Towards a Regular Theory of Parameterized Concurrent Systems

Benedikt Bollig

Laboratoire Spécification et Vérification
ENS Cachan & CNRS, France

Reports on joint works with Paul Gastin, Akshay Kumar, and Jana Schubert.

ACTS 2015
Chennai Mathematical Institute

The verification problem for parmeterized systems:

«Is a system correct independently of
the number of processes / the way they are arranged?»

Talks by Arnaud Sangnier and Pierre Ganty.

The verification problem for parmeterized systems:

«Is a system correct independently of
the number of processes / the way they are arranged?»

→ Talks by Arnaud Sangnier and Pierre Ganty.

In this talk, we study language-theoretic questions / expressiveness:

- Complementation
- Equivalent characterization in terms of MSO logic
- Nonemptiness

We are looking for «robust» models of parameterized systems.

The verification problem for parmeterized systems:

«Is a system correct independently of
the number of processes / the way they are arranged?»

→ Talks by Arnaud Sangnier and Pierre Ganty.

In this talk, we study language-theoretic questions / expressiveness:

- Complementation
- Equivalent characterization in terms of MSO logic
- Nonemptiness

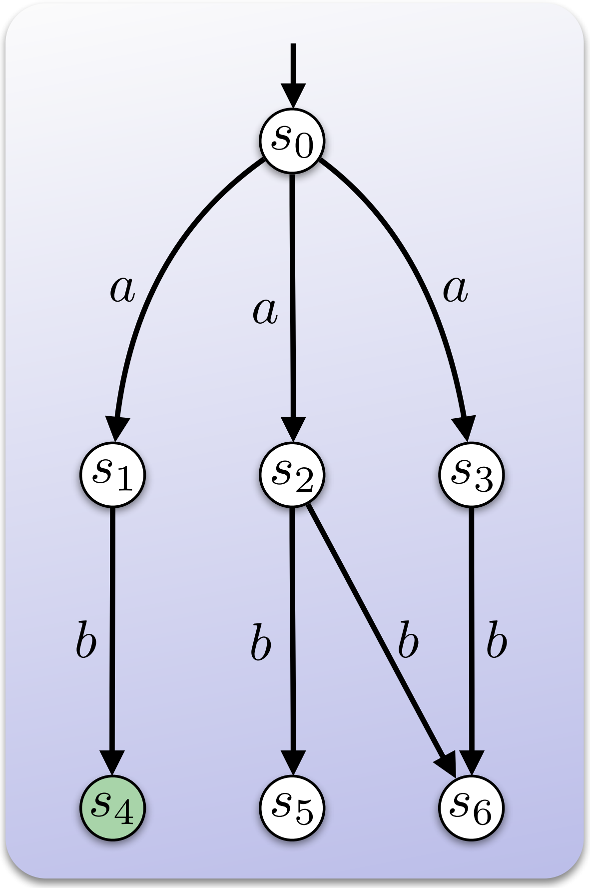
We are looking for «robust» models of parameterized systems.

There have been robust models for fixed process architectures:

- Thomas: *On logical definability of trace languages.* ASMICS 1990.
- Henriksen-Mukund-Narayan Kumar-Sohoni-Thiagarajan: *A Theory of Regular MSC Languages.* I&C 2005.
- Genest-Kuske-Muscholl: *A Kleene theorem and model checking algorithms for existentially bounded communicating automata.* I&C 2006.

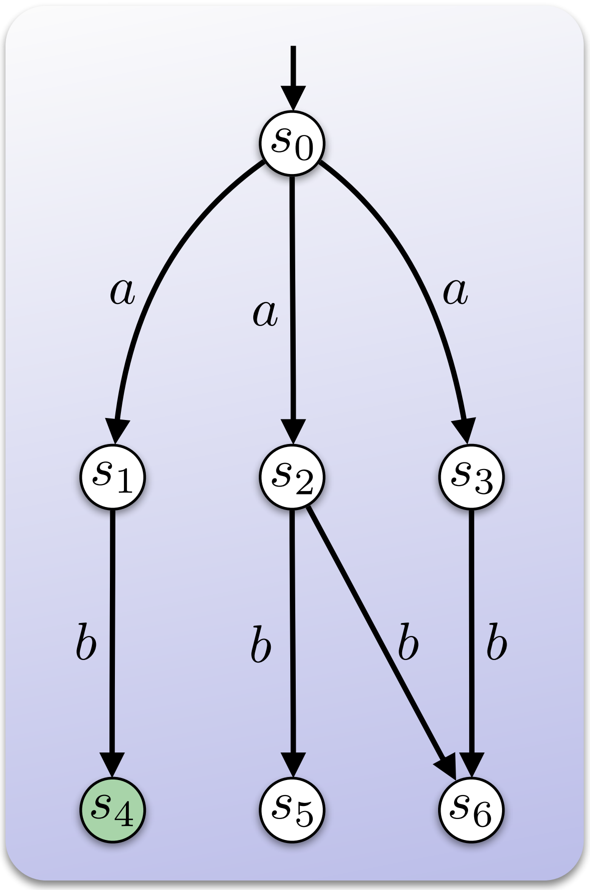
Finite Automata

finite automaton

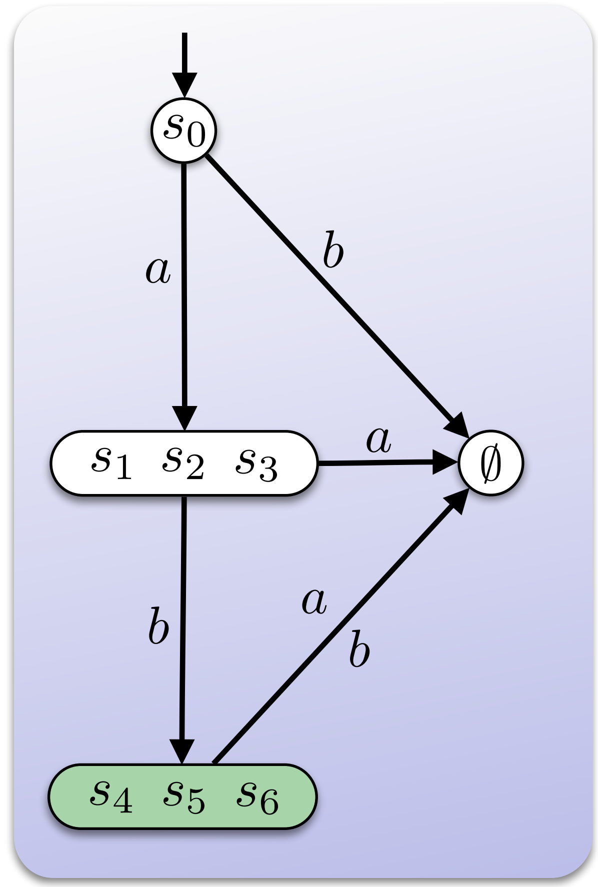


Finite Automata

finite automaton

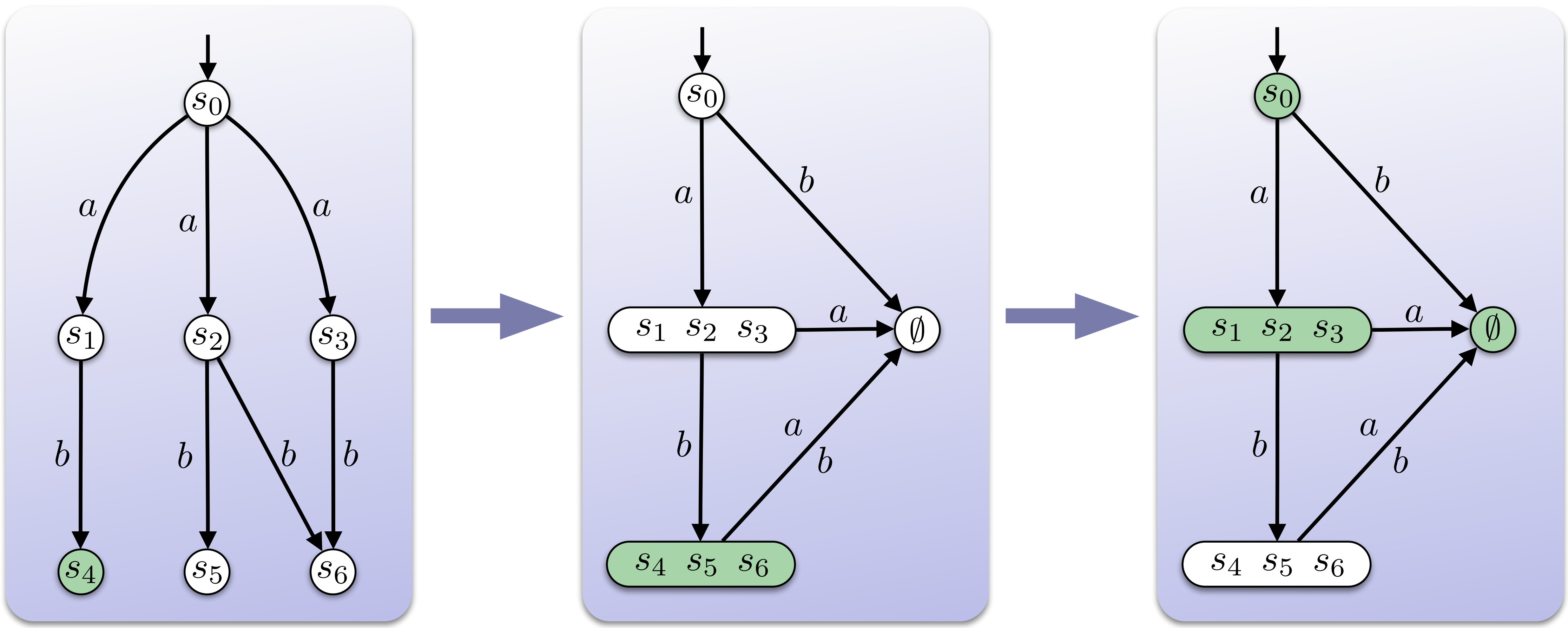


determinization



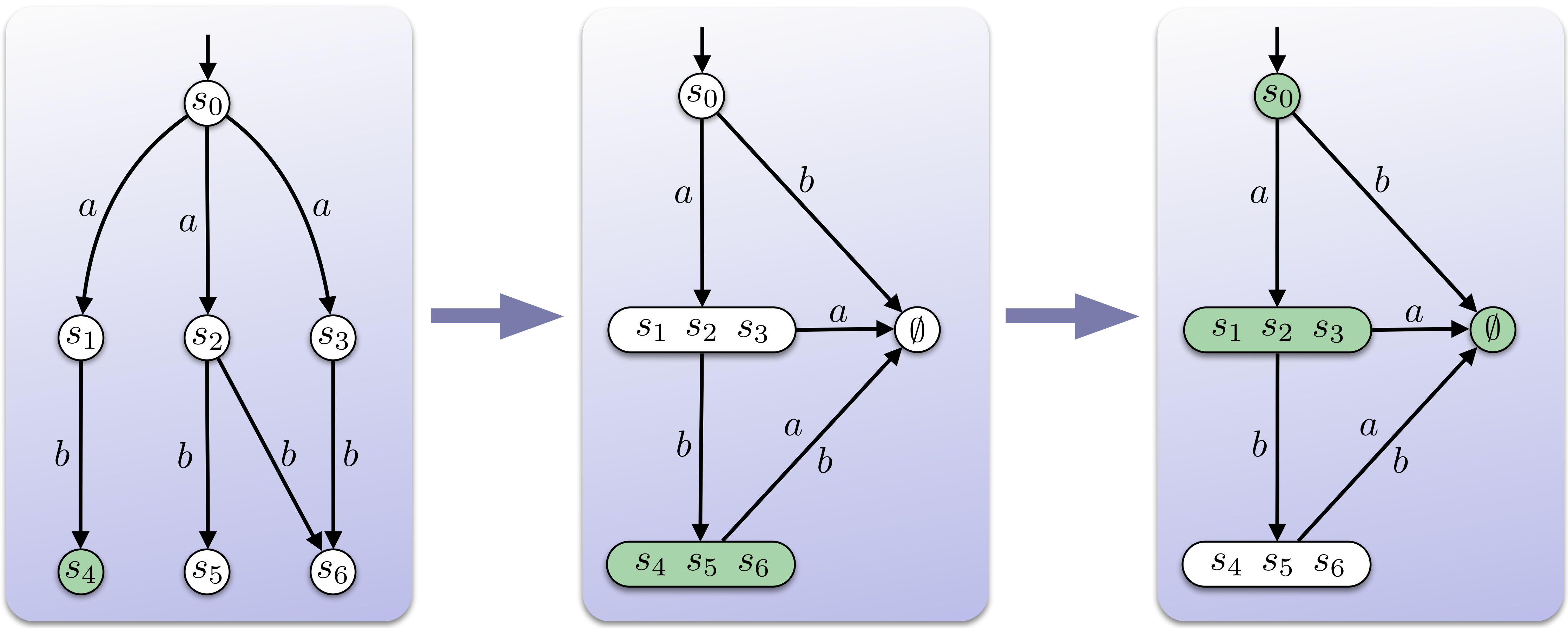
Finite Automata

finite automaton | determinization | complementation



Finite Automata

finite automaton — determinization — complementation

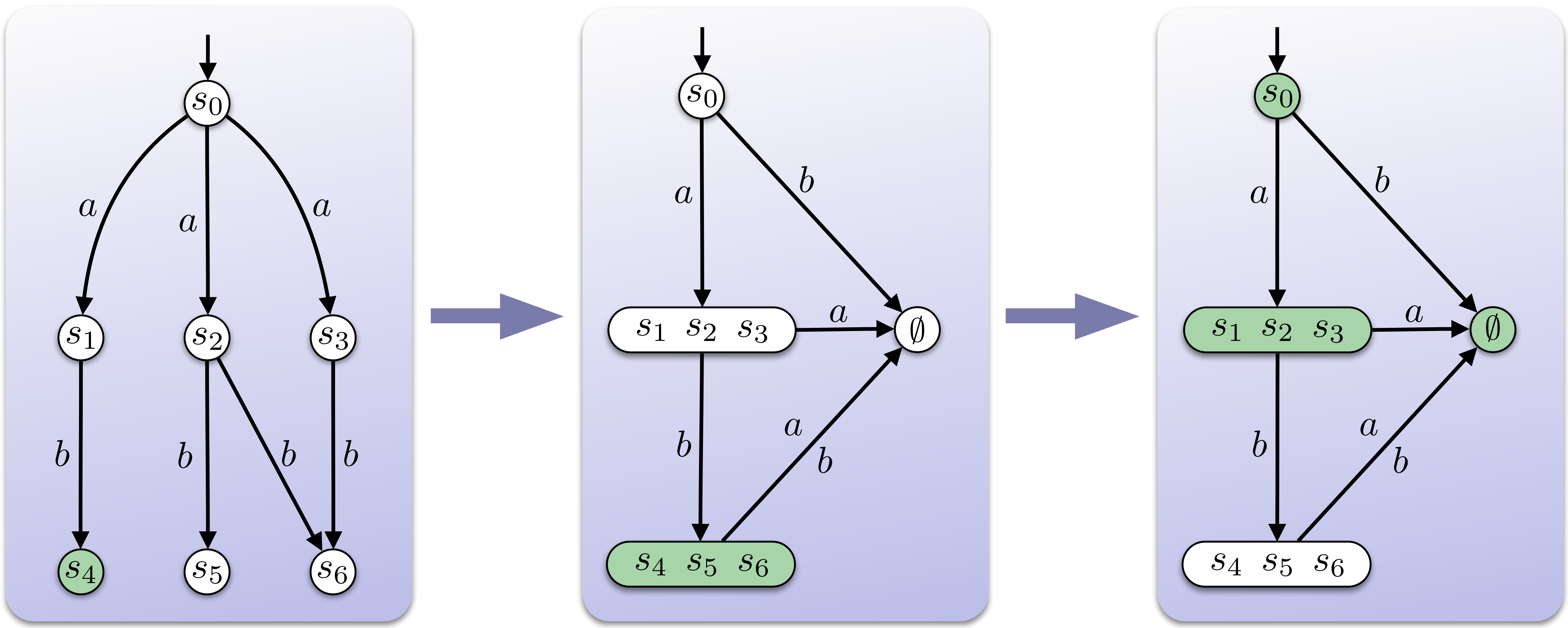


Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

$$\forall x(a(x) \rightarrow \exists y(\mathsf{succ}(x,y) \wedge b(y)))$$

Finite Automata

finite automaton | determinization | complementation



Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

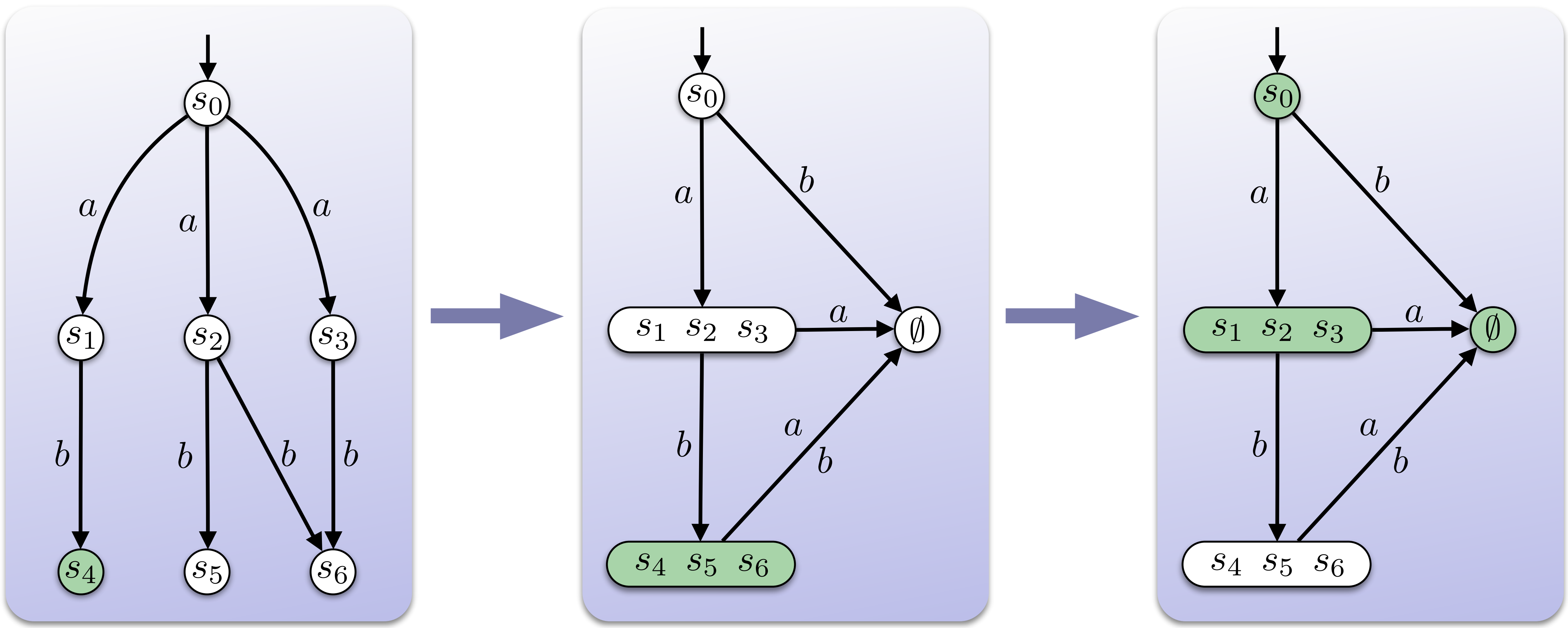
$$\forall x(a(x) \rightarrow \exists y(\mathsf{succ}(x, y) \wedge b(y)))$$

Proof:

- free variables $\Rightarrow$ extended alphabet

Finite Automata

finite automaton　　determinization　　complementation



Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

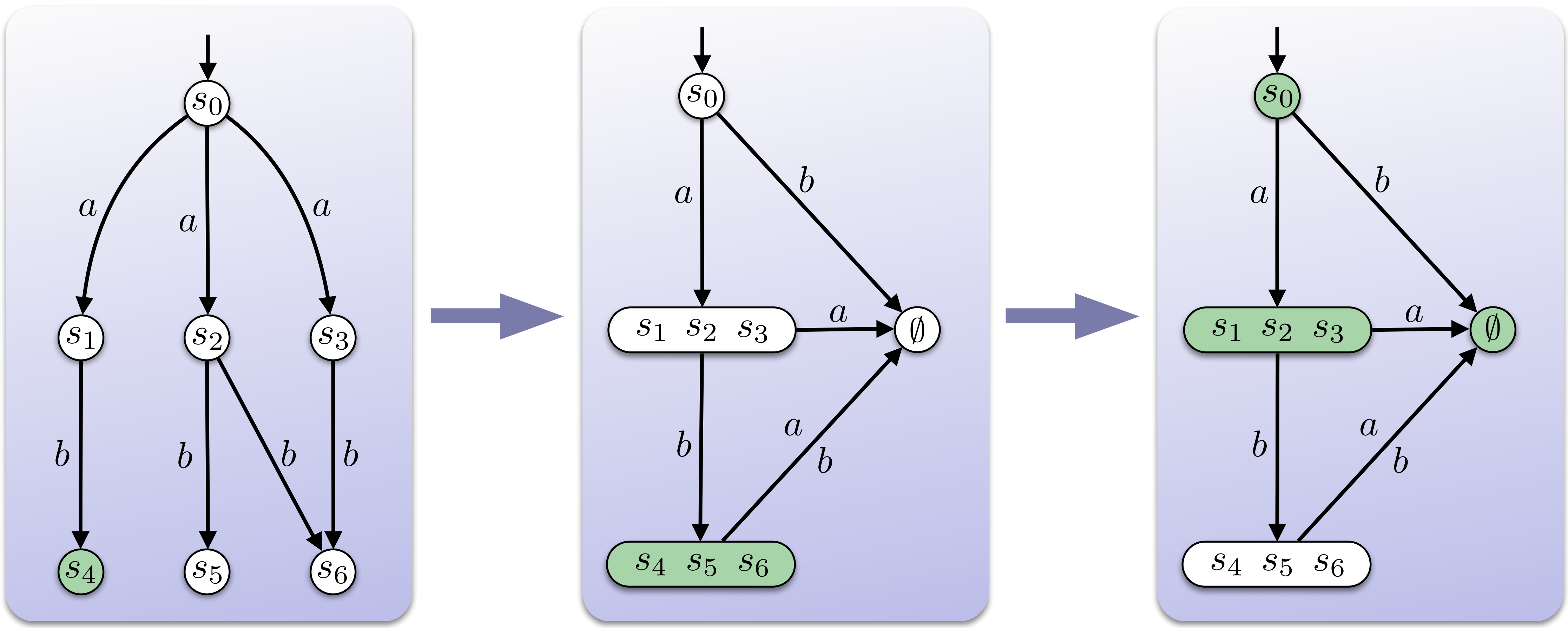
$$\forall x(a(x) \to \exists y(\mathsf{succ}(x,y) \wedge b(y)))$$

Proof:

- free variables → extended alphabet
- existential quantification → projection

Finite Automata

finite automaton | determinization | complementation



Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

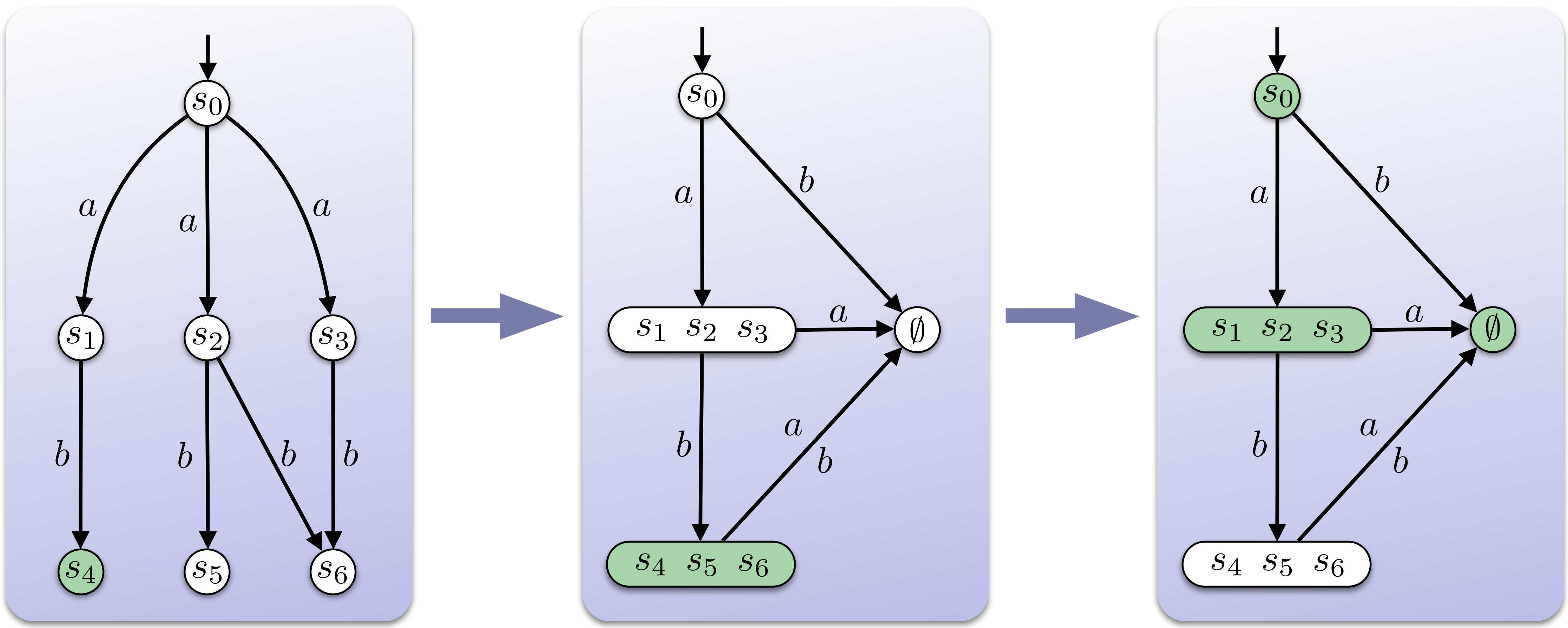
$$\forall x(a(x) \rightarrow \exists y(\mathsf{succ}(x,y) \wedge b(y)))$$

Proof:

- free variables → extended alphabet
- existential quantification → projection
- negation → complementation

Fi Outline ata

finite automaton determinization complementation



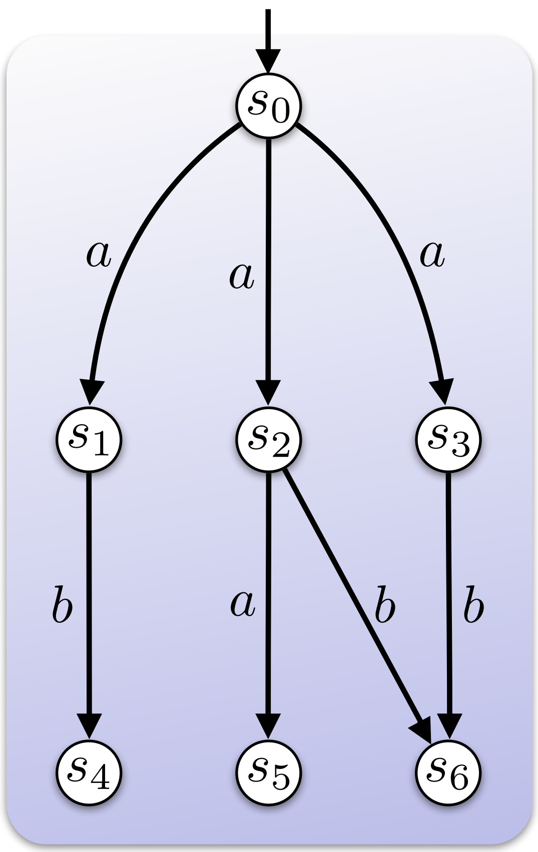
Theorem [Büchi-Elgot-Trakhtenbrot 1960s]:
Finite Automata = MSO

$$\forall x(a(x) \rightarrow \exists y(\mathsf{succ}(x,y) \wedge b(y)))$$

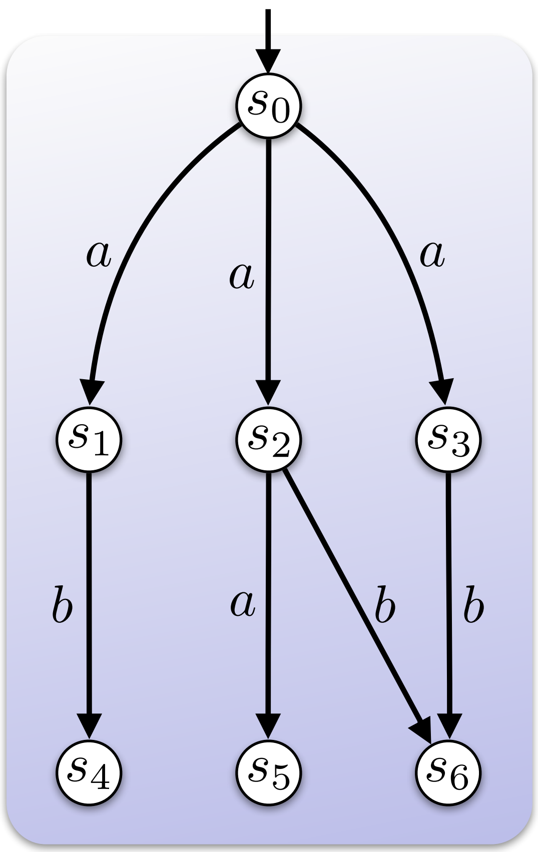
Proof:

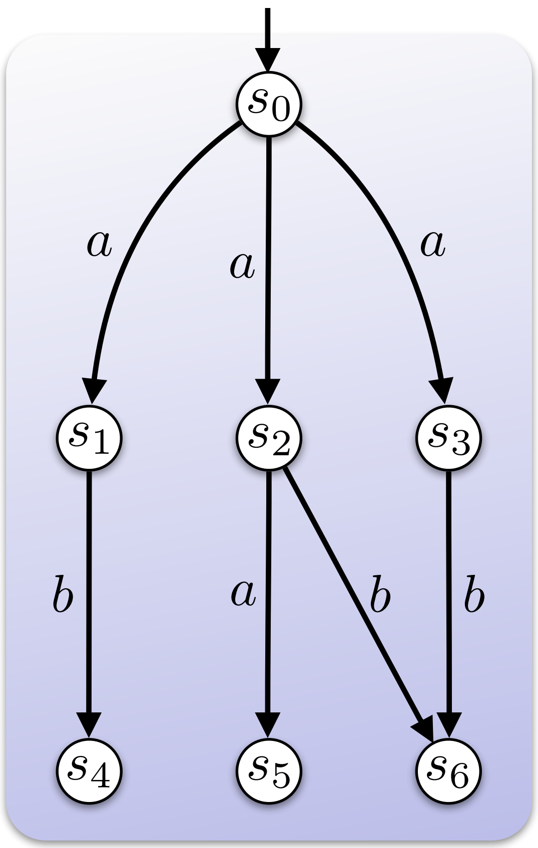
- free variables $\rightarrow$ extended alphabet
- existential quantification $\rightarrow$ projection
- negation $\rightarrow$ complementation

Parameterized Communicating Automata (PCA)

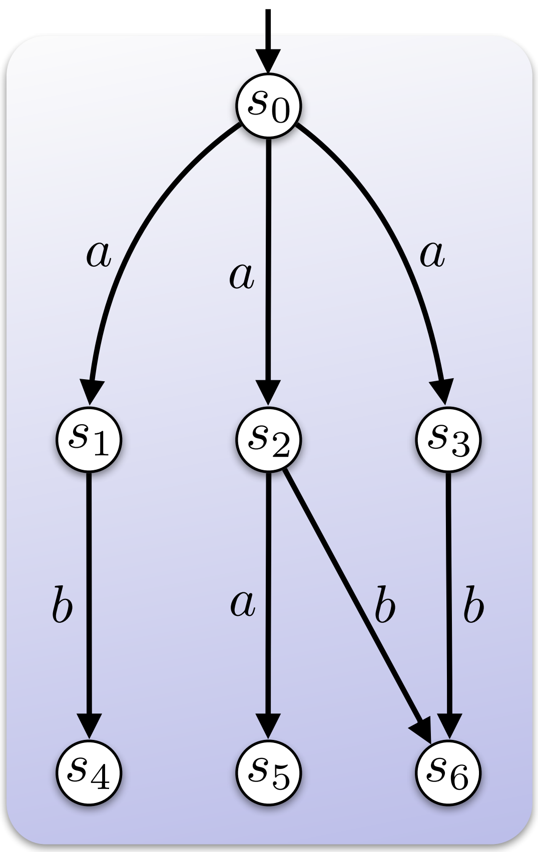


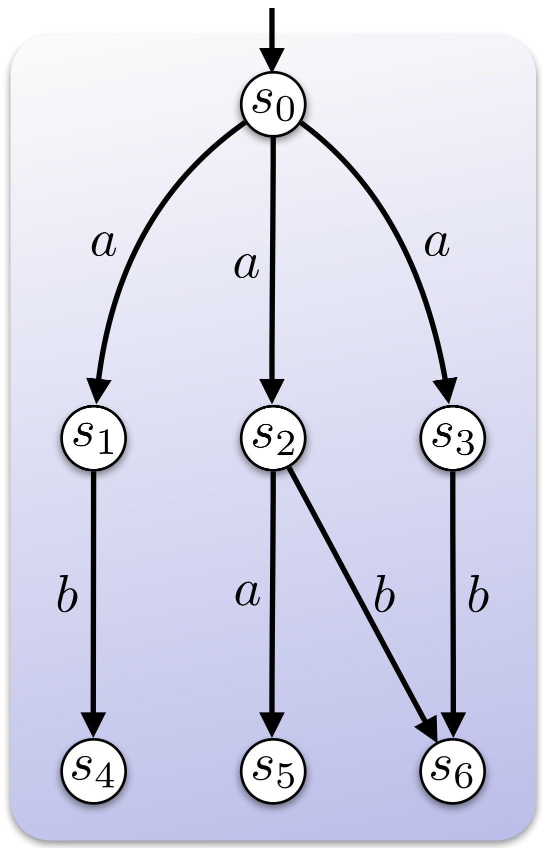
Parameterized Communicating Automata (PCA)

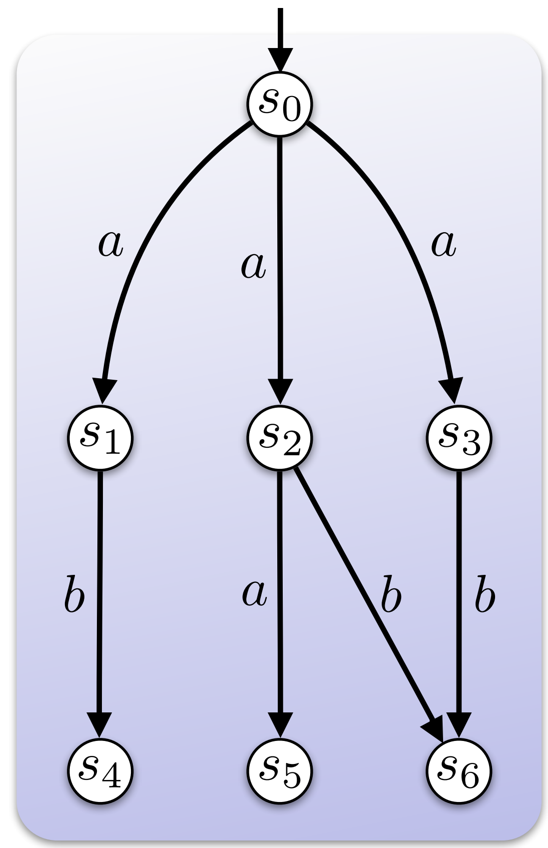




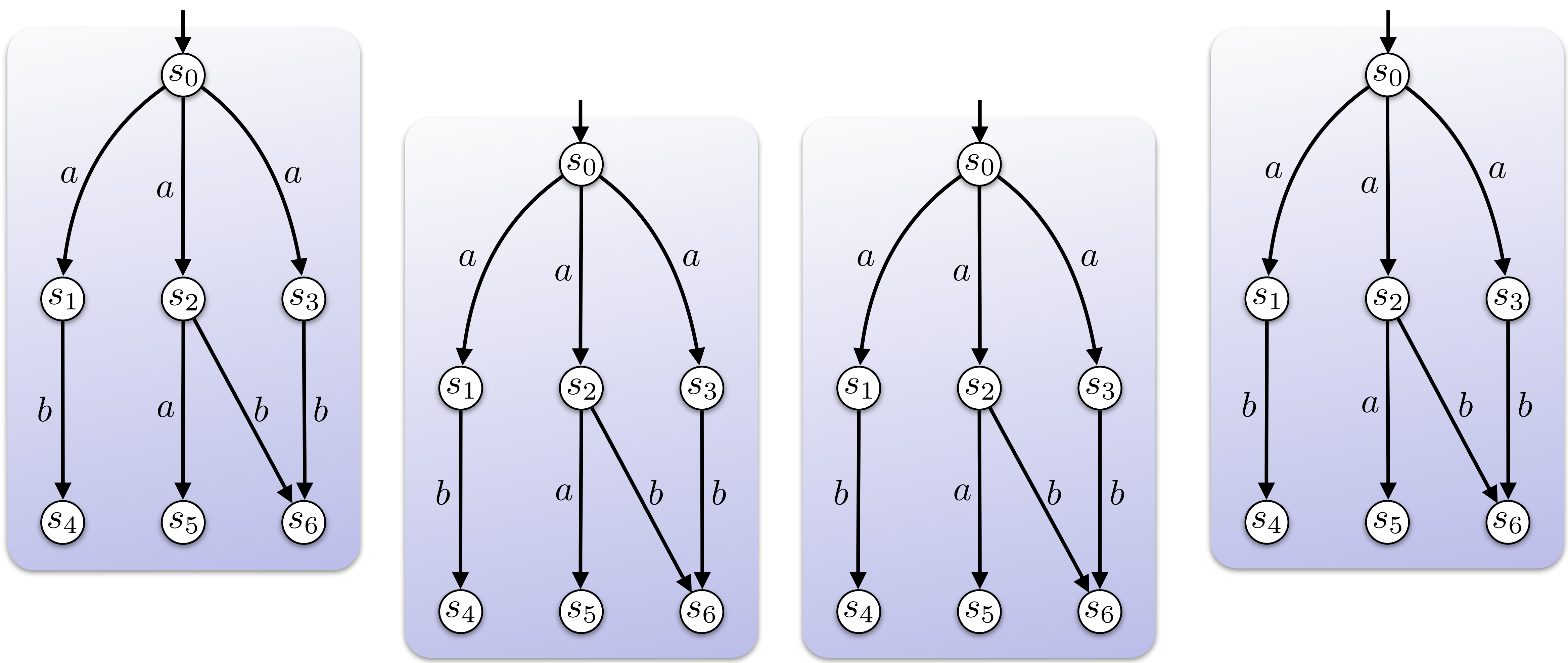
Parameterized Communicating Automata (PCA)



