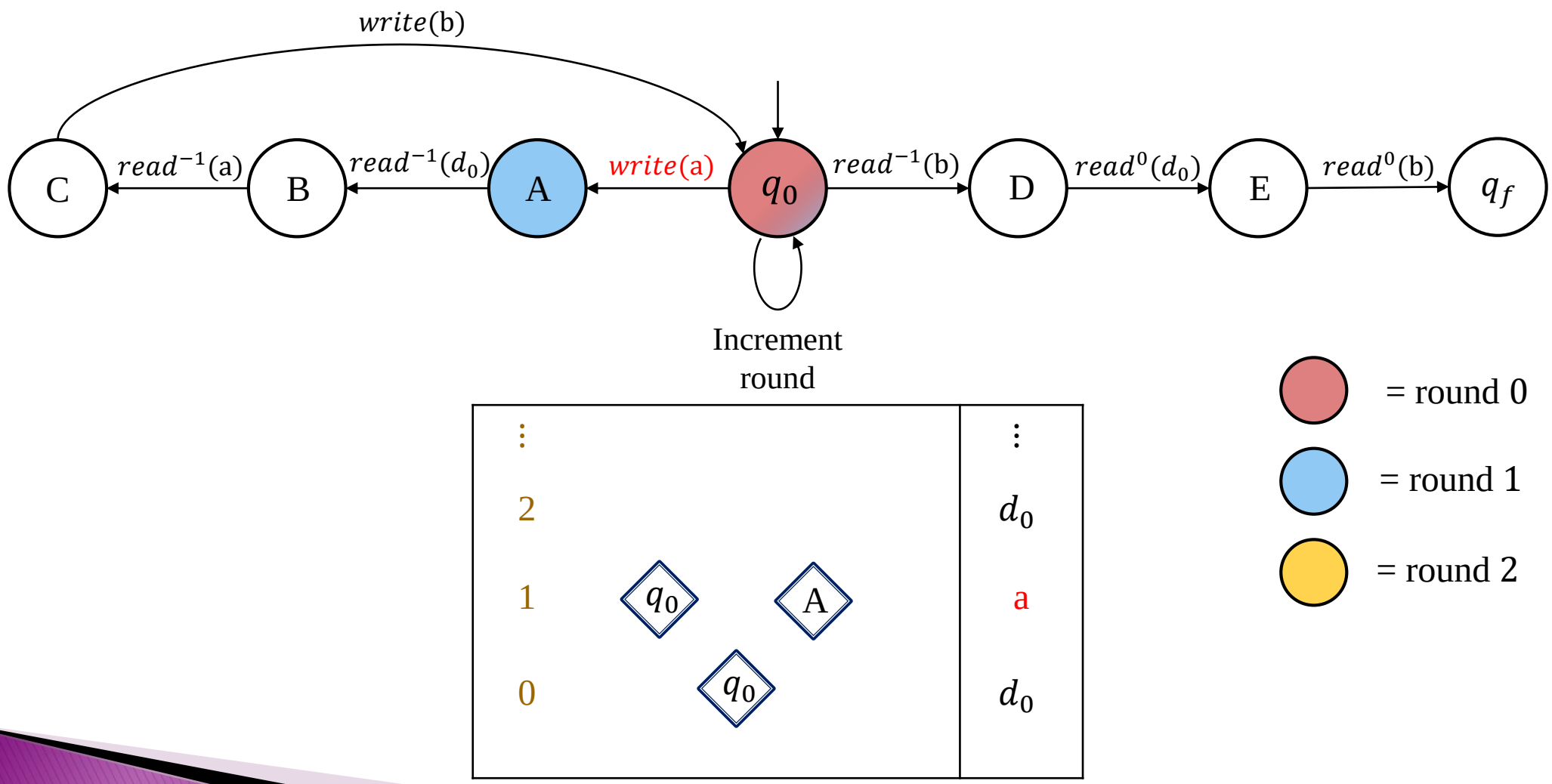
An example of round-based register protocol



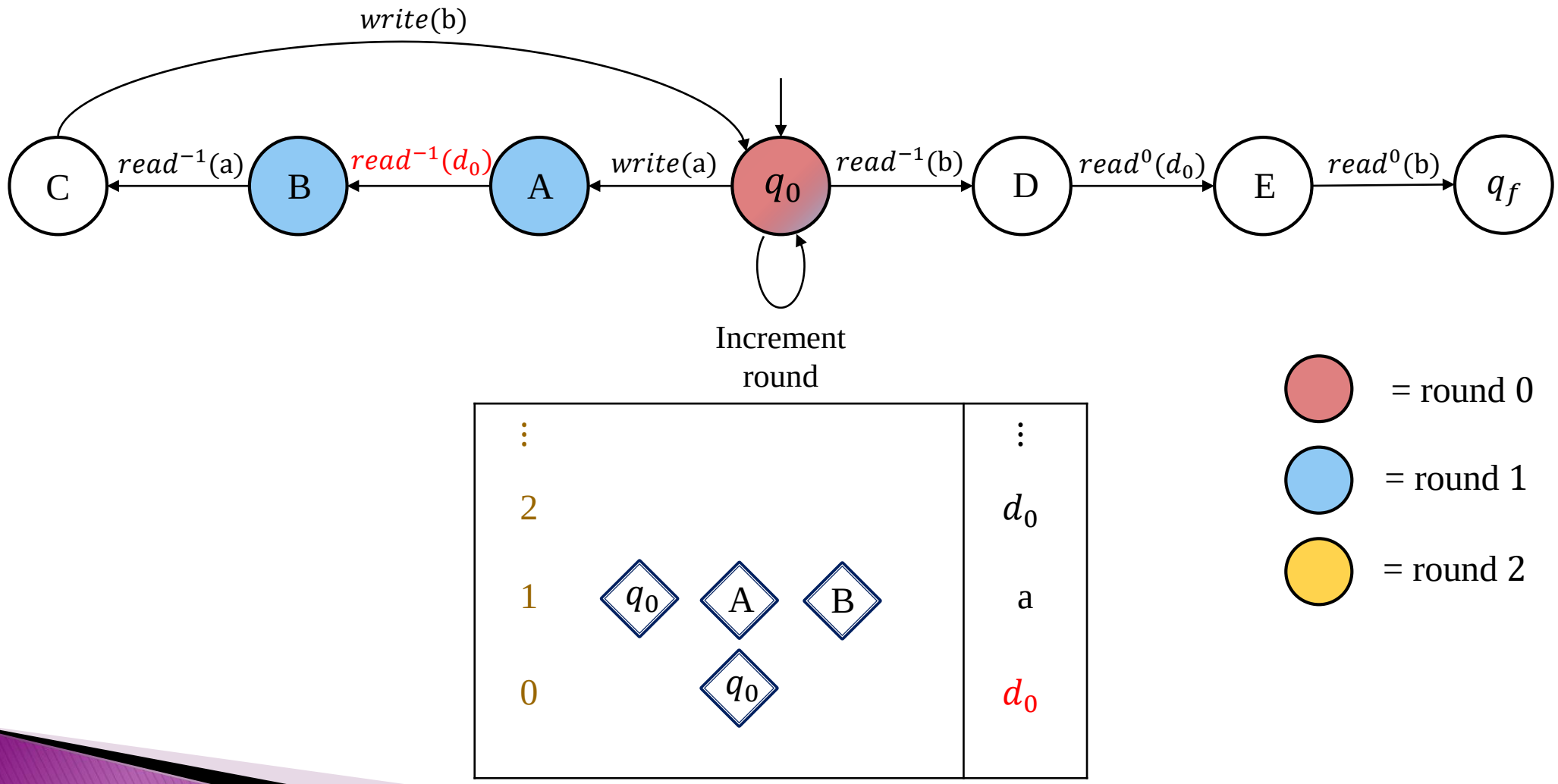
An example of round-based register protocol



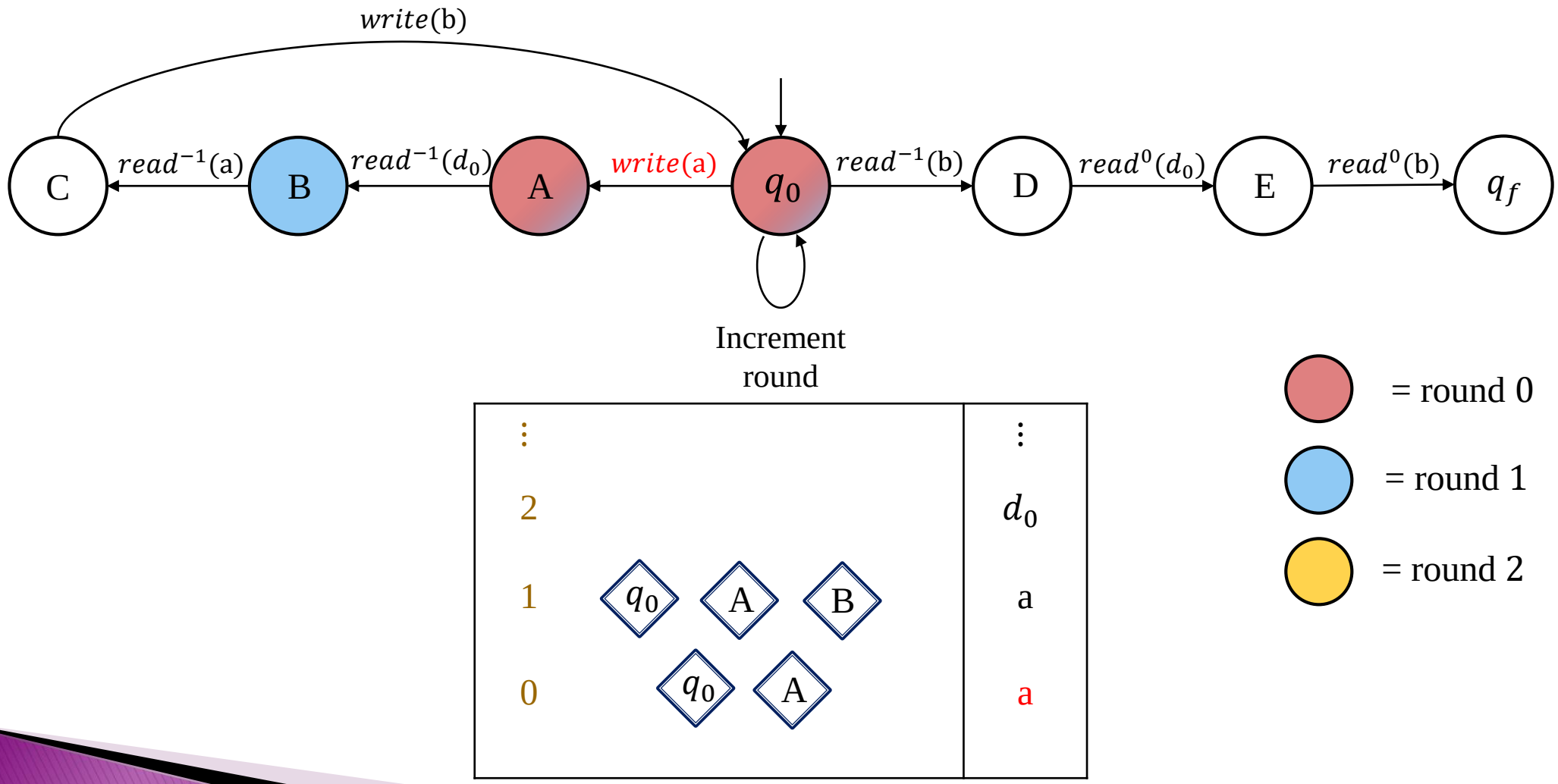
An example of round-based register protocol



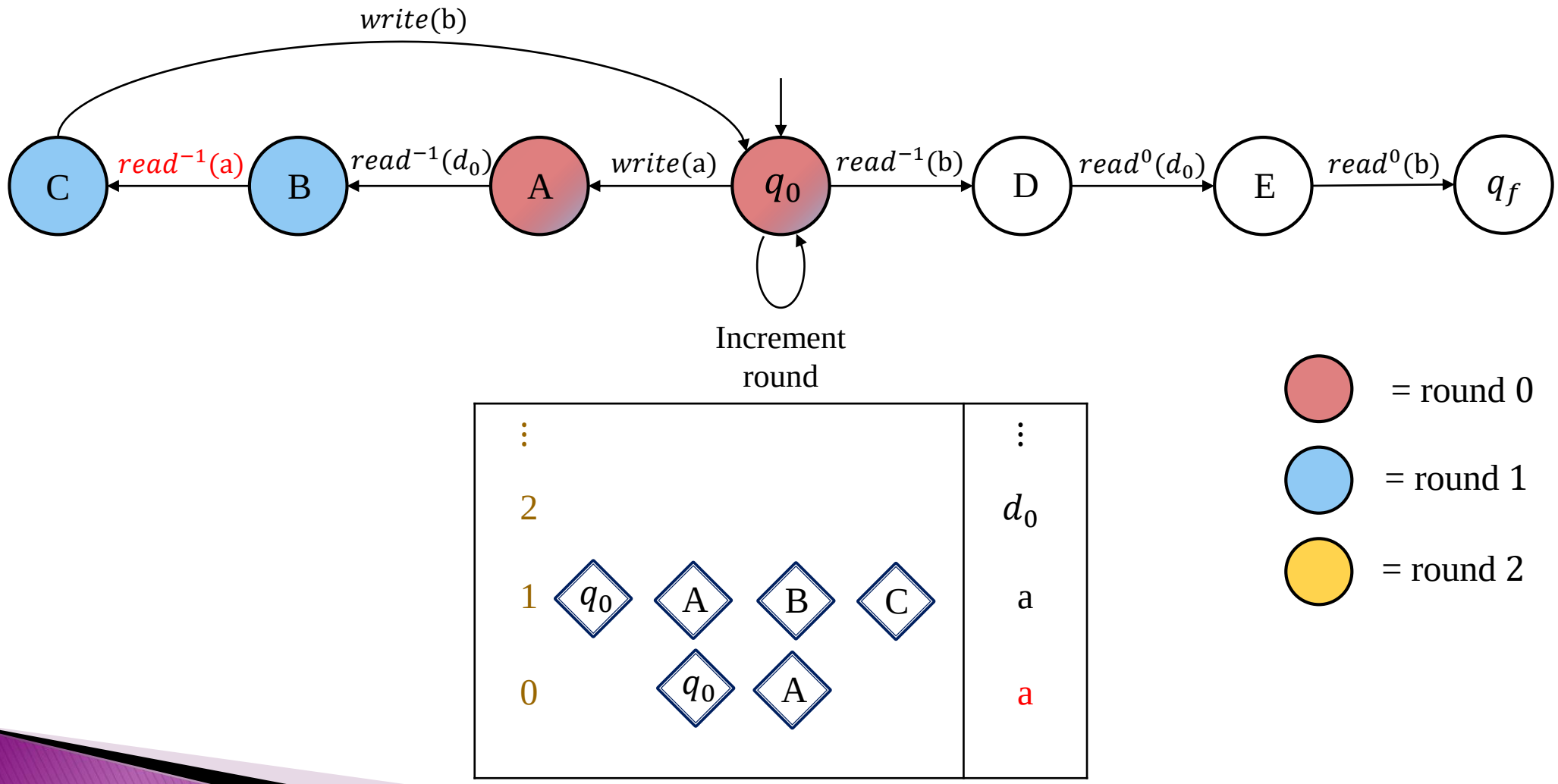
An example of round-based register protocol



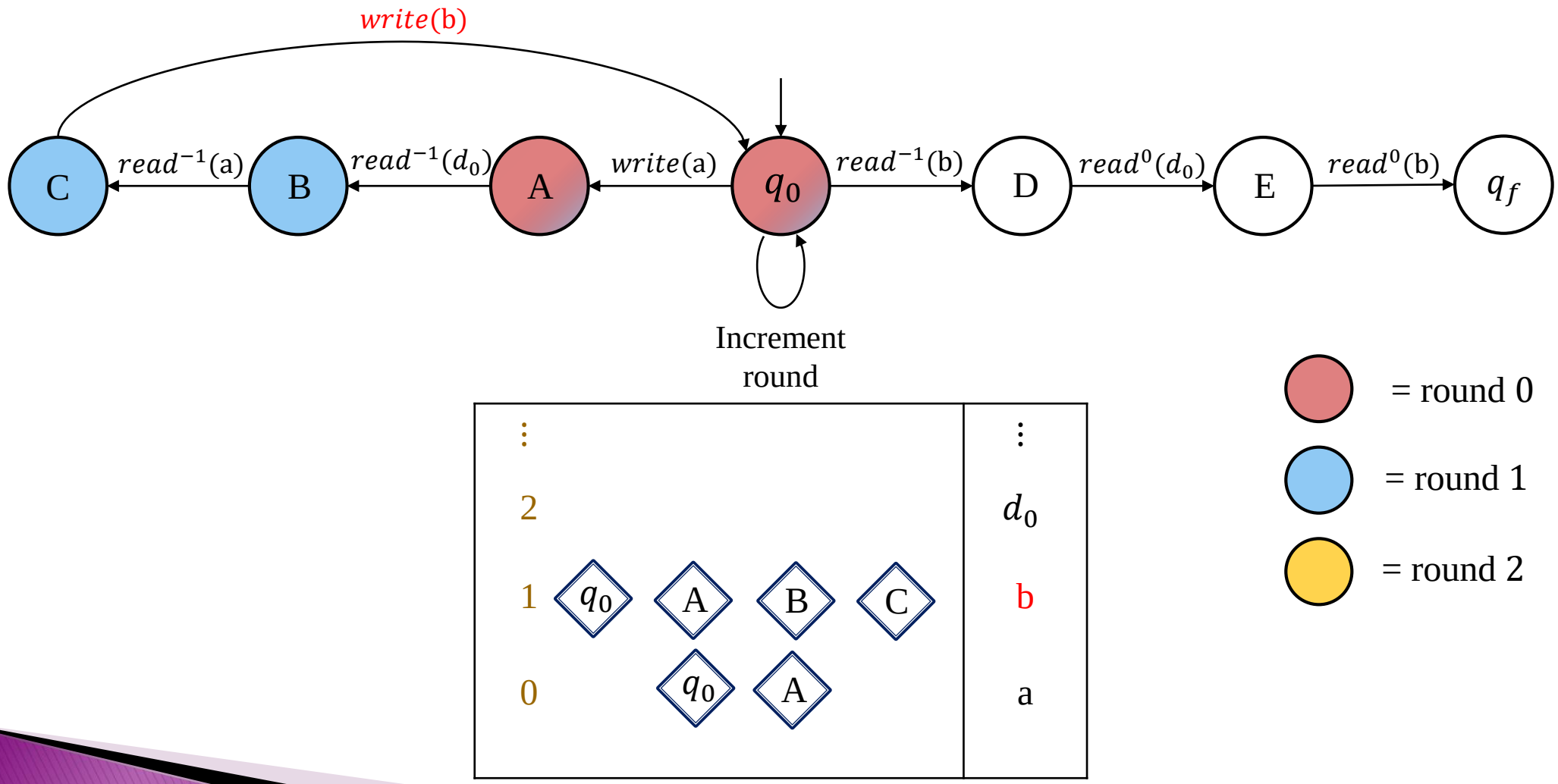
An example of round-based register protocol



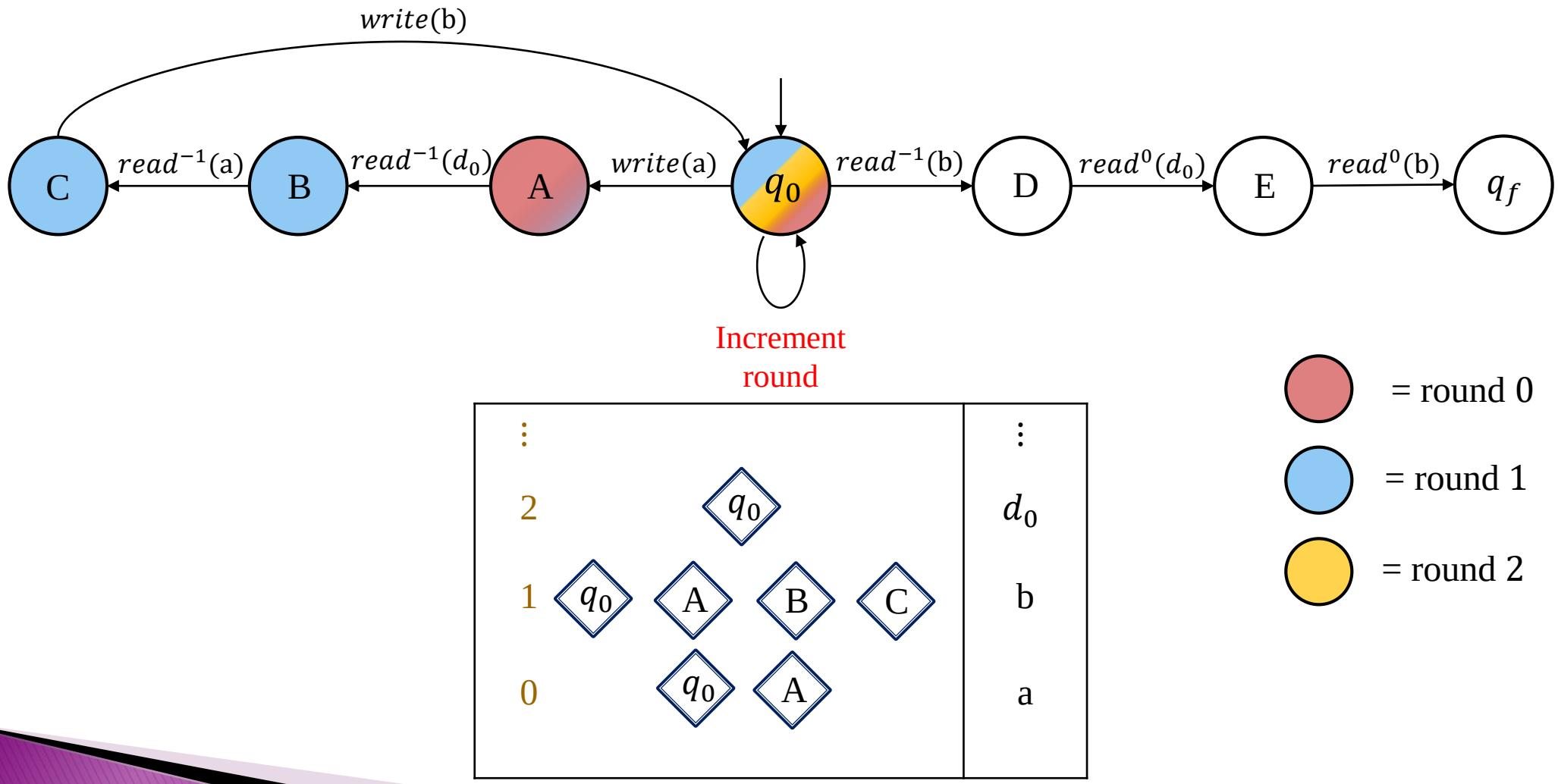
An example of round-based register protocol



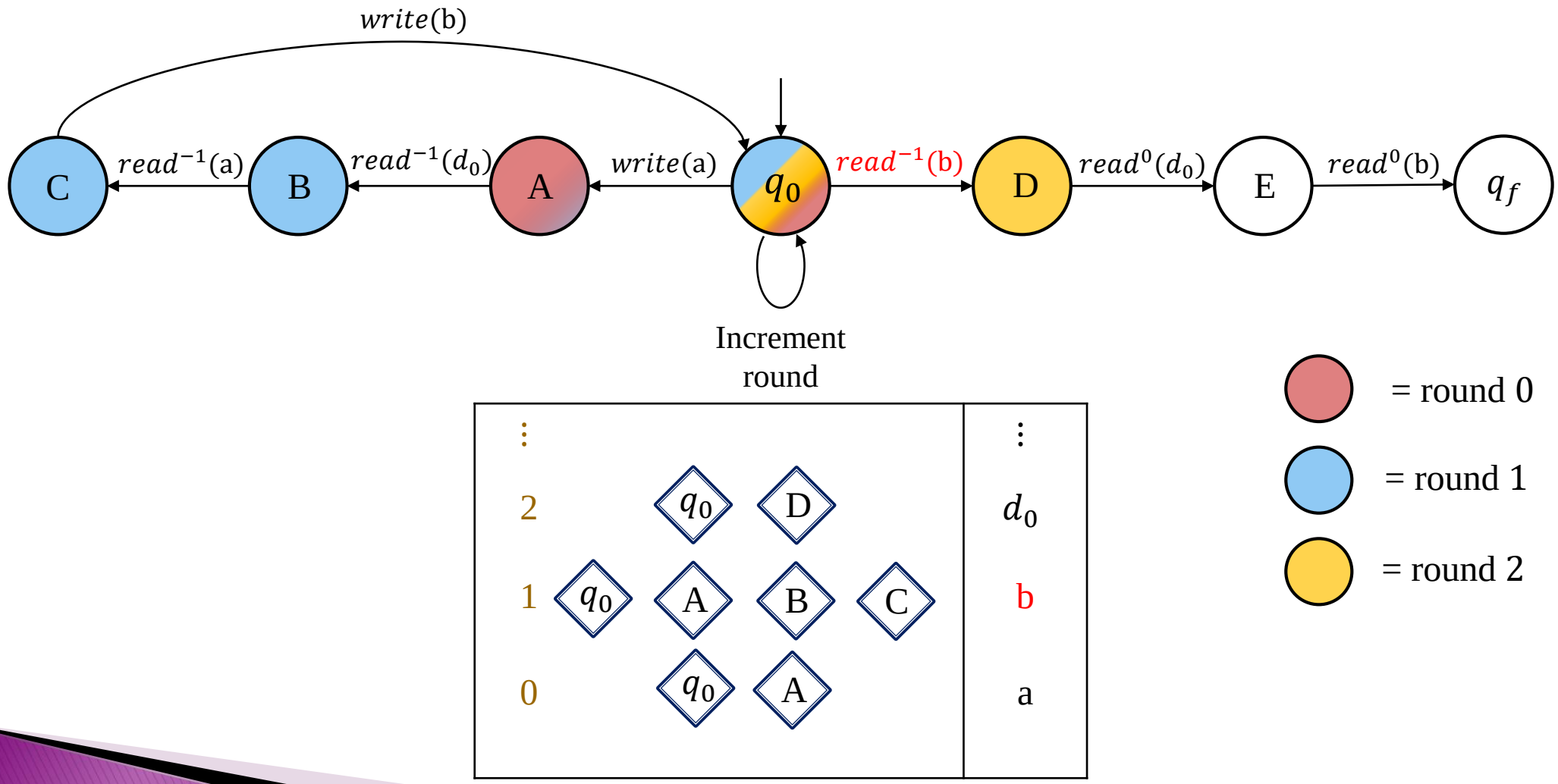
An example of round-based register protocol



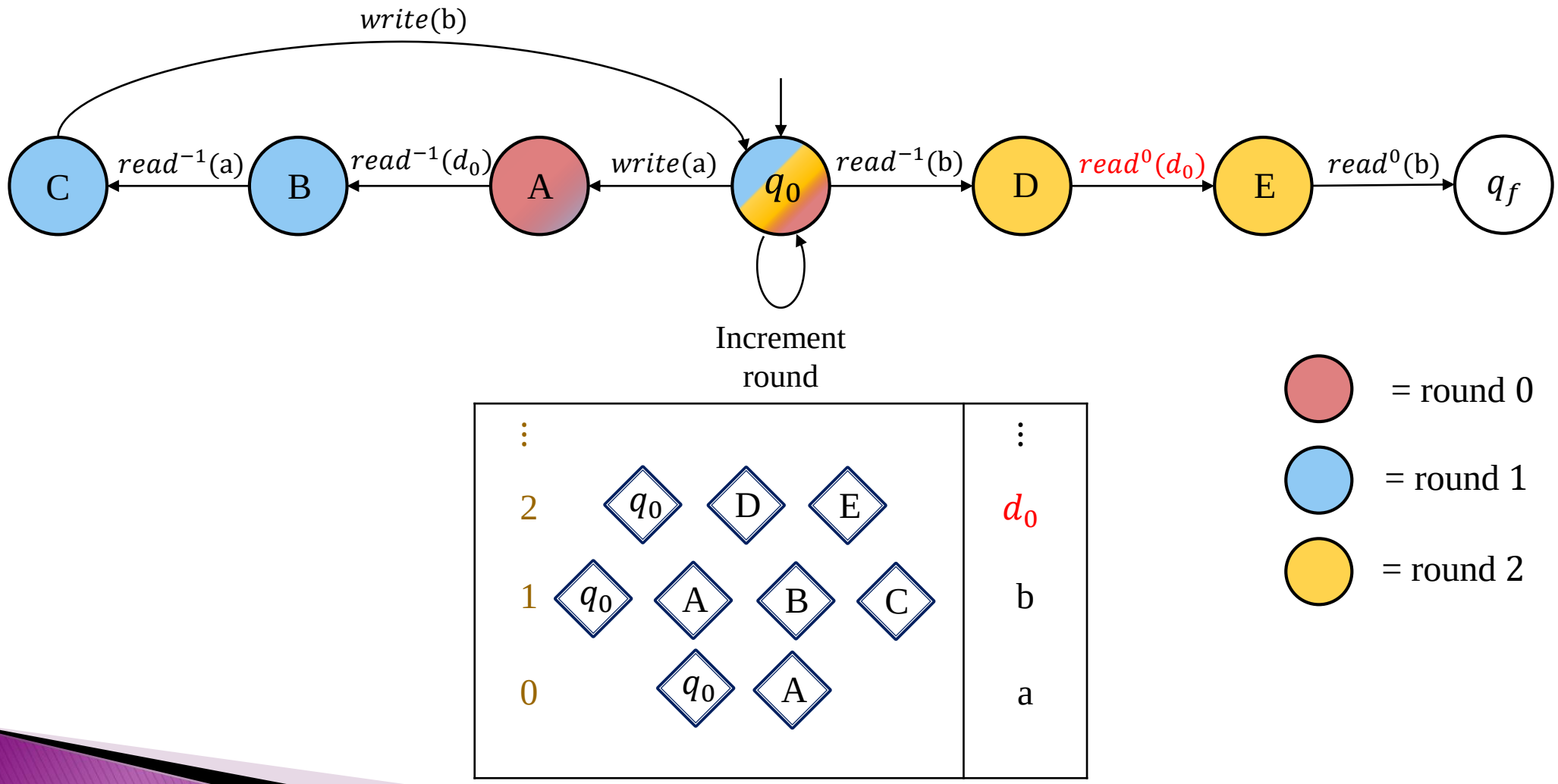
An example of round-based register protocol



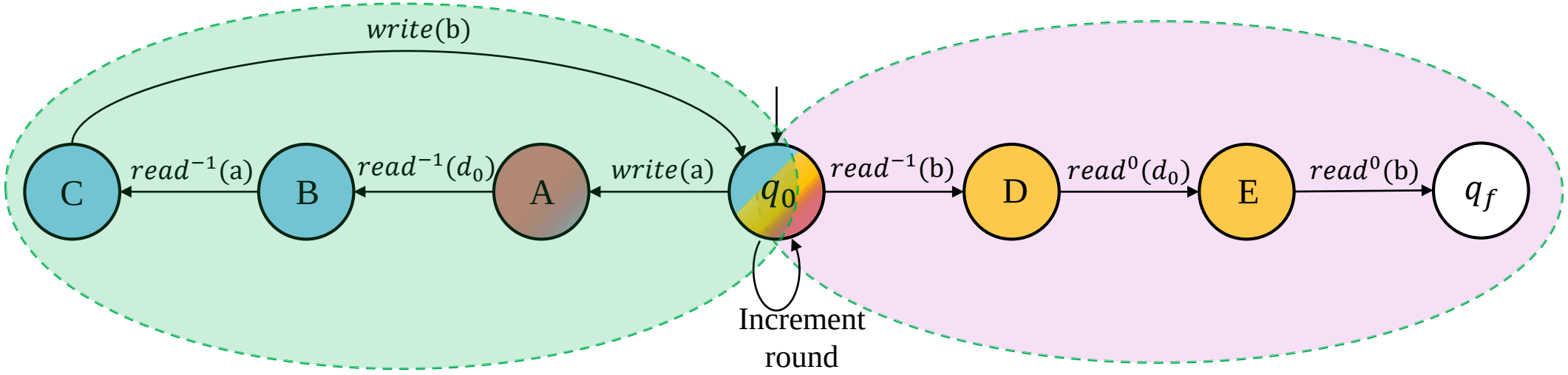
An example of round-based register protocol



An example of round-based register protocol



An example of round-based register protocol



To write b to **reg**$[k]$, one must write to **reg**$[k]$ while **reg**$[k-1]$ still has value d_0

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

q_f cannot be covered at any round!

Reachability problems in round-based setting

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

$\uparrow$

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

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

$\uparrow$

Every process is on state q_f regardless of its round

Reachability problems in round-based setting

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

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

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

with ψ a first-order formula on rounds with no nested quantifiers, whose terms are in $\{\#(q, k+c) = 0, \#(q, k+c > 0\} \cup \{\mathbf{reg}_i[k+c] = d, \mathbf{reg}_i[k+c] \neq d\}$

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

Complexity results

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

This is already true when:

- the reachability objective is a coverability objective,
- the visibility range is equal to 0 (processes cannot see previous rounds),
- there is only one register per round,
- the registers are uninitialized.

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 PRP is PSPACE-hard.