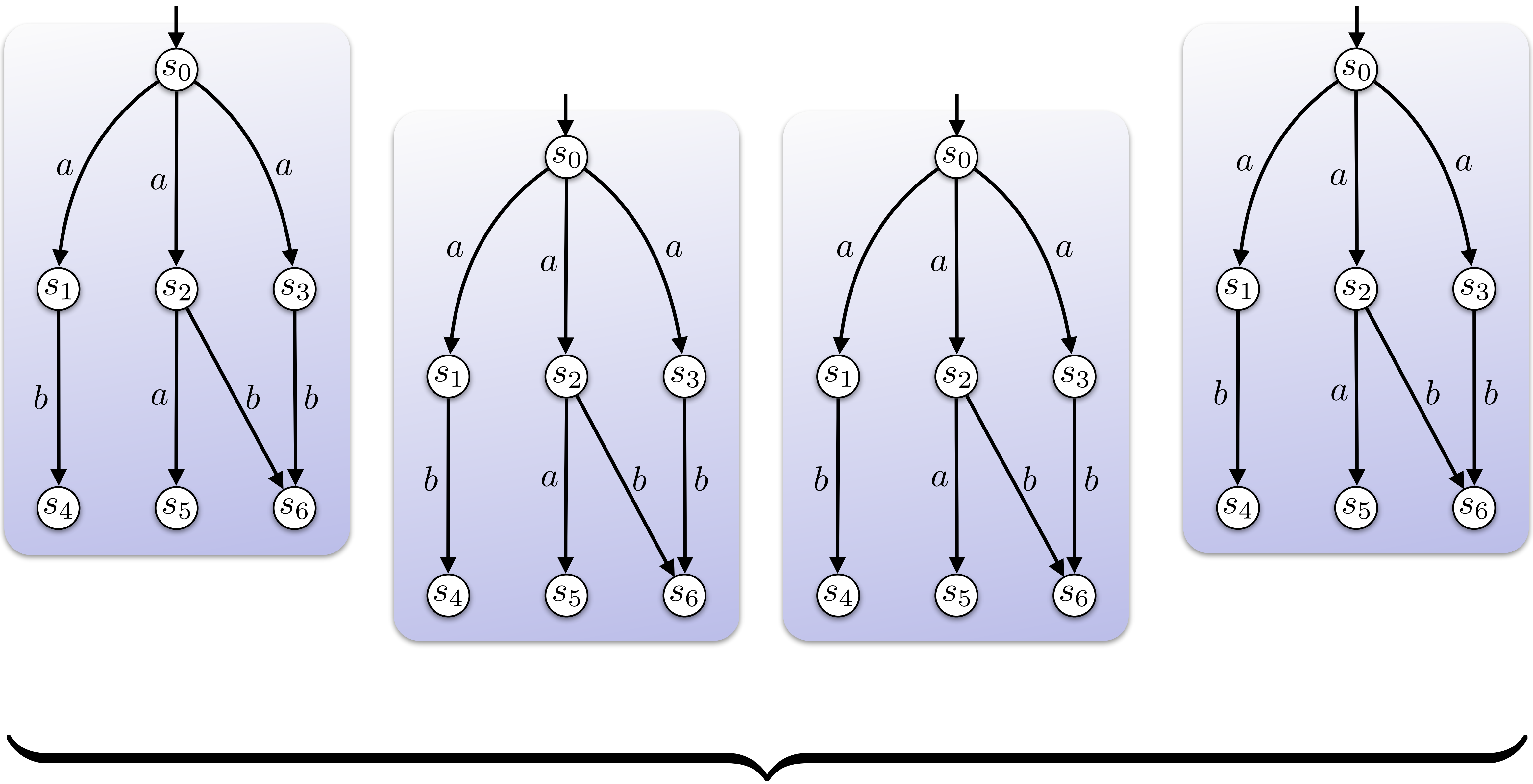




Parameterized Communicating Automata (PCA)

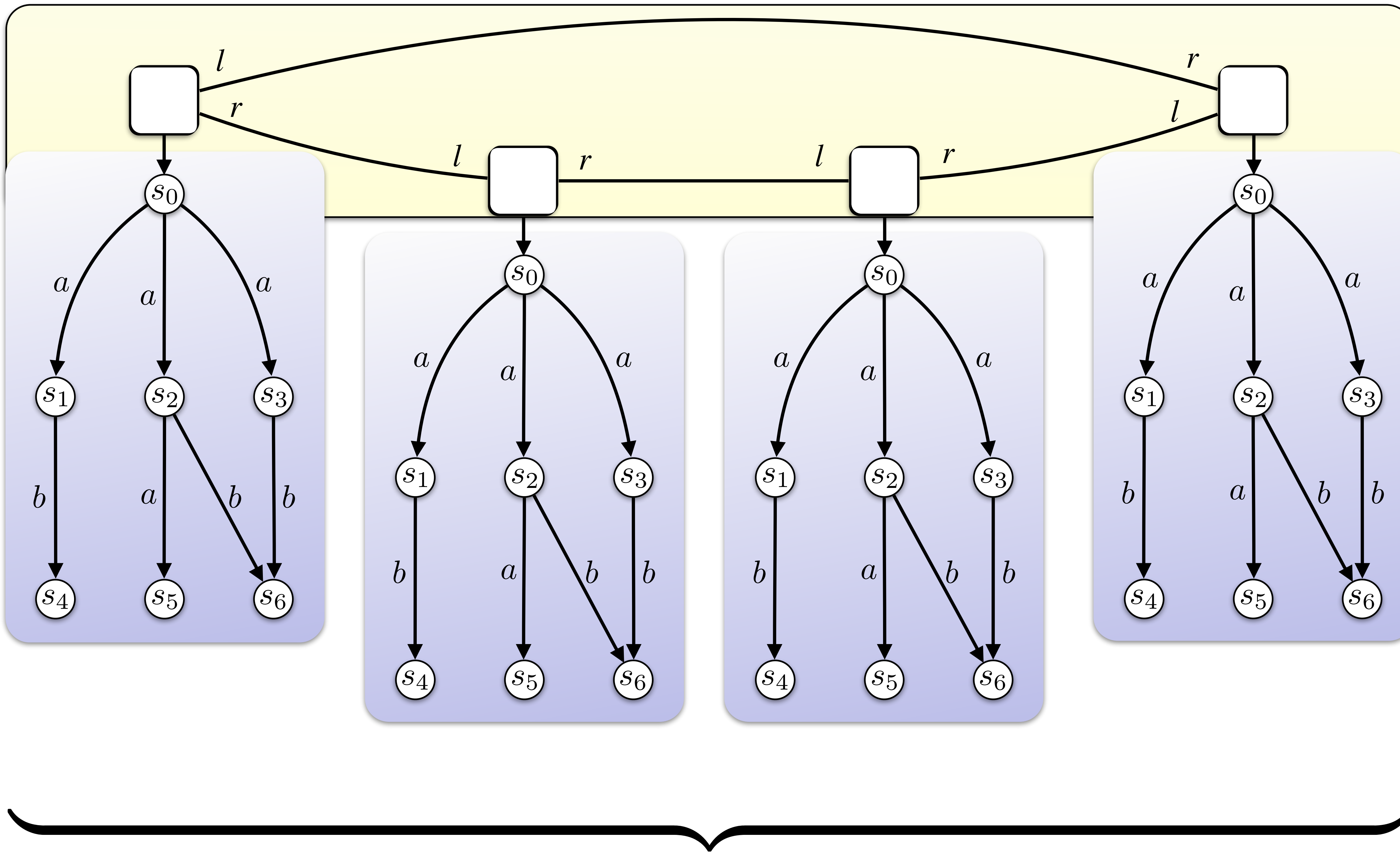


Parameterized Communicating Automata (PCA)



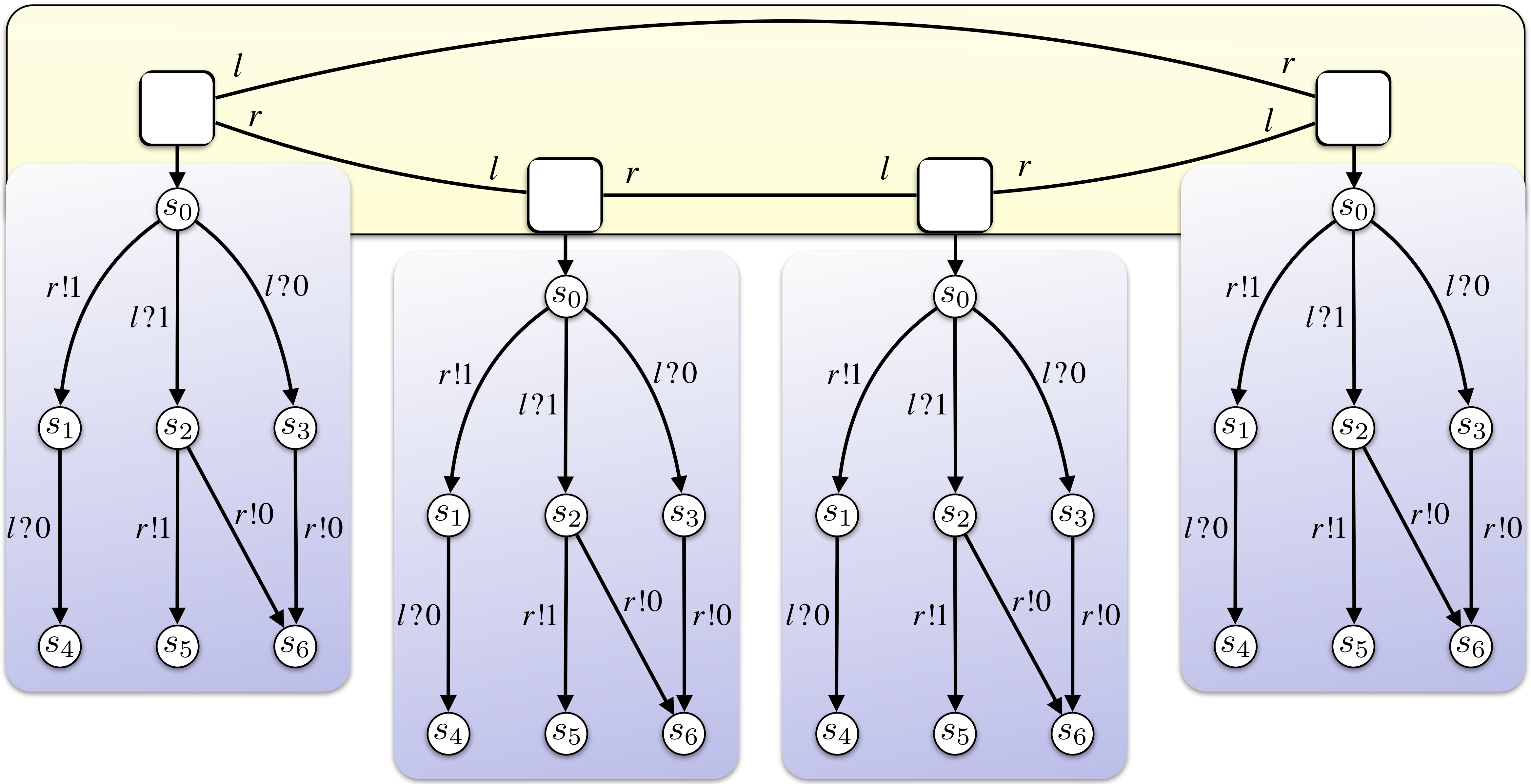
non-fixed & unbounded

Parameterized Communicating Automata (PCA) over Rings



non-fixed & unbounded

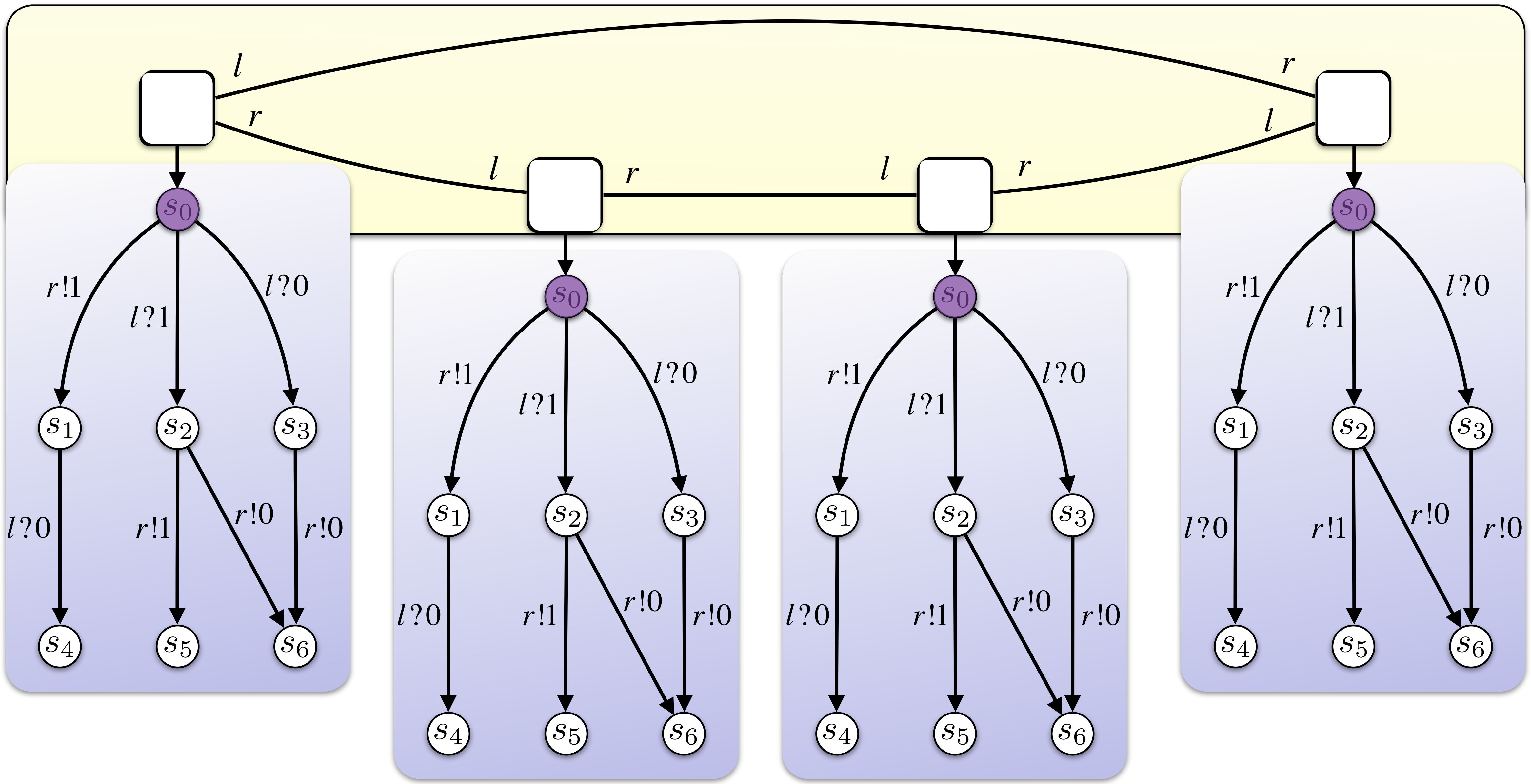
Parameterized Communicating Automata (PCA) over Rings



A PCA is given by:

- finite automaton over $\{l, r\} \times \{!, ?\} \times Msg$ (here: $Msg = \{0, 1\}$)
- acceptance condition

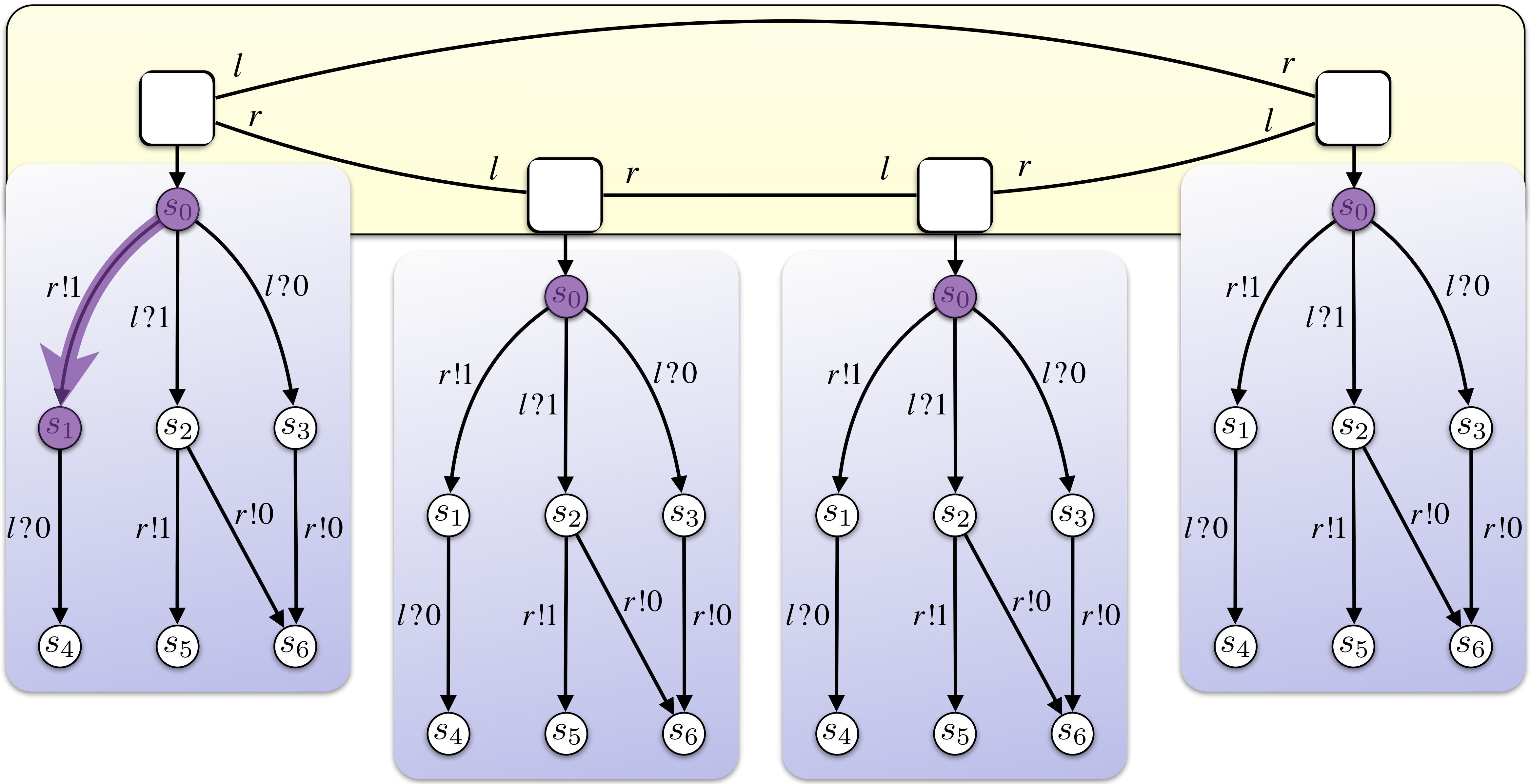
Parameterized Communicating Automata (PCA) over Rings



A PCA is given by:

- finite automaton over $\{l, r\} \times \{!, ?\} \times Msg$ (here: $Msg = \{0, 1\}$)
- acceptance condition

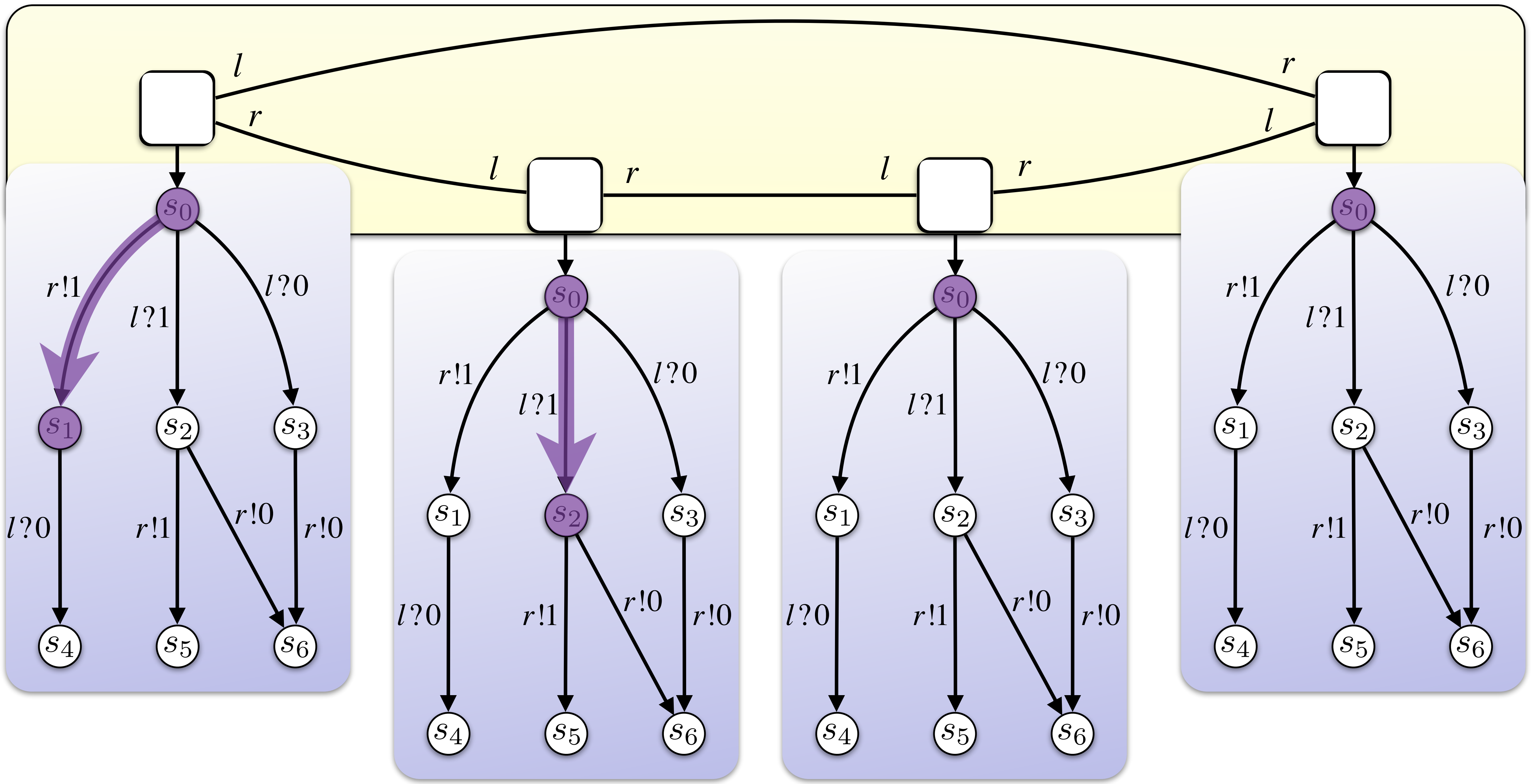
Parameterized Communicating Automata (PCA) over Rings



A PCA is given by:

- finite automaton over $\{l, r\} \times \{!, ?\} \times Msg$ (here: $Msg = \{0, 1\}$)
- acceptance condition

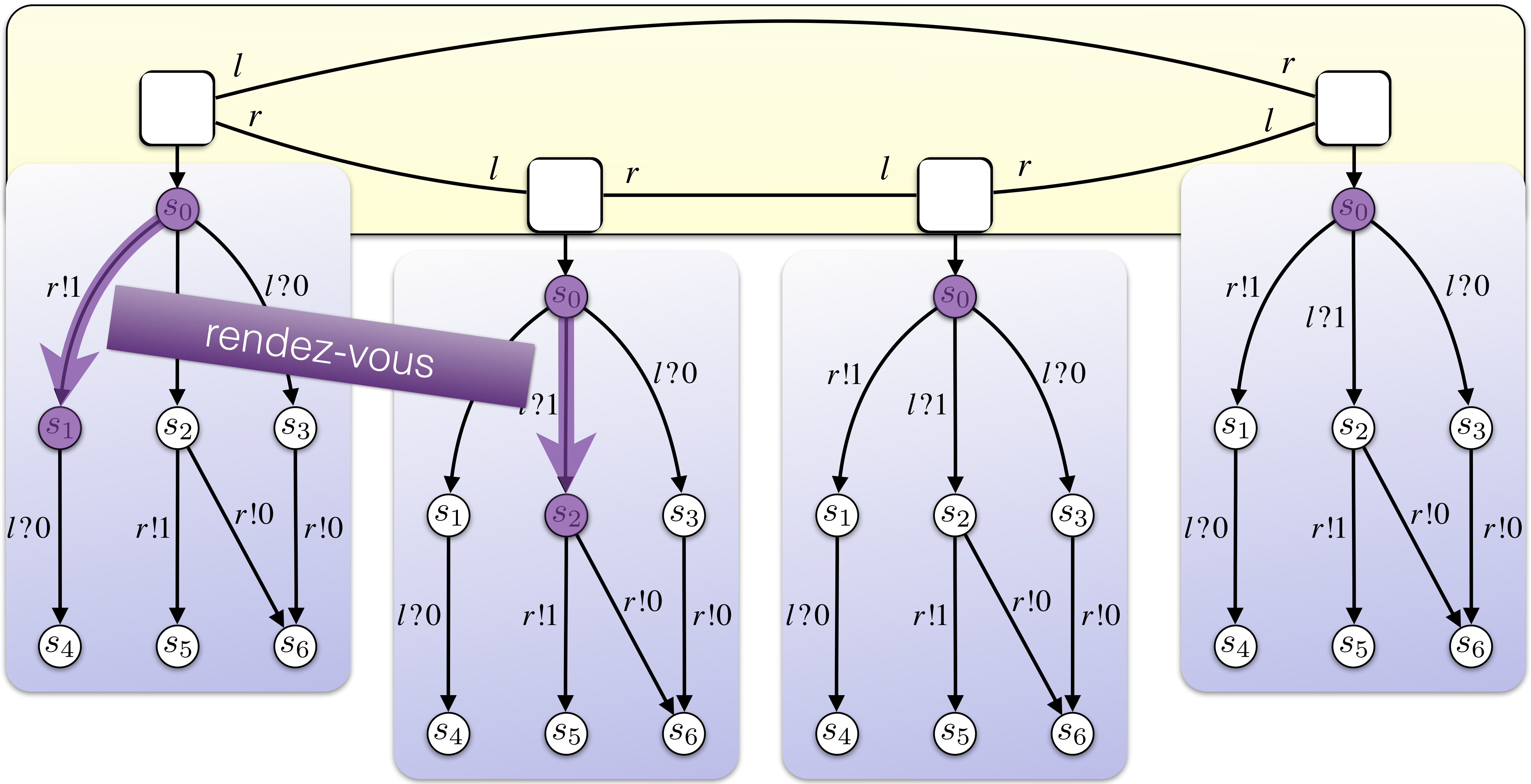
Parameterized Communicating Automata (PCA) over Rings



A PCA is given by:

- finite automaton over $\{l, r\} \times \{!, ?\} \times Msg$ (here: $Msg = \{0, 1\}$)
- acceptance condition

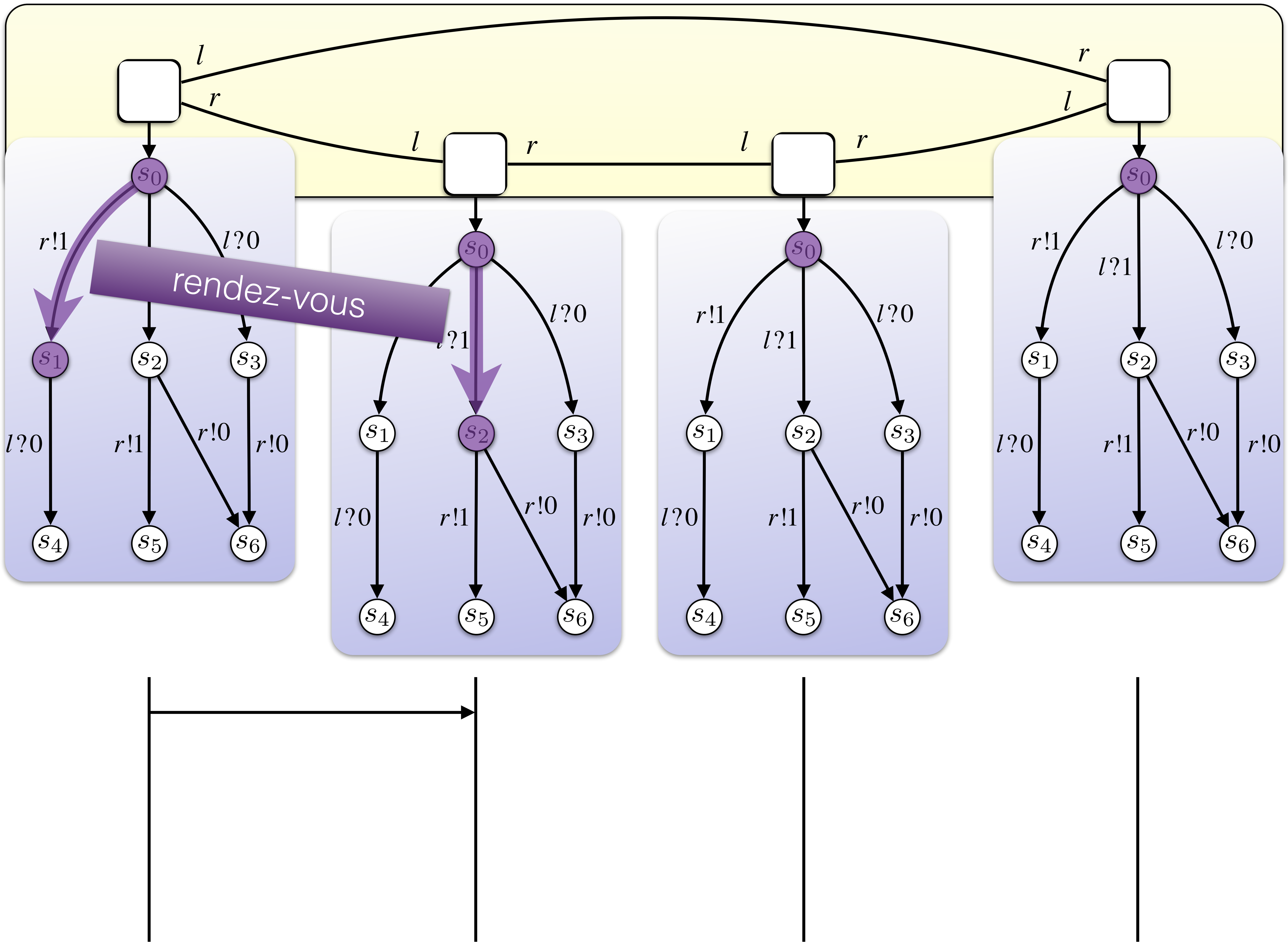
Parameterized Communicating Automata (PCA) over Rings



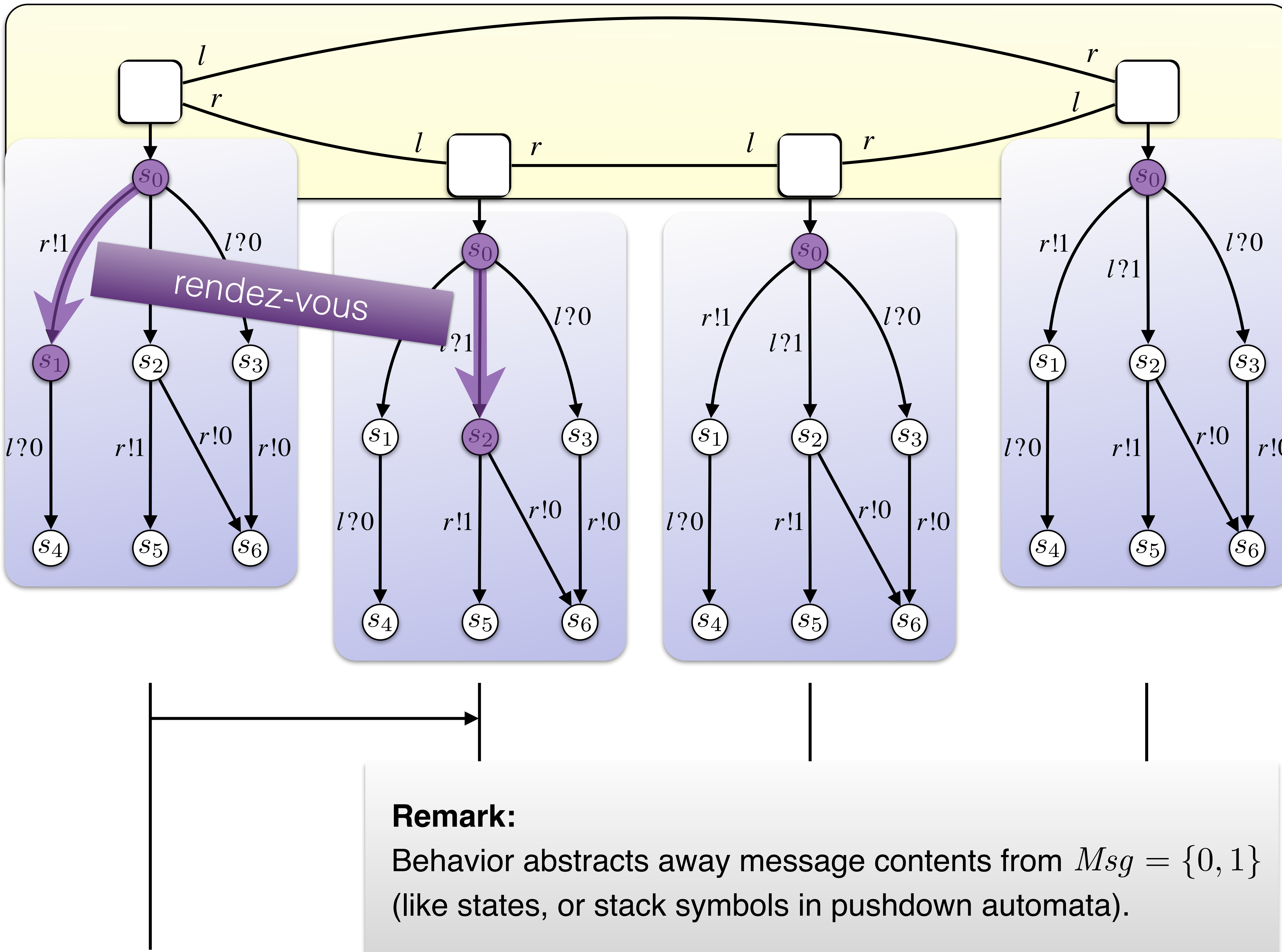
A PCA is given by:

- finite automaton over $\{l, r\} \times \{!, ?\} \times Msg$ (here: $Msg = \{0, 1\}$)
- acceptance condition