This is already true when:

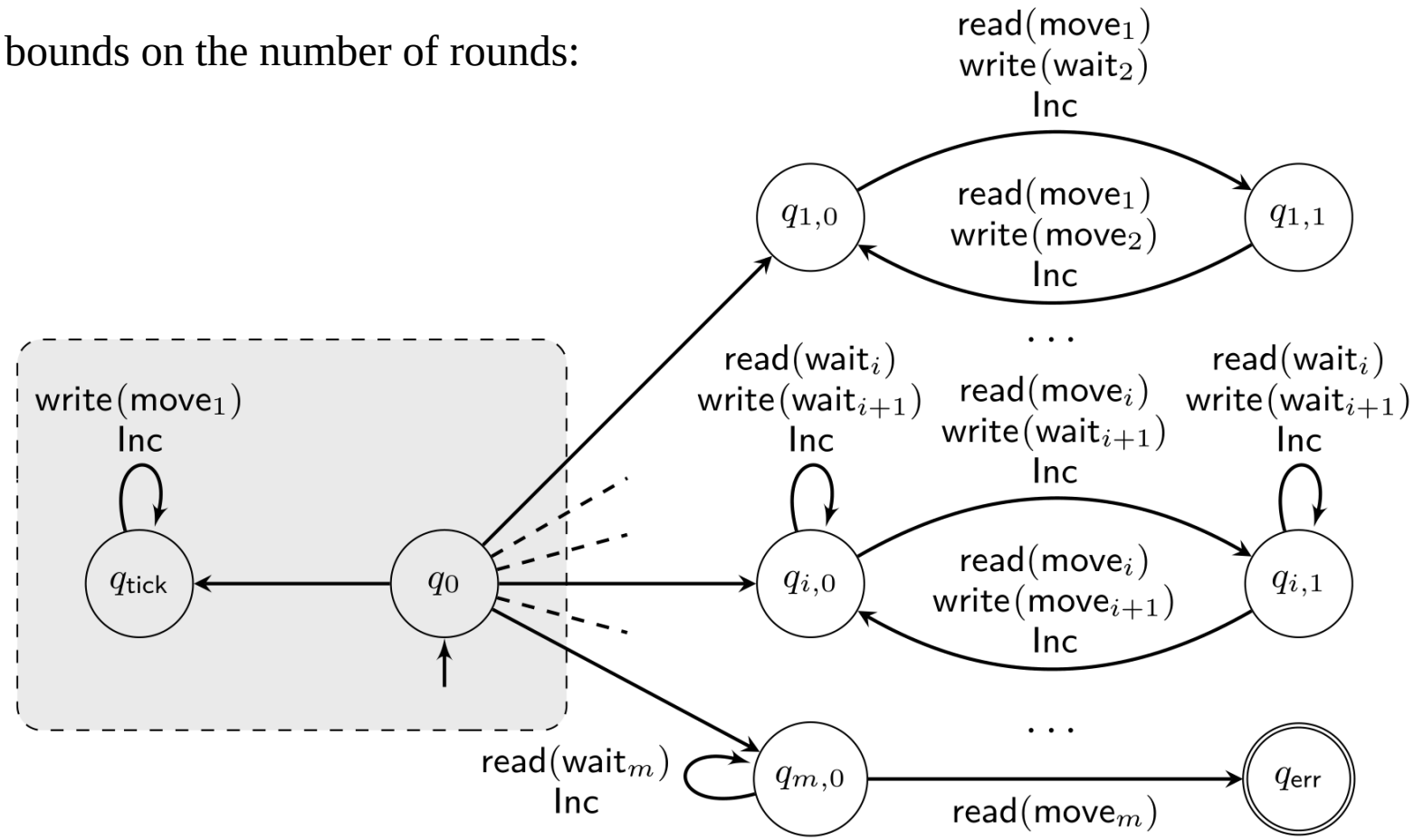
- the reachability objective is a coverability objective,
- the visibility range is equal to 0 (processes cannot see previous rounds),
- there is only one register per round,
- the registers are uninitialized.

Theorem[910]*:* 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

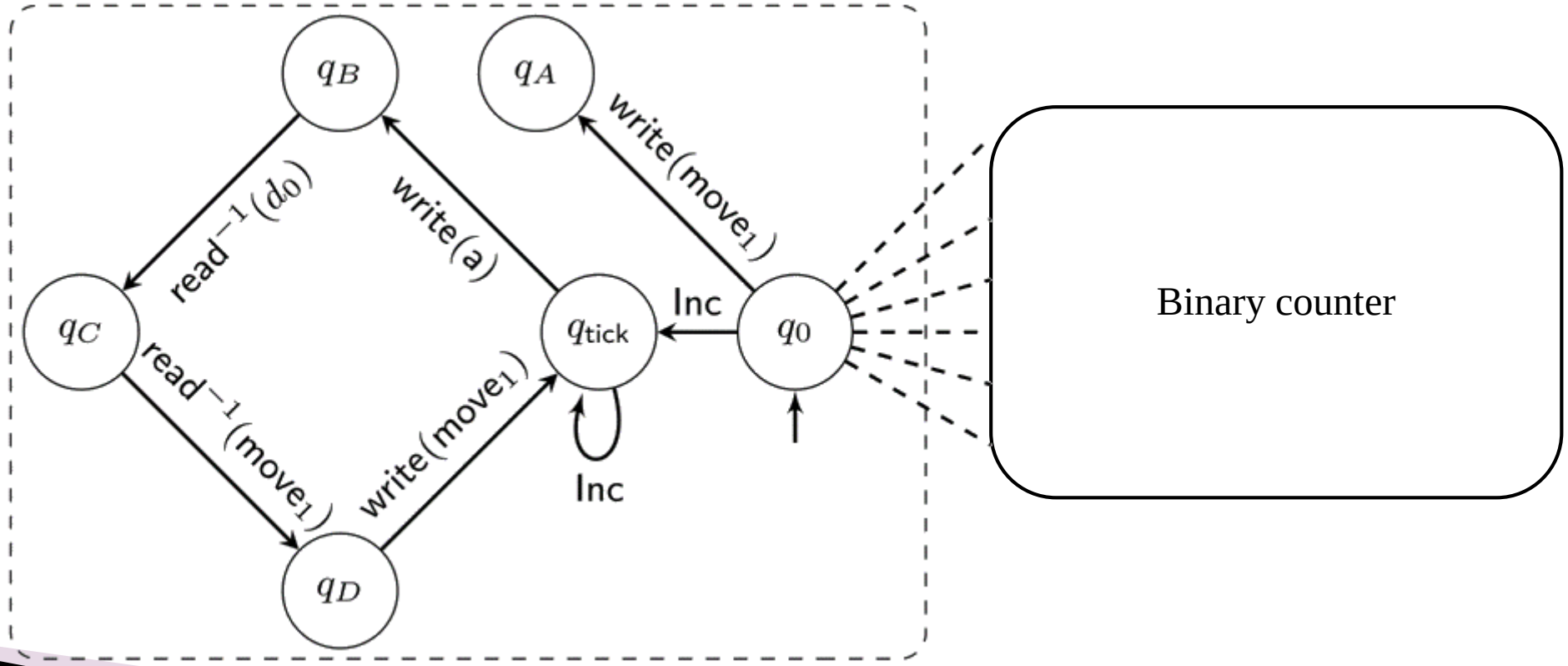
A challenge: exponential lower bounds

Exponential lower bounds on the number of rounds:



A challenge: exponential lower bounds

Exponential lower bounds on the number *active* rounds:



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

$$\Downarrow$$

Actions:	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Rounds:	1	4	3	2	0	1	4

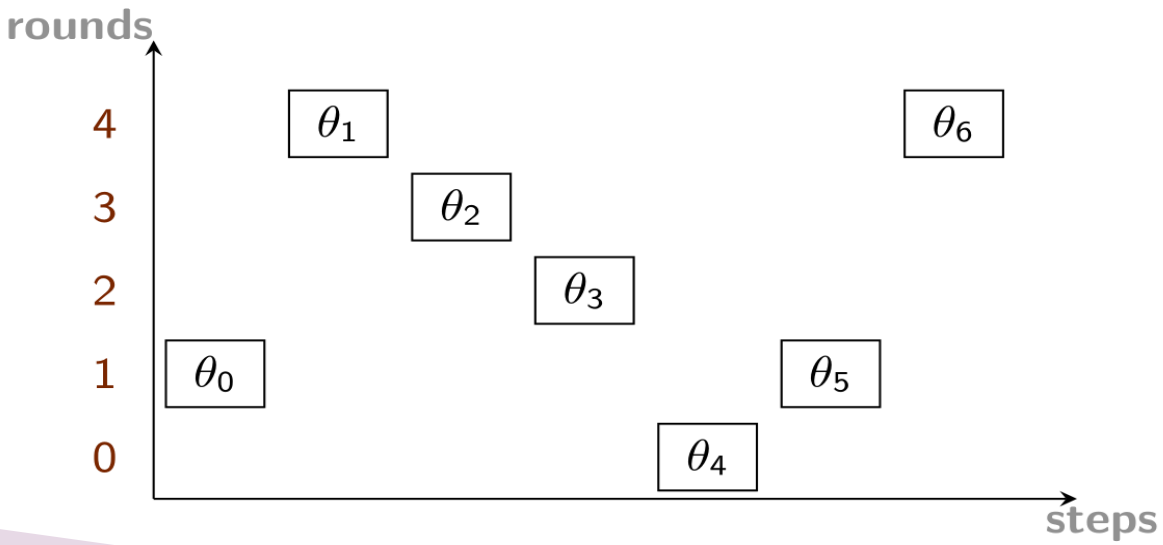
A non-deterministic polynomial-space algorithm

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

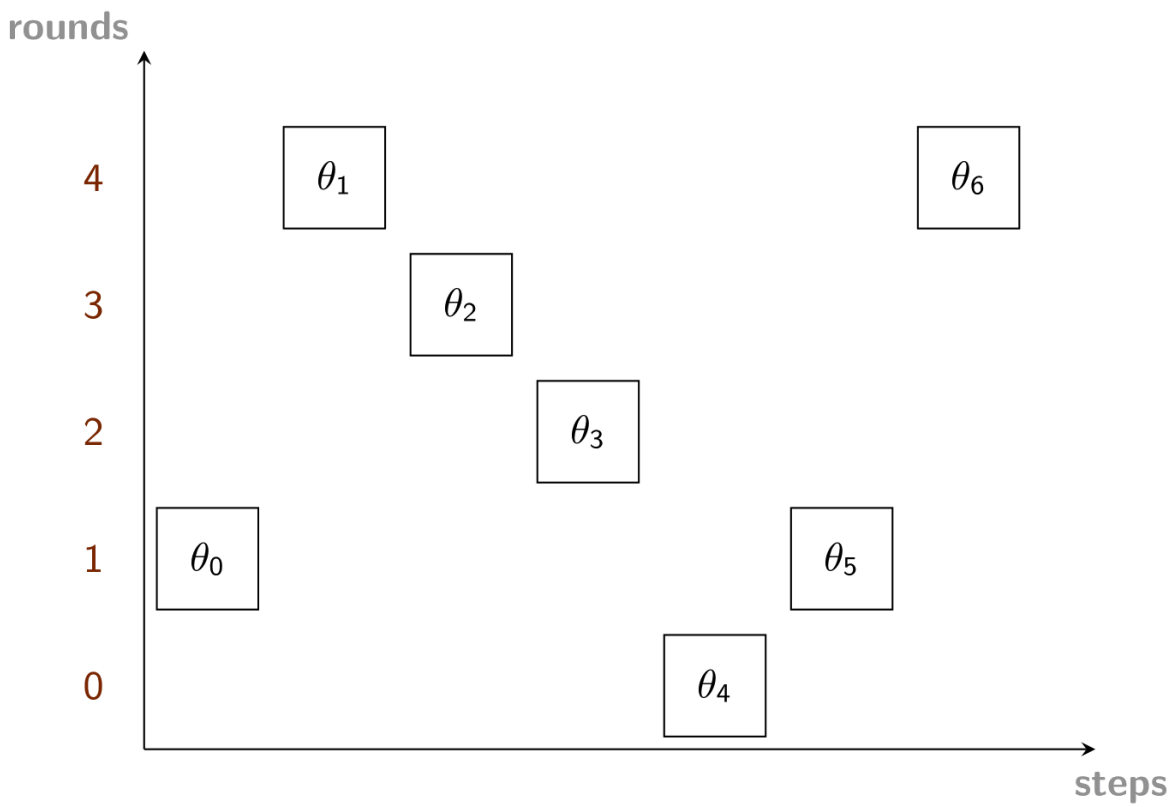
$\Downarrow$

Actions:	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
Rounds:	1	4	3	2	0	1	4

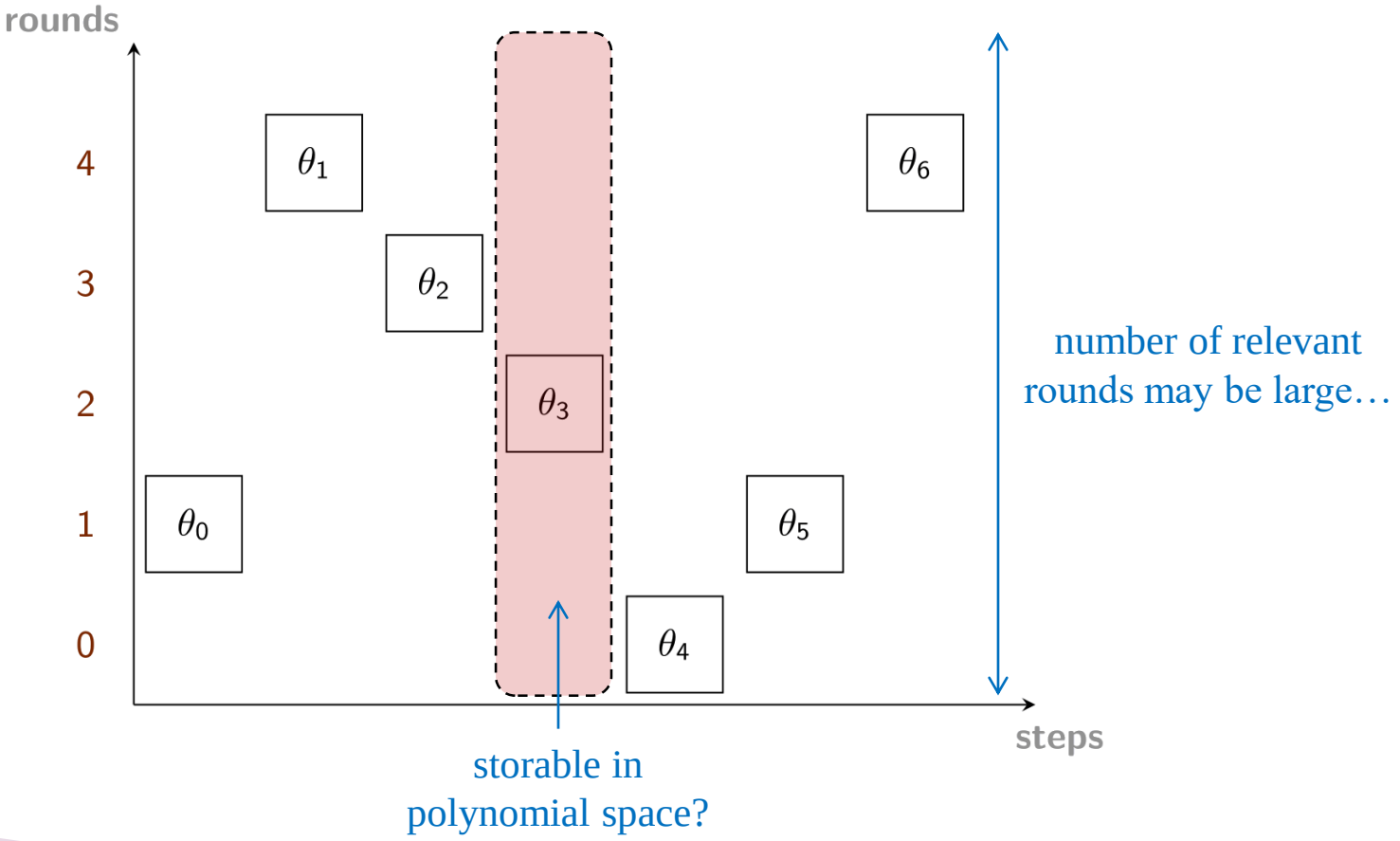
$\Downarrow$



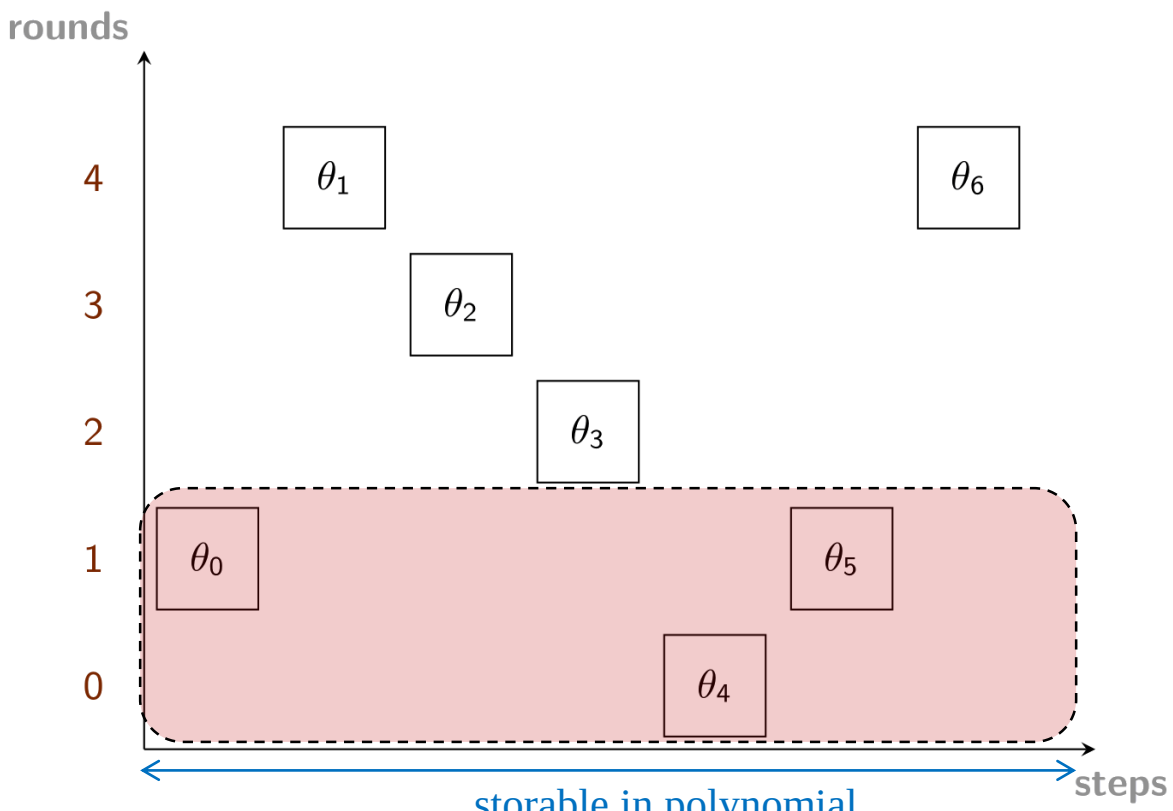
A non-deterministic polynomial-space algorithm