Parameterized Communicating Automata (PCA) over Rings

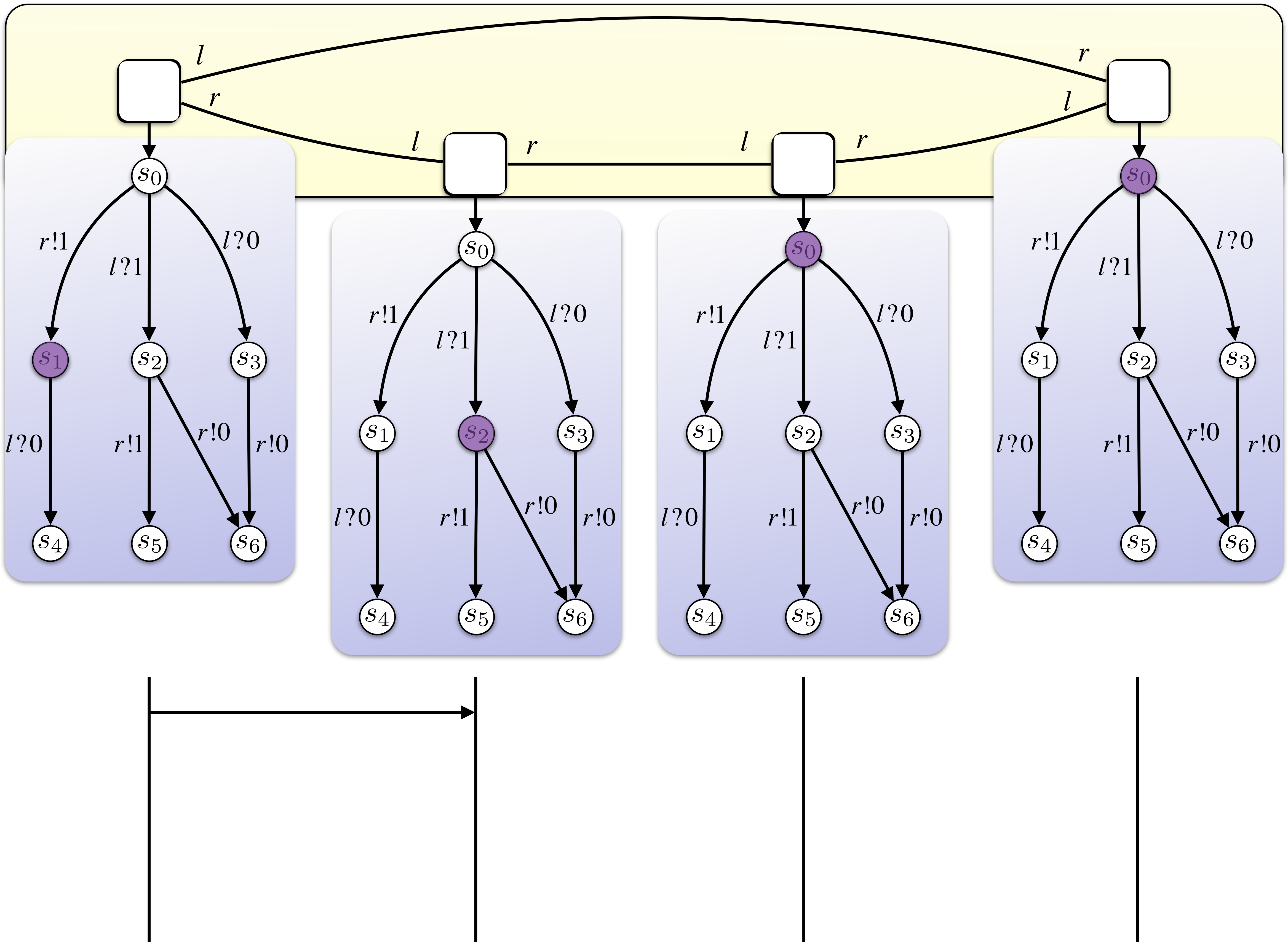


Parameterized Communicating Automata (PCA) over Rings

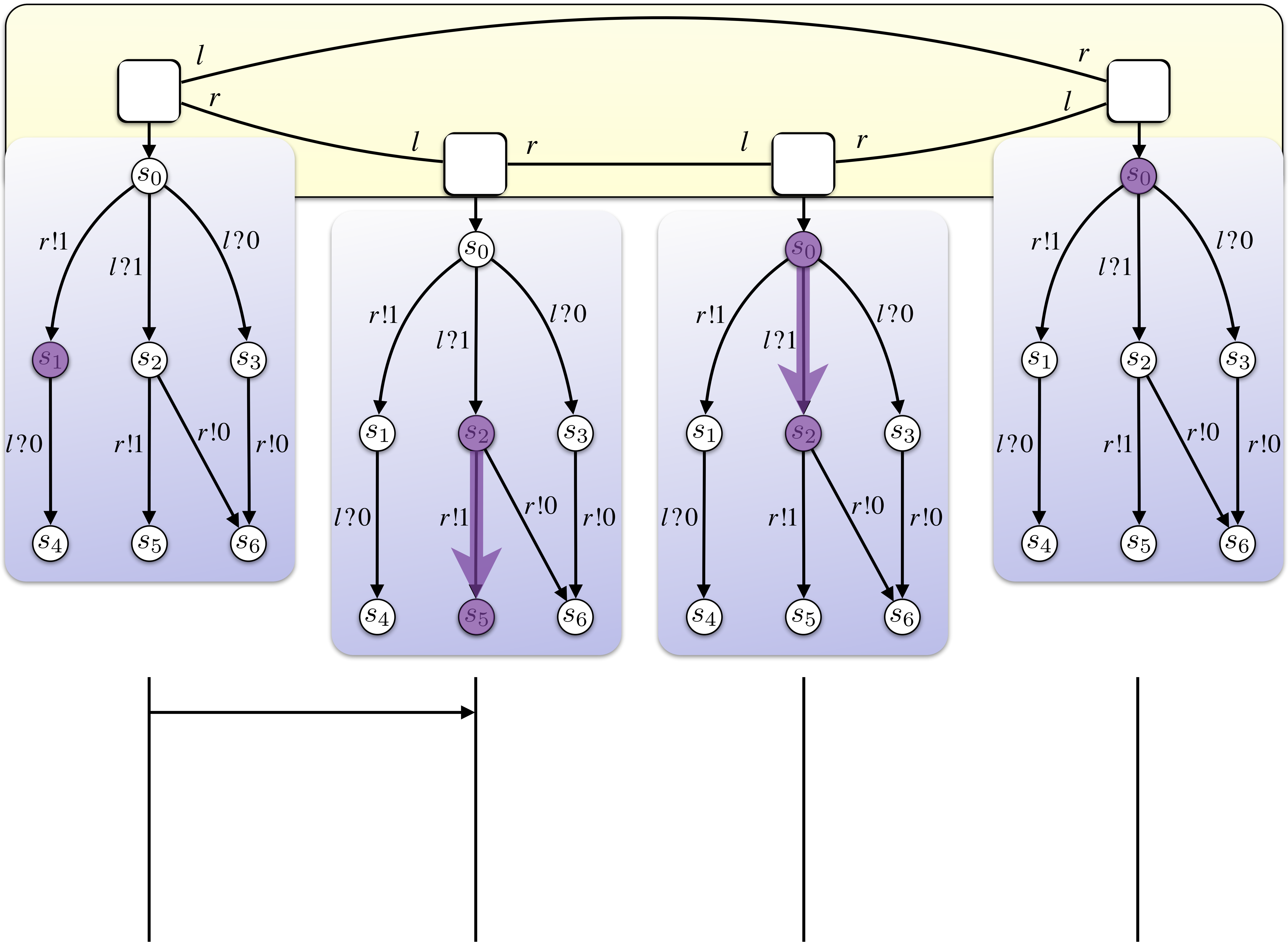


Remark:
Behavior abstracts away message contents from $Msg = \{0, 1\}$ (like states, or stack symbols in pushdown automata).

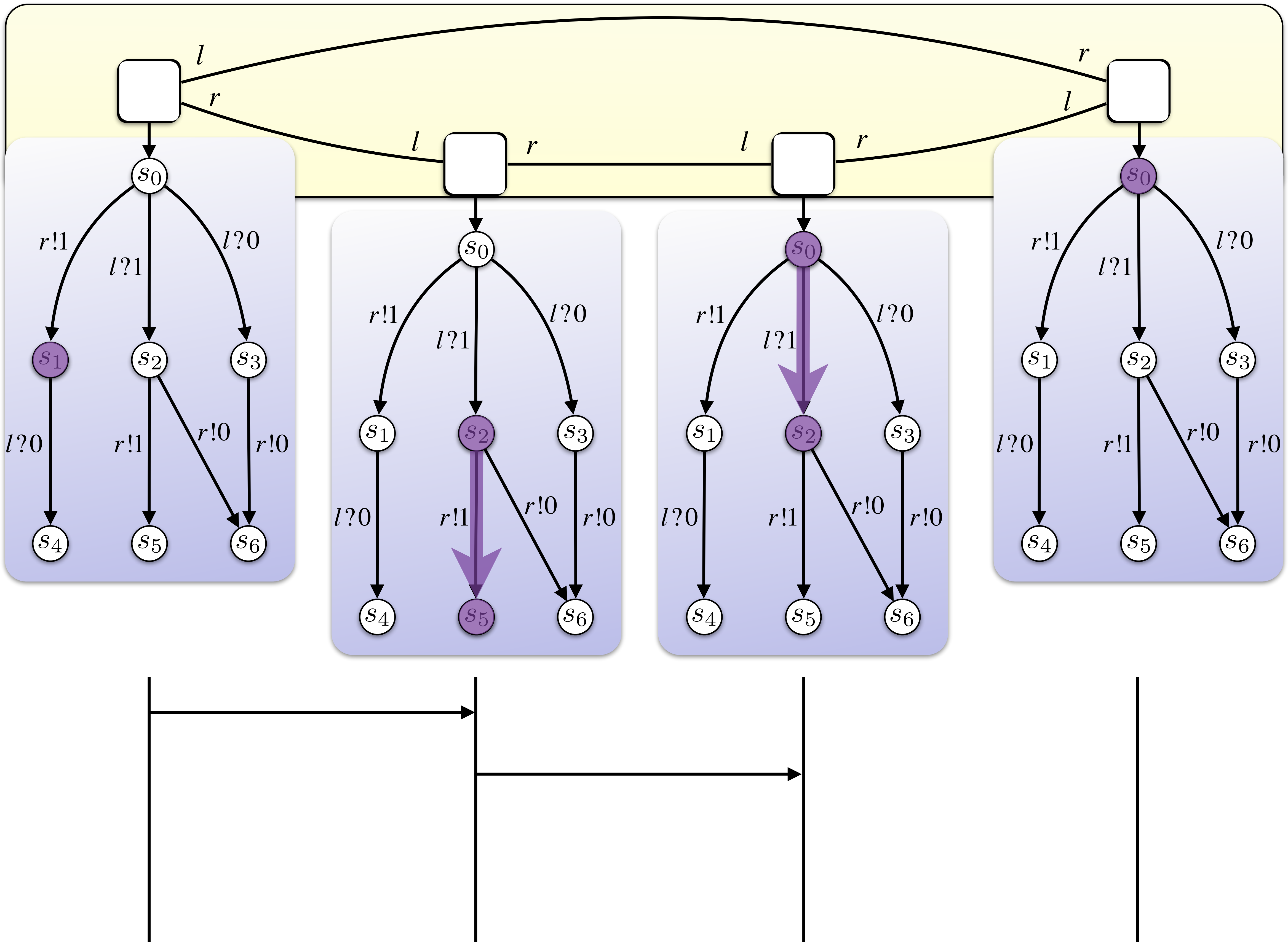
Parameterized Communicating Automata (PCA) over Rings



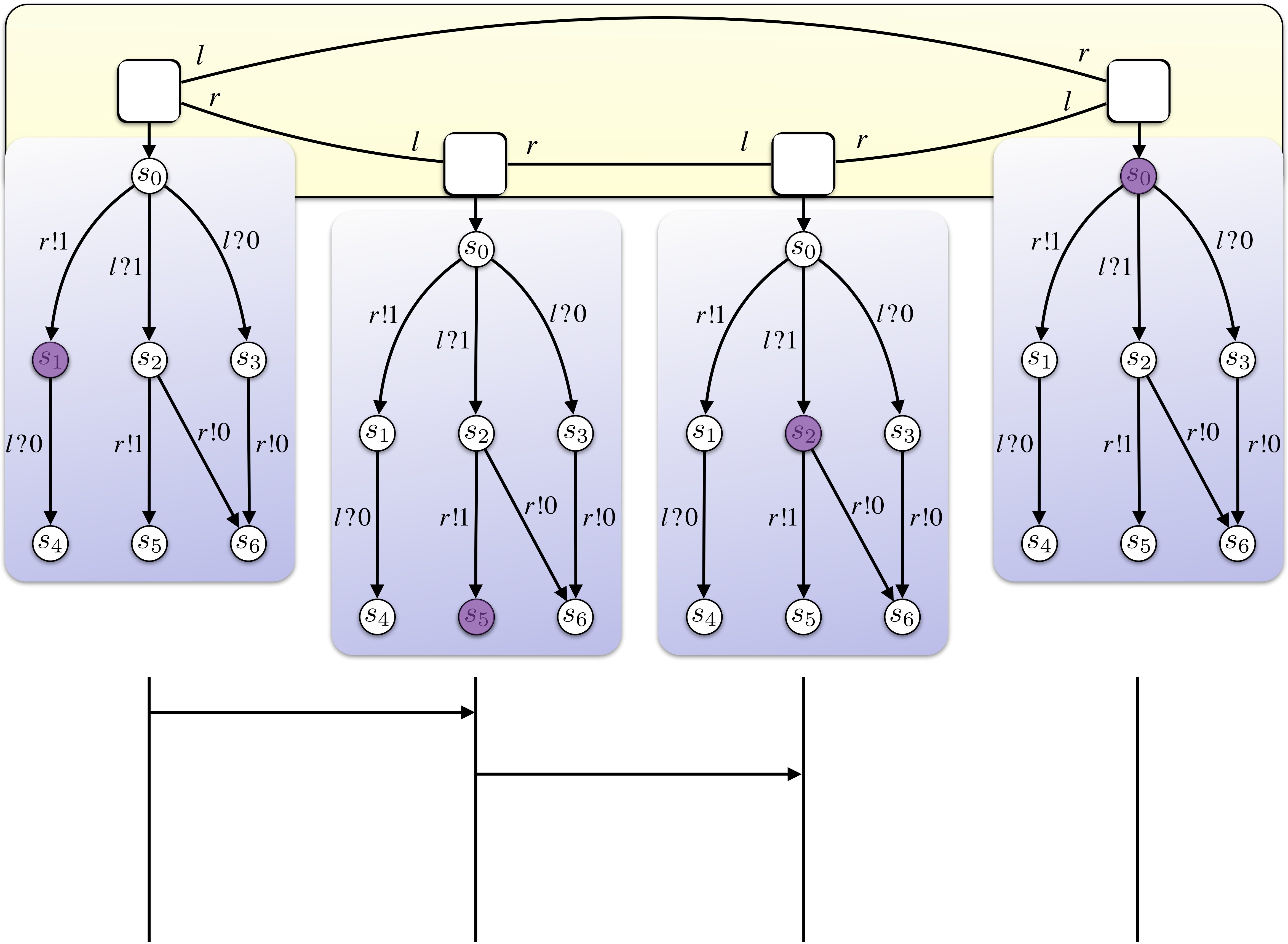
Parameterized Communicating Automata (PCA) over Rings



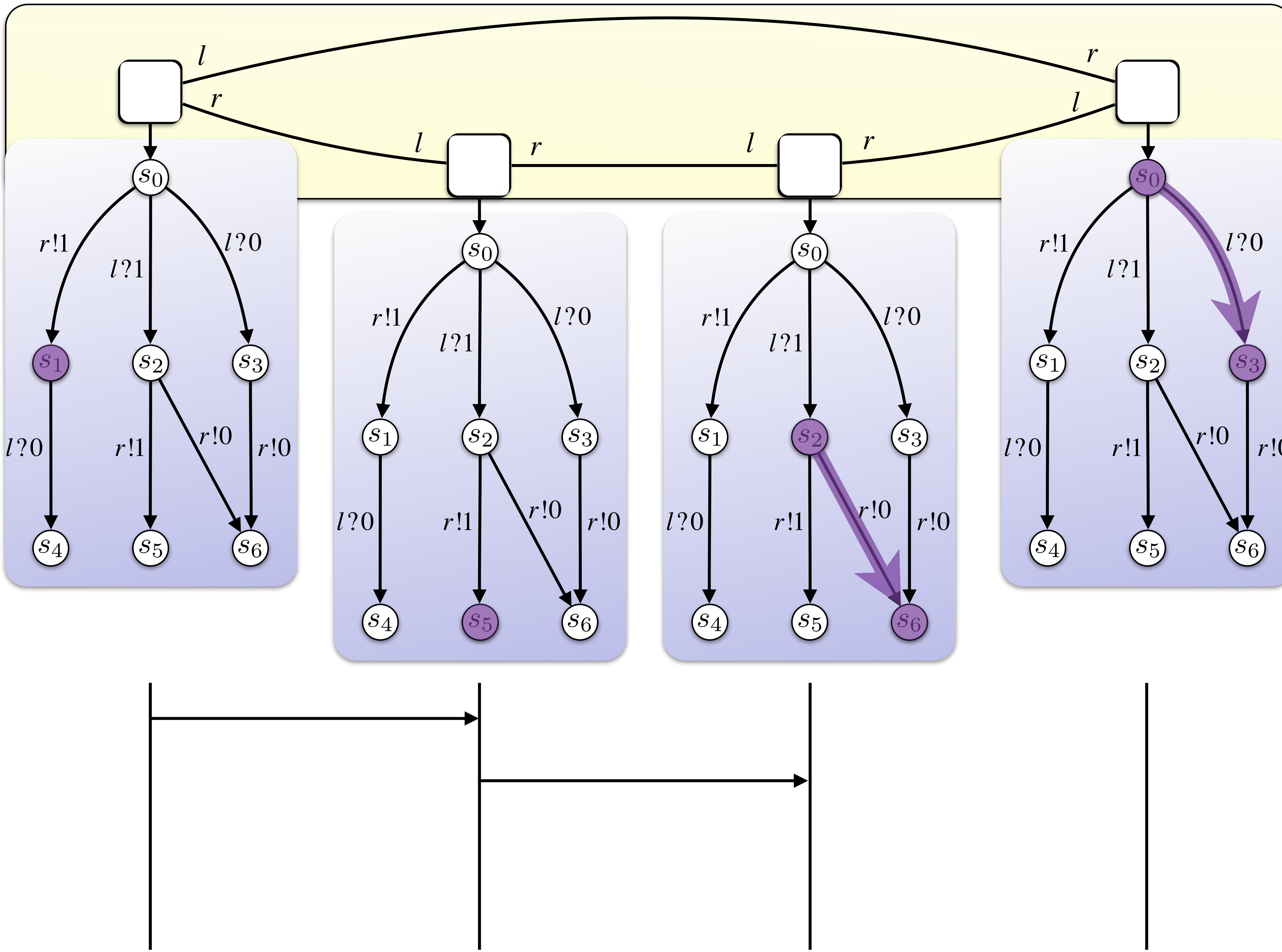
Parameterized Communicating Automata (PCA) over Rings



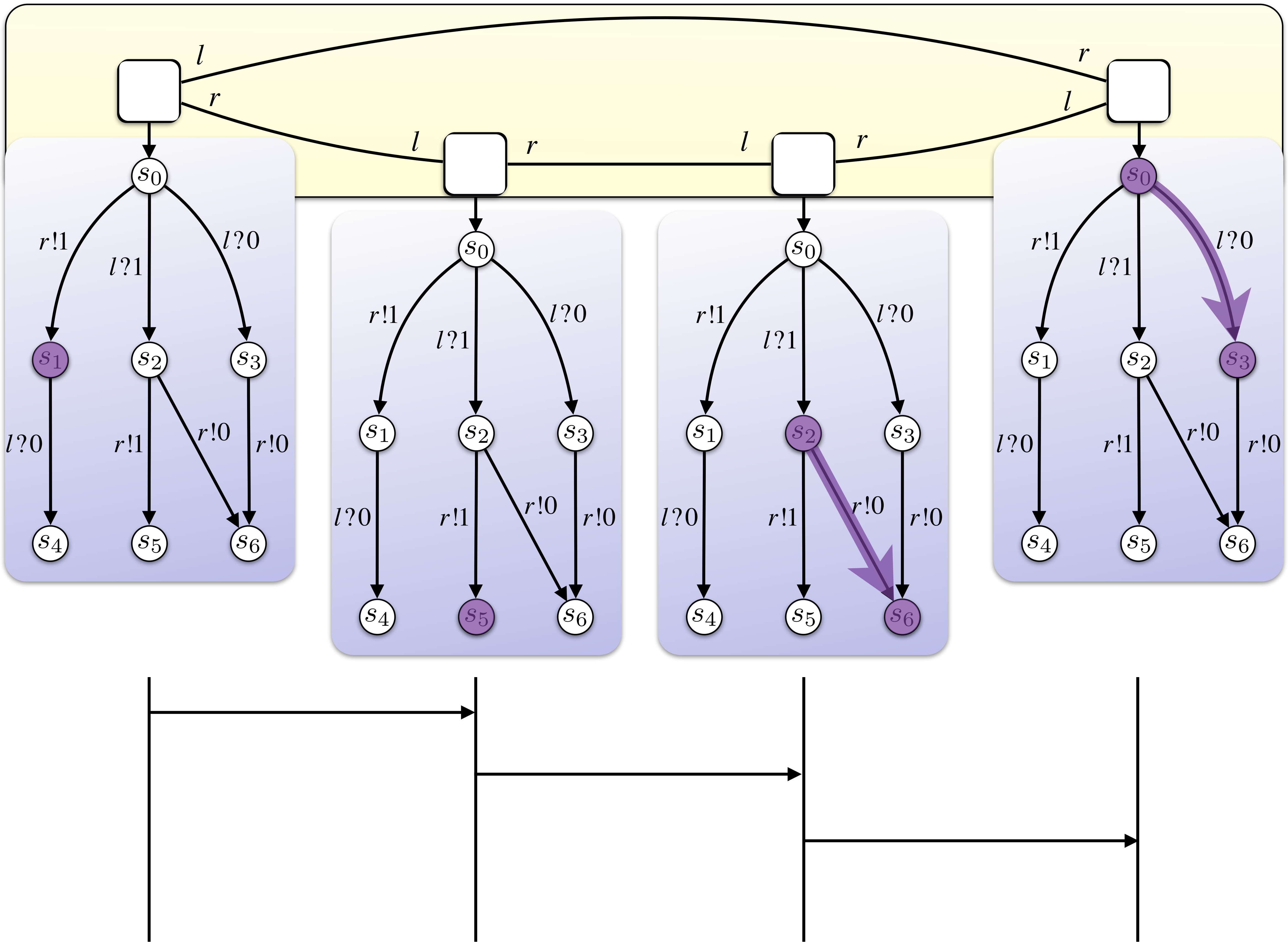
Parameterized Communicating Automata (PCA) over Rings



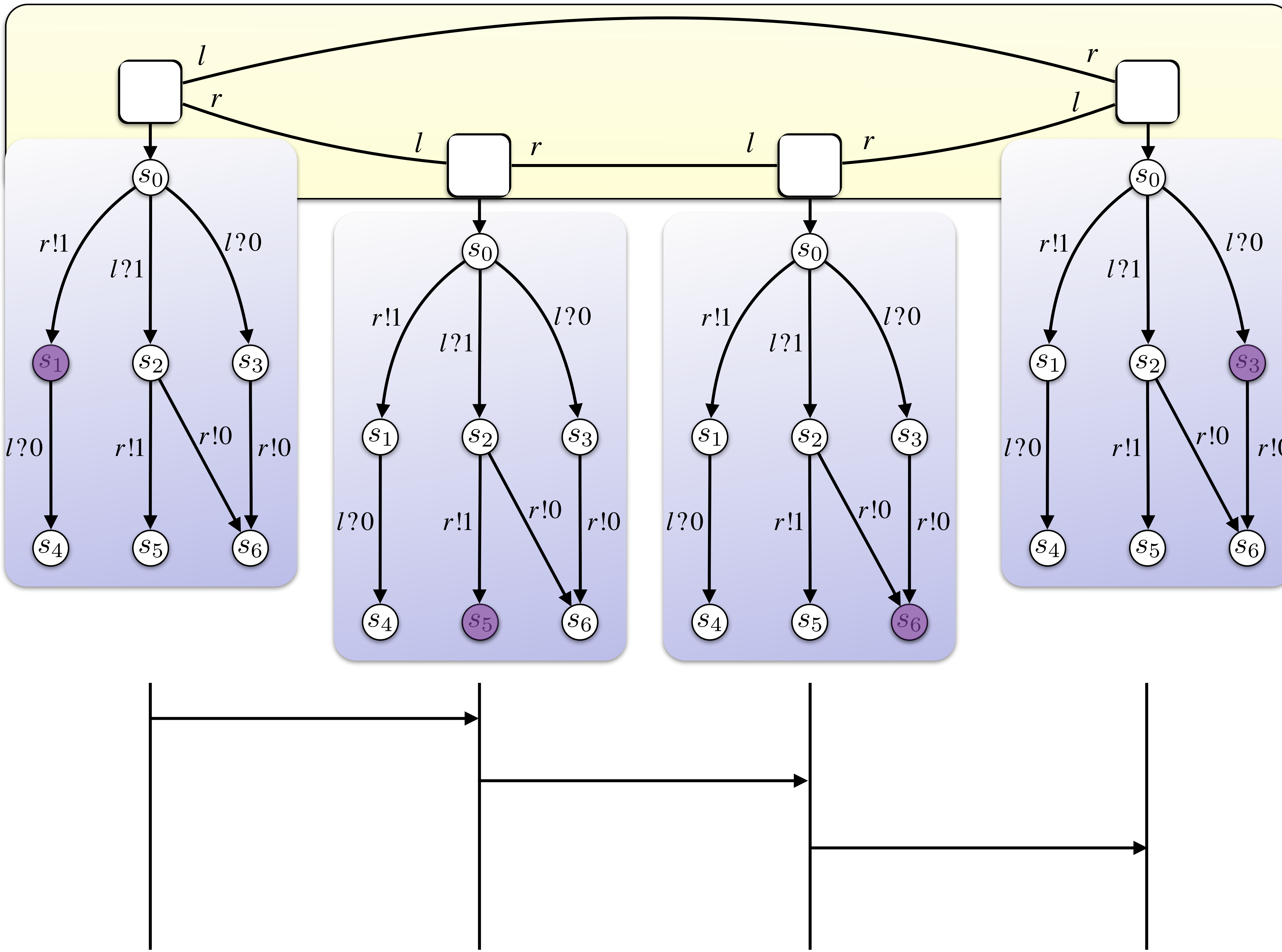
Parameterized Communicating Automata (PCA) over Rings



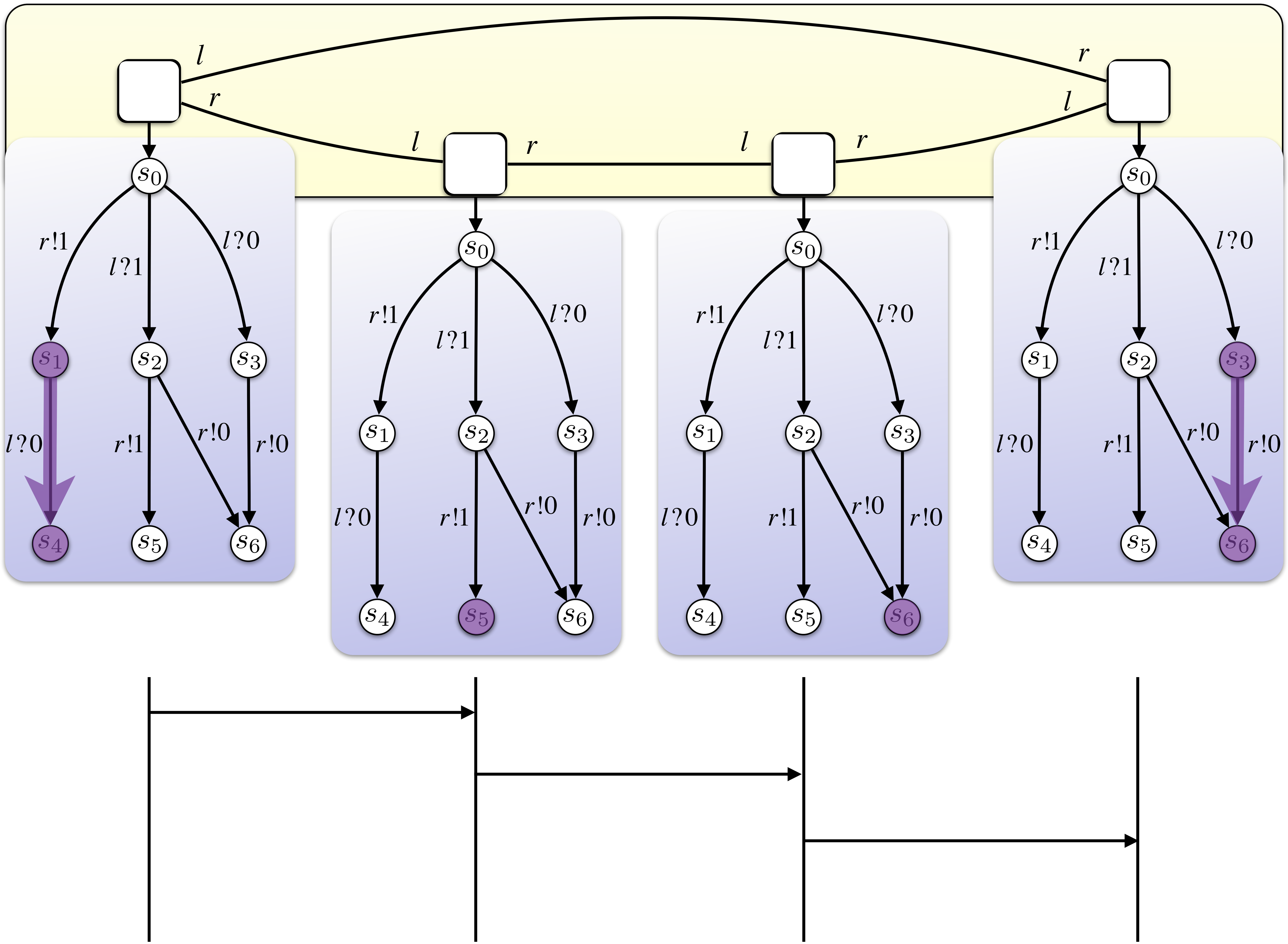
Parameterized Communicating Automata (PCA) over Rings



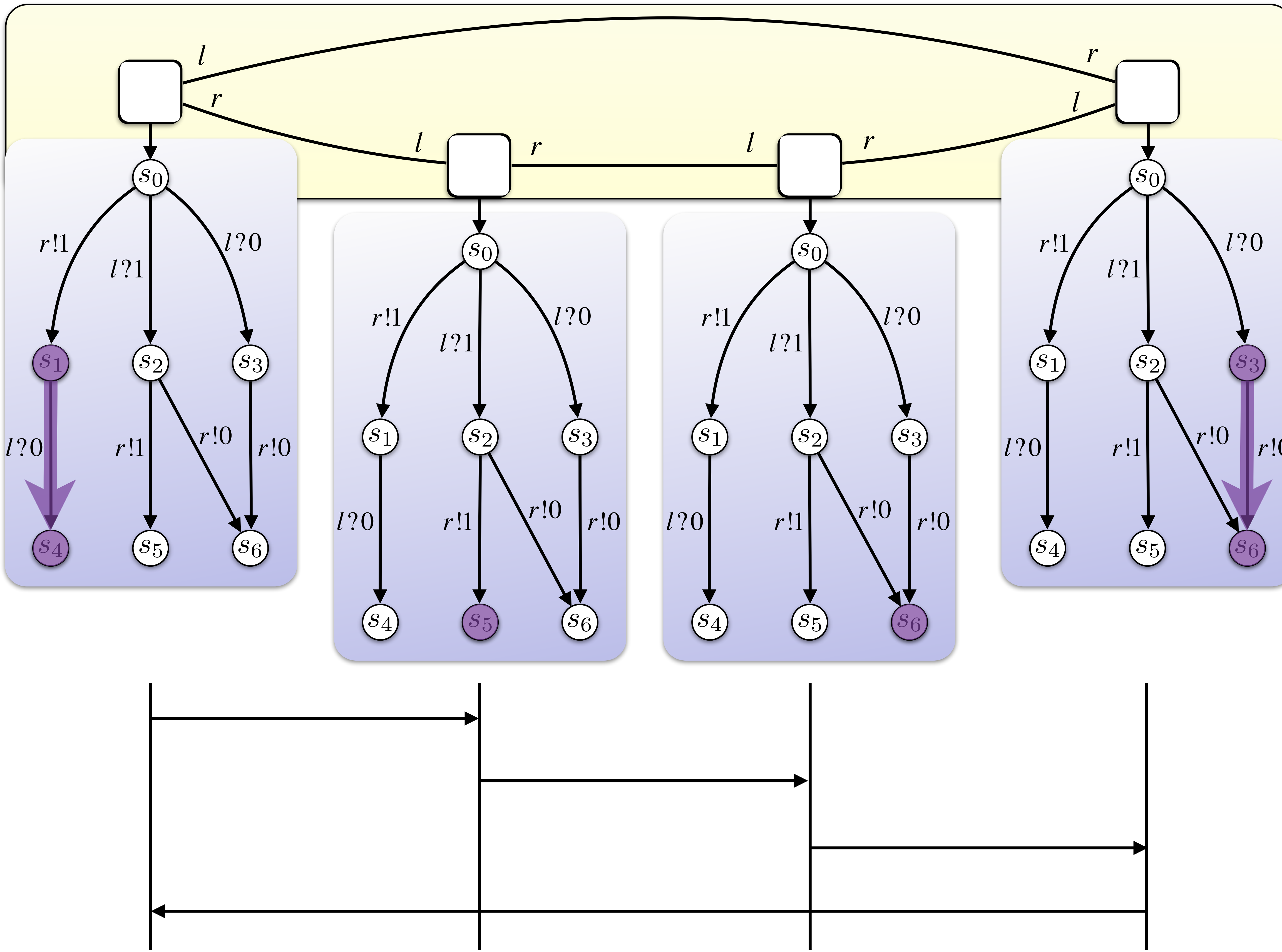
Parameterized Communicating Automata (PCA) over Rings



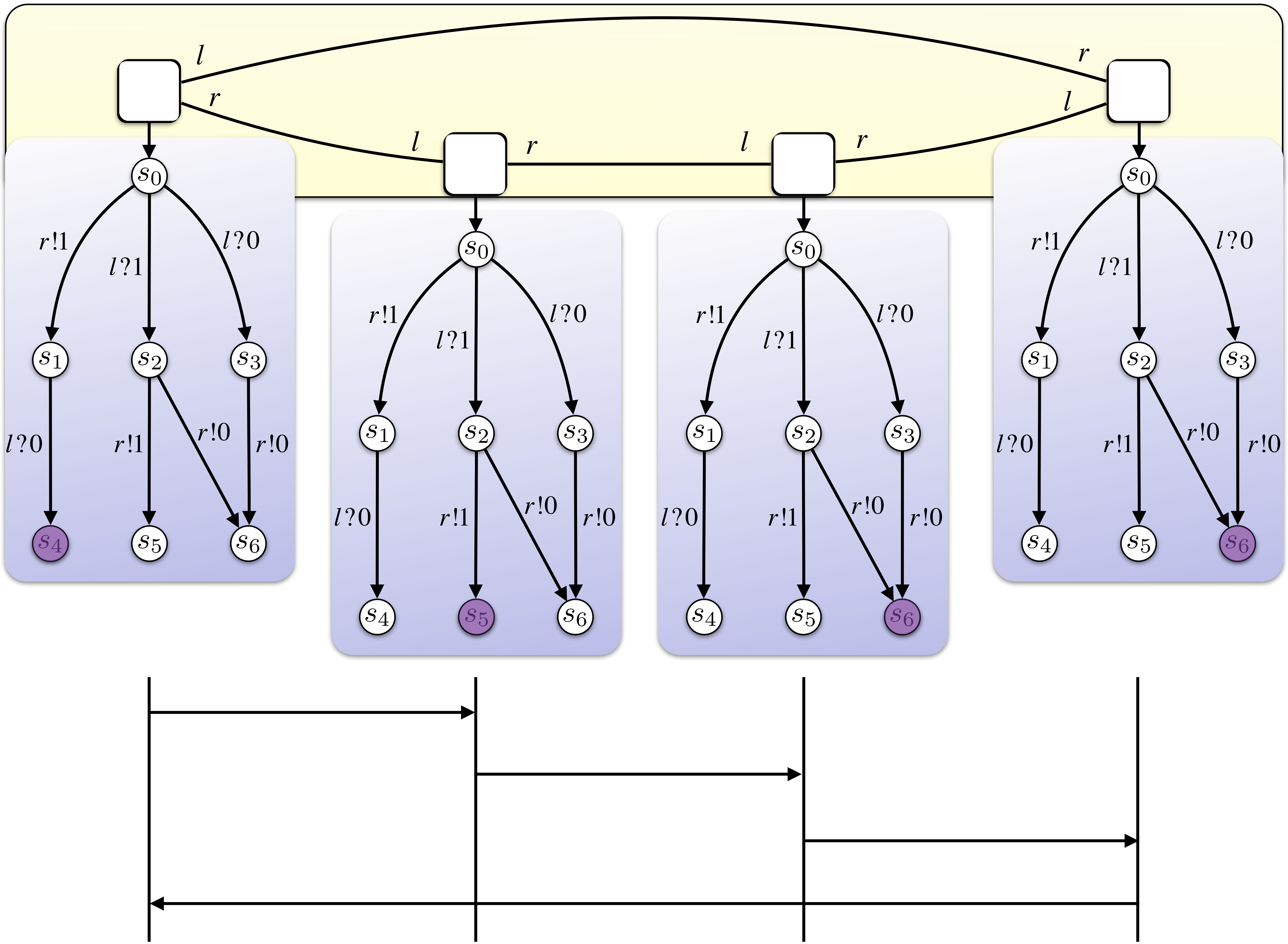
Parameterized Communicating Automata (PCA) over Rings



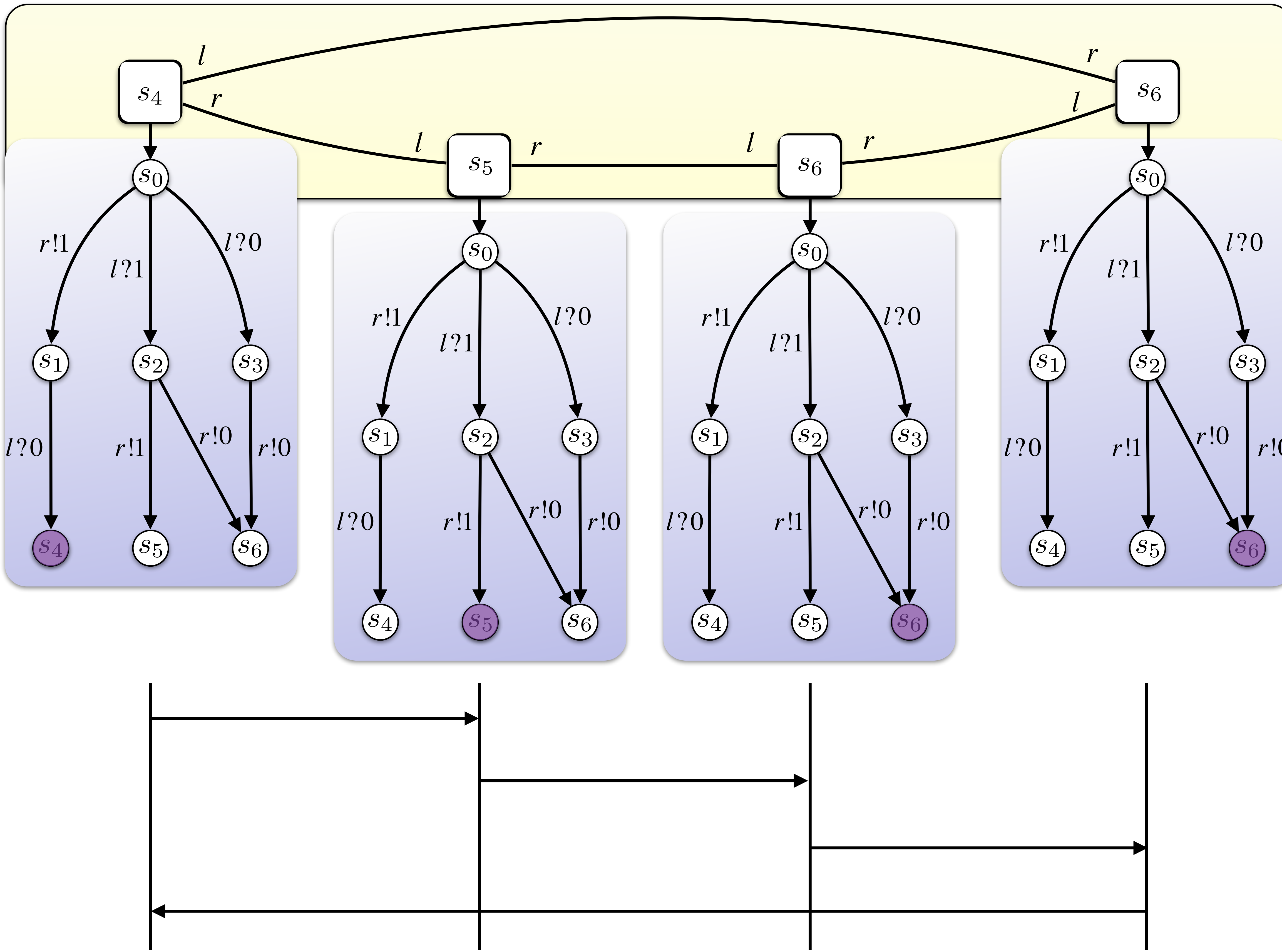
Parameterized Communicating Automata (PCA) over Rings



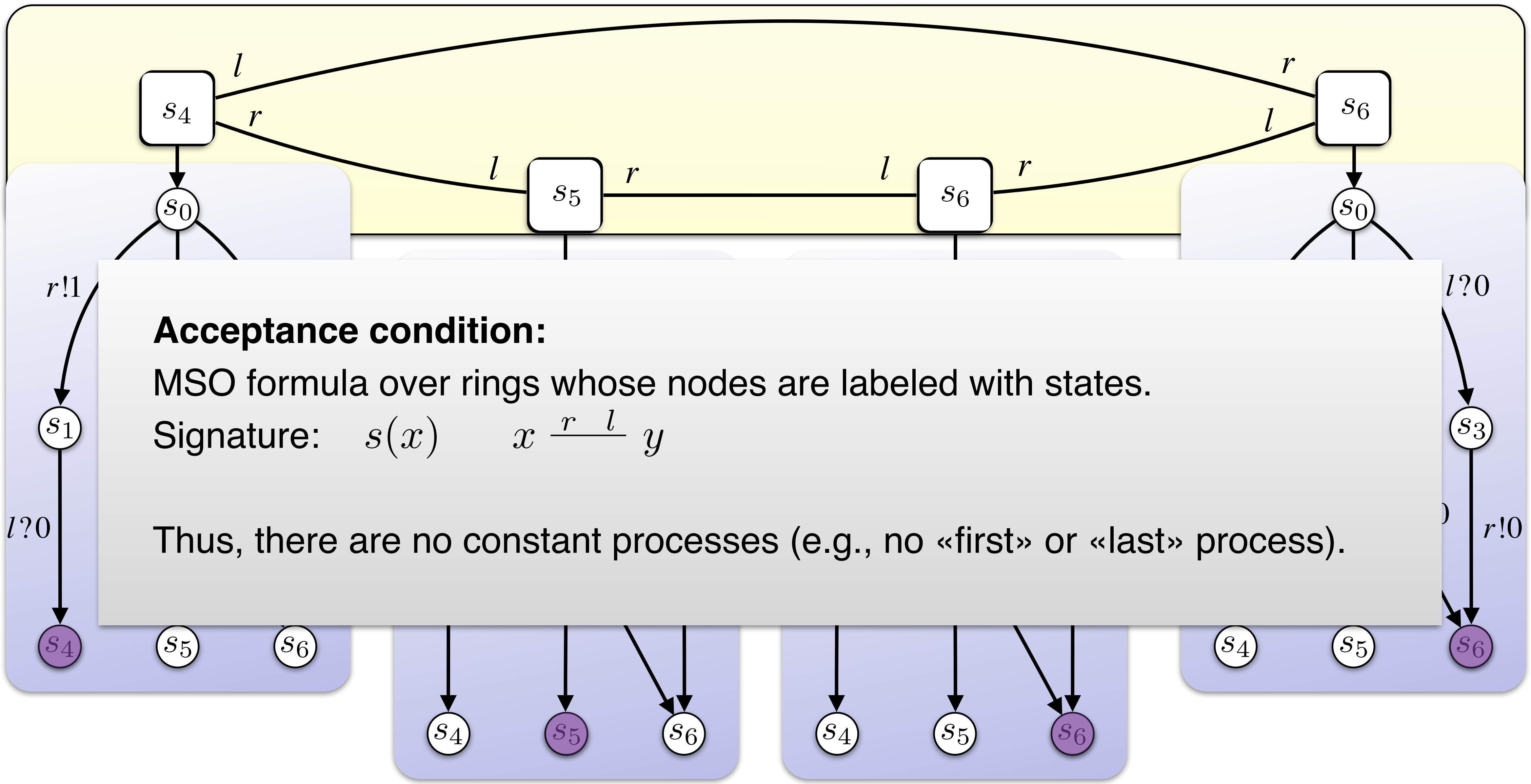
Parameterized Communicating Automata (PCA) over Rings



Parameterized Communicating Automata (PCA) over Rings



Parameterized Communicating Automata (PCA) over Rings

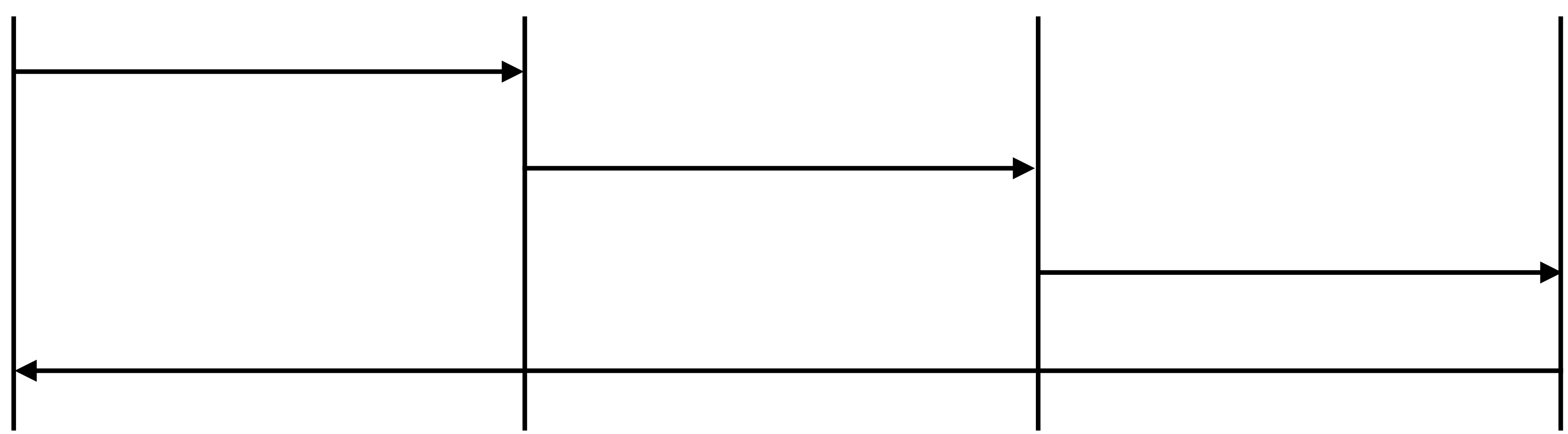


Acceptance condition:

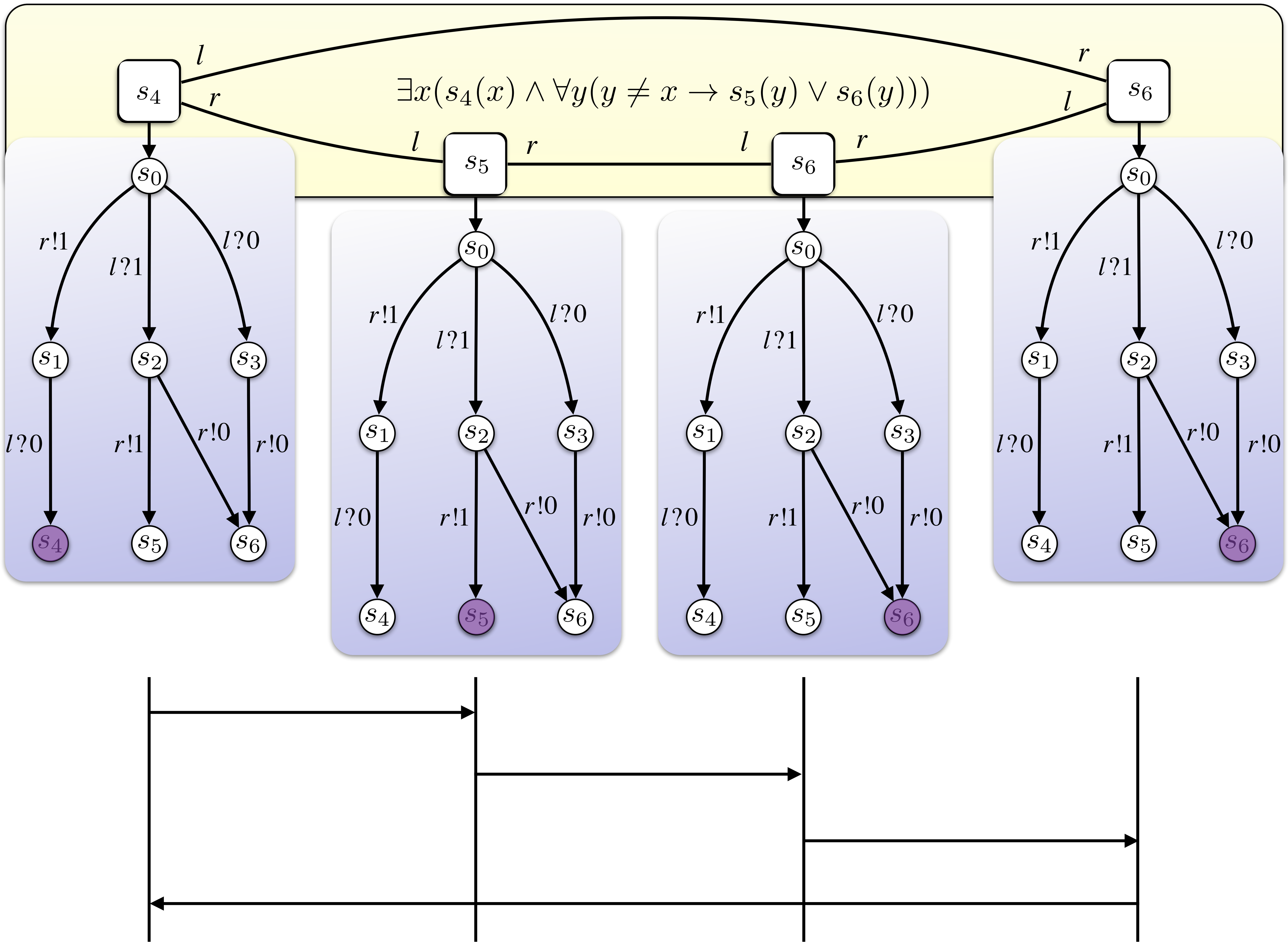
MSO formula over rings whose nodes are labeled with states.

Signature: $s(x)$ $\quad x \xrightarrow{r \quad l} y$

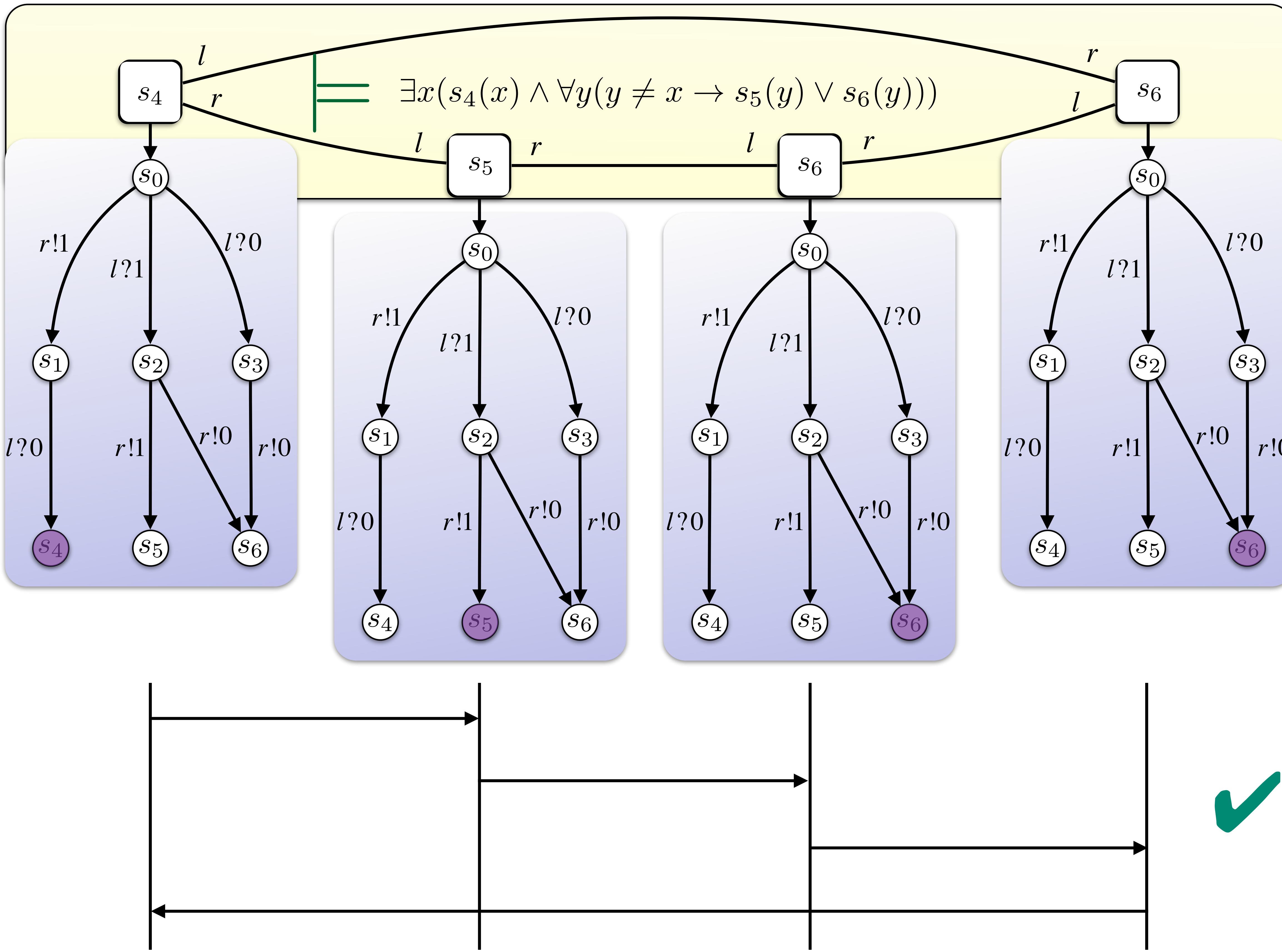
Thus, there are no constant processes (e.g., no «first» or «last» process).



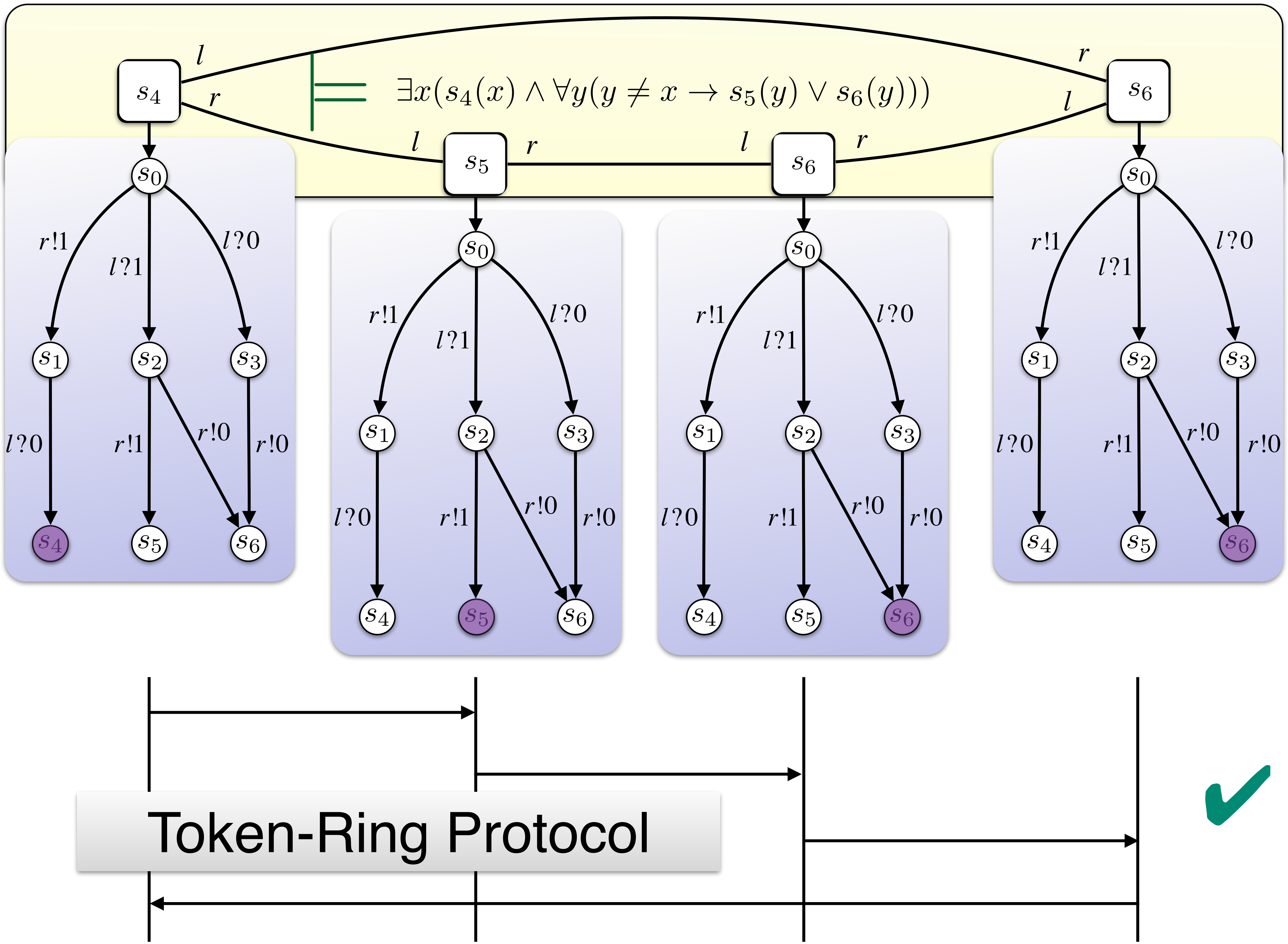
Parameterized Communicating Automata (PCA) over Rings



Parameterized Communicating Automata (PCA) over Rings



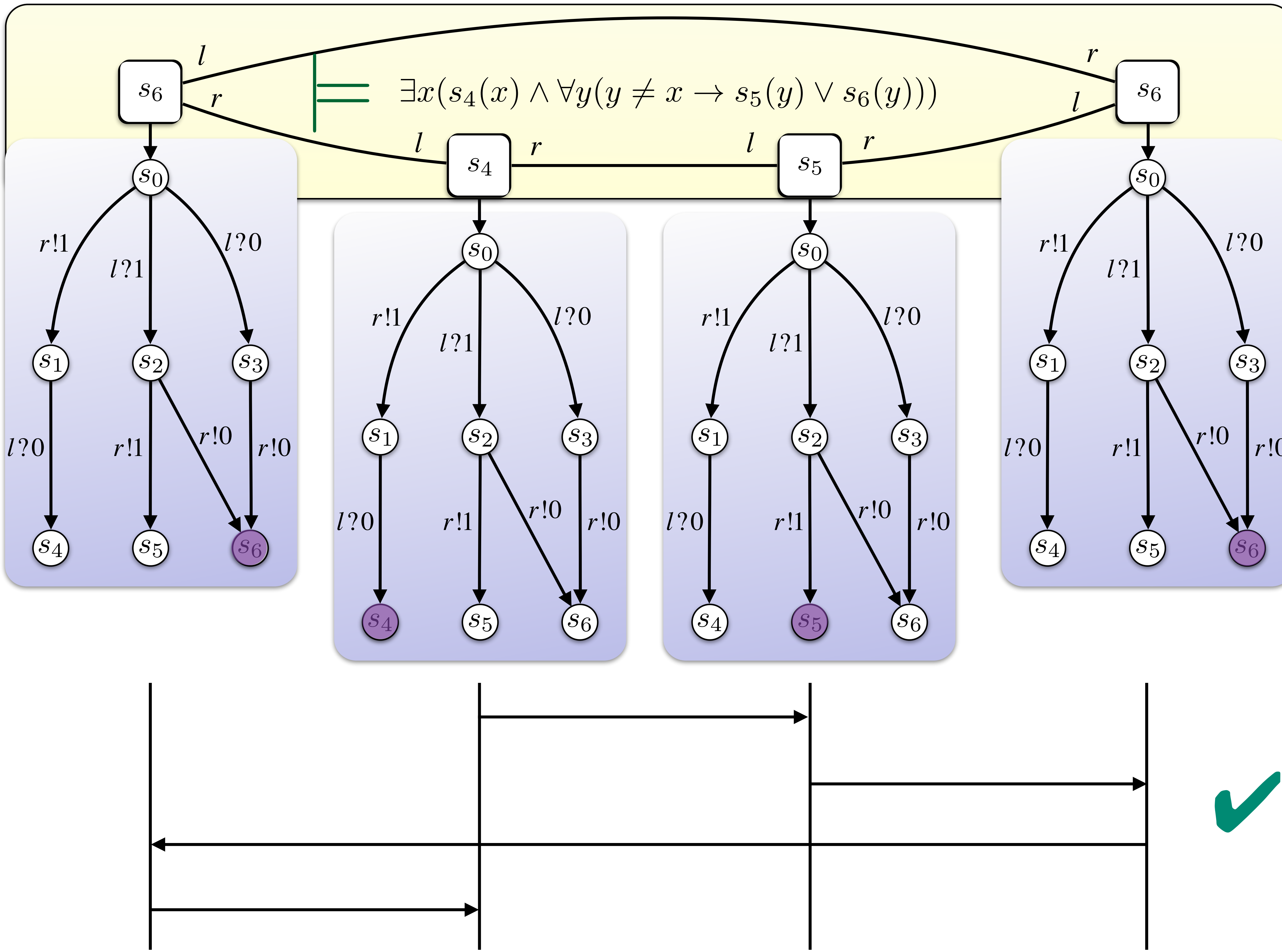
Parameterized Communicating Automata (PCA) over Rings



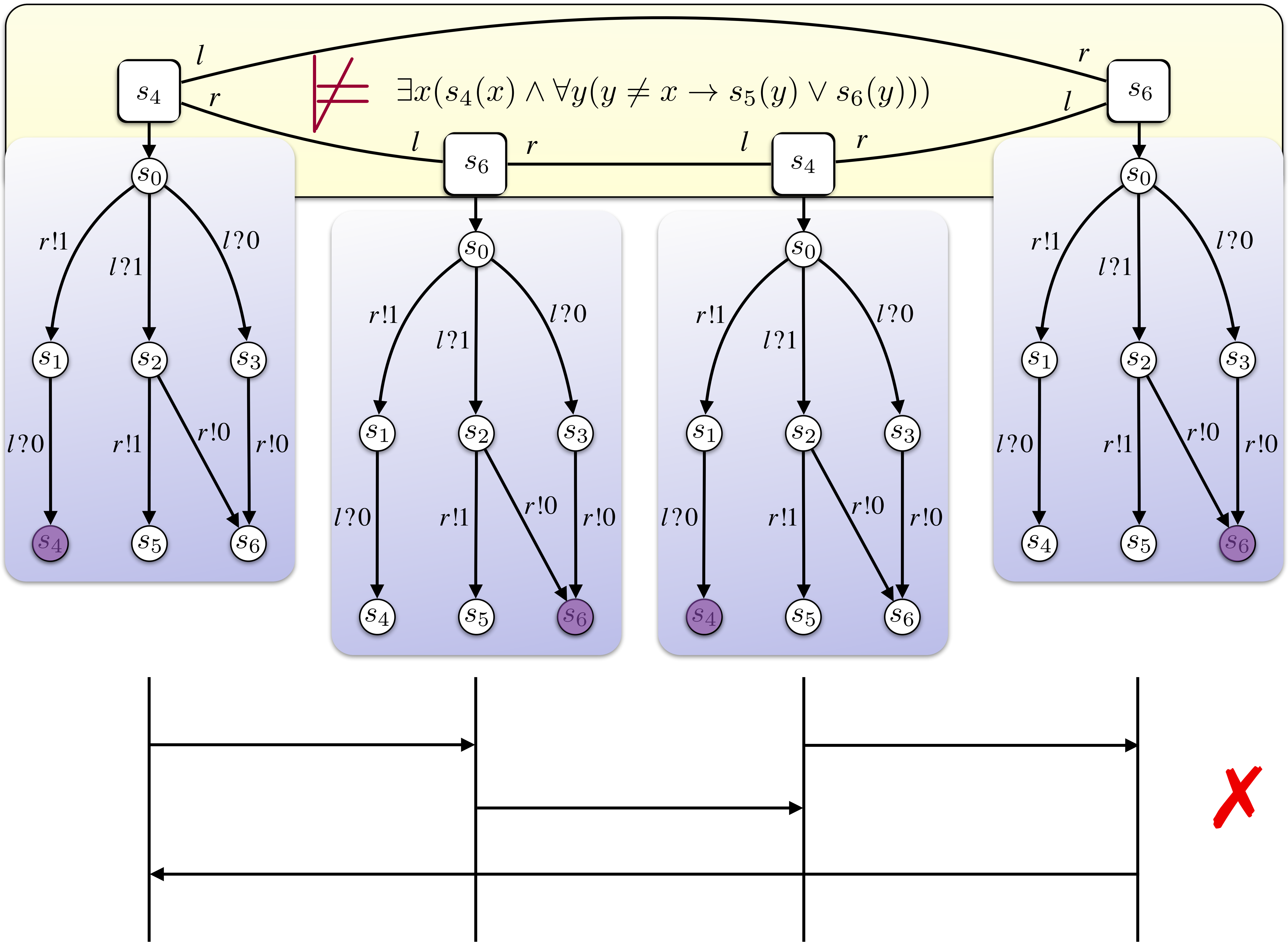
Token-Ring Protocol

✔

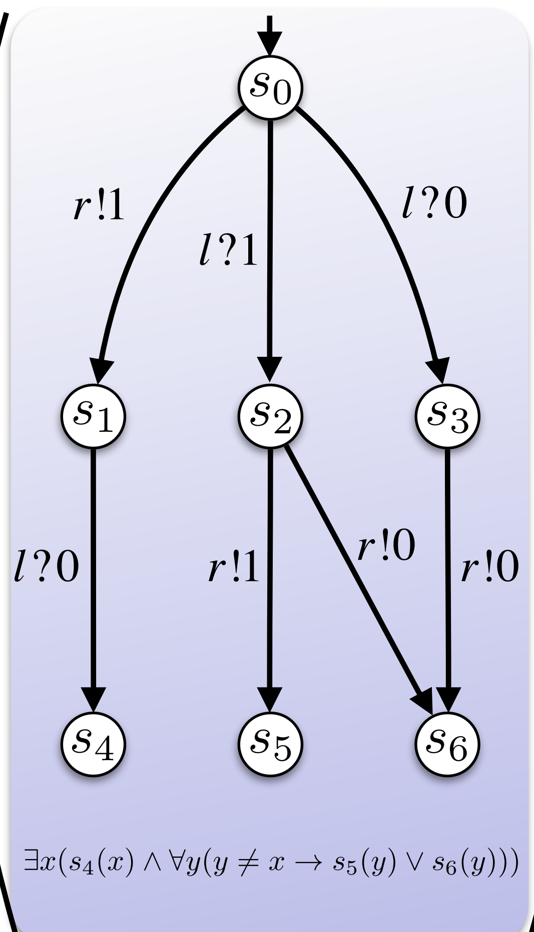
Parameterized Communicating Automata (PCA) over Rings



Parameterized Communicating Automata (PCA) over Rings



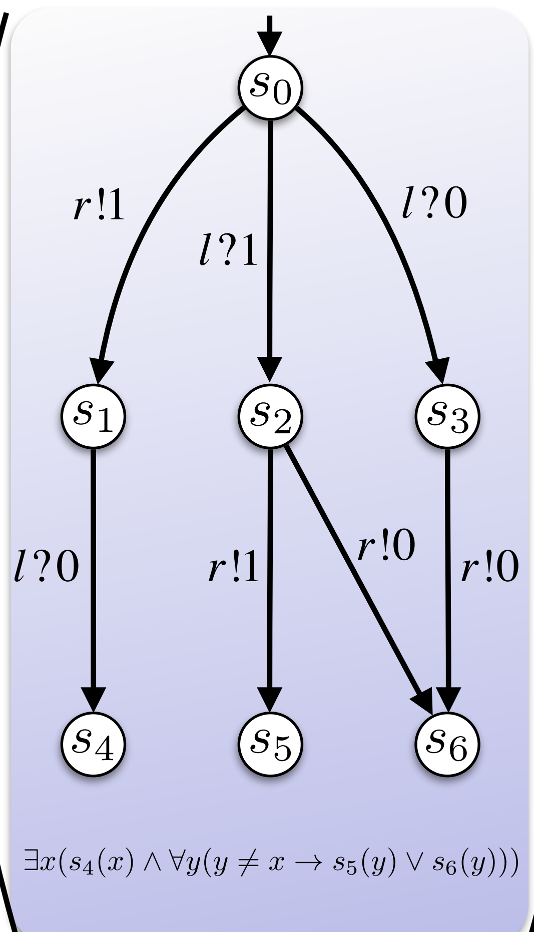
Parameterized Communicating Automata (PCA) over Rings

L (


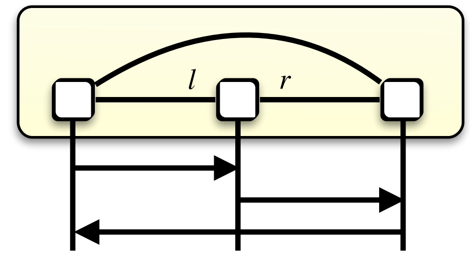
)