A non-deterministic polynomial-space algorithm



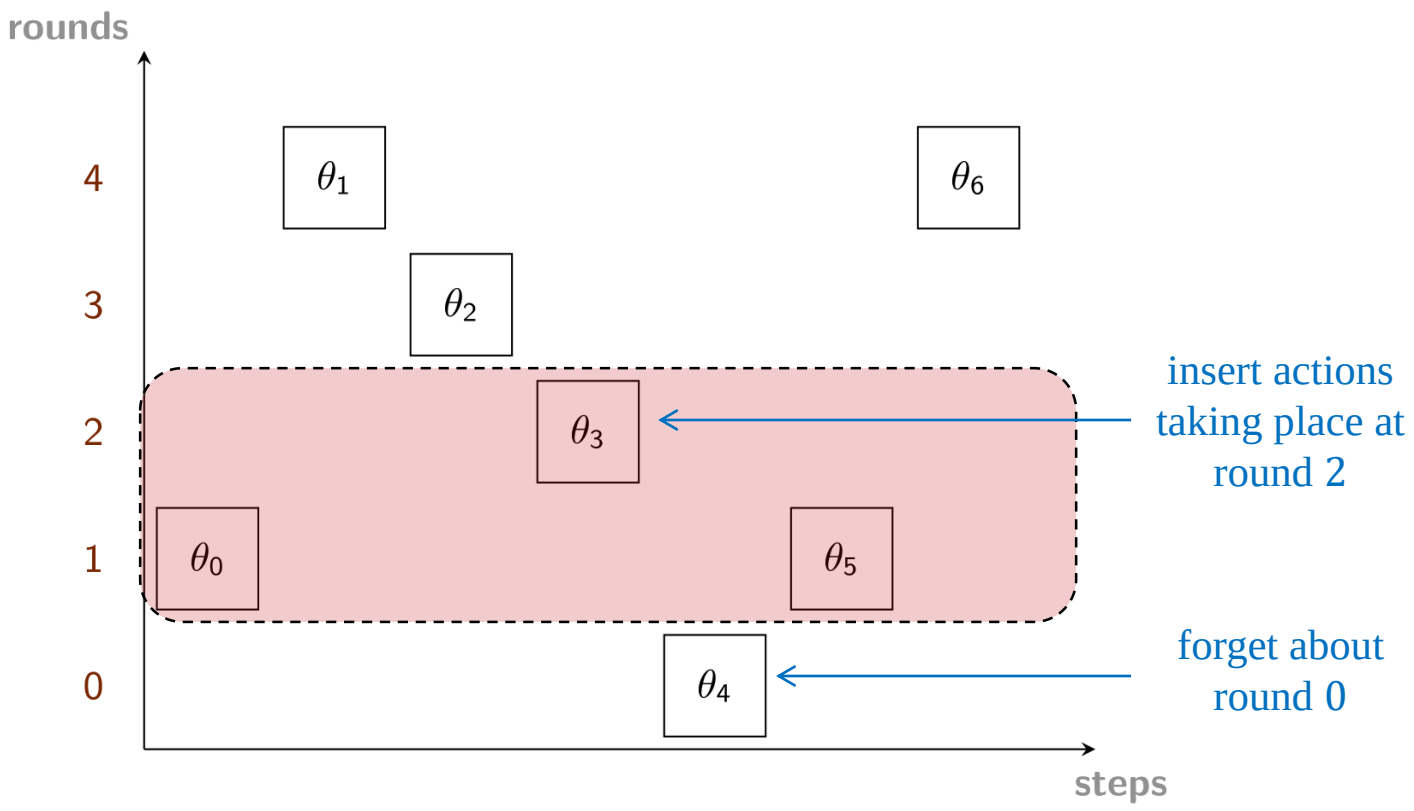
A non-deterministic polynomial-space algorithm



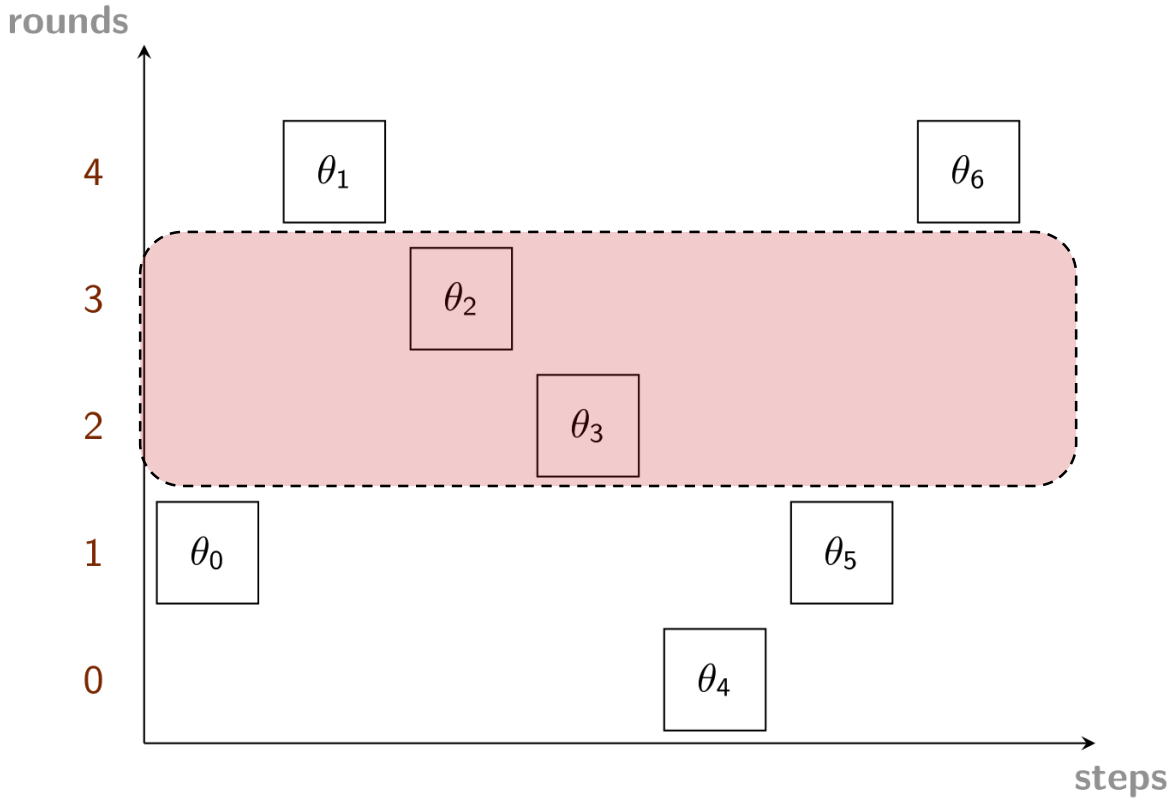
sliding window
on $v + 1$ rounds
(here $v = 1$)

storable in polynomial
space using abstract
representation

A non-deterministic polynomial-space algorithm

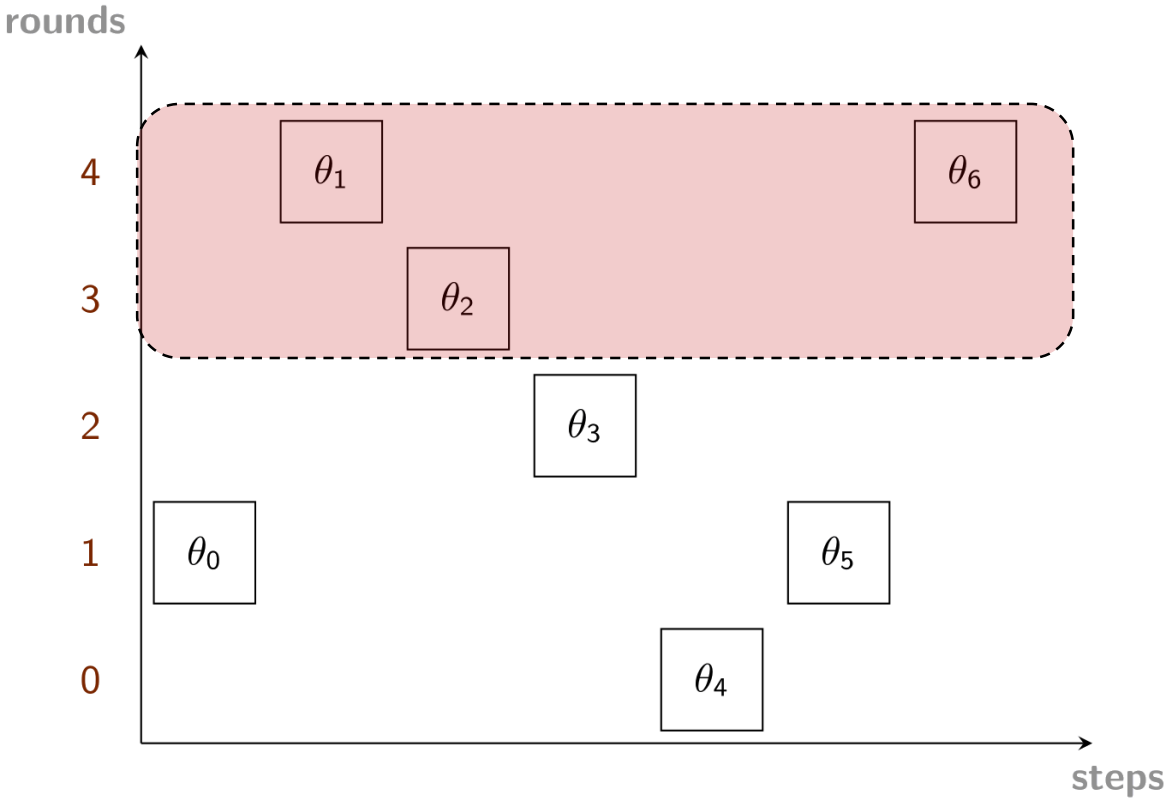


A non-deterministic polynomial-space algorithm



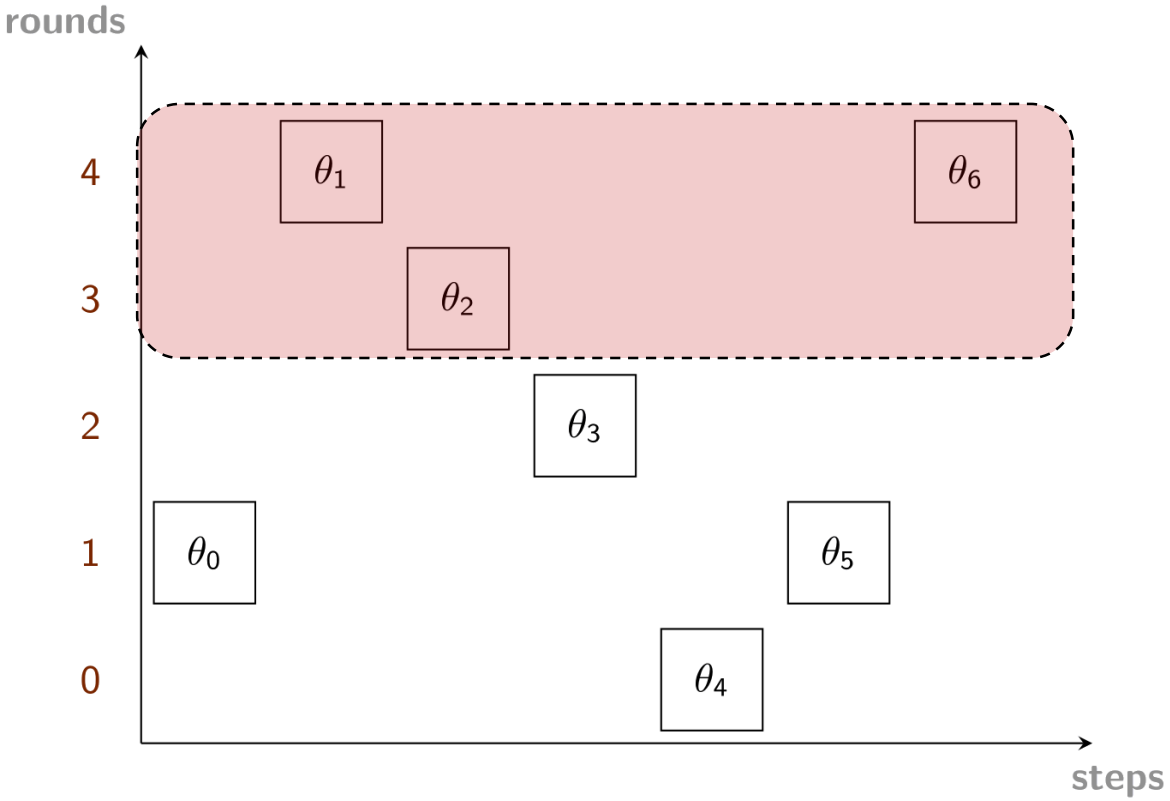
A non-deterministic polynomial-space algorithm

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.

Conclusion

Summary

- Two models in this talk: *roundless* register protocols and *round-based* register protocols.
- Properties studied are *reachability properties* which do not "count" processes. Two classical such problems are COVER and TARGET; PRP is a general class which encompasses these two problems.
- In the first model, despite its simplicity, PRP is NP-complete, but some restrictions make it PTIME.
- In the second model, PRP is PSPACE-complete, and similar restrictions do not decreases the complexity. The polynomial-space algorithm relies, among others, on a *sliding window along the rounds*.

Future work

- Almost-sure reachability in round-based register protocols with stochastic schedulers
- Cube reachability (= initial and final constraints may "count" processes), link with almost-sure reachability
- Weak memory

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

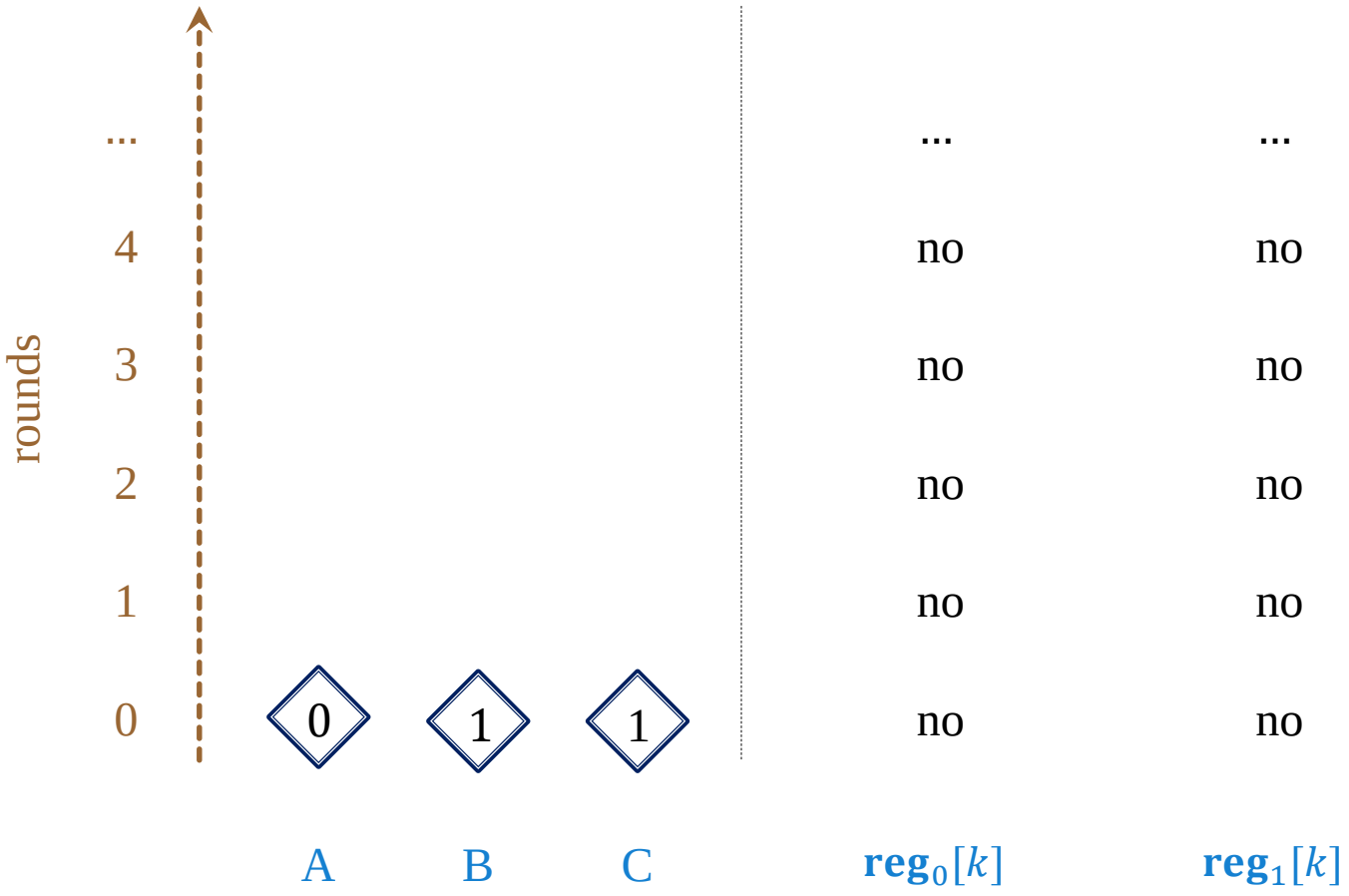
Aspnes' consensus algorithm[3]:

```
int k := 0, bool p ∈ {0,1}, (rg_b[r])_{b∈{0,1},r∈ℕ} all initialized to no;
while true do
    read from rg_0[k] and rg_1[k];
    if rg_0[k] = yes and rg_1[k] = no then p := 0;
    else if rg_0[k] = no and rg_1[k] = yes then p := 1;
    write yes to rg_p[k];
    if k > 0 then
        read from rg_{1-p}[k-1];
        if rg_{1-p}[k-1] = no then return p;
    k := k+1;
```