Parameterized Communicating Automata (PCA) over Rings

L

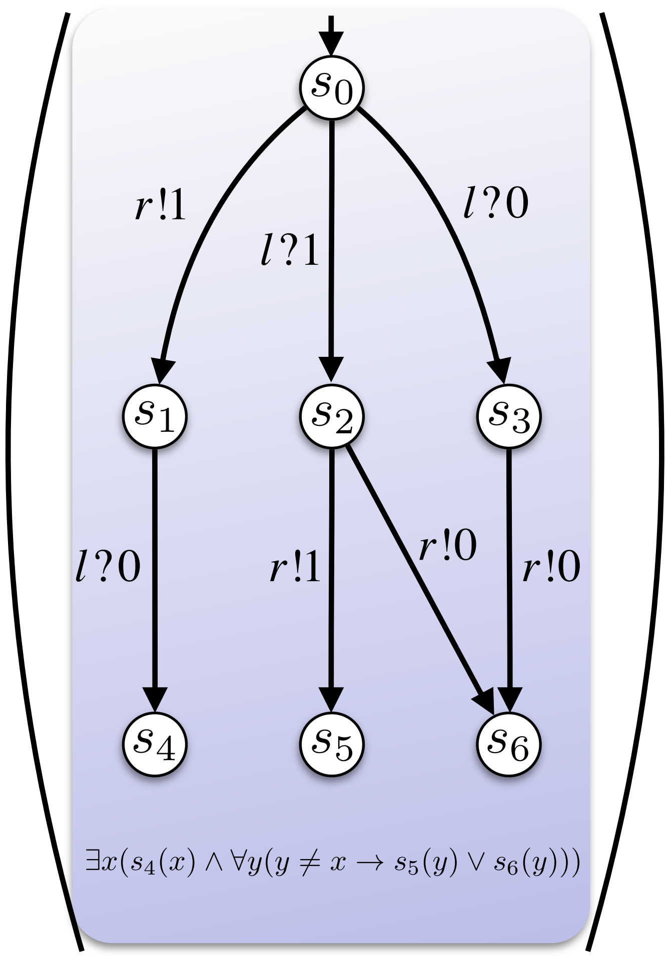


$=$

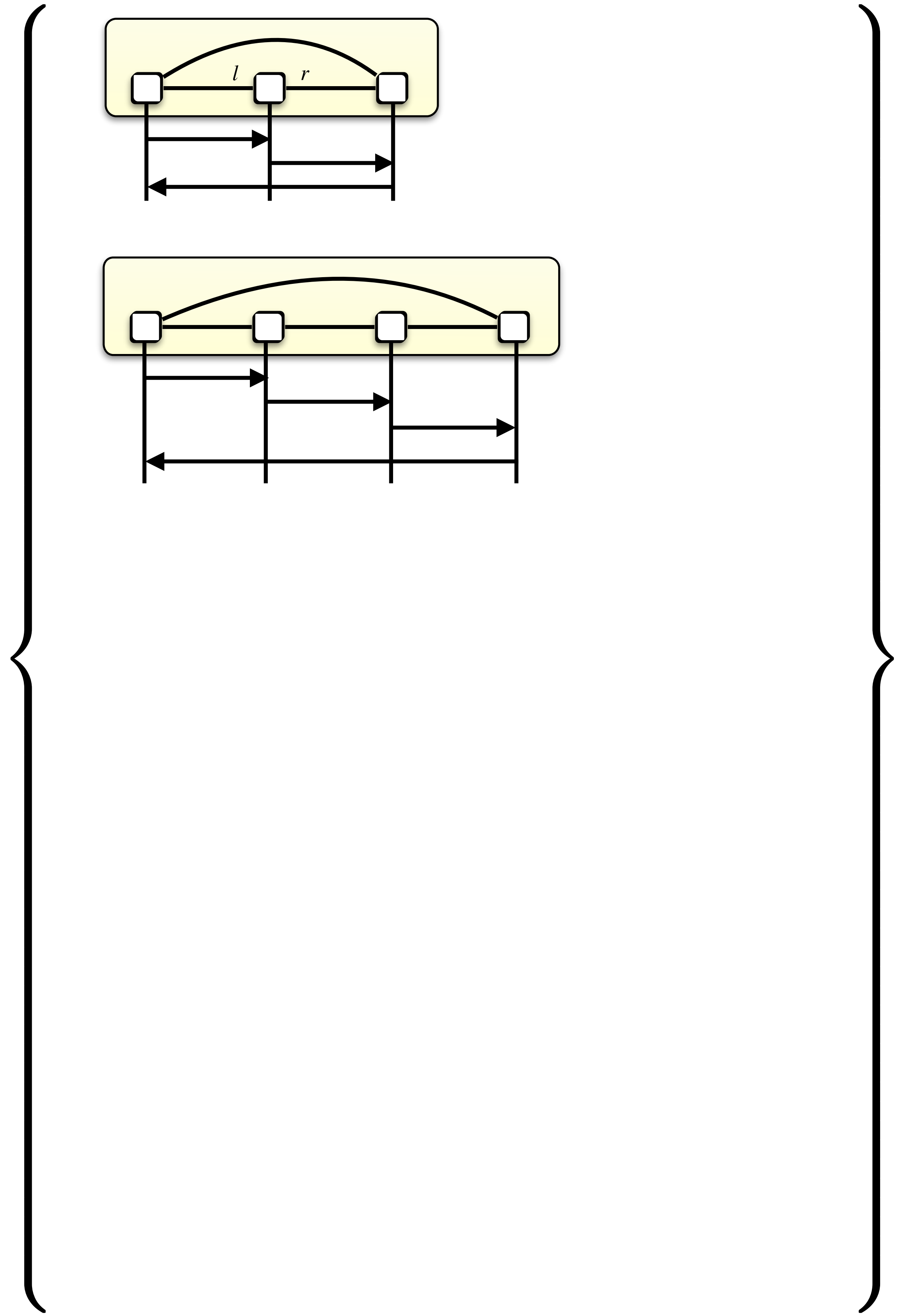


Parameterized Communicating Automata (PCA) over Rings

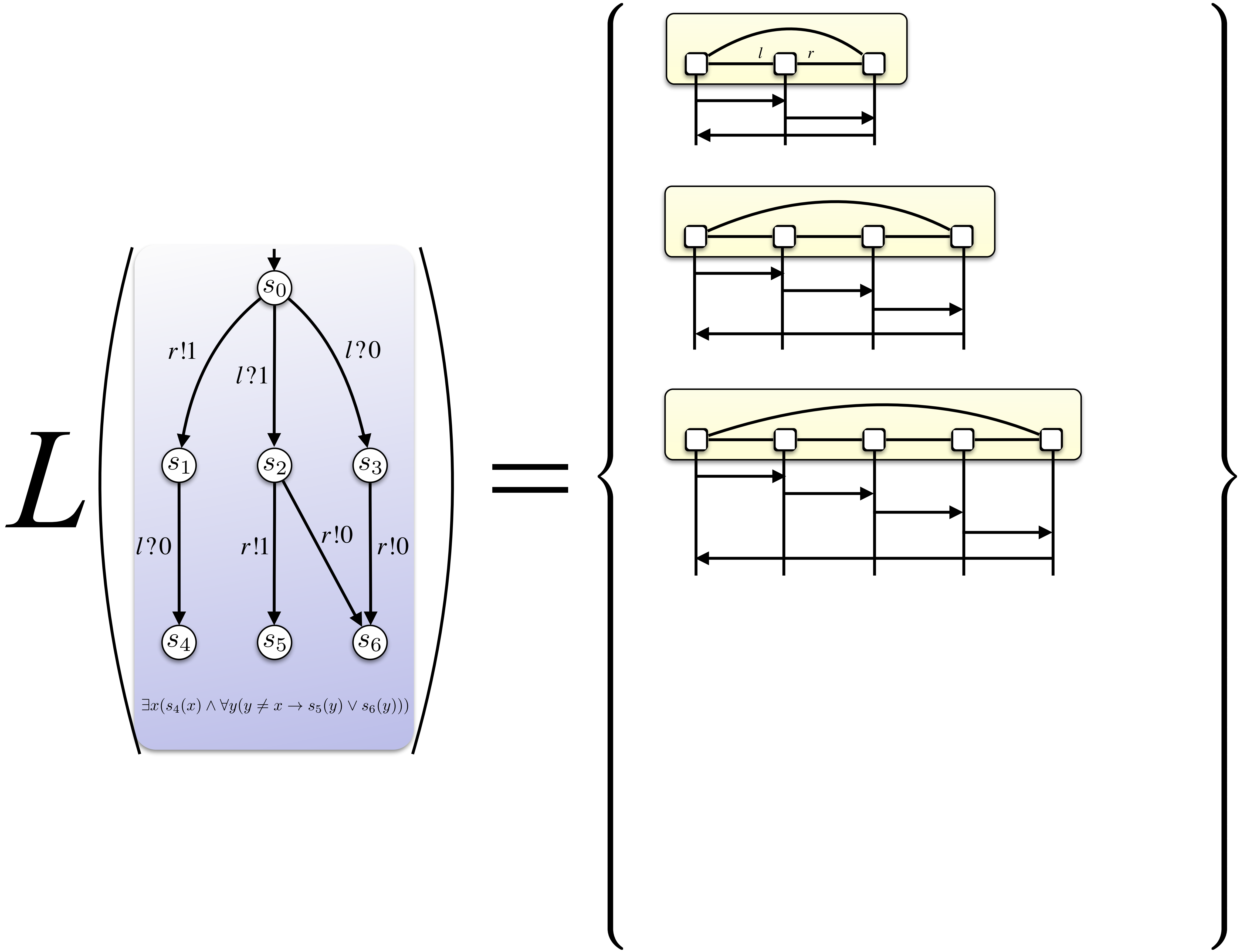
L



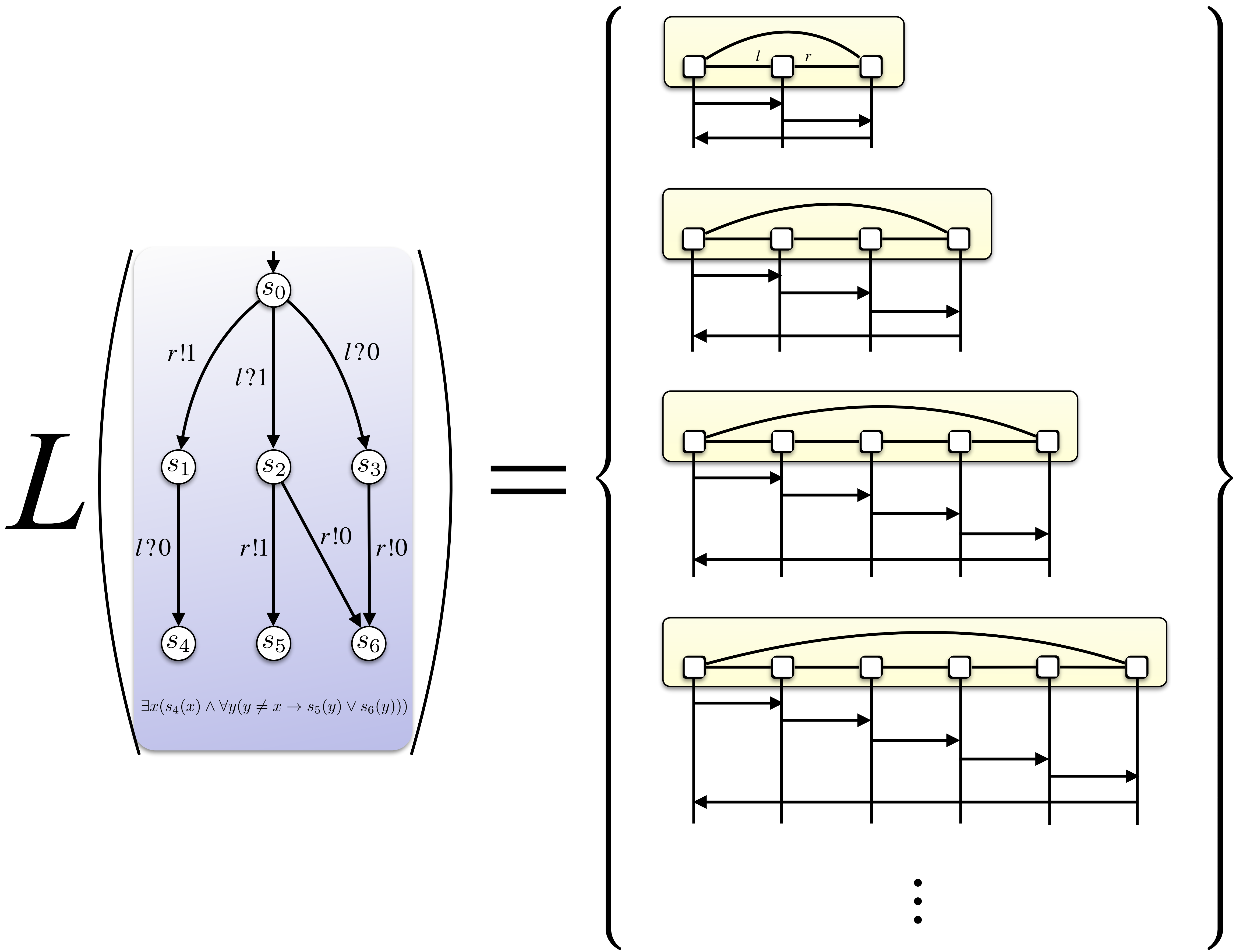
$=$



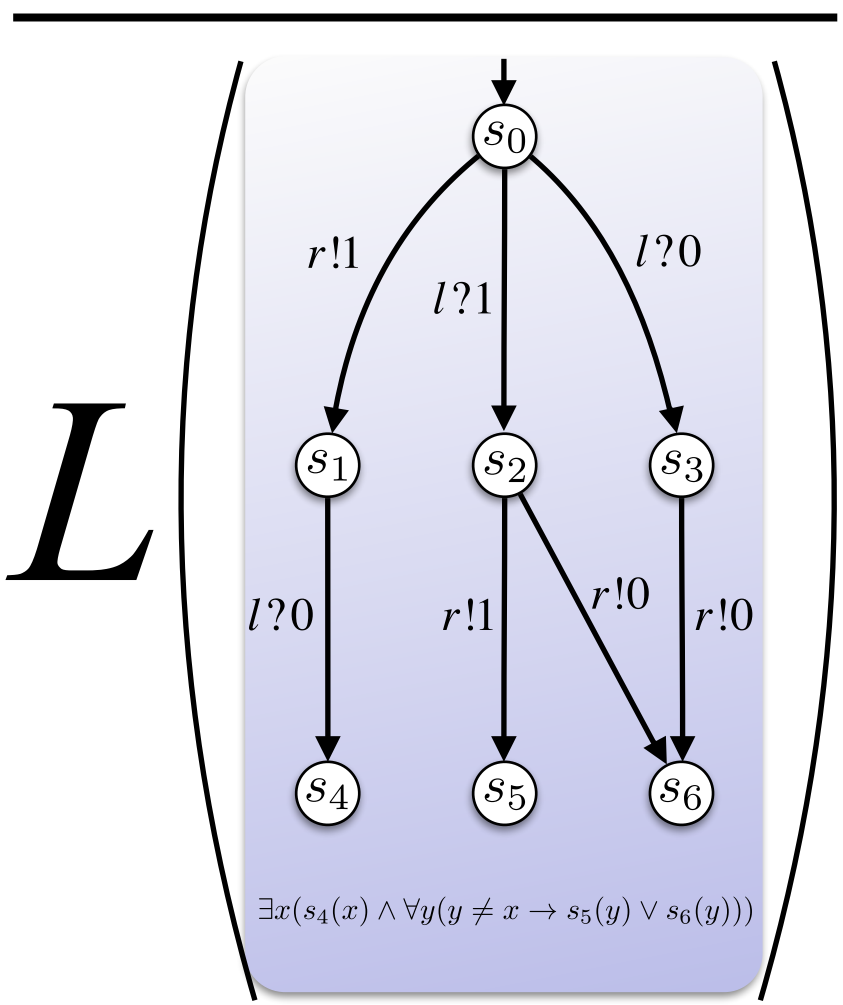
Parameterized Communicating Automata (PCA) over Rings



Parameterized Communicating Automata (PCA) over Rings

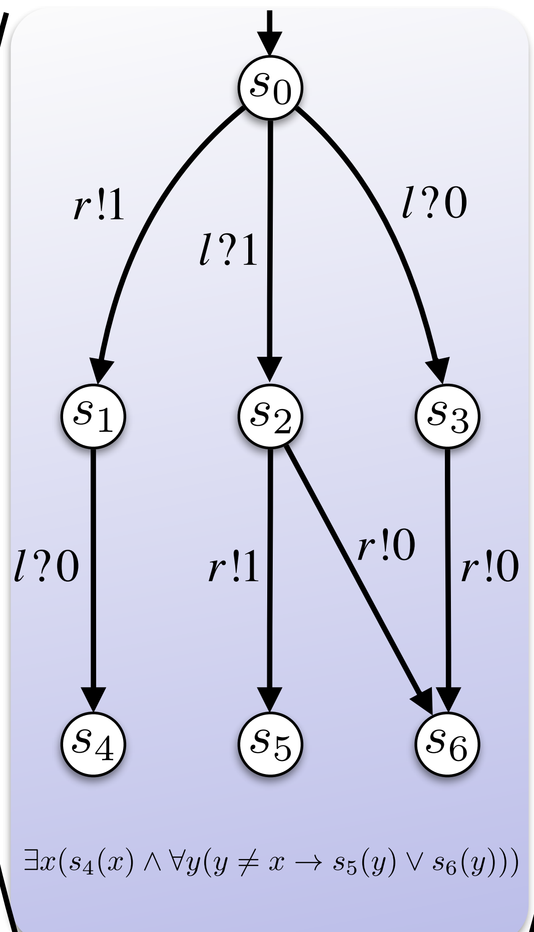


Complementation

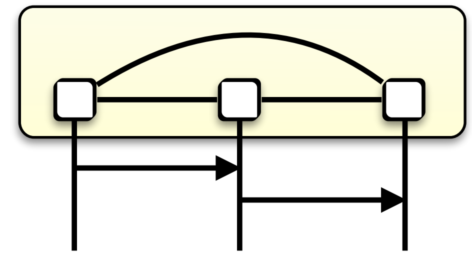


Complementation

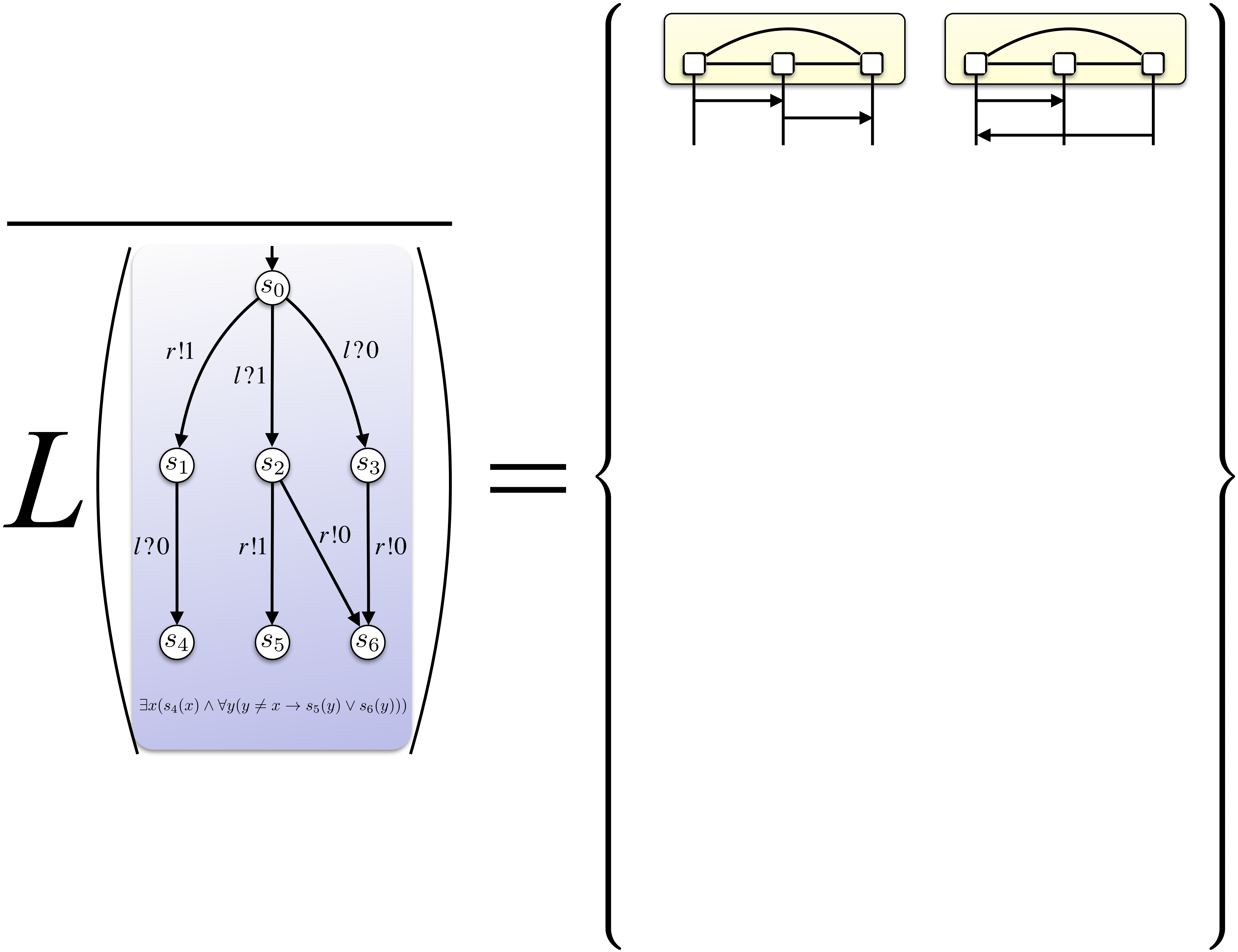
L



$=$

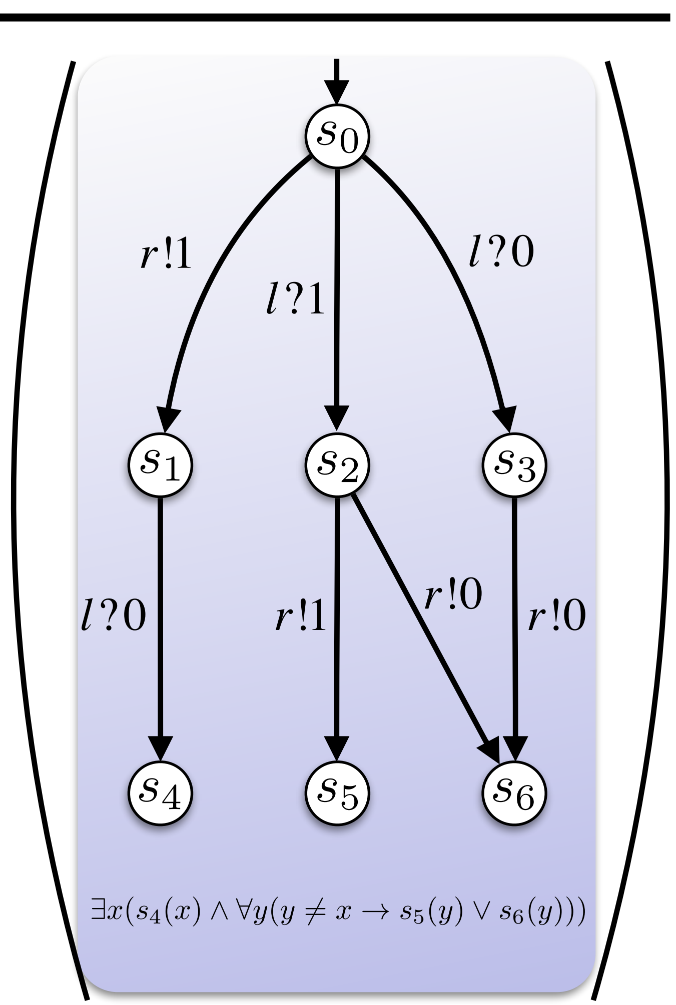


Complementation

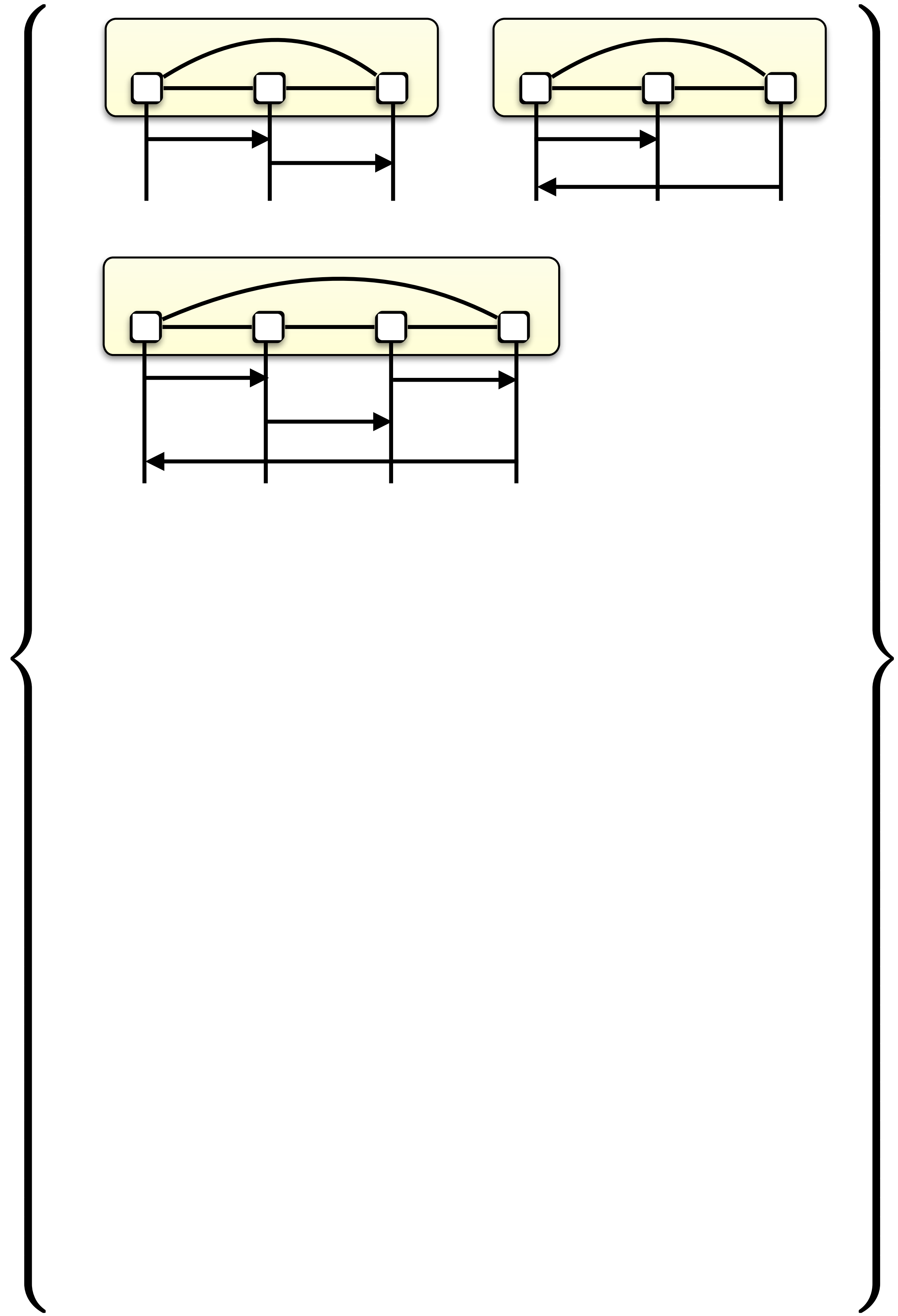


Complementation

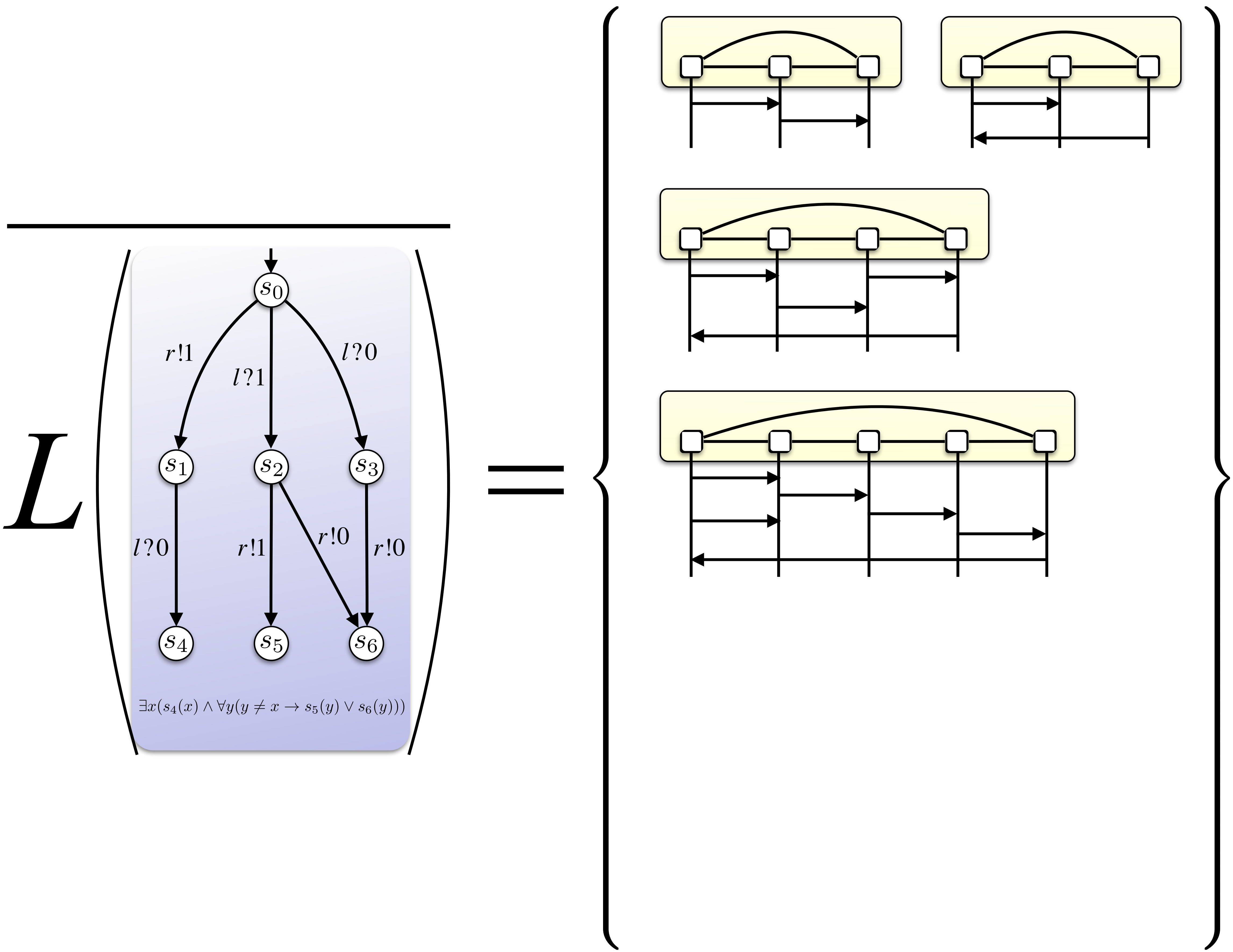
L



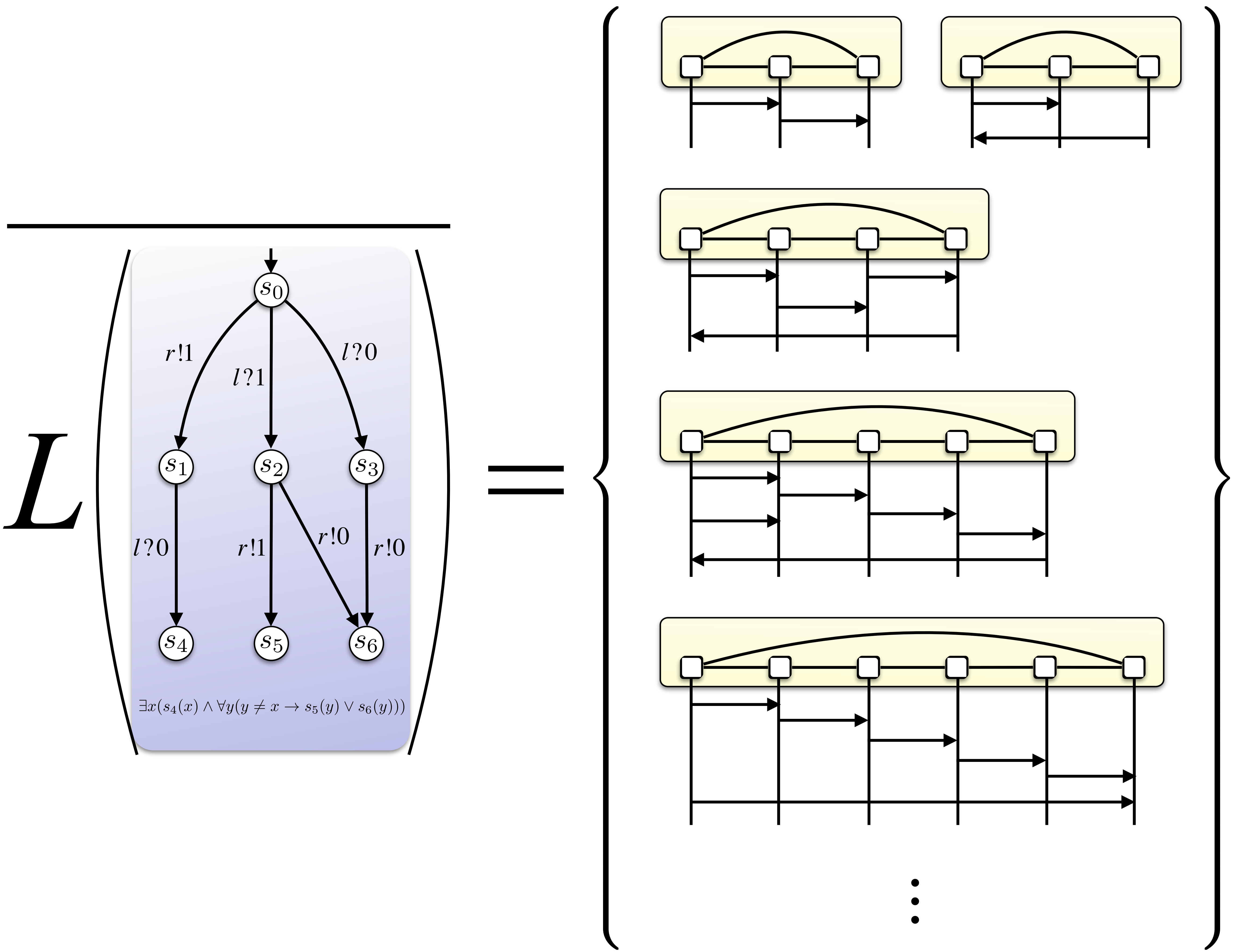
$=$



Complementation



Complementation



Negative Results

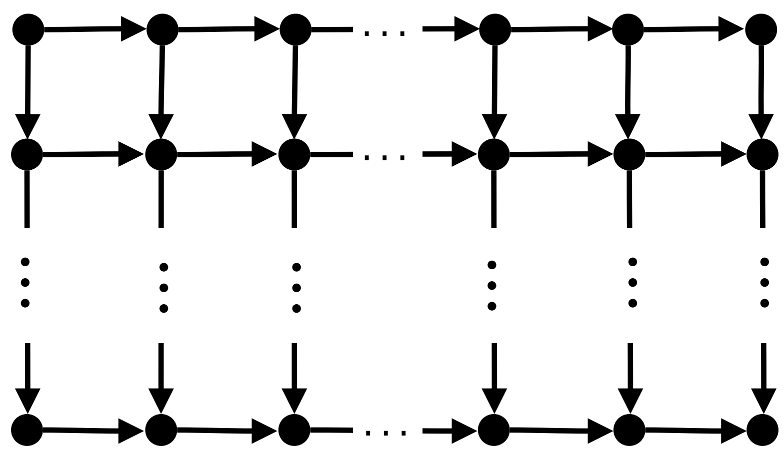
Theorem [B.-Gastin-Kumar; FSTTCS 2014]:

PCAs over rings are not complementable.

Negative Results

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

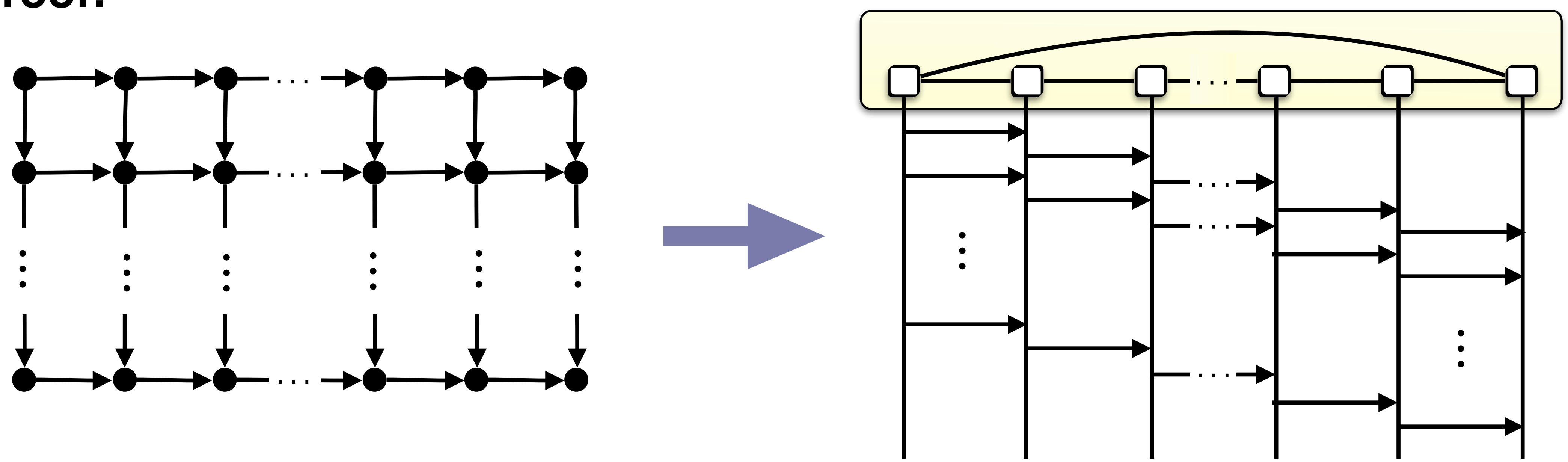
Proof:



Negative Results

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

Proof:

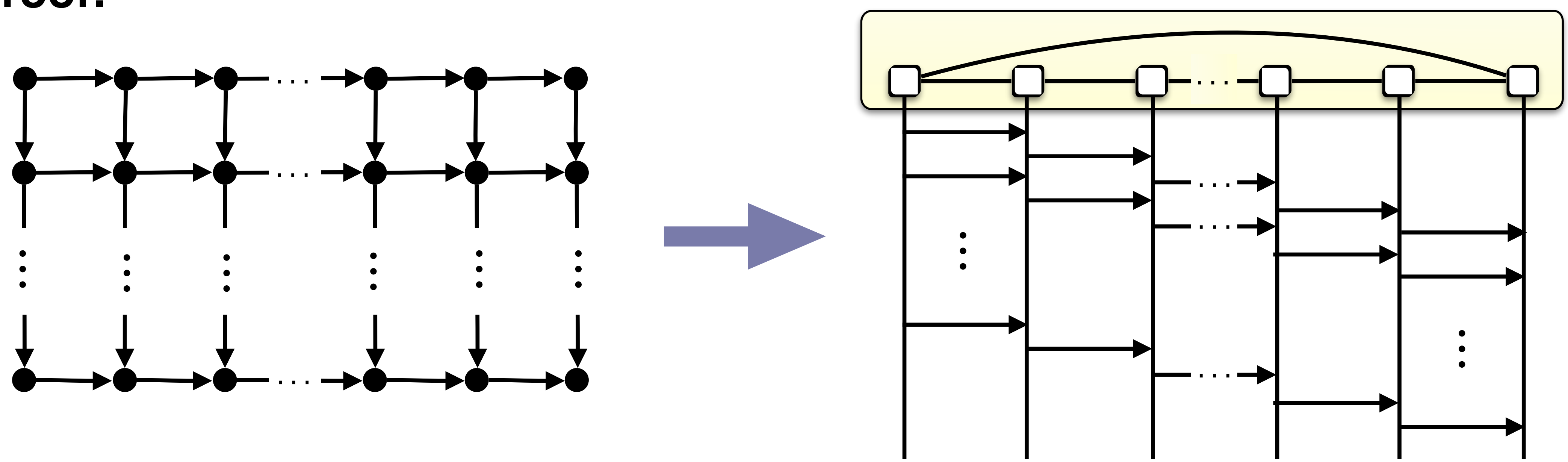


- Behaviors encode grids.

Negative Results

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

Proof:

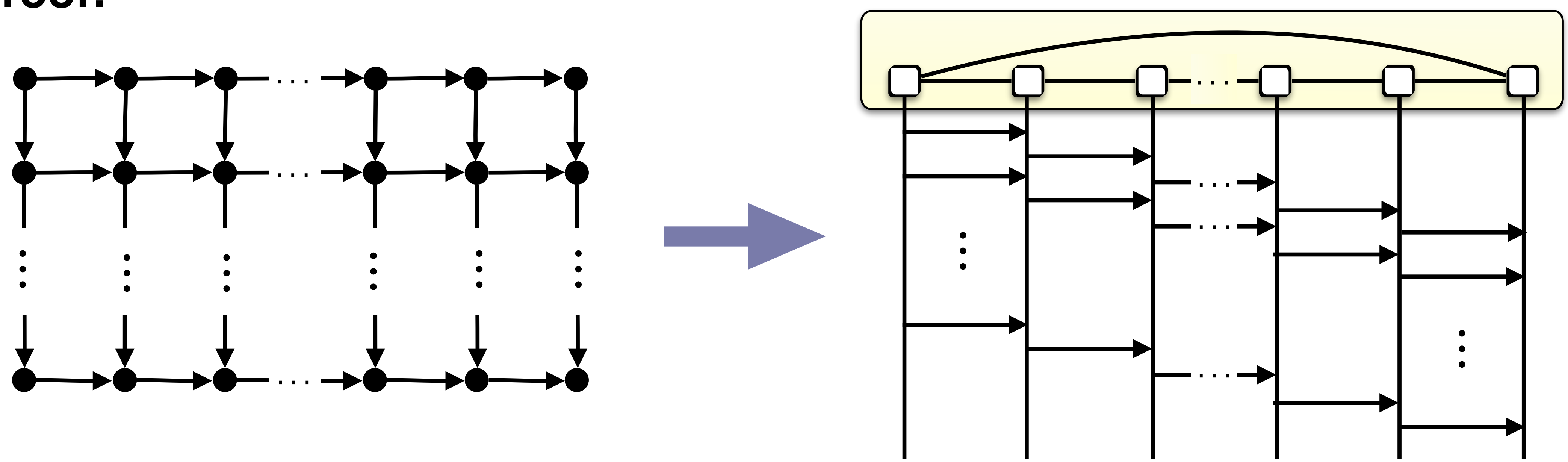


- Behaviors encode grids.
- Grid automata are not closed under complementation [Matz-Schweikardt-Thomas '02]. □

Negative Results

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

Proof:



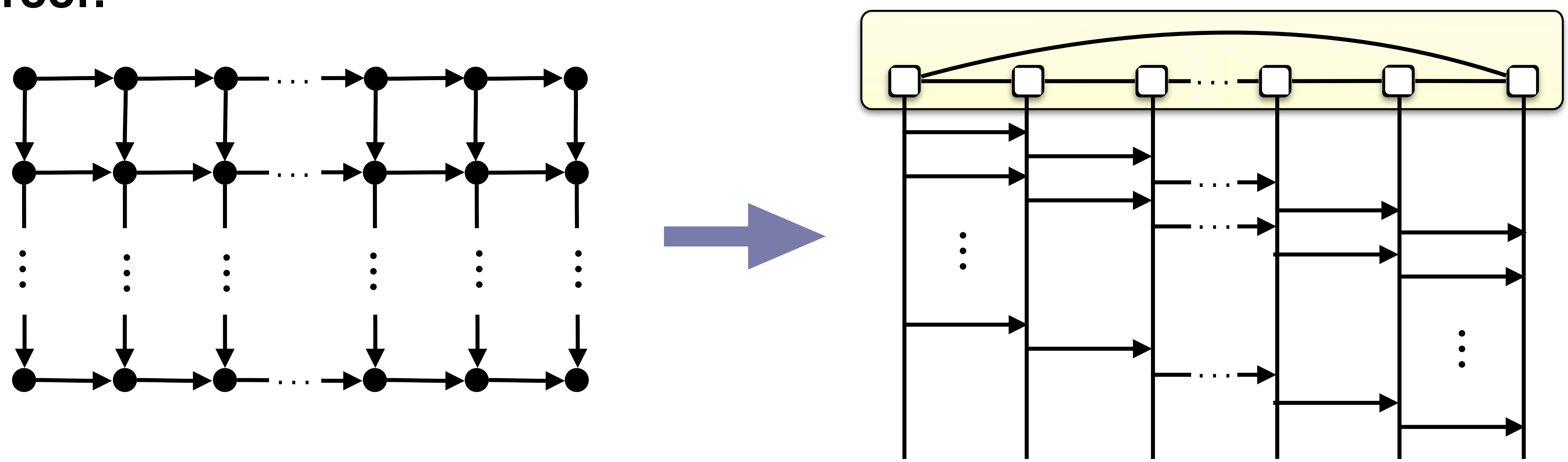
- Behaviors encode grids.
- Grid automata are not closed under complementation [Matz-Schweikardt-Thomas '02]. □

Theorem [Emerson-Namjoshi 2003]:
Emptiness is undecidable for PCAs over rings
(even token-passing systems, unless $|Msg| = 1$).