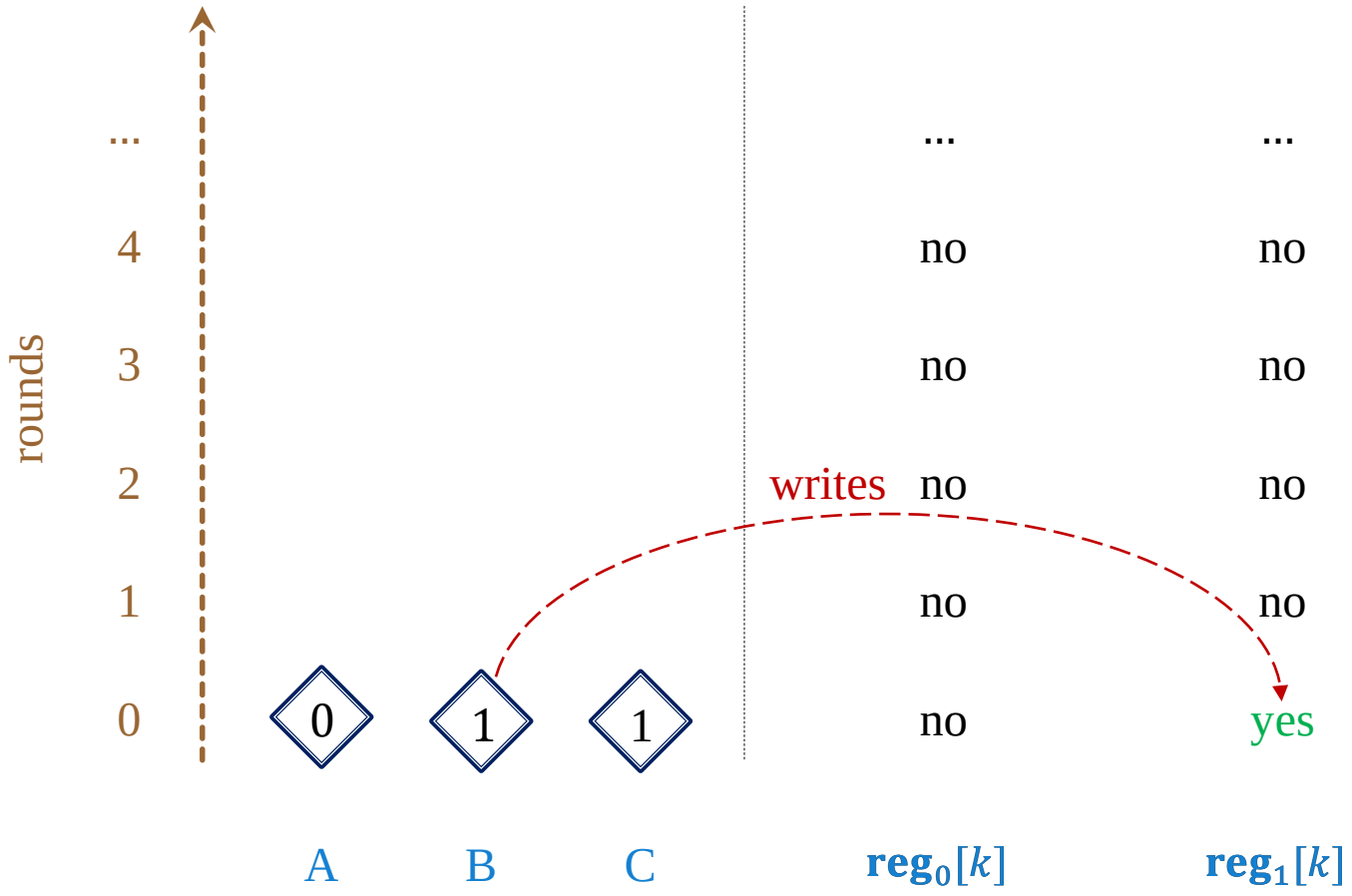
read from registers of rounds k and $k-1$

write to registers of round k

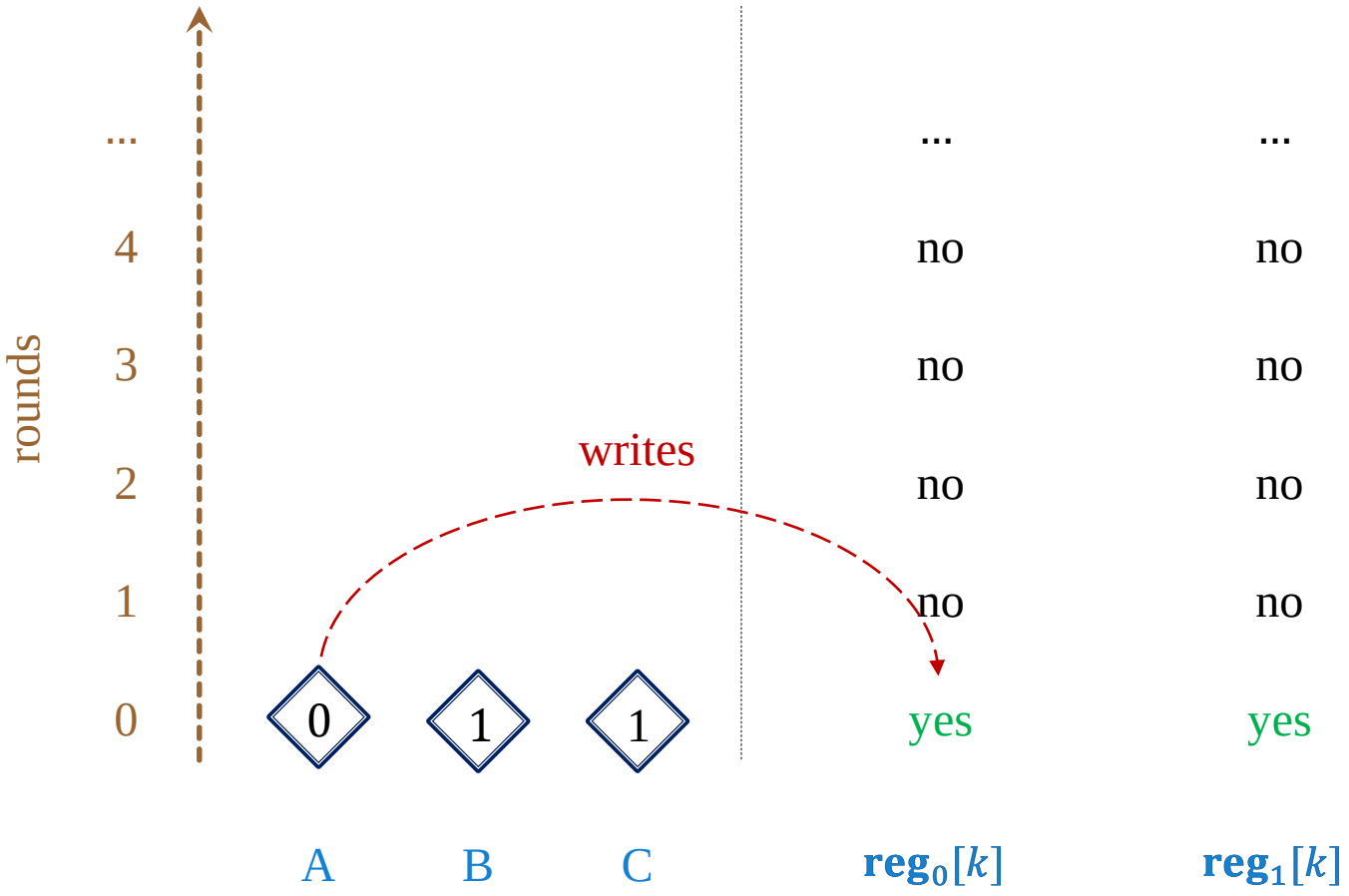
Example of execution of the algorithm



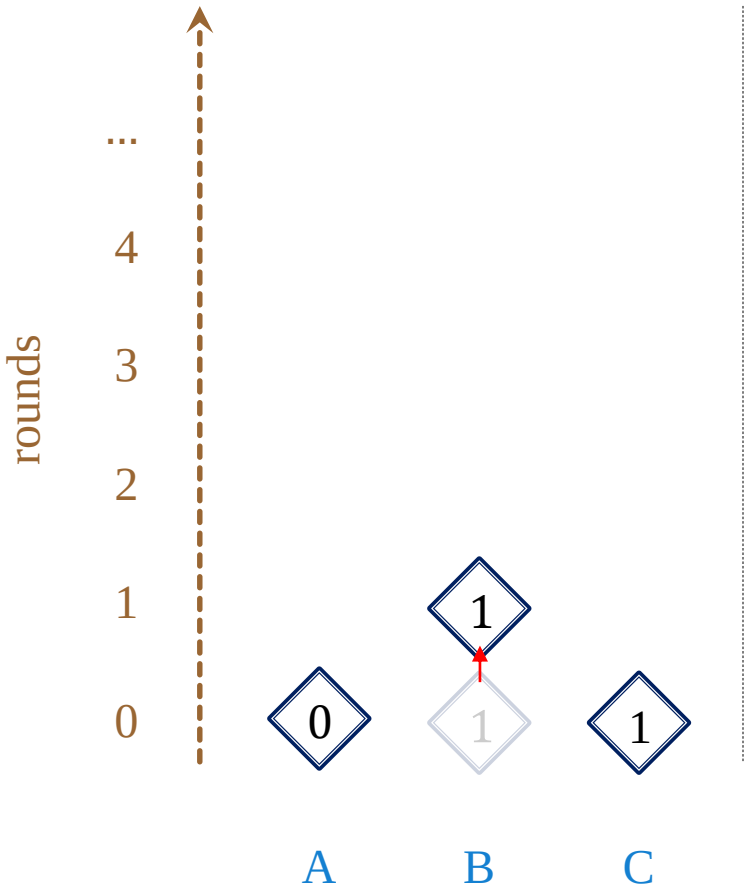
Example of execution of the algorithm



Example of execution of the algorithm

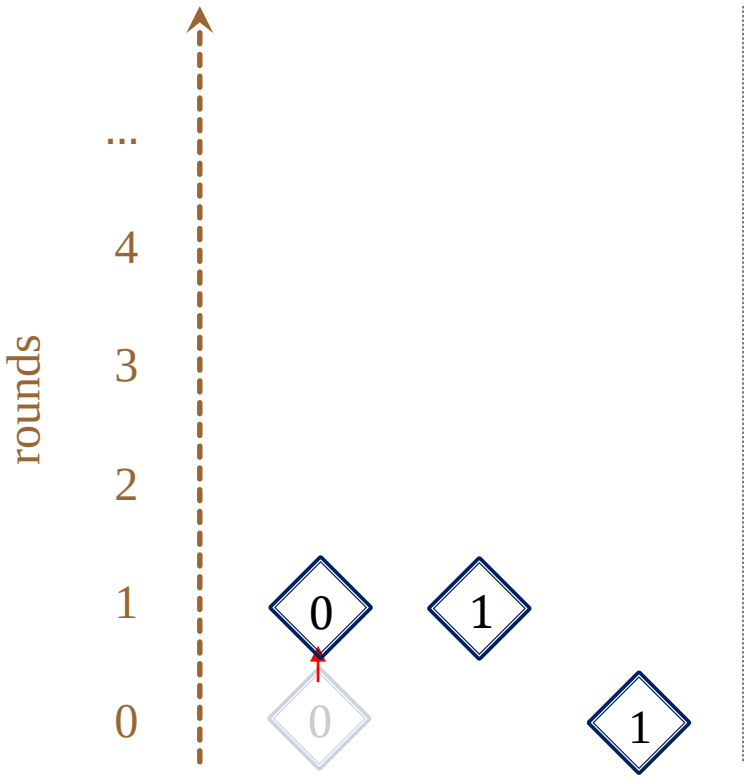


Example of execution of the algorithm



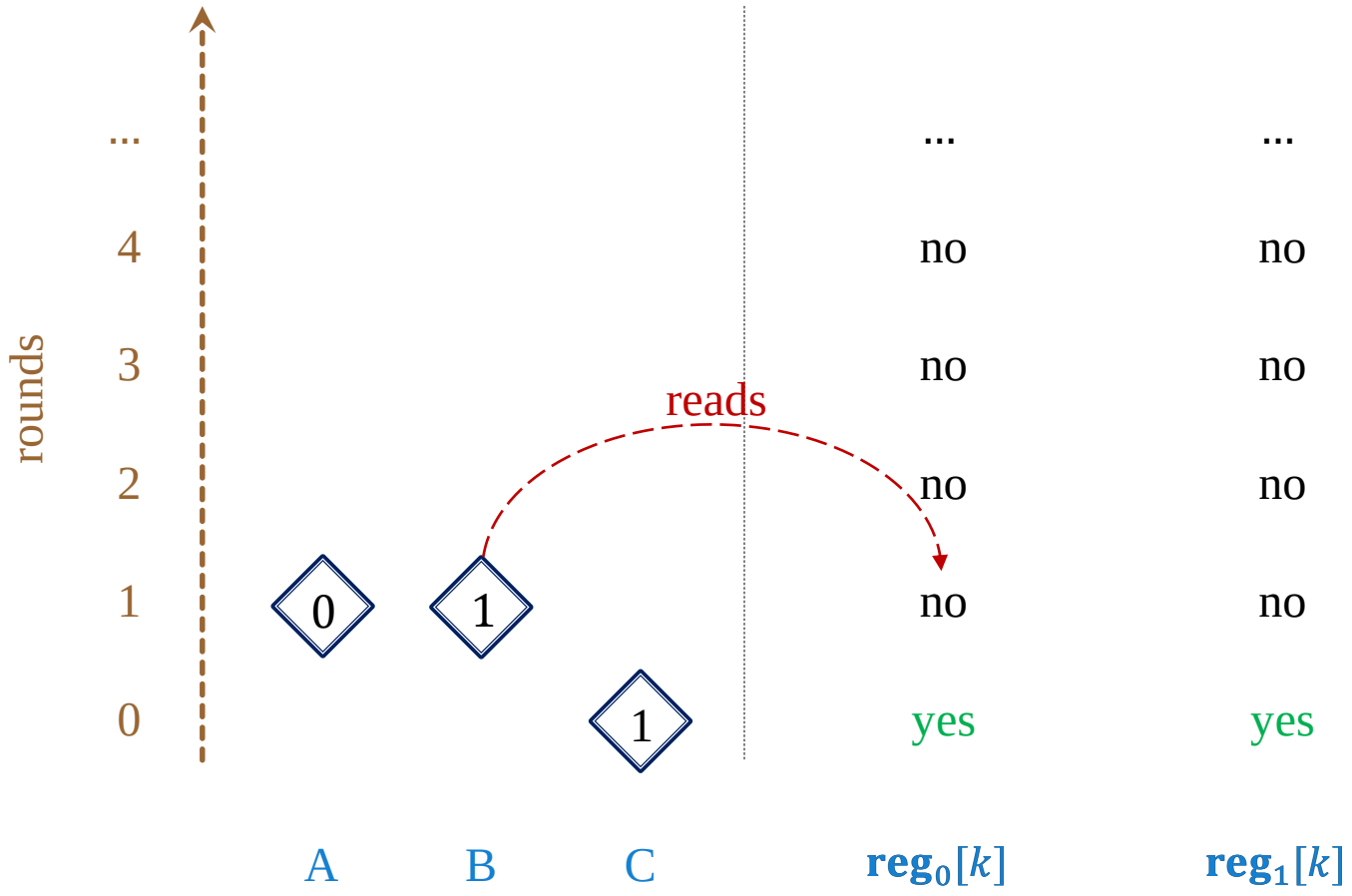
	$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...	...
4	no	no
3	no	no
2	no	no
1	no	no
0	yes	yes