Negative Results

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
PCAs over rings are not complementable.

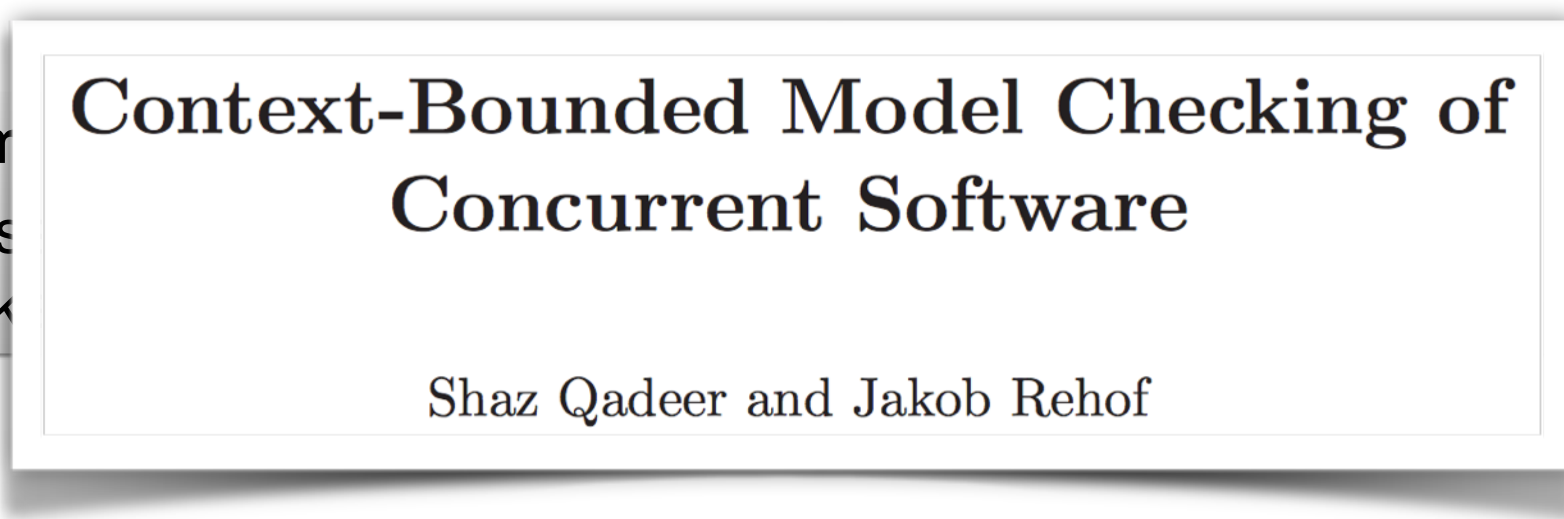
Proof:



- Behaviors encode grids.
- Grid automata are not closed under complementation [Matz-Schweikardt-Thomas '02].

□

Theorem
Emptines
(even tok



Context-Bounded Model Checking of Concurrent Software

Shaz Qadeer and Jakob Rehof

Context-Bounded PCAs

Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.

Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

Context-Bounded PCAs

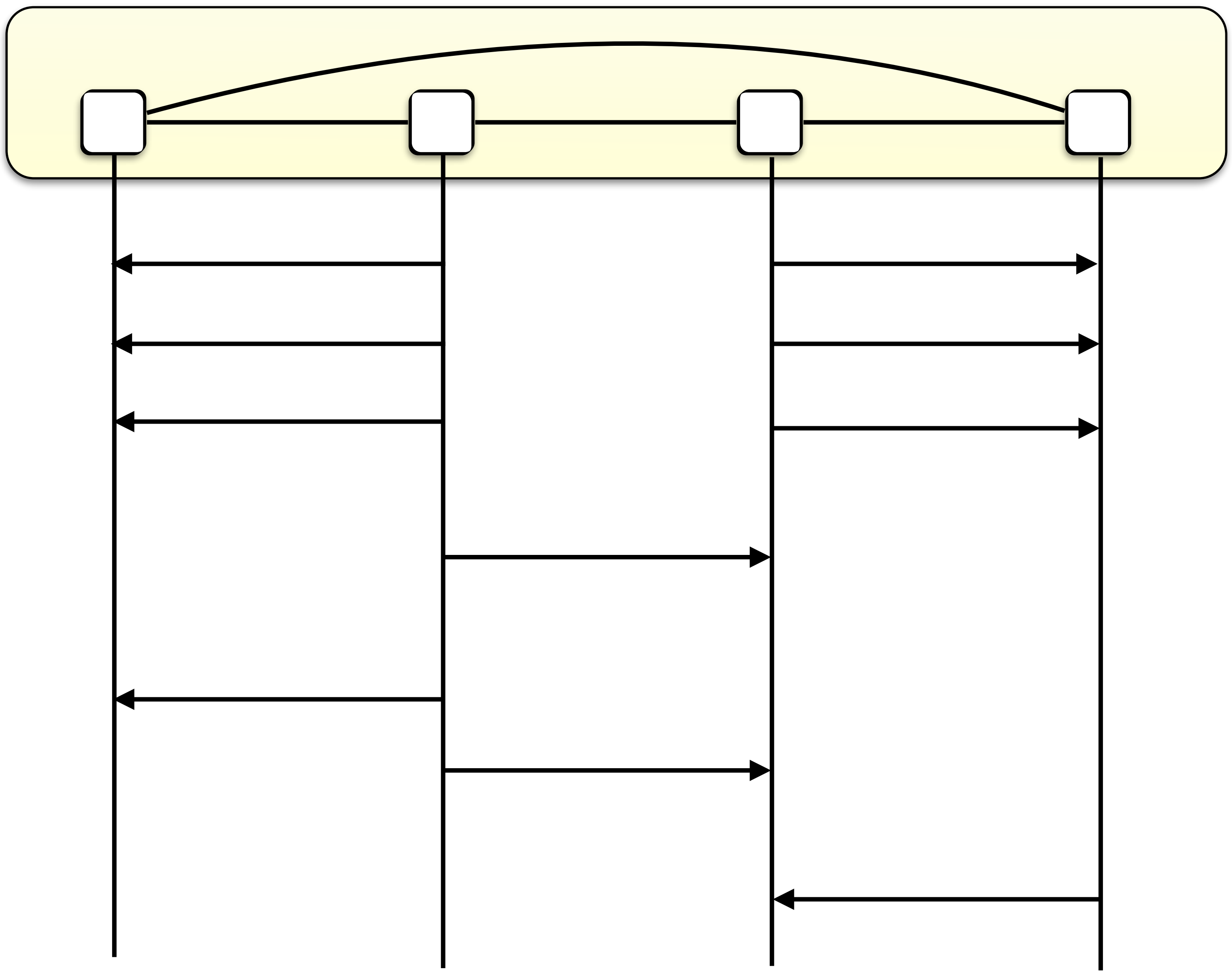
Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.

Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

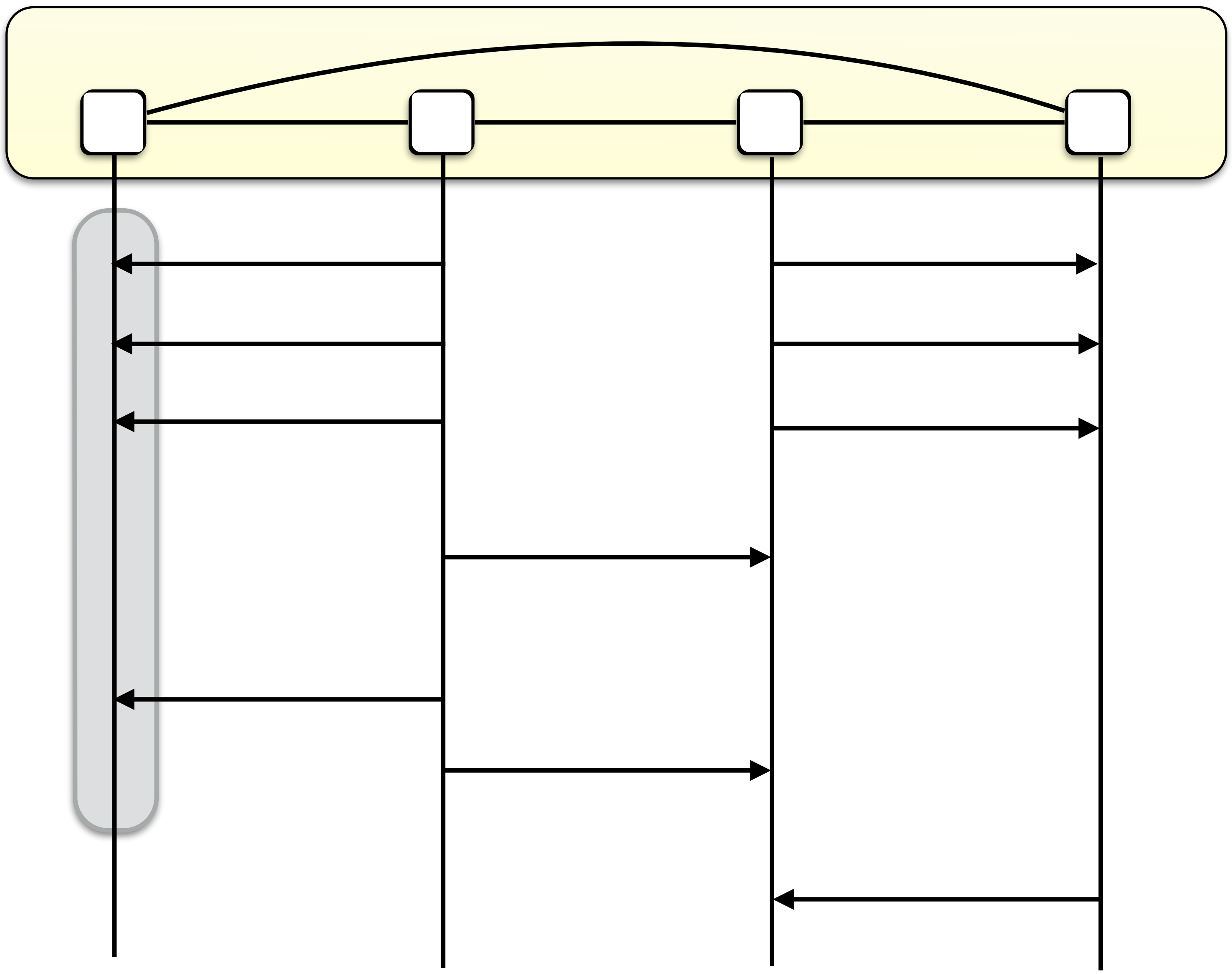
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

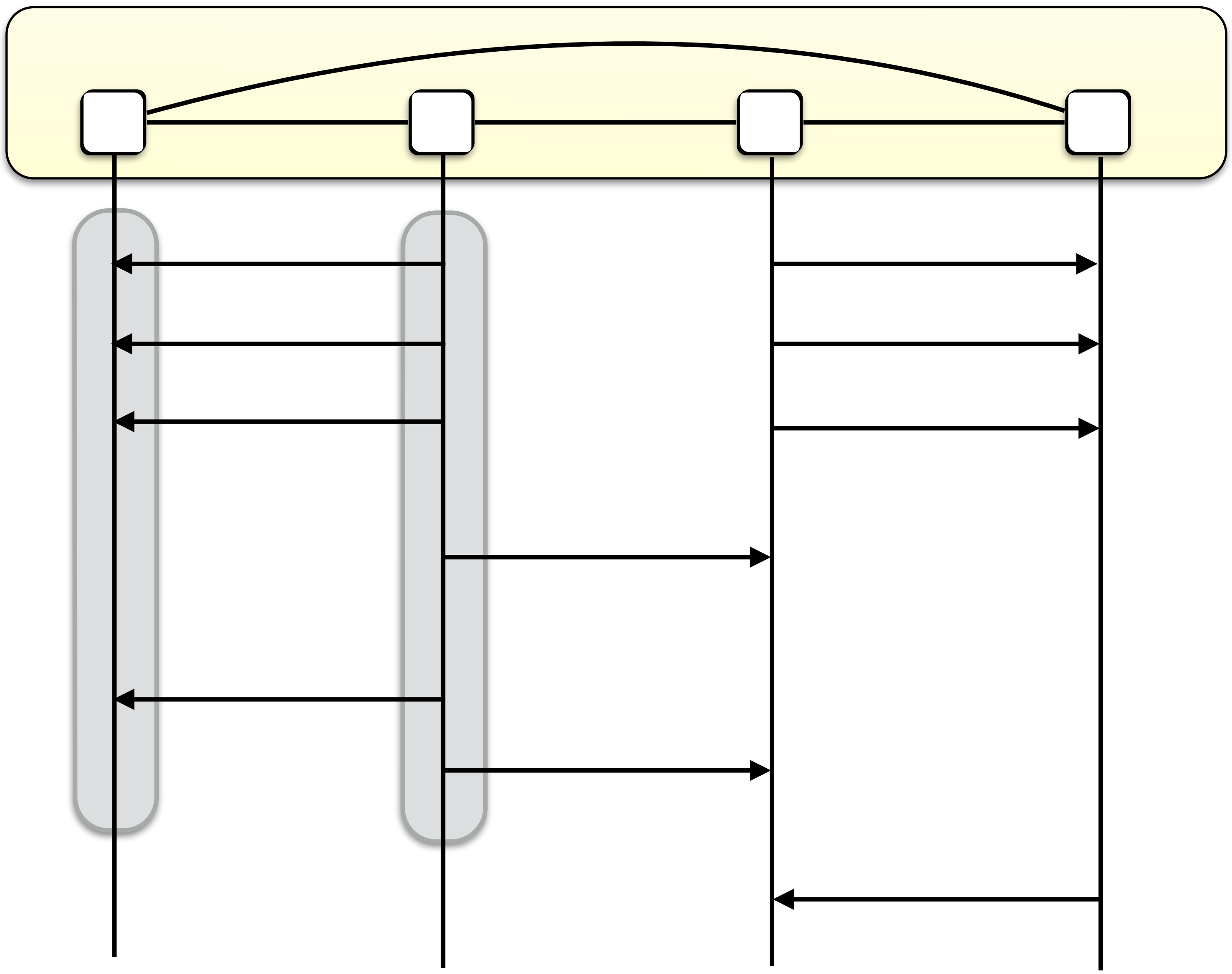
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

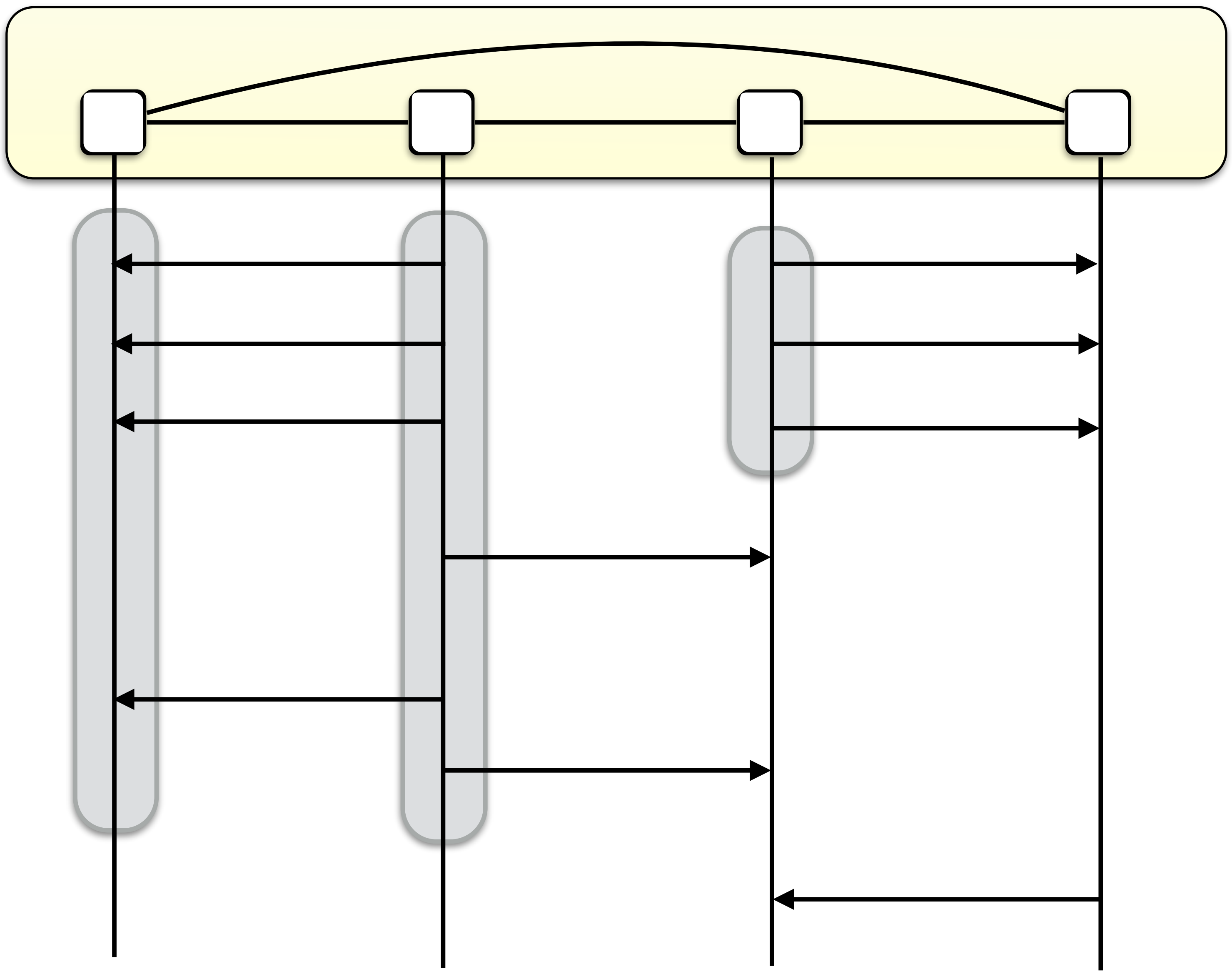
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

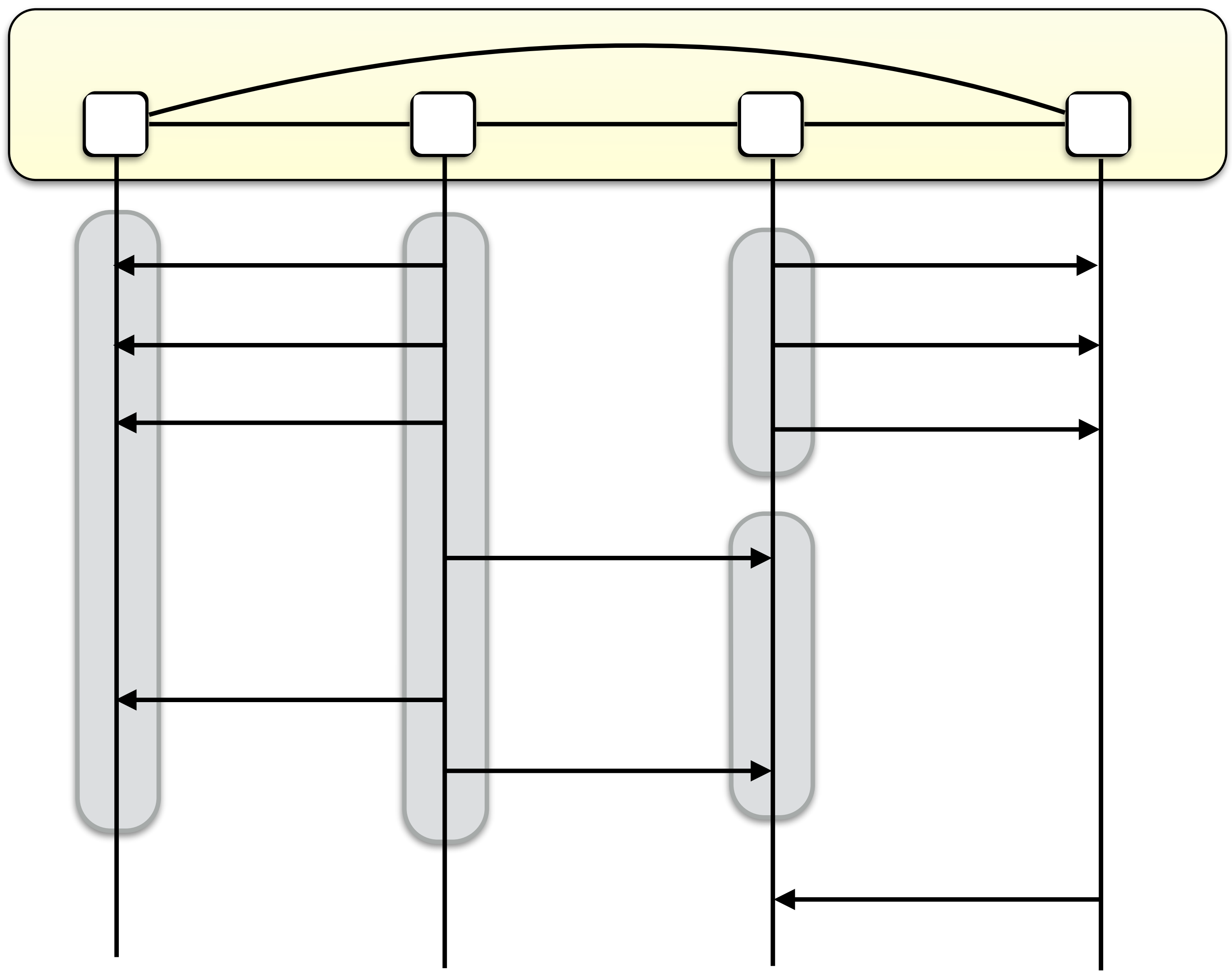
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

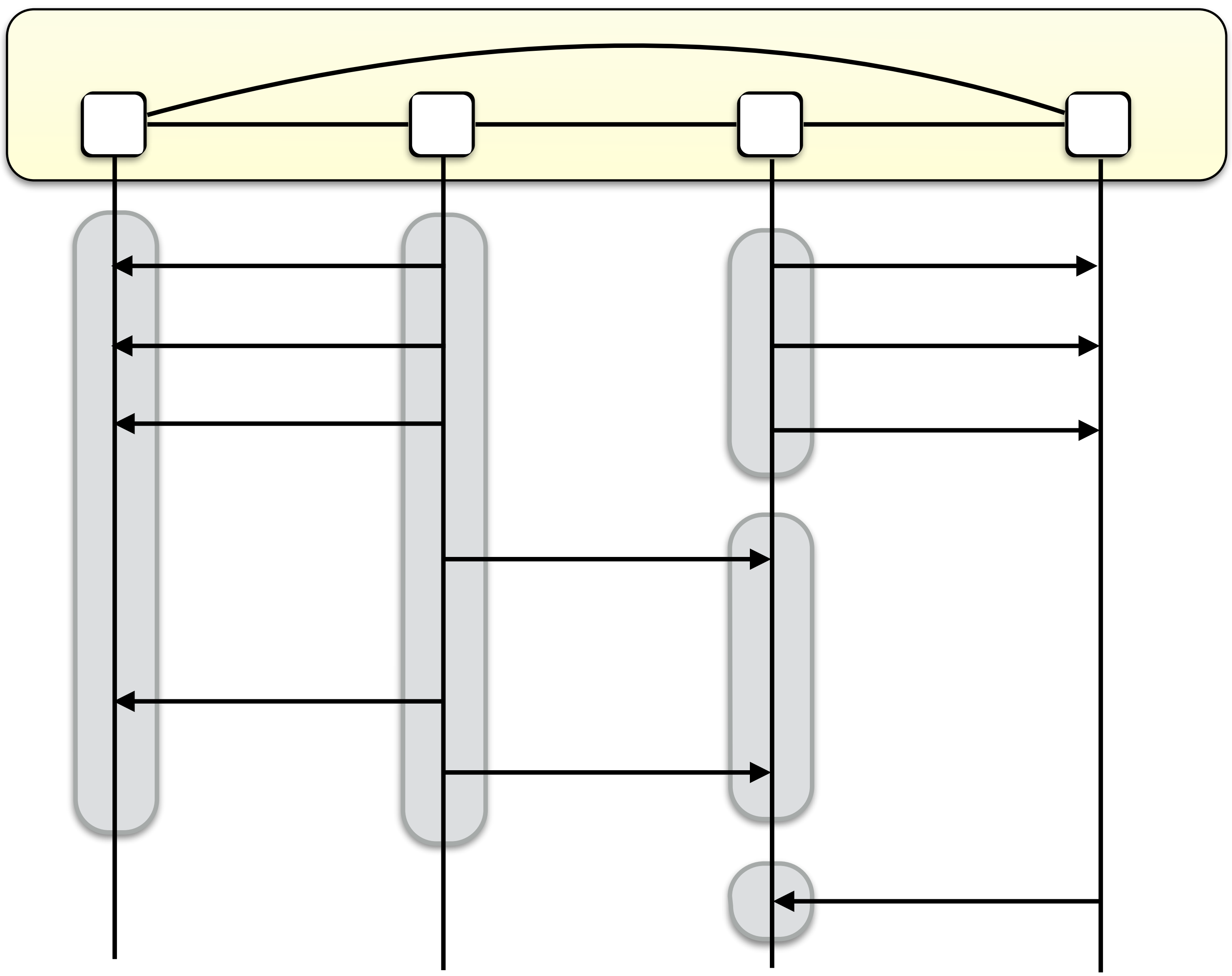
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

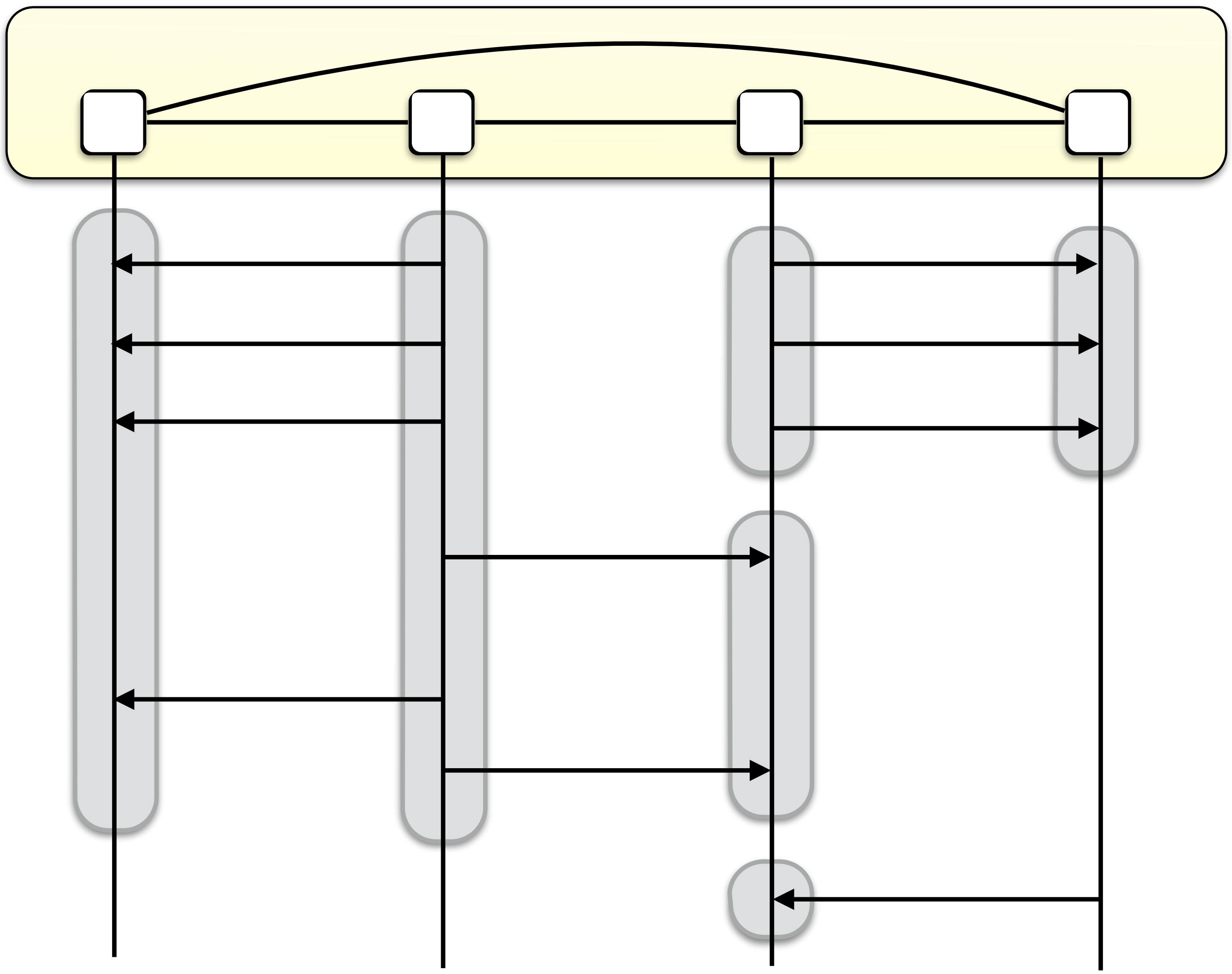
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

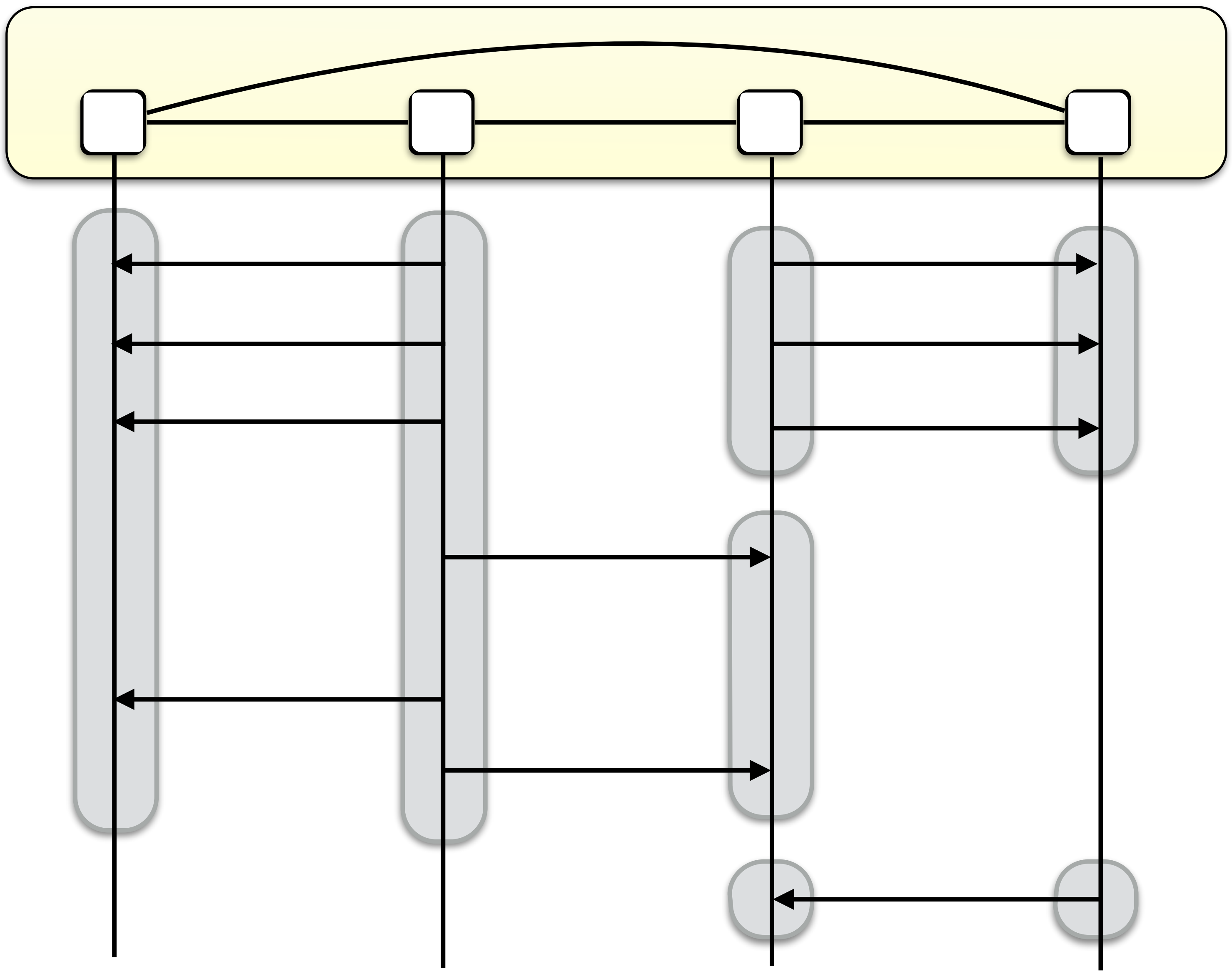
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

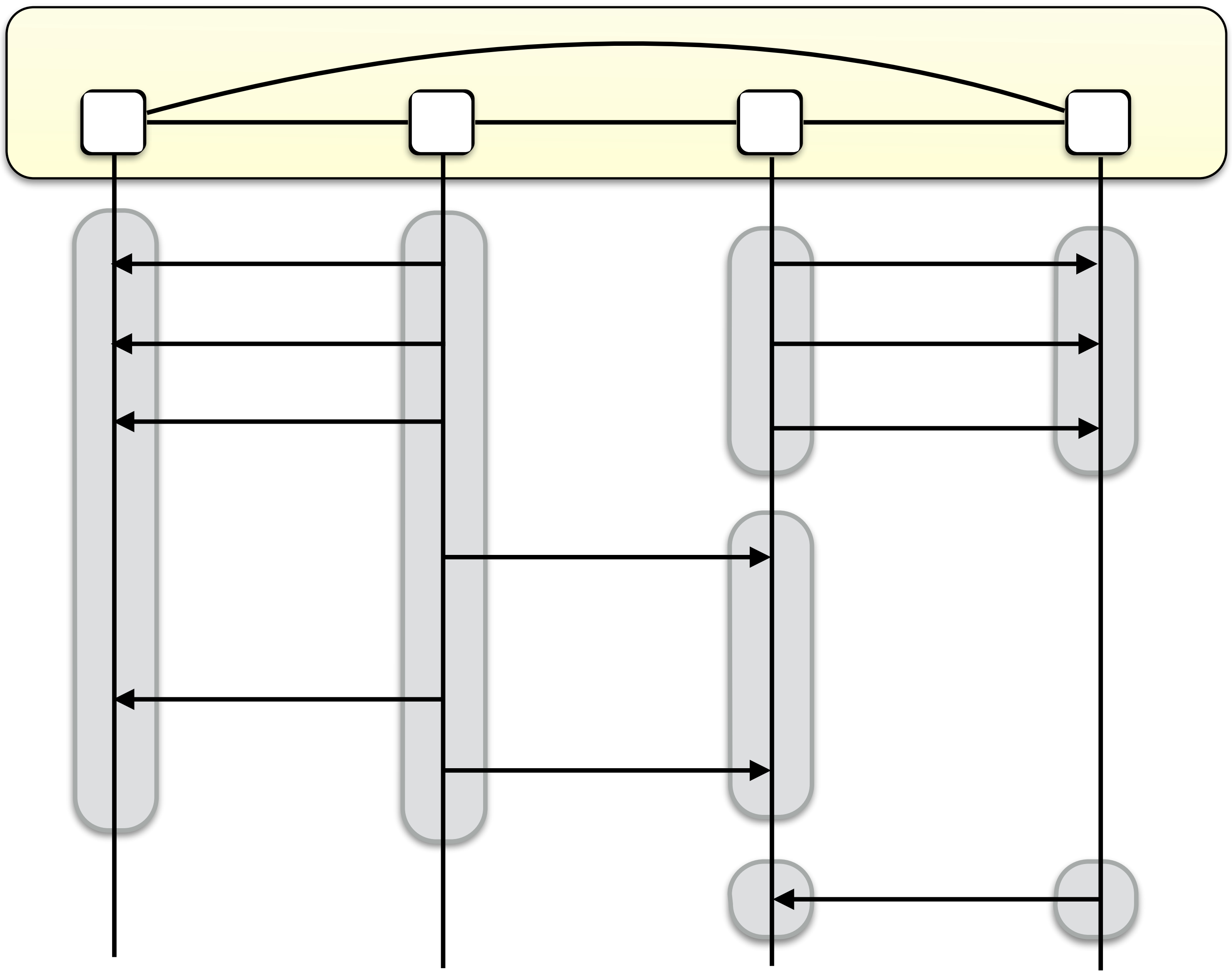
Here: Process only sends XOR only receives from one fixed neighbor.