Example of execution of the algorithm

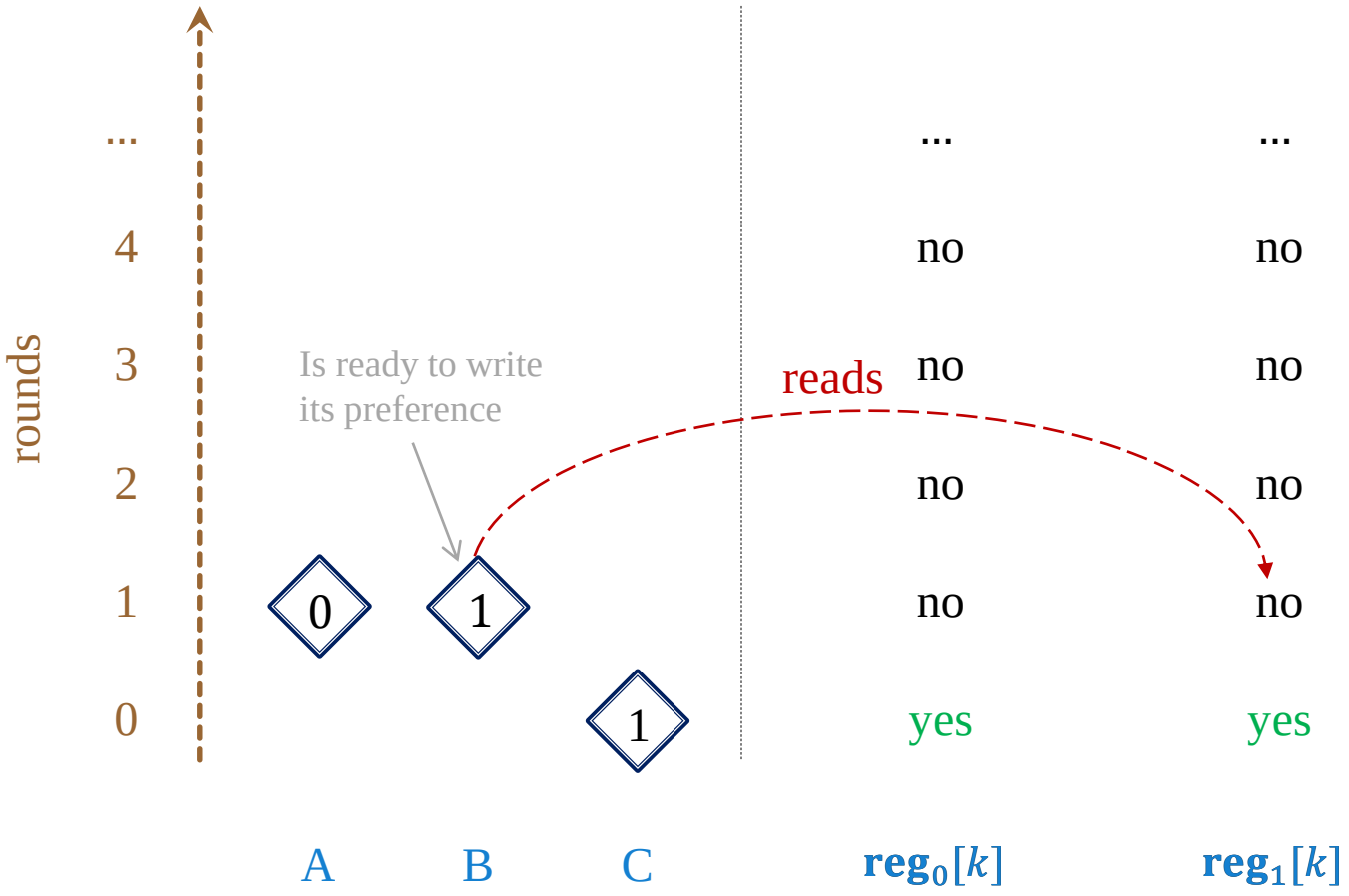


	$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...	...
4	no	no
3	no	no
2	no	no
1	no	no
0	yes	yes