Context-Bounded PCAs

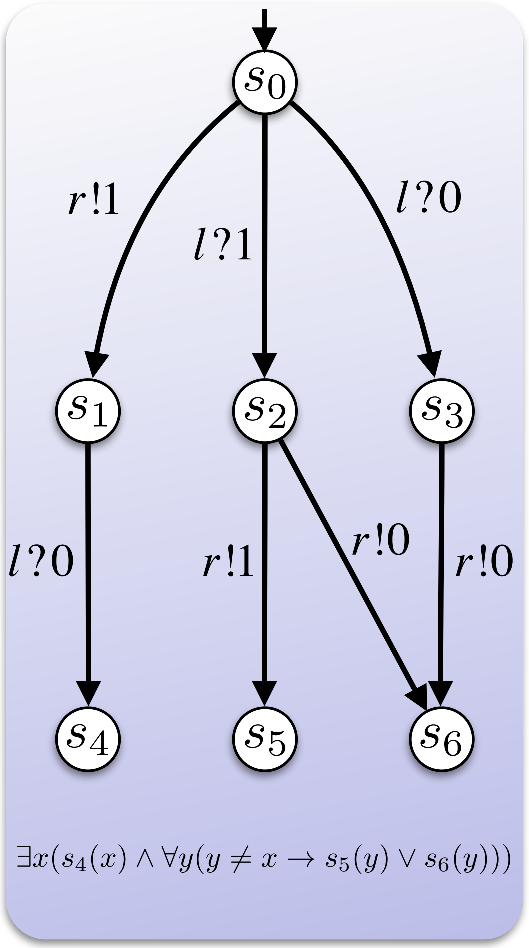
Idea: Every process is contrained to a bounded number of contexts.
There are several possible definitions of a context that lead to positive results.

Here: Process only sends XOR only receives from one fixed neighbor.



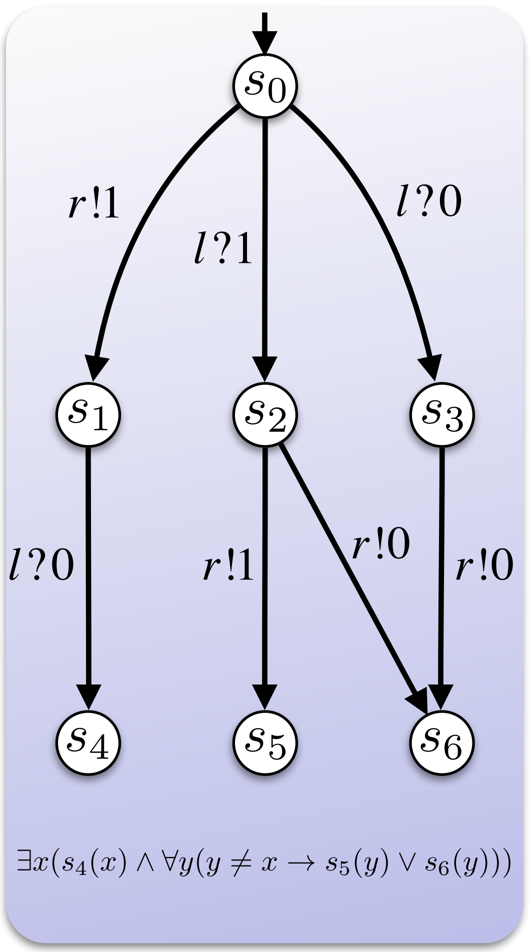
3-bounded

Context-Bounded PCAs



Definition: A PCA is k-bounded if the finite automaton restricts to k contexts.

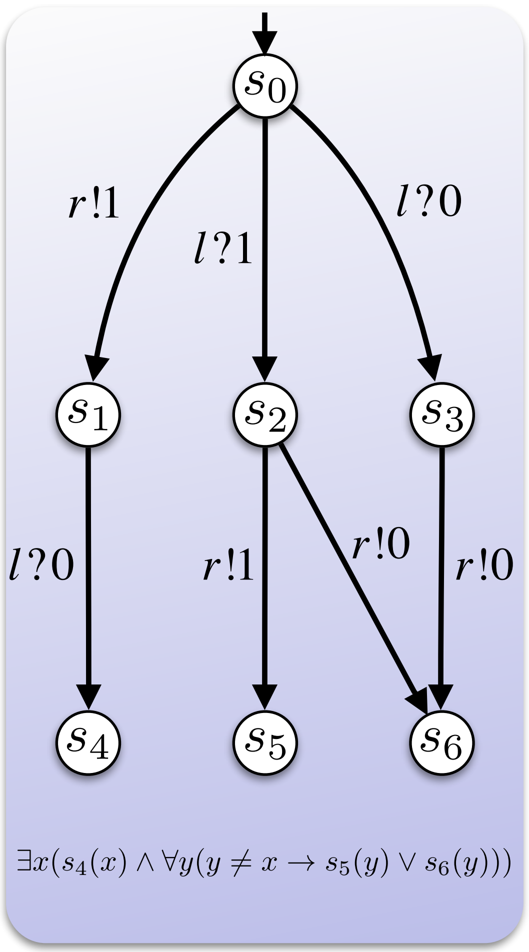
Context-Bounded PCAs



2-bounded PCA

Definition: A PCA is k-bounded if the finite automaton restricts to k contexts.

Context-Bounded PCAs



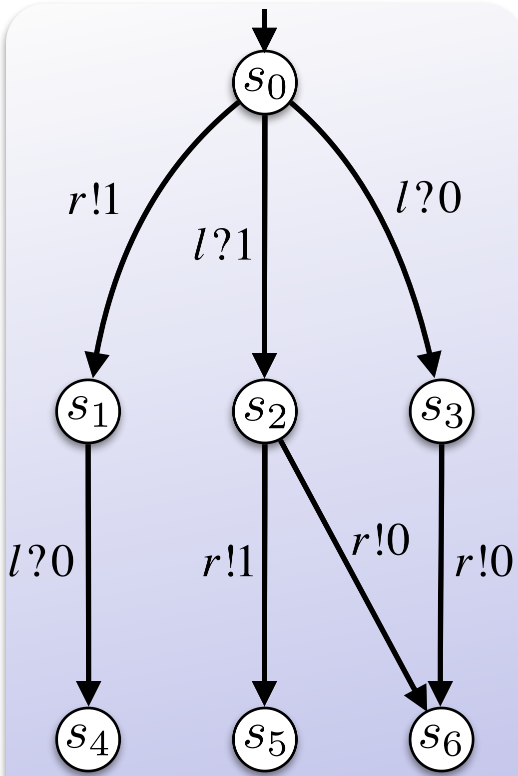
2-bounded PCA

Definition: A PCA is k-bounded if the finite automaton restricts to k contexts.

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
For every bounded PCA $\mathcal{A}$, there is a PCA $\mathcal{B}$ such that $L(\mathcal{B}) = \overline{L(\mathcal{A})}$.

Proof Outline

nondeterminism



$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$

k-bounded

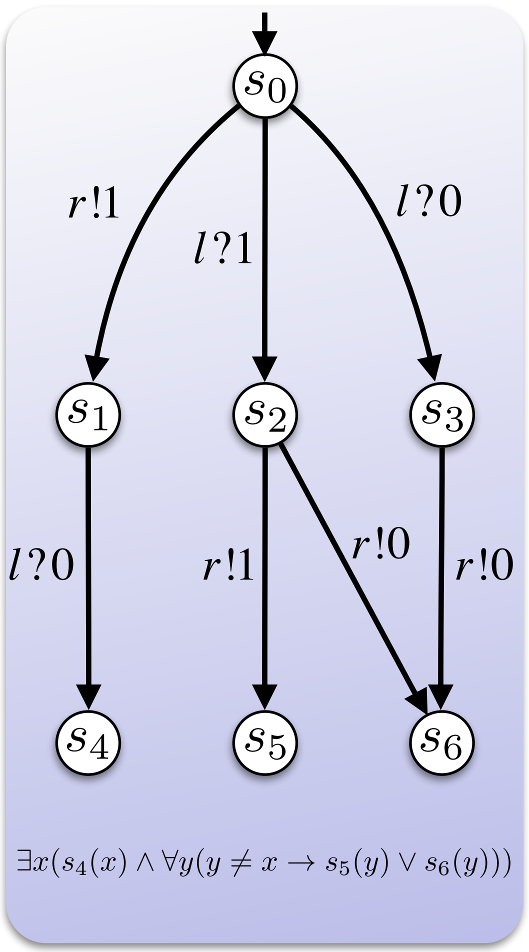
disambiguation

every behavior has a unique run

complementation

Proof Outline

nondeterminism



k-bounded

disambiguation

every behavior has a unique run

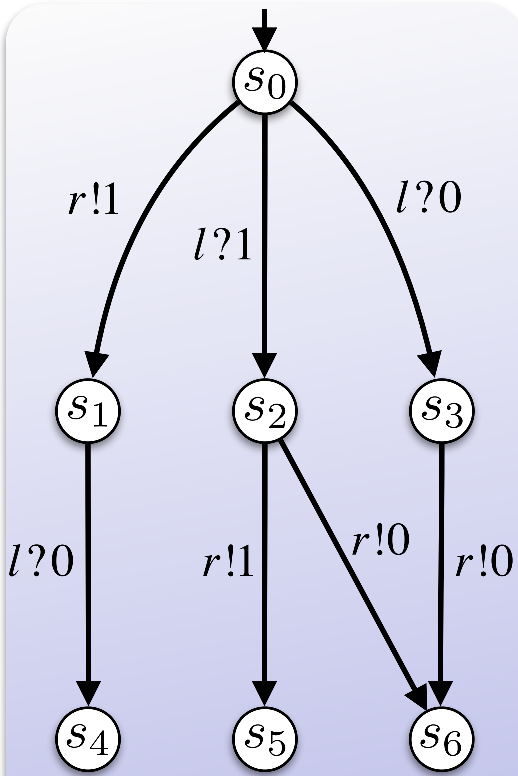
$\mathcal{A}$

φ

complementation

Proof Outline

nondeterminism



$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$

k-bounded

disambiguation

every behavior has a unique run

$\mathcal{A}$

φ

!

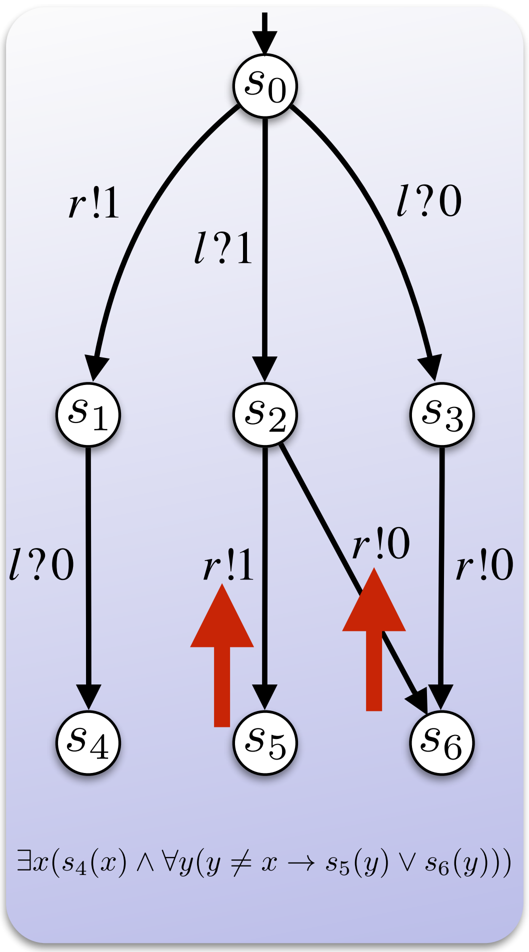
complementation

$\mathcal{A}$

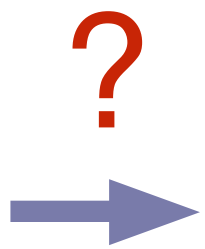
$\neg\varphi$

Proof Outline

nondeterminism

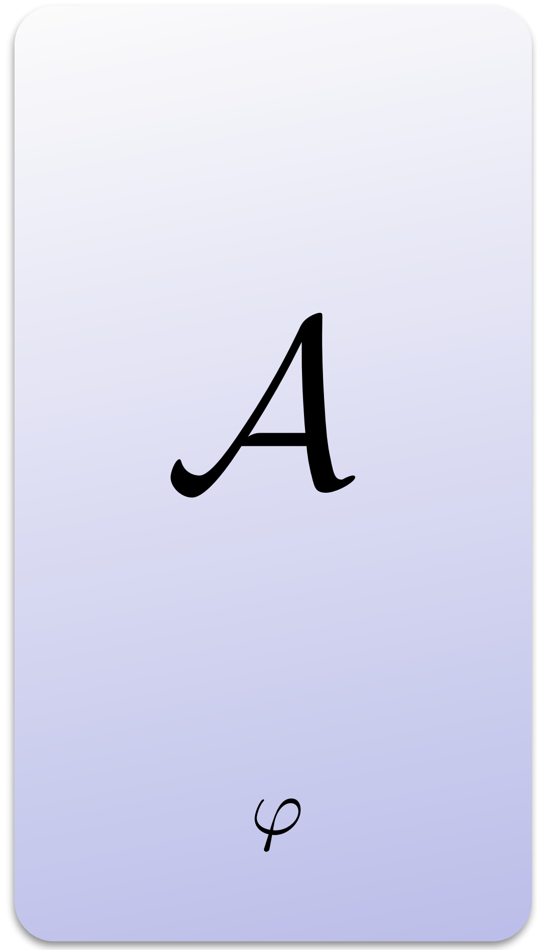


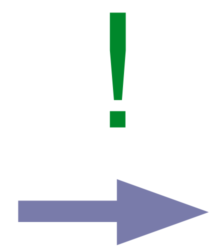
k-bounded



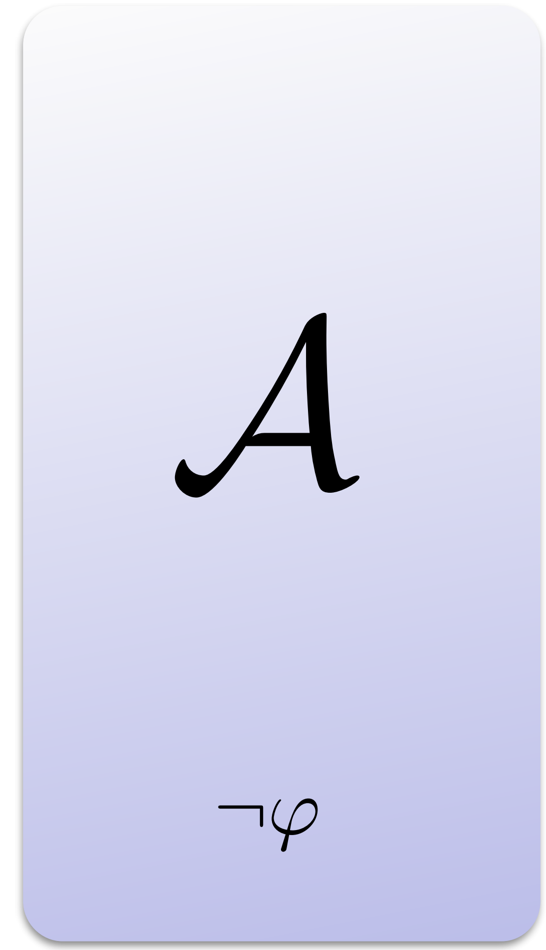
disambiguation

every behavior has a unique run



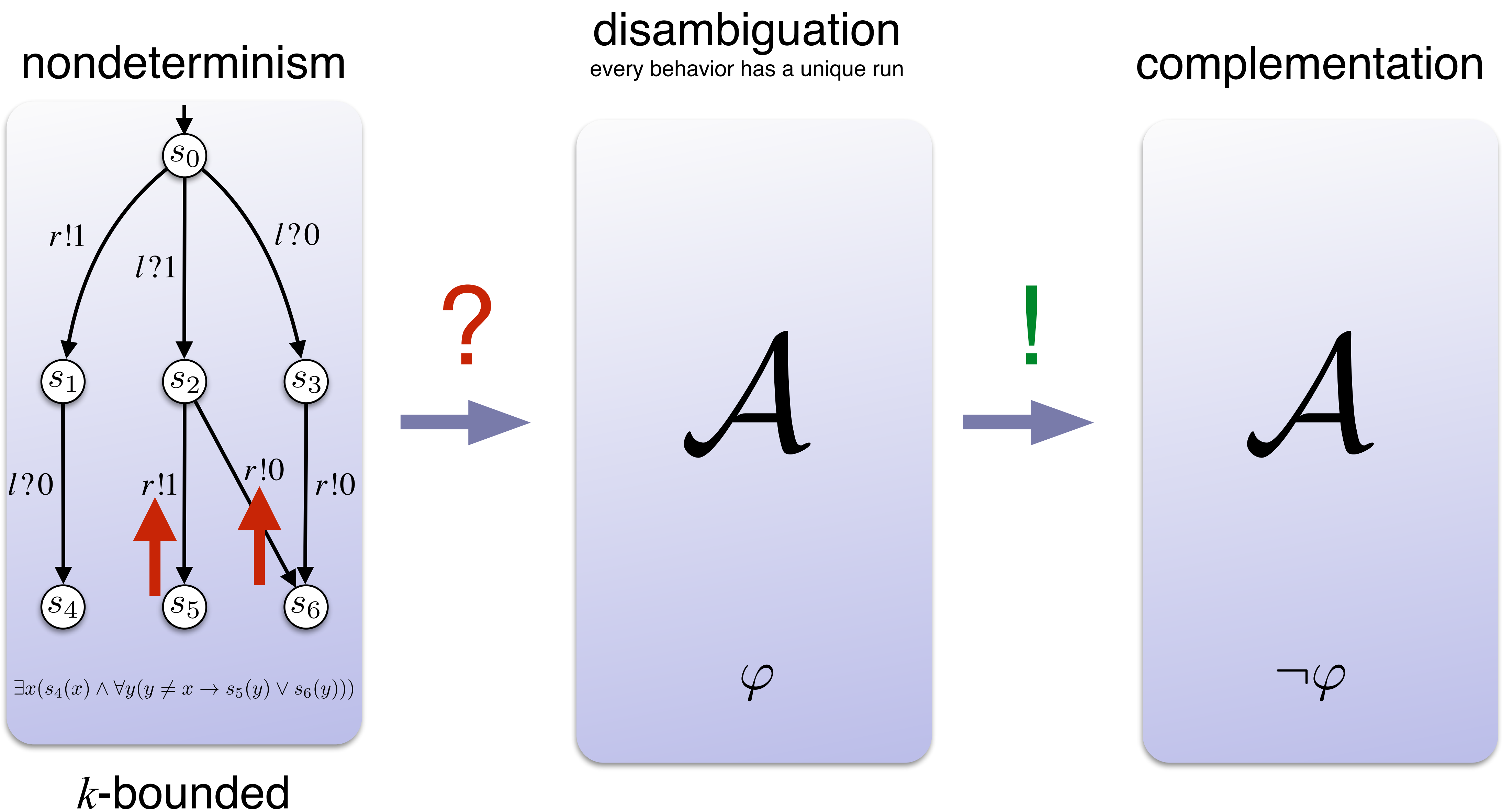


complementation



Powerset construction not applicable due to message contents.

Proof Outline

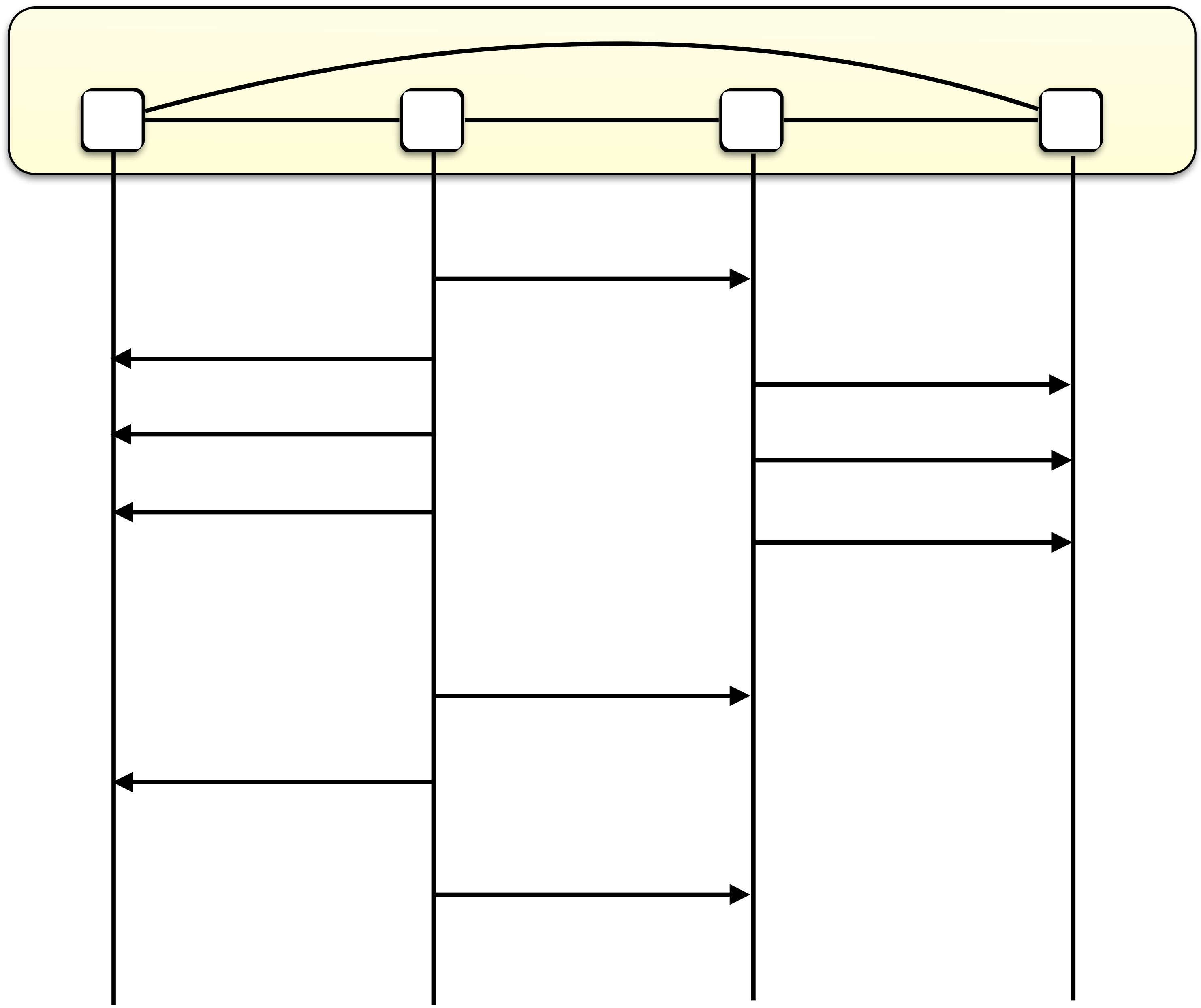


Powerset construction not applicable due to message contents.

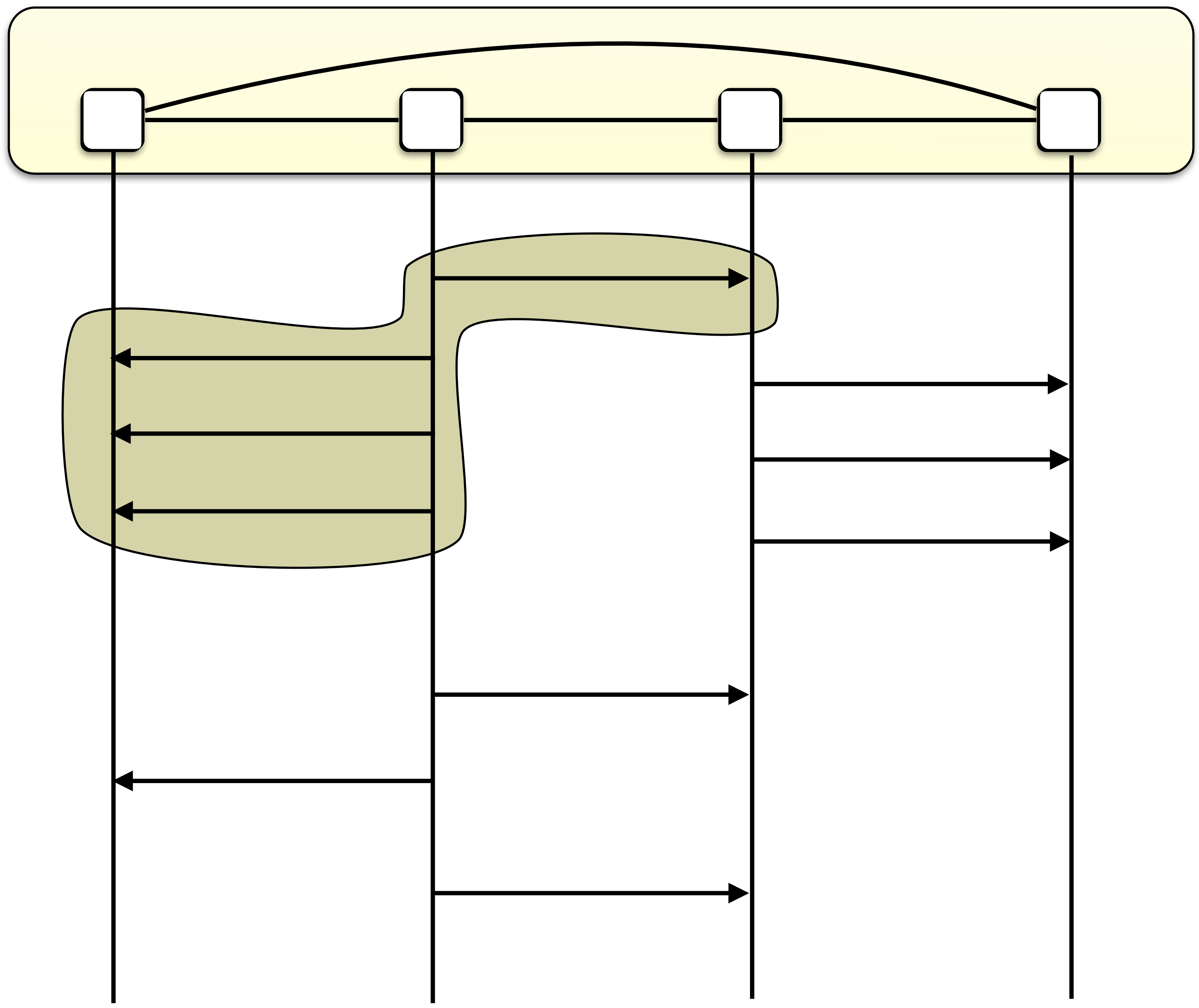
Disambiguation through summaries:

- Alur-Madhusudan: *Visibly pushdown languages.* STOC 2004.
- La Torre-Madhusudan-Parlato: *The language theory of bounded context switching.* LATIN 2010.
- La Torre-Napoli-Parlato: *Scope-bounded pushdown languages.* DLT 2014.