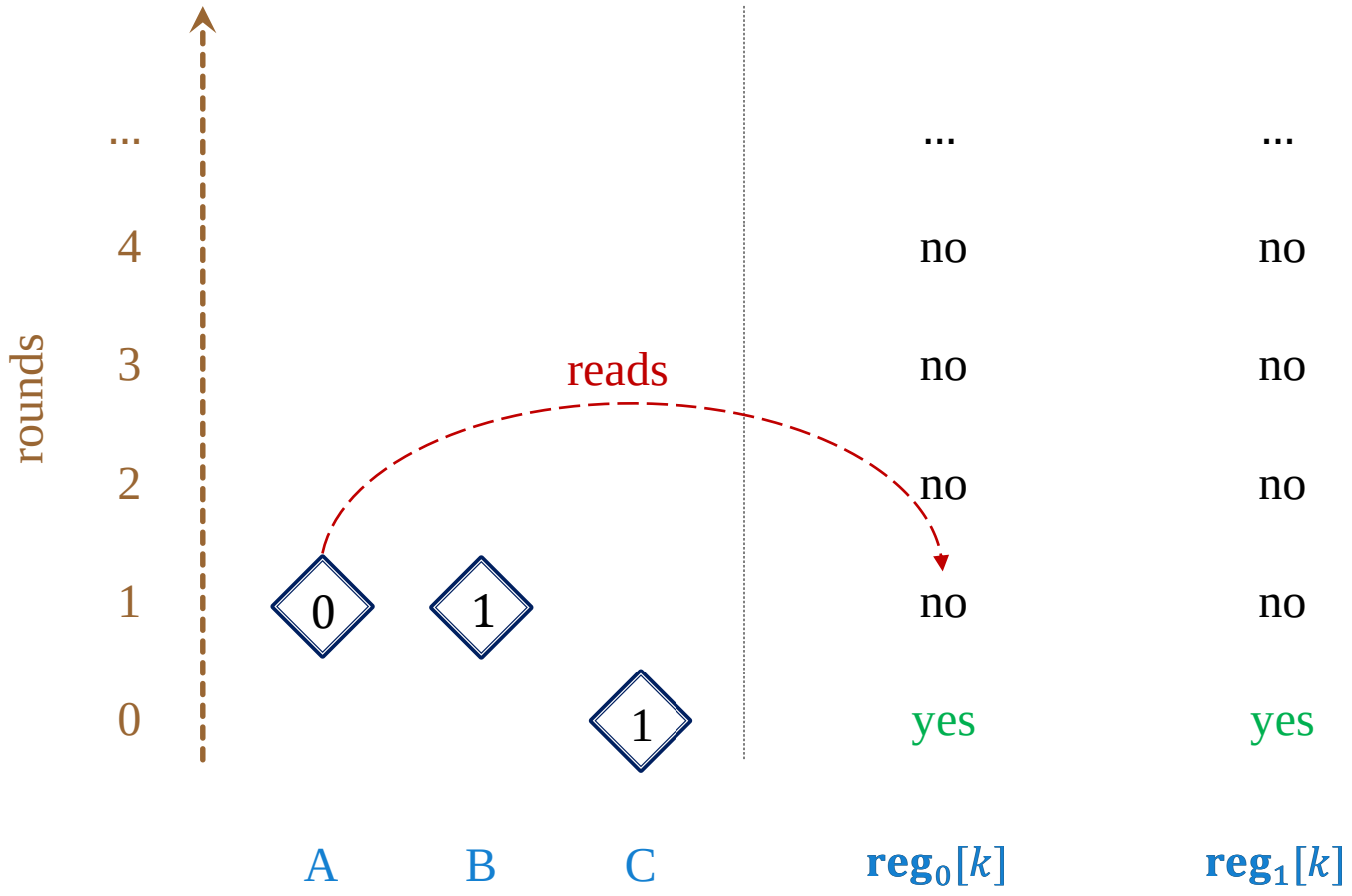
Example of execution of the algorithm



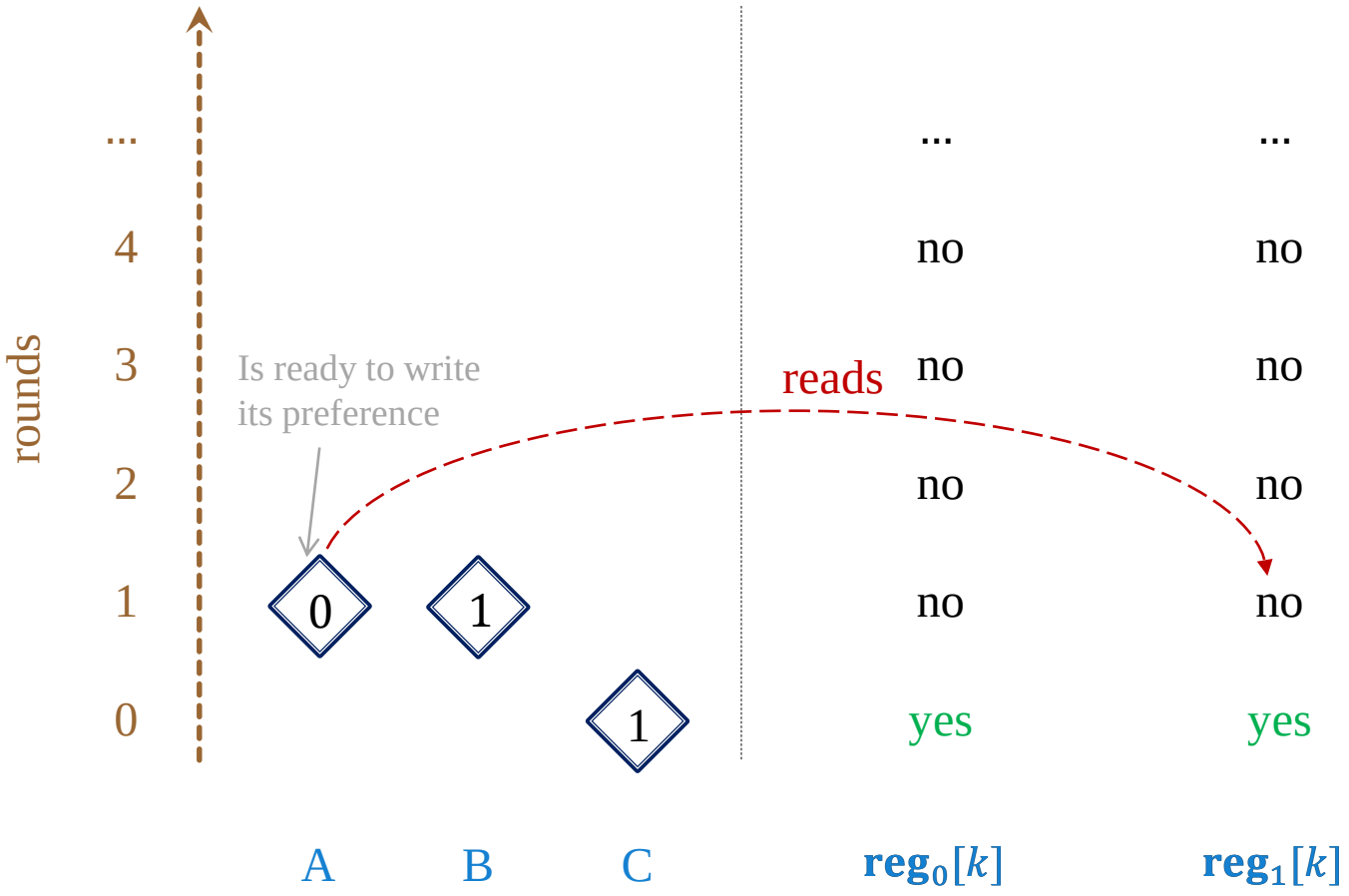
Example of execution of the algorithm



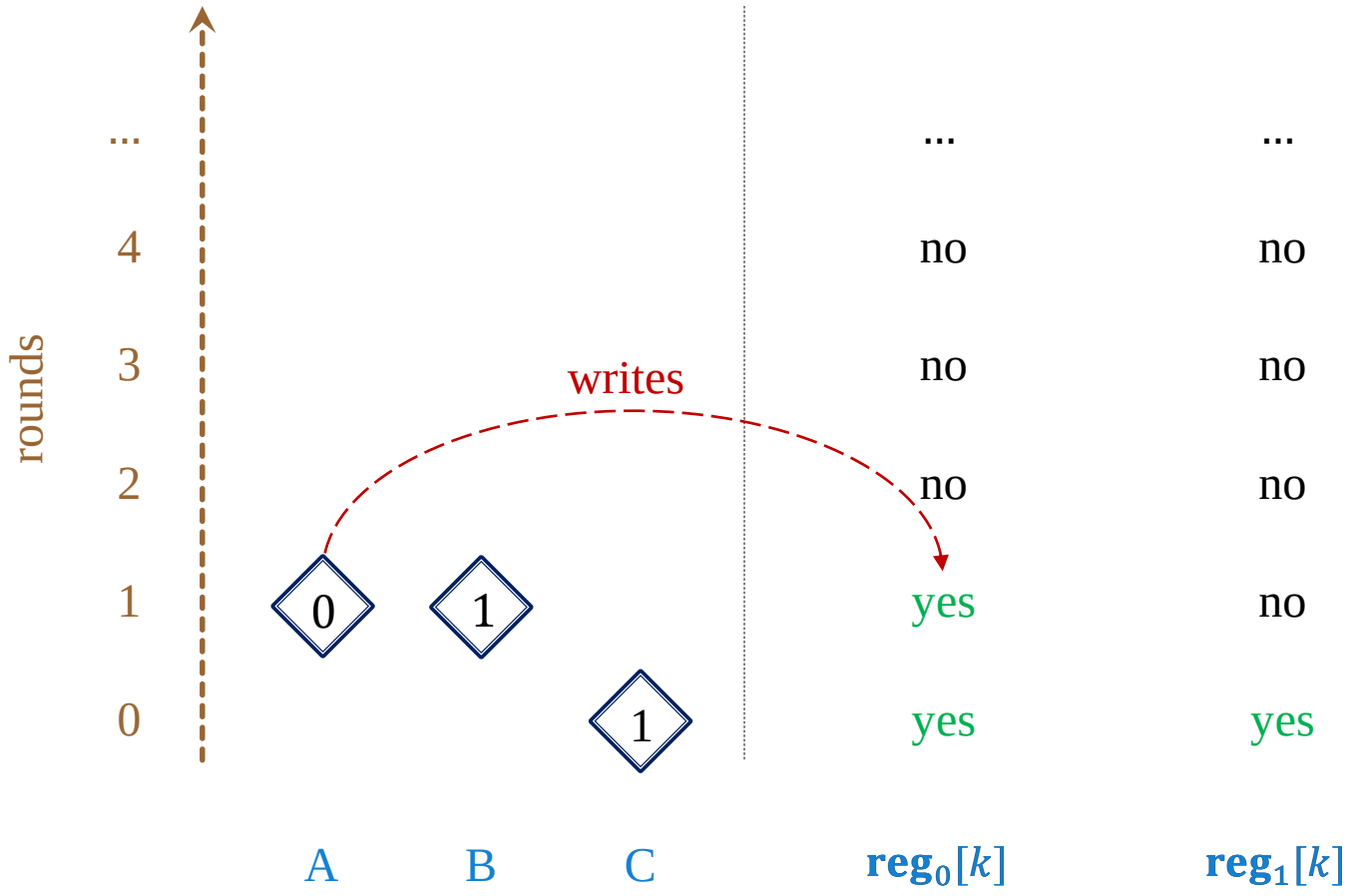
Example of execution of the algorithm



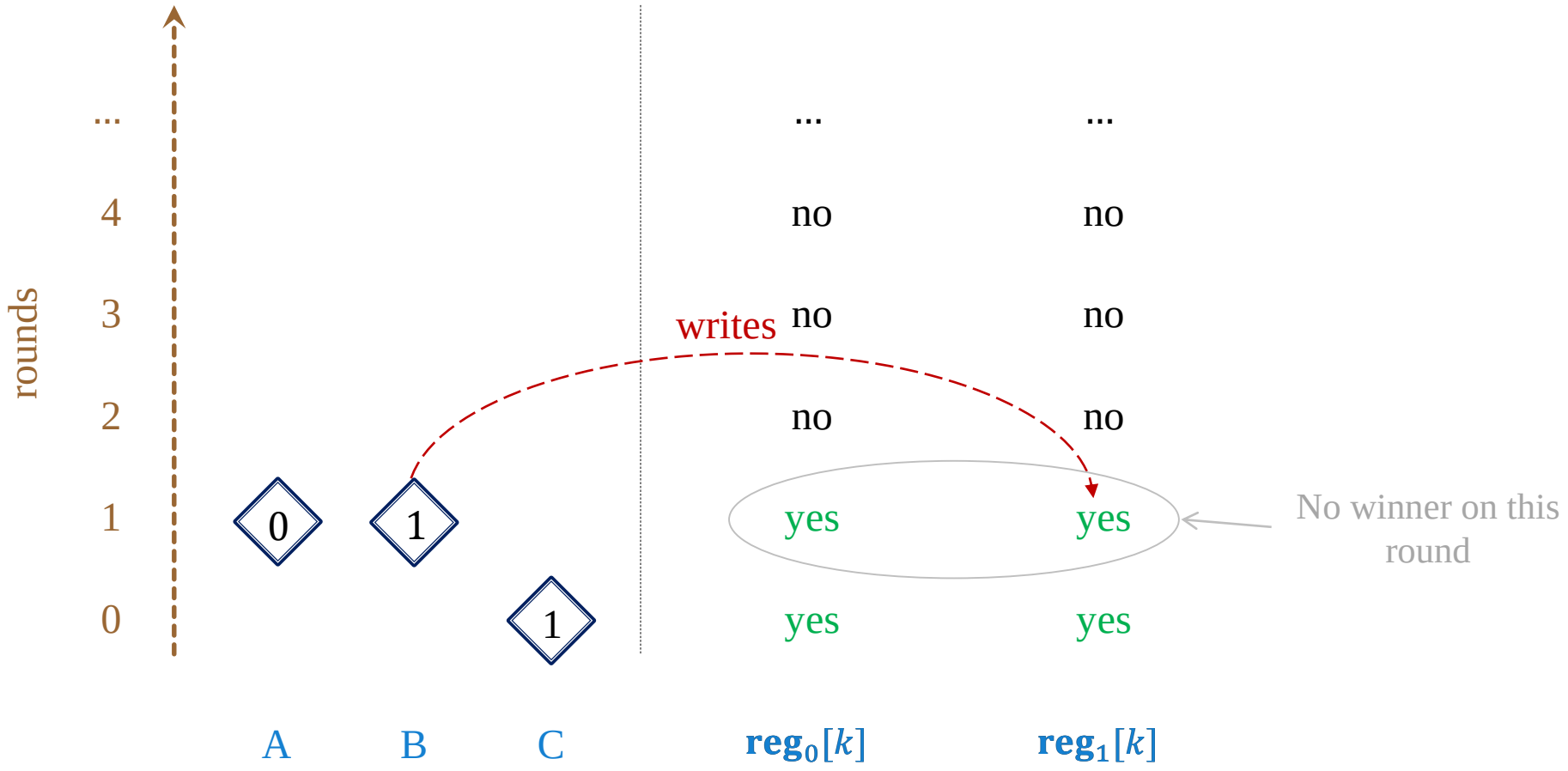
Example of execution of the algorithm



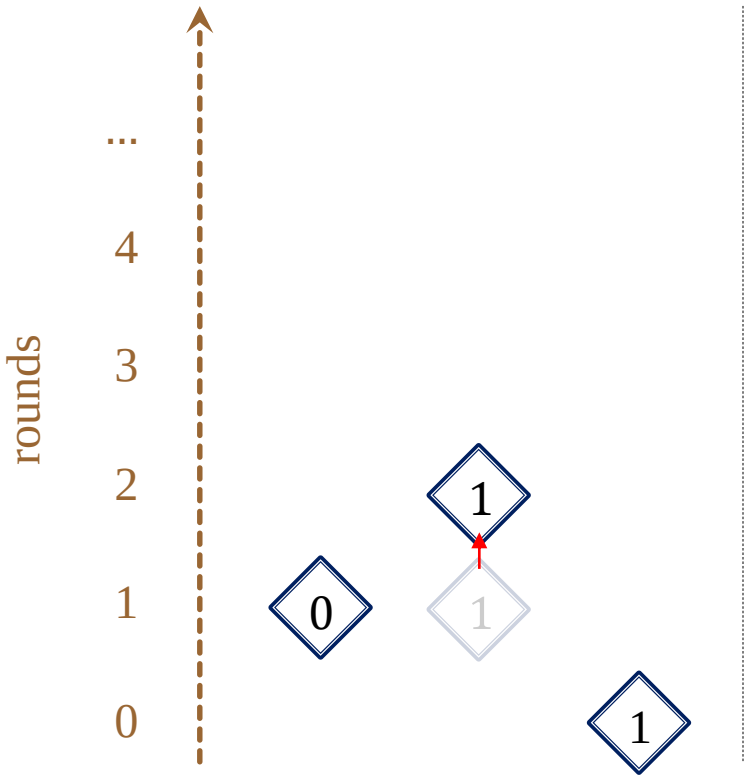
Example of execution of the algorithm



Example of execution of the algorithm

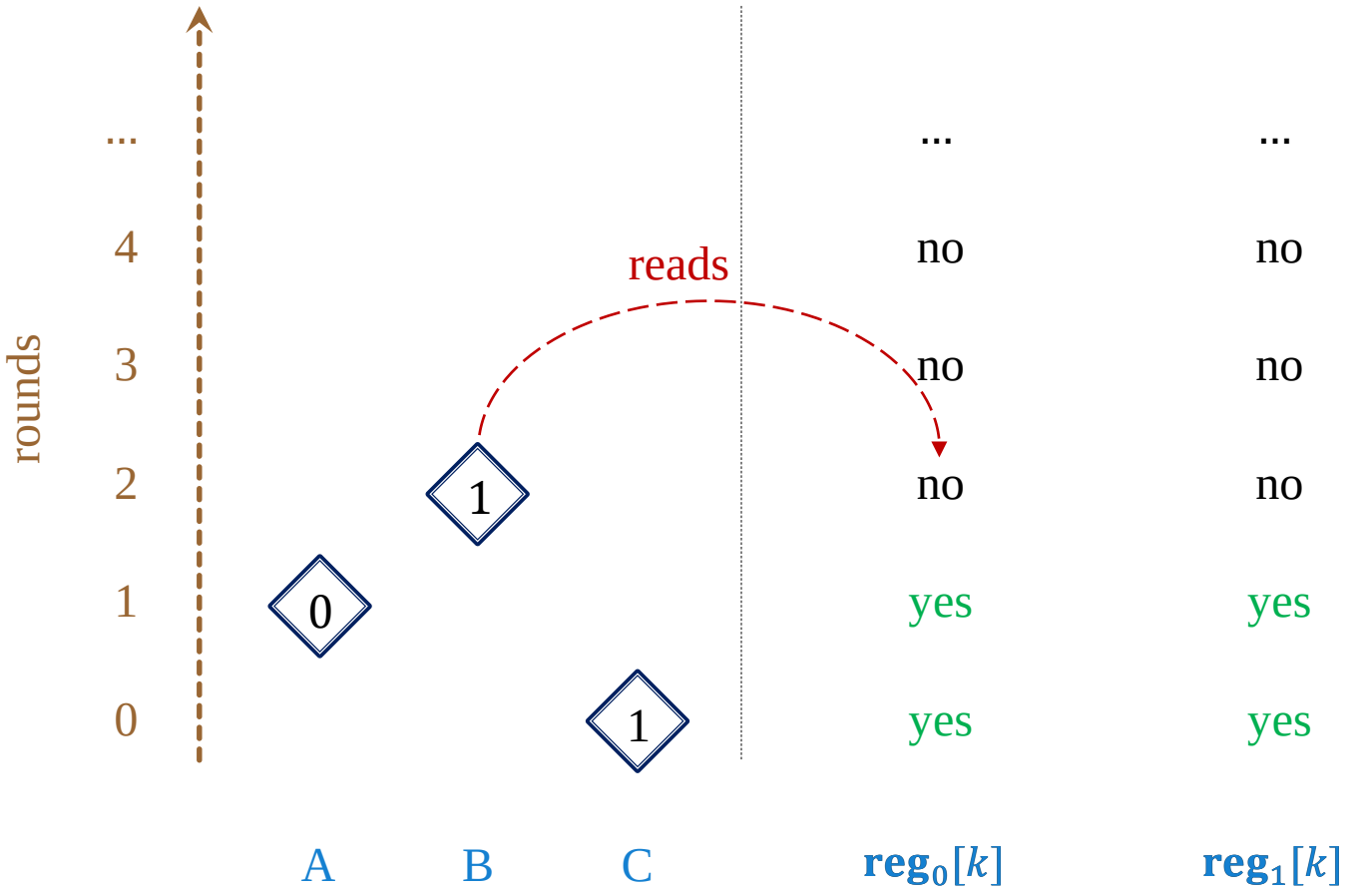


Example of execution of the algorithm

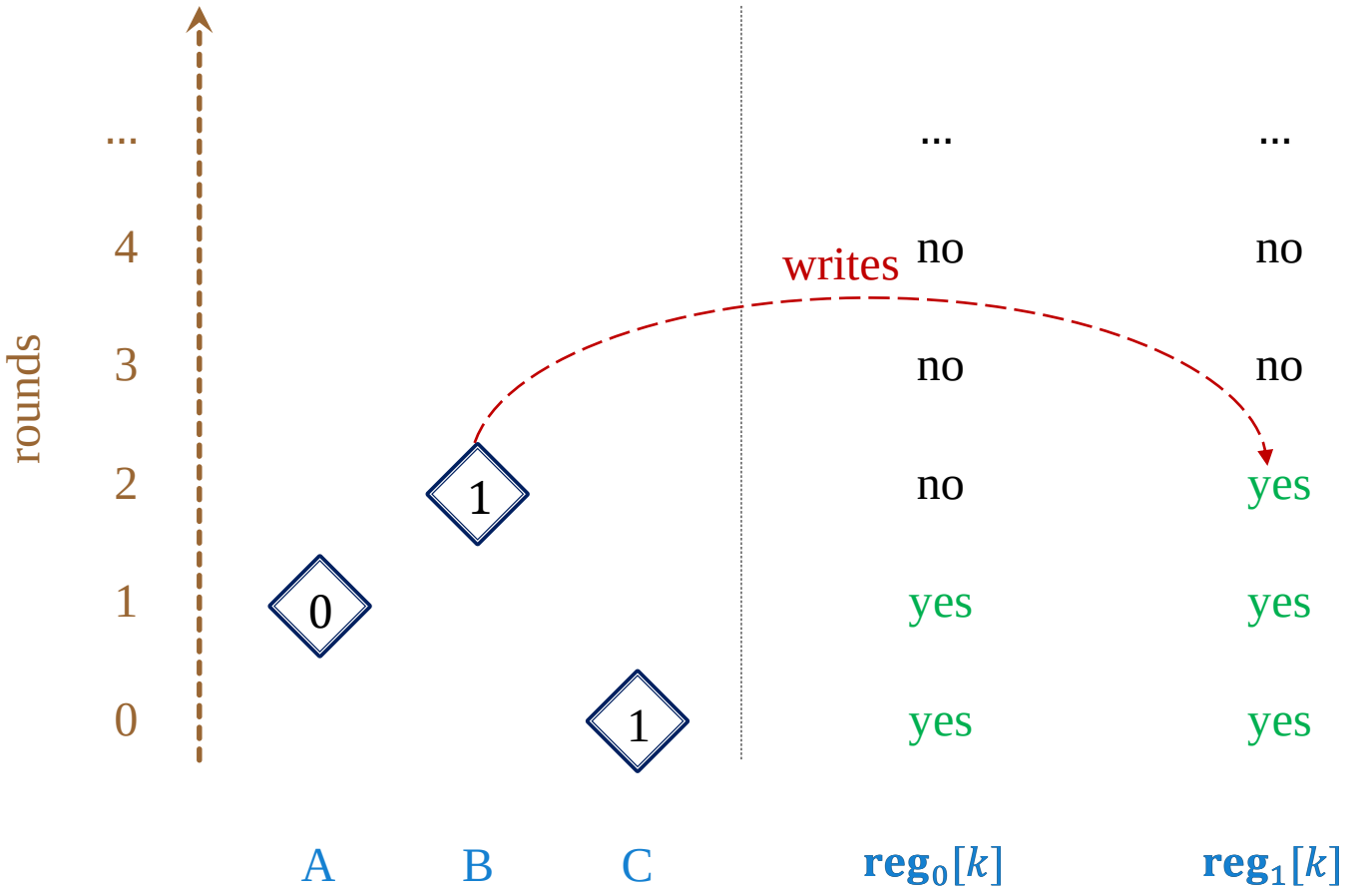


	$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$
...	...	...
4	no	no
3	no	no
2	no	no
1	yes	yes
0	yes	yes

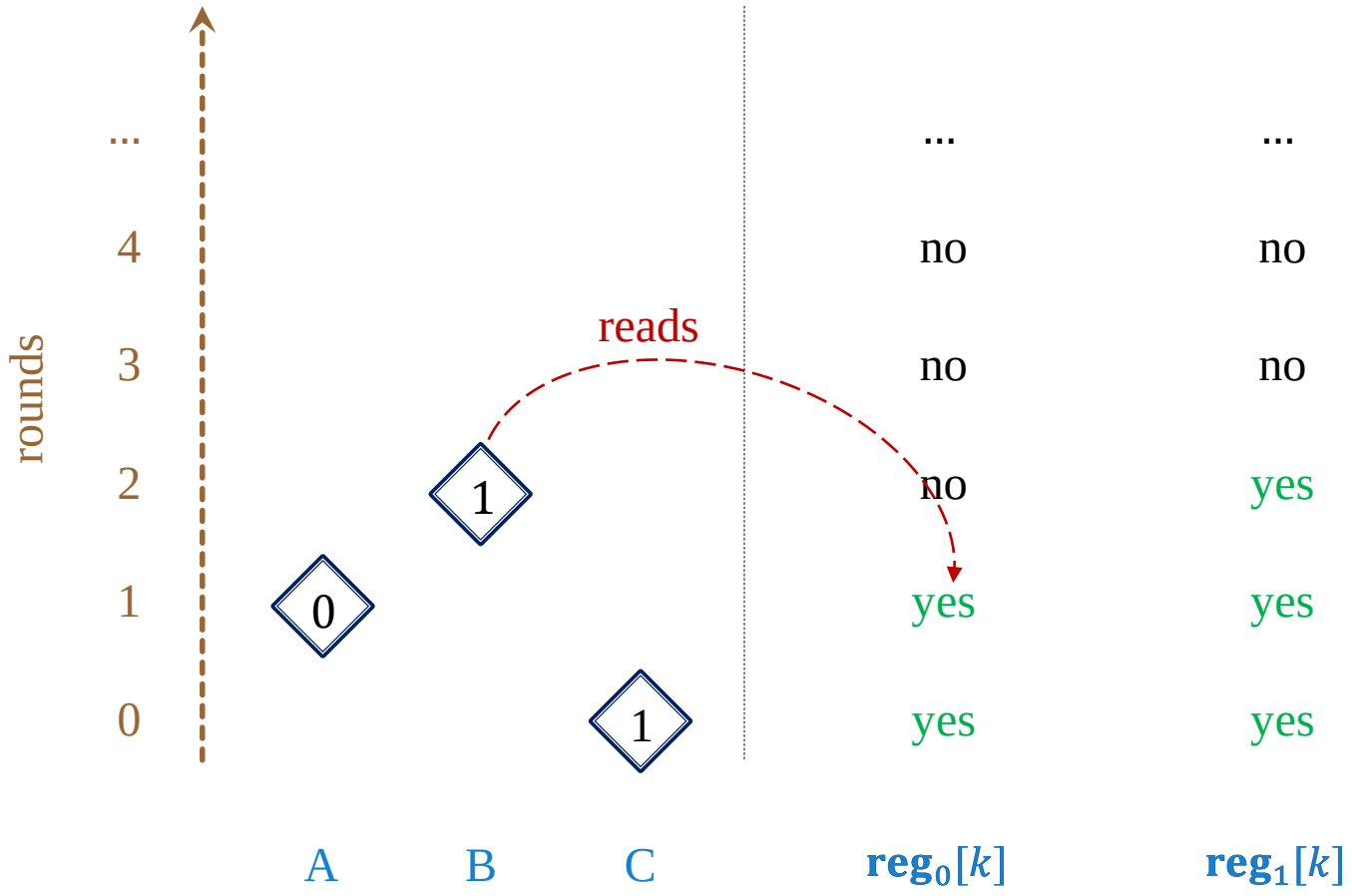
Example of execution of the algorithm



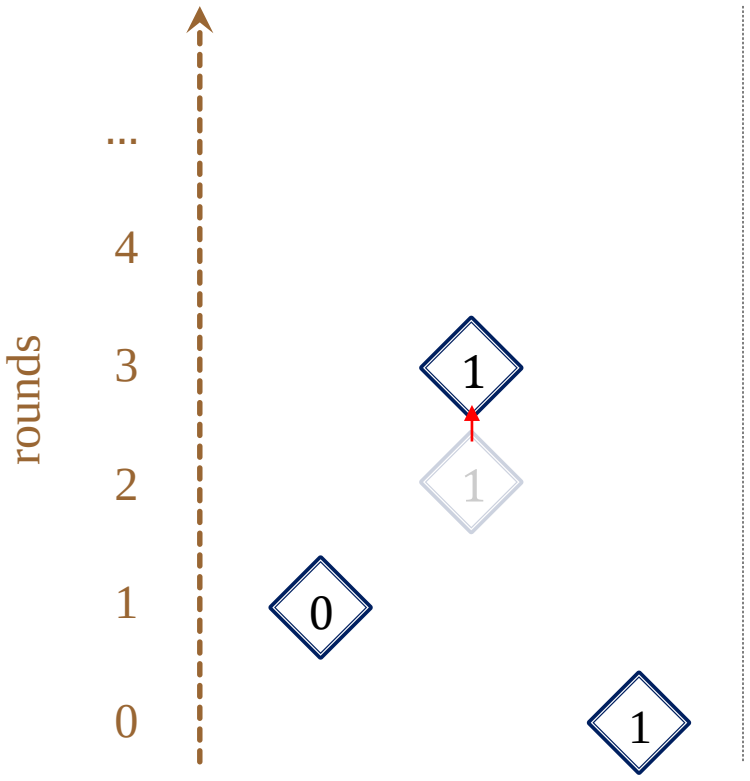
Example of execution of the algorithm



Example of execution of the algorithm

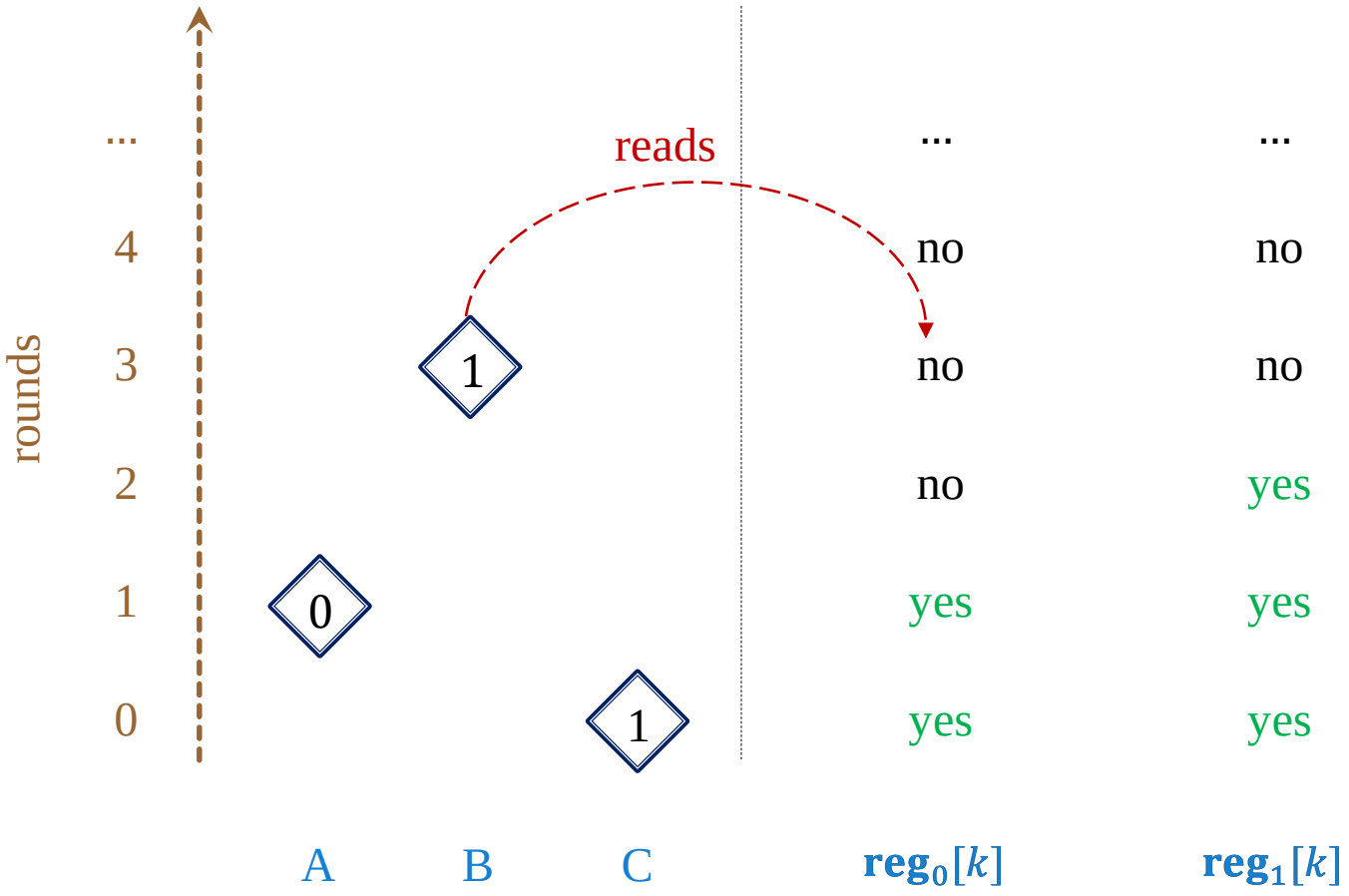


Example of execution of the algorithm

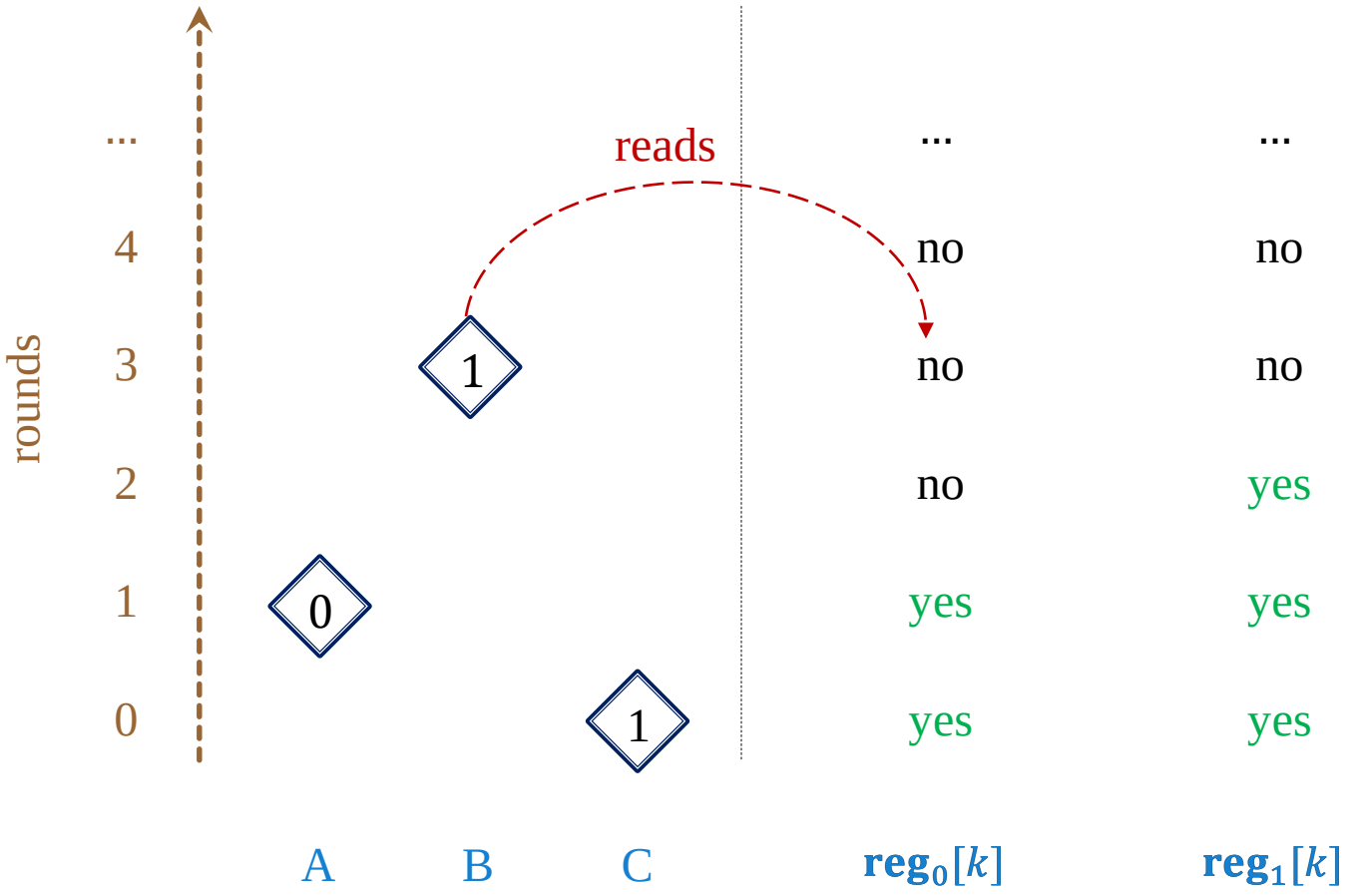


...	...	...
4	no	no
3	no	no
2	no	yes
1	yes	yes
0	yes	yes
	$\mathbf{reg}_0[k]$	$\mathbf{reg}_1[k]$

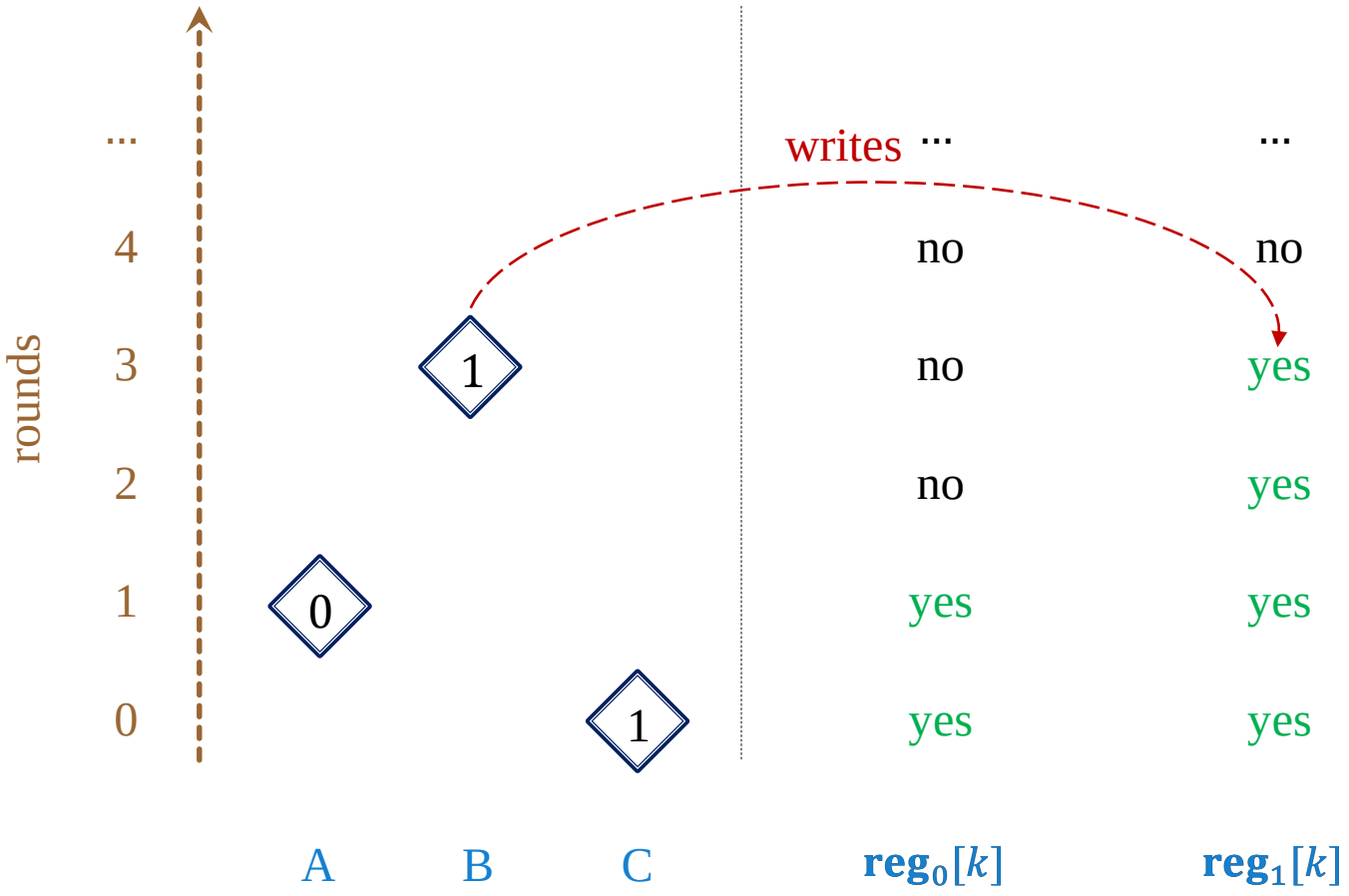
Example of execution of the algorithm



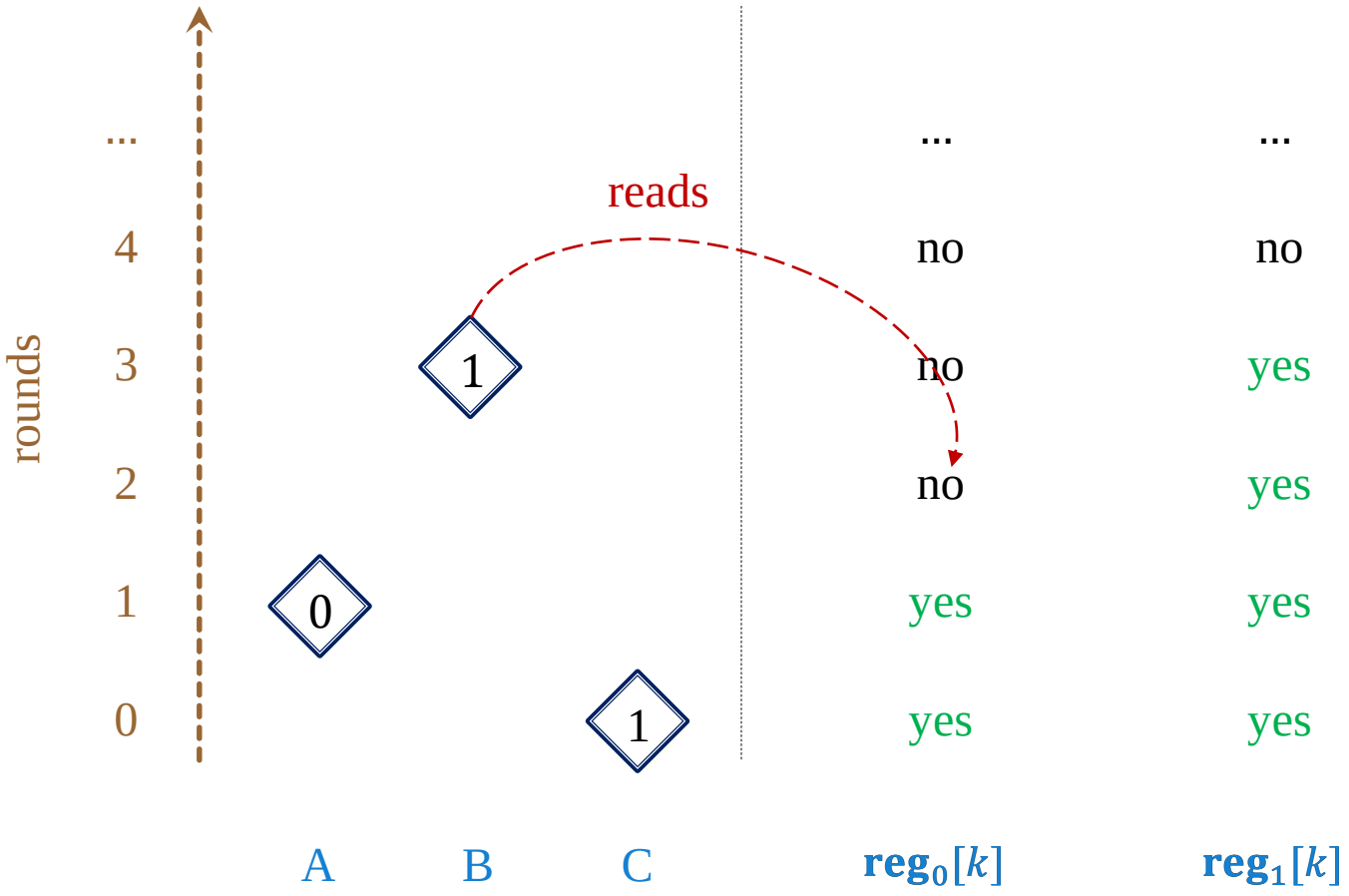
Example of execution of the algorithm



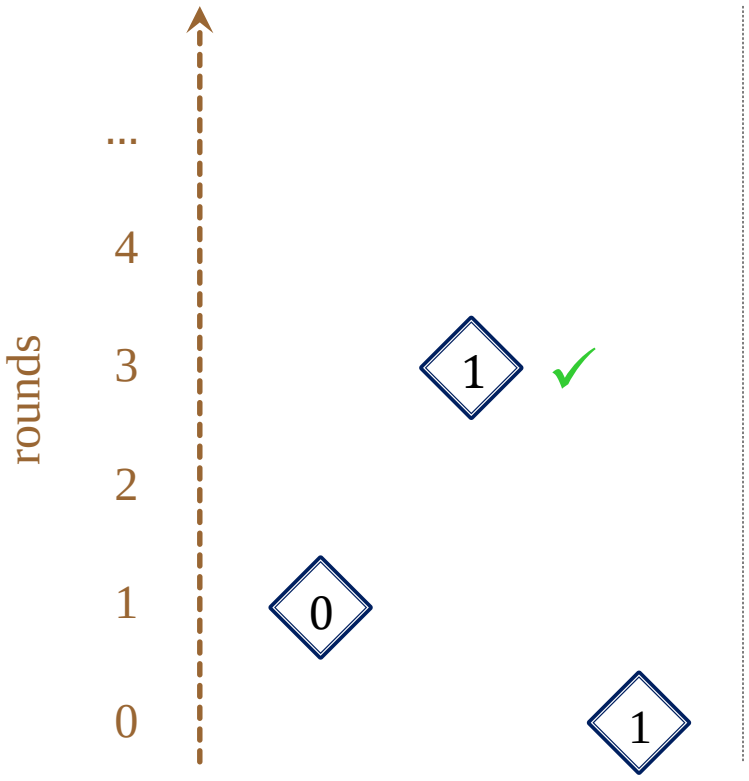
Example of execution of the algorithm



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