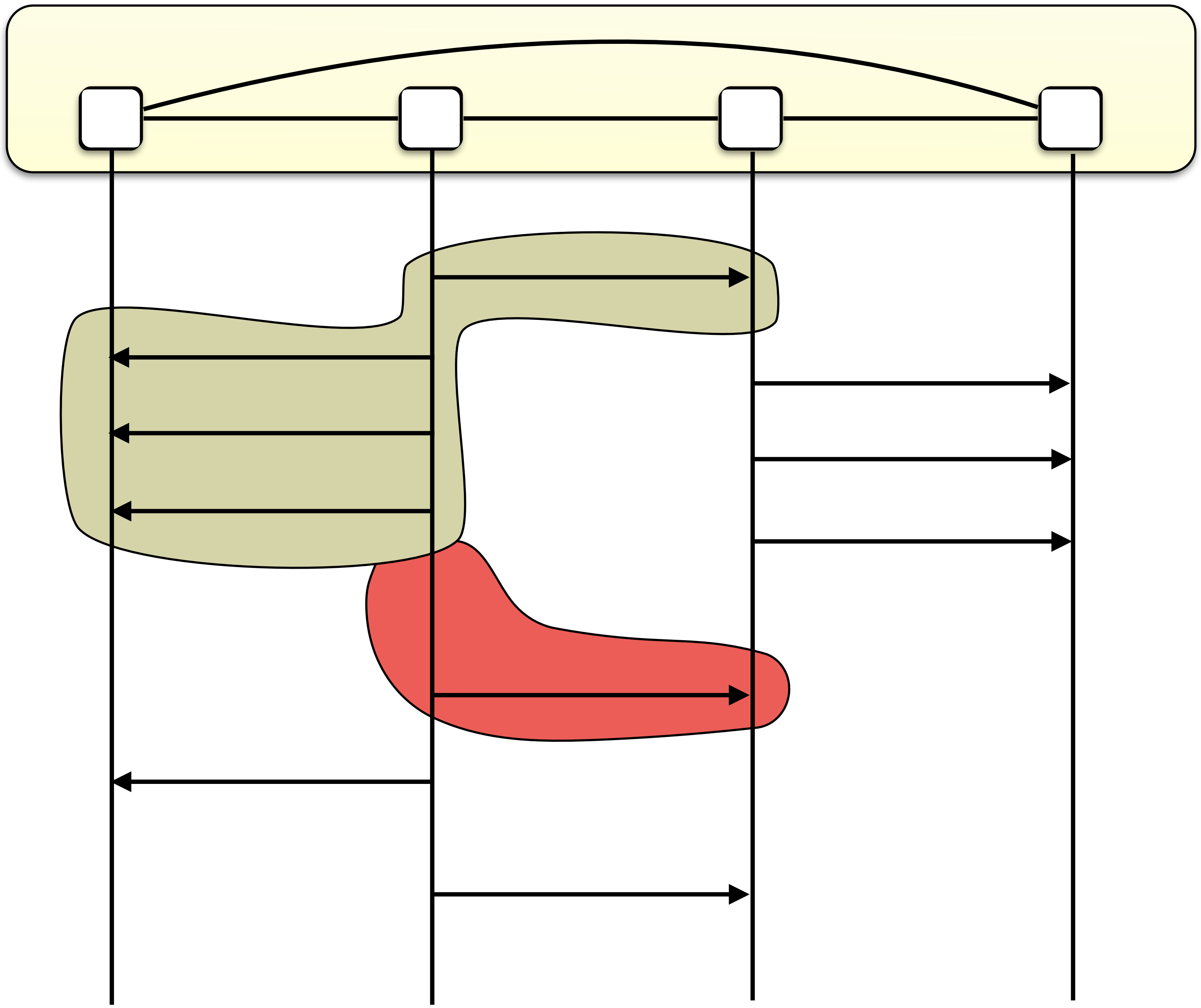
Disambiguation of context-bounded PCAs



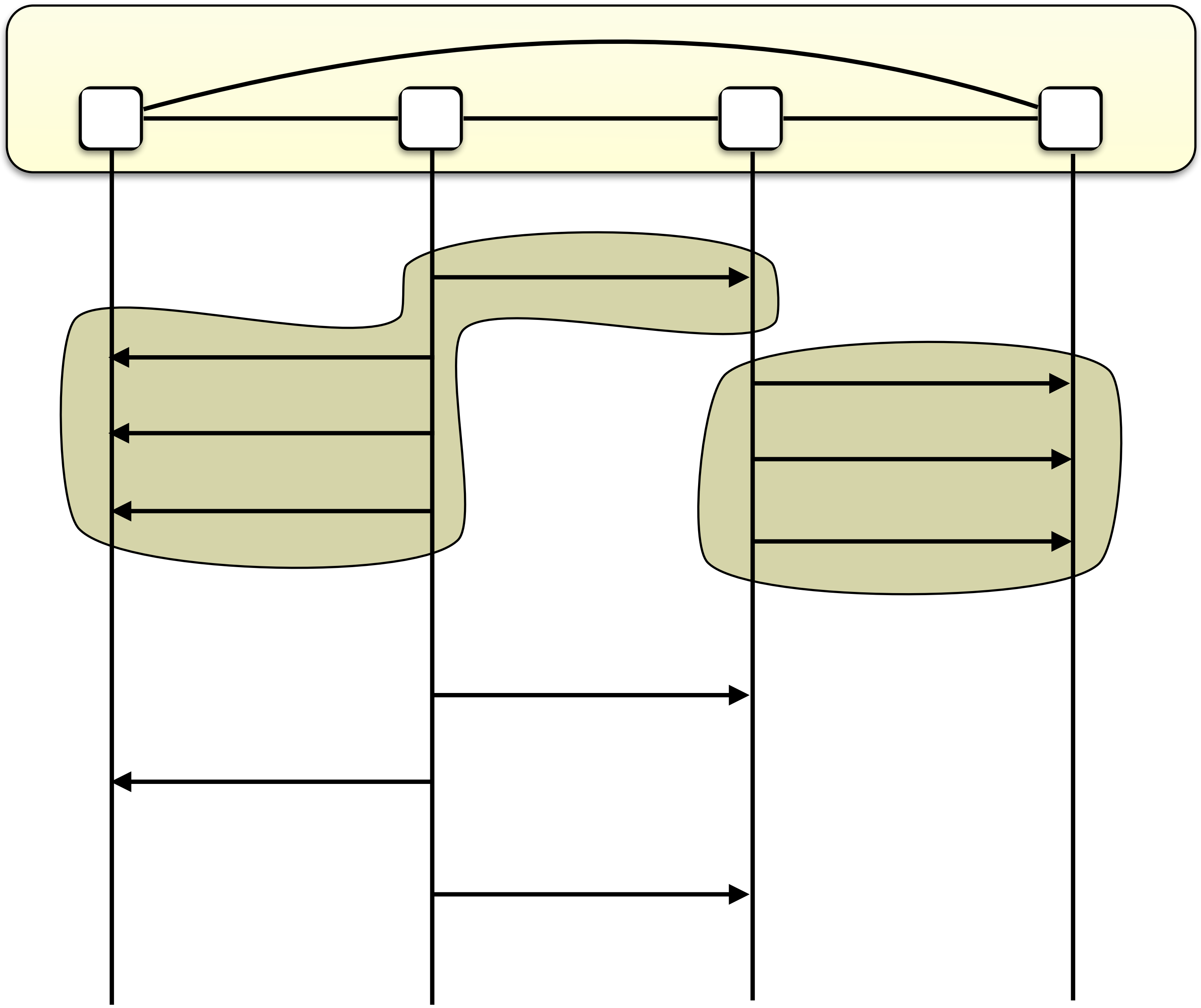
Disambiguation of context-bounded PCAs



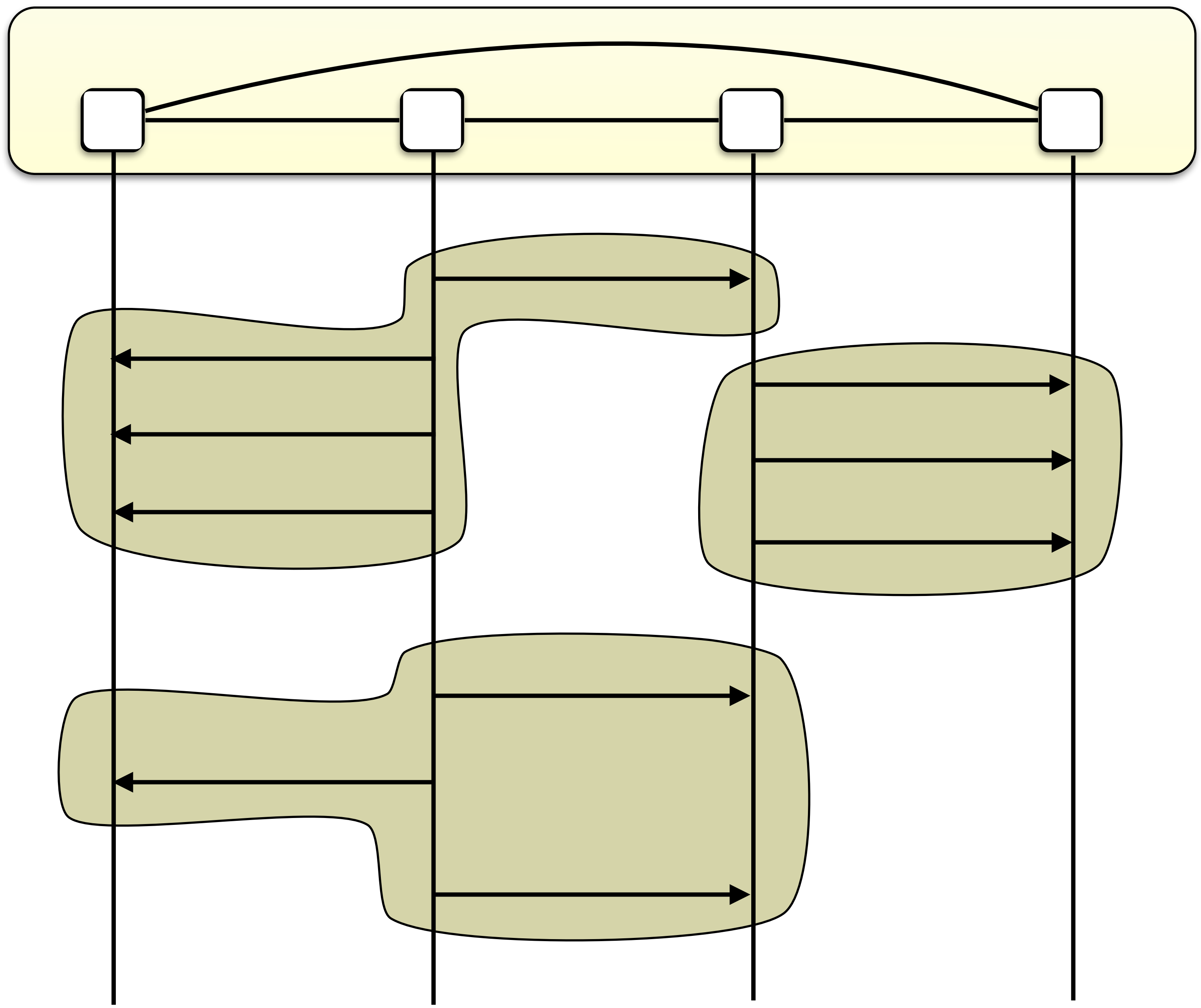
Disambiguation of context-bounded PCAs



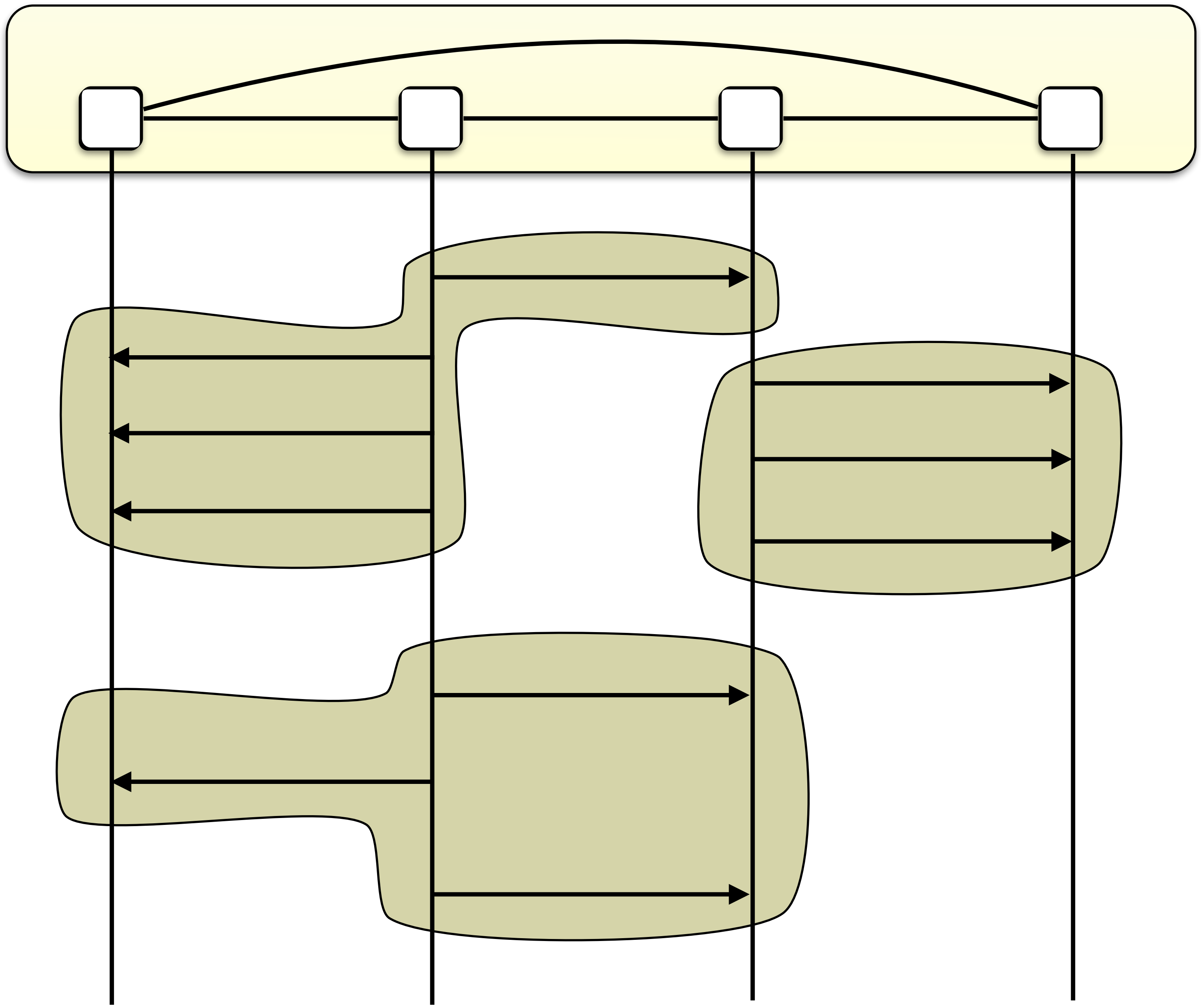
Disambiguation of context-bounded PCAs



Disambiguation of context-bounded PCAs

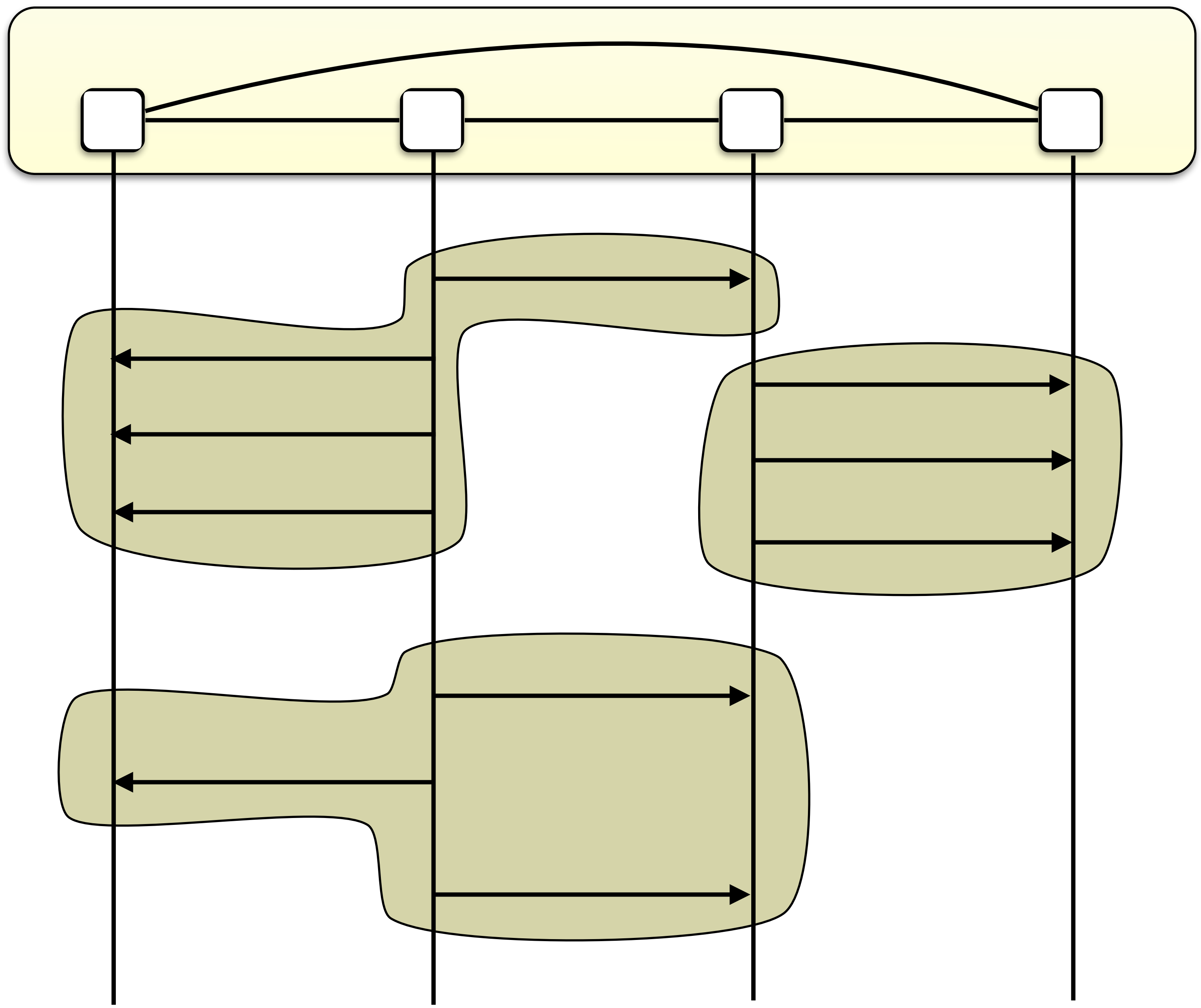


Disambiguation of context-bounded PCAs



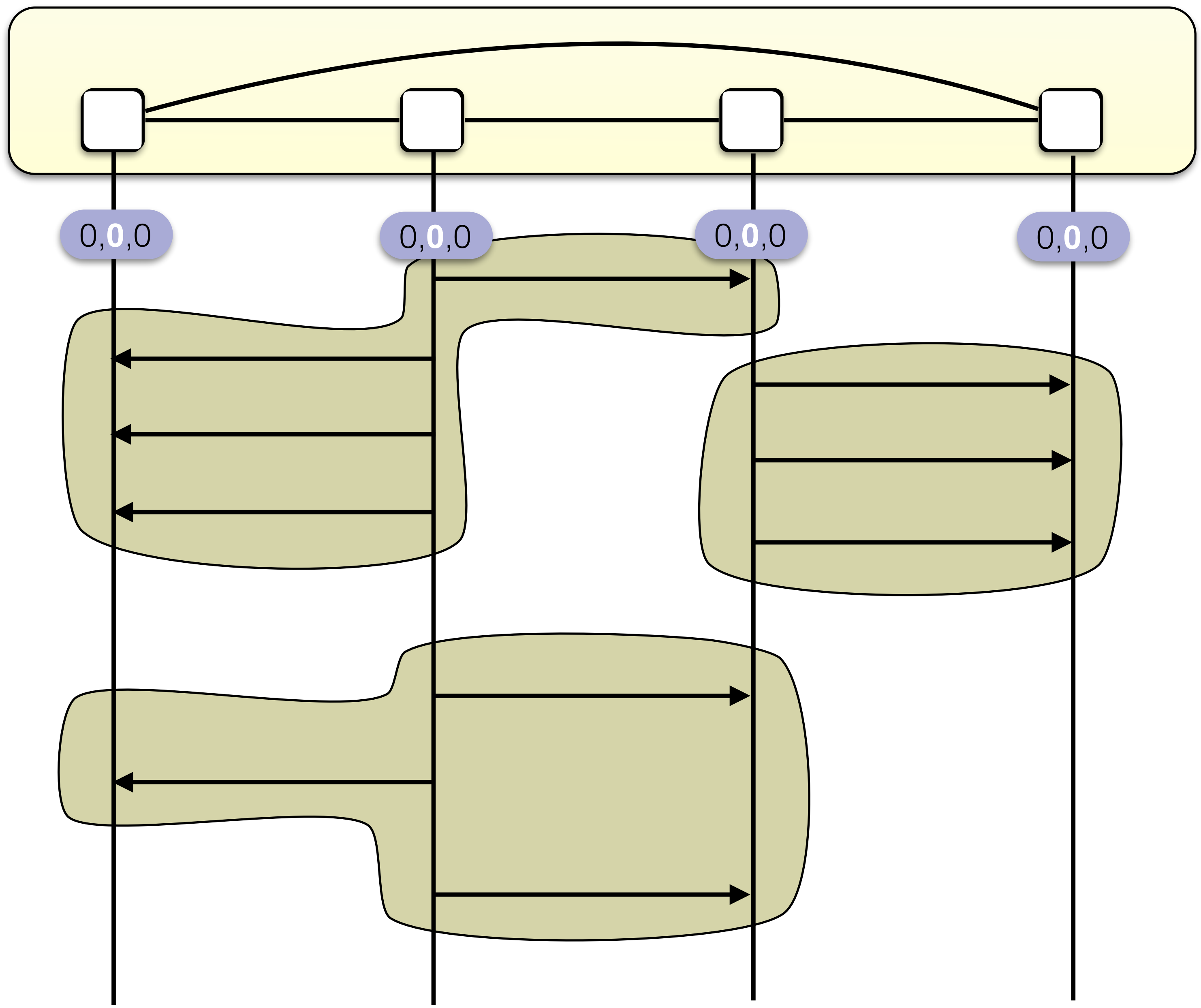
- Every process traverses a bounded number of zones.

Disambiguation of context-bounded PCAs



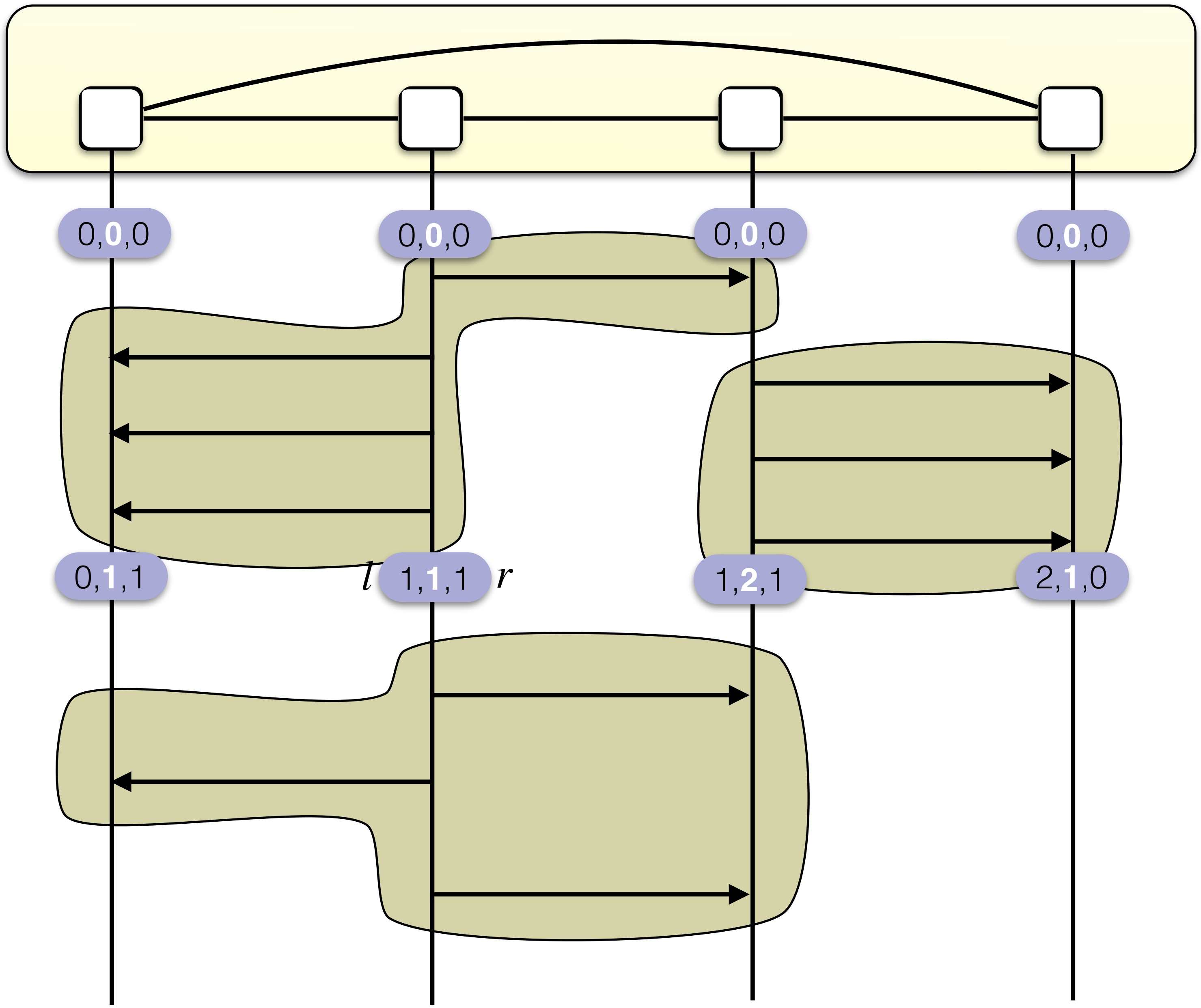
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



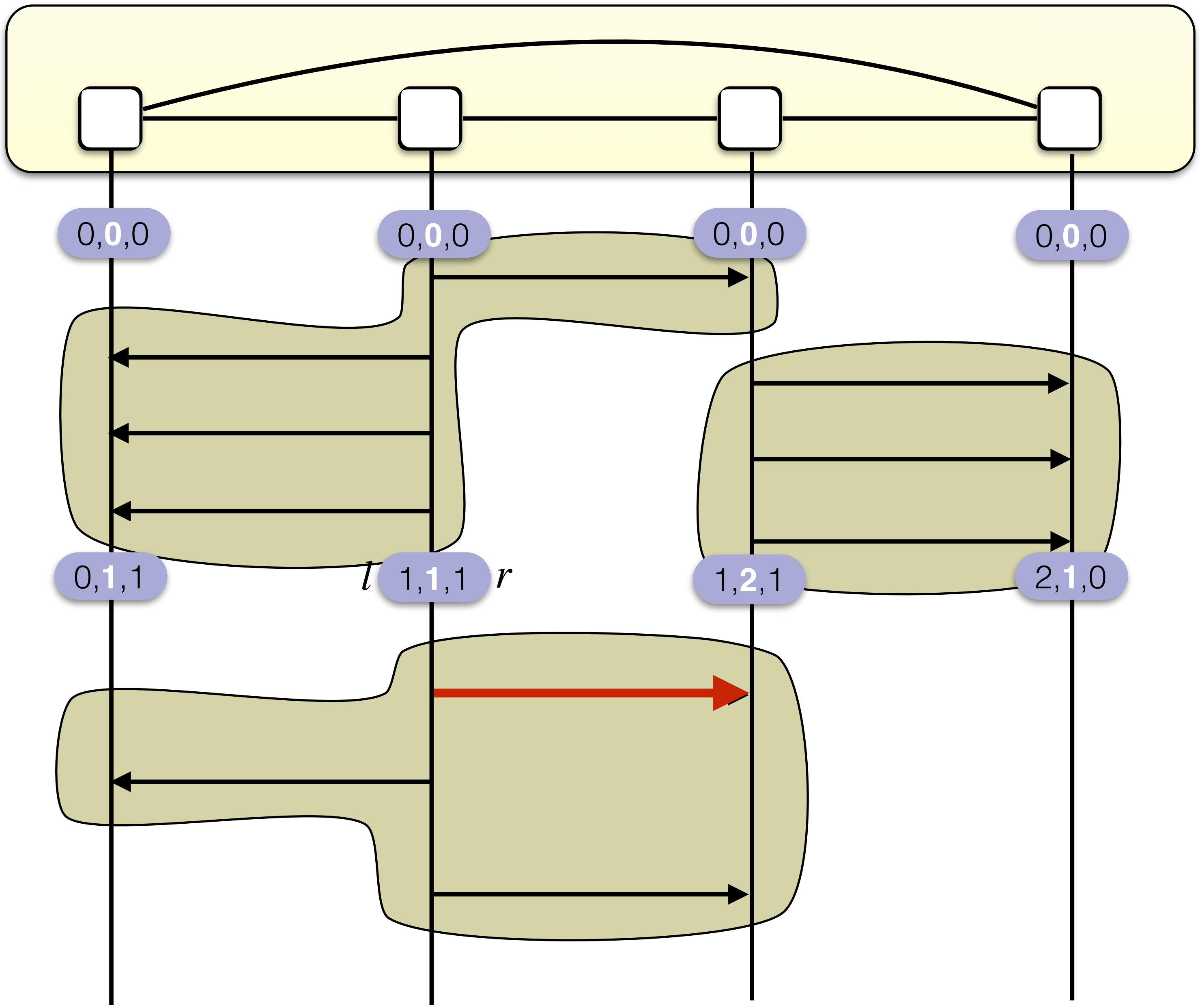
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



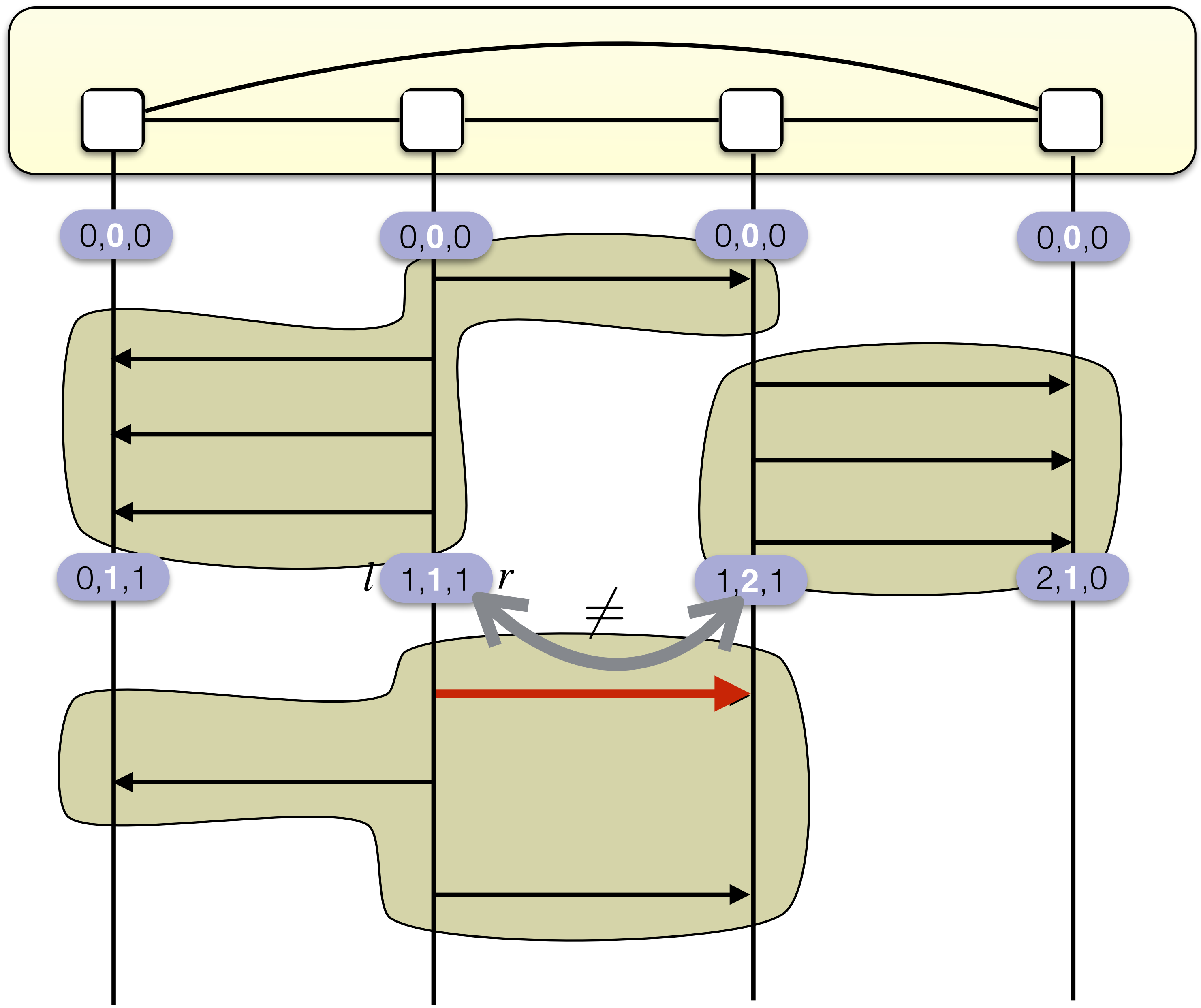
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



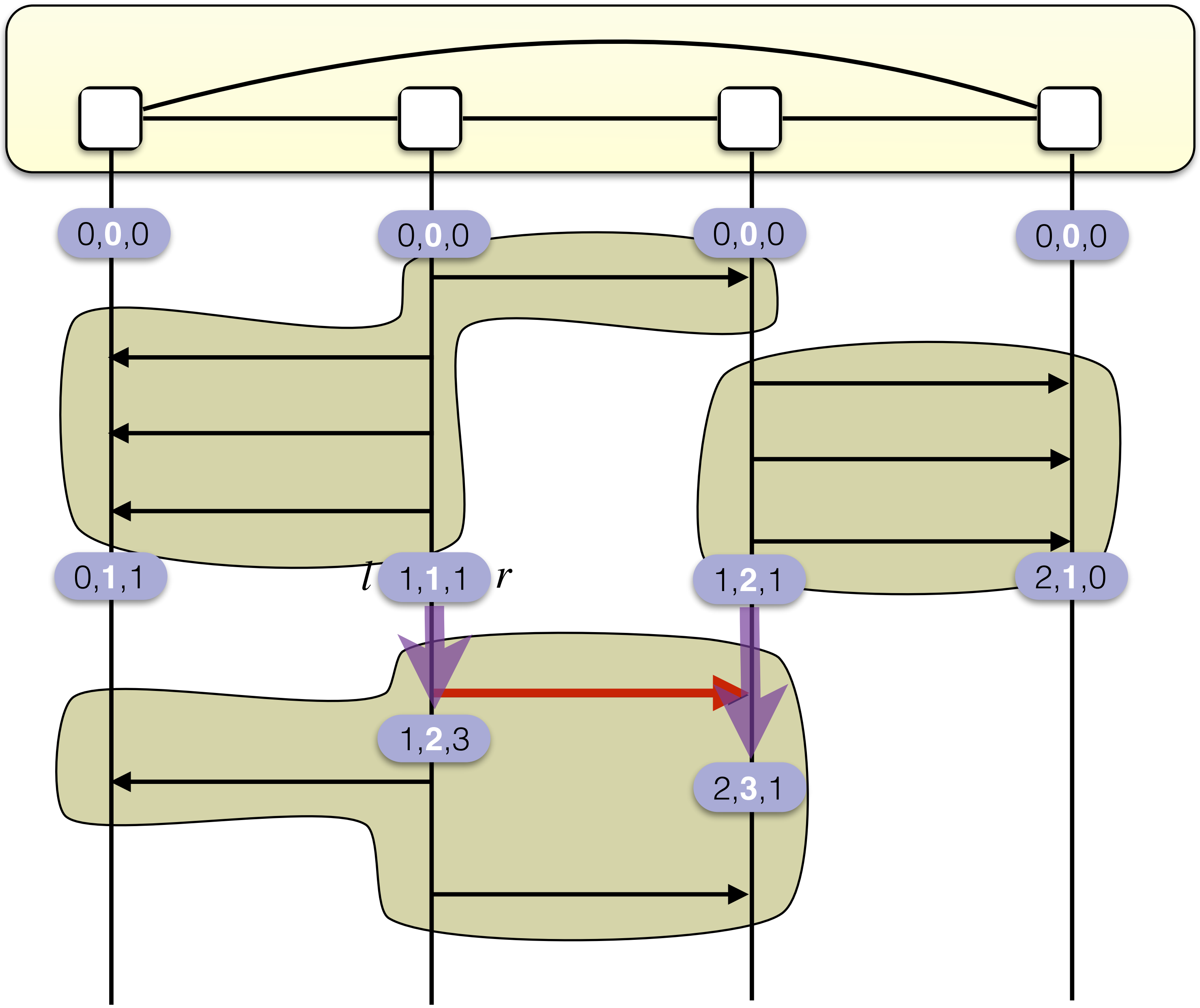
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



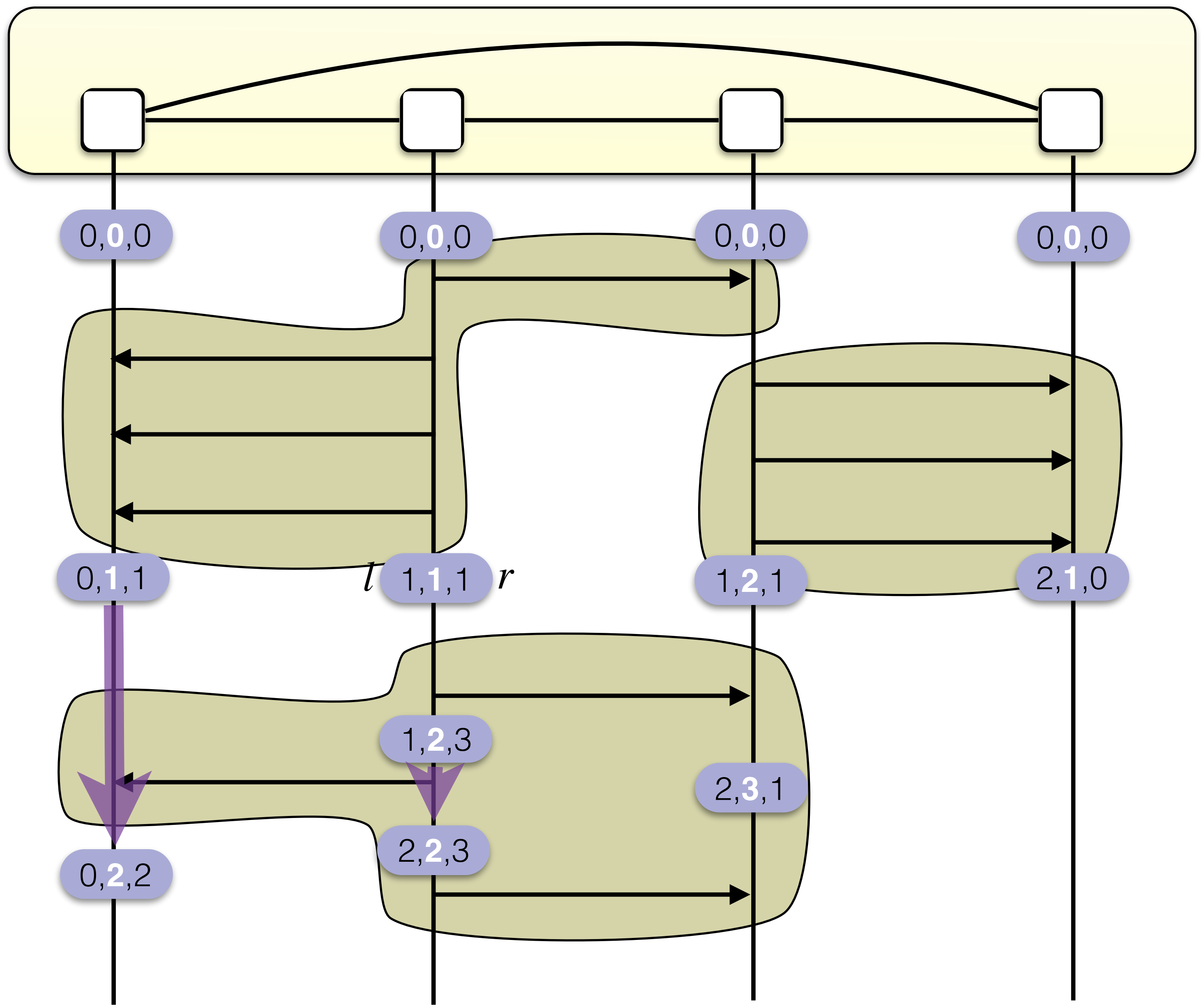
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



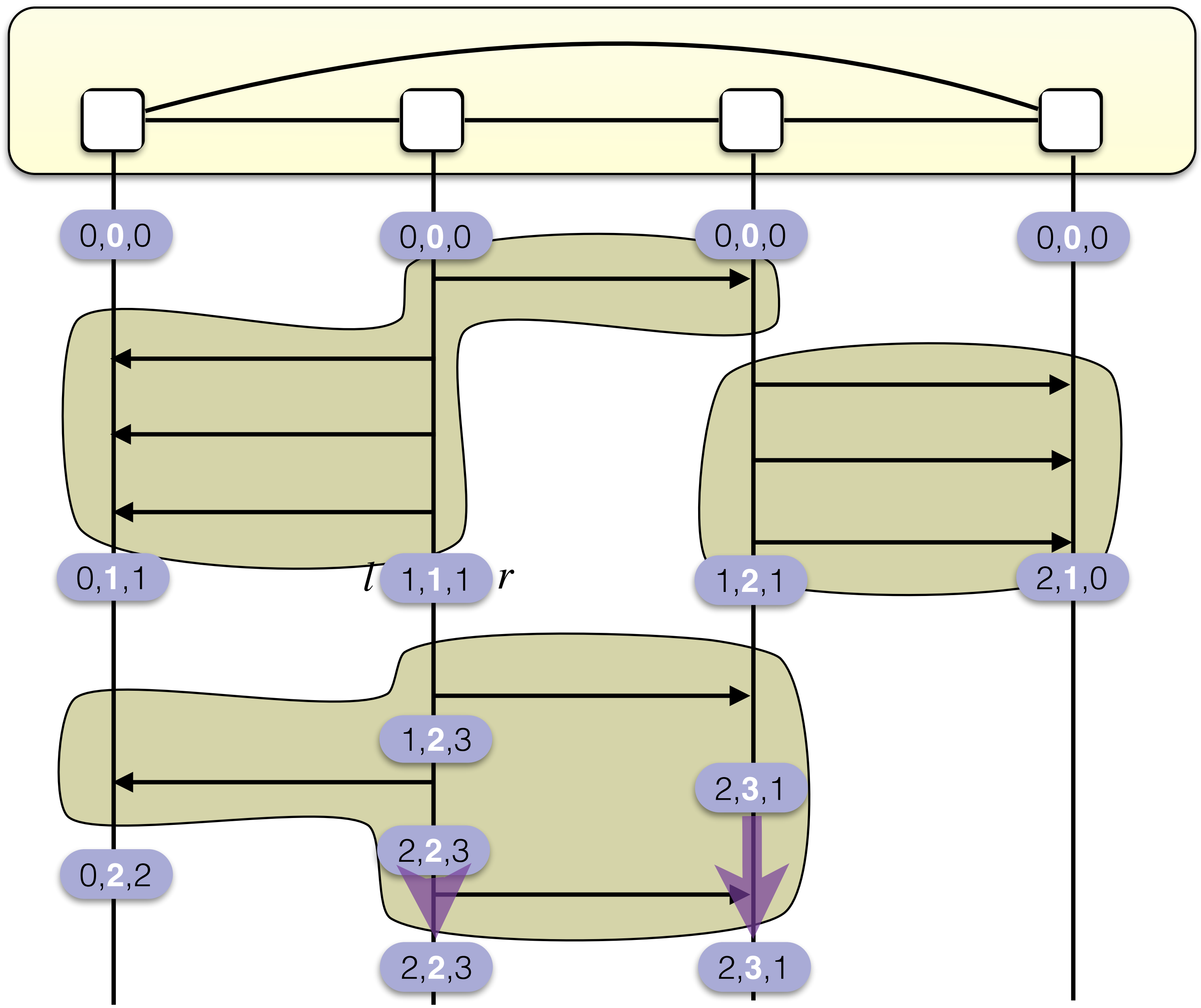
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



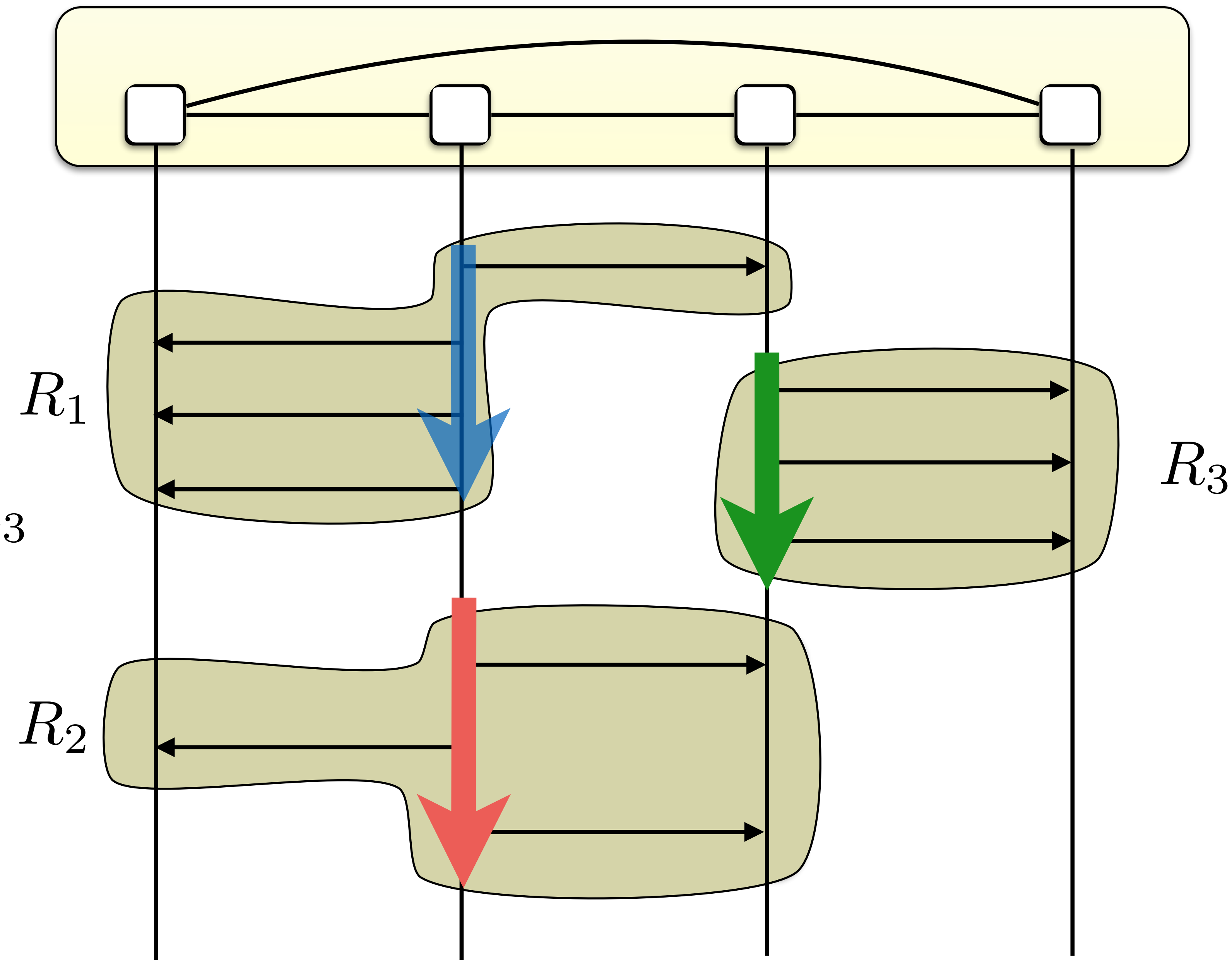
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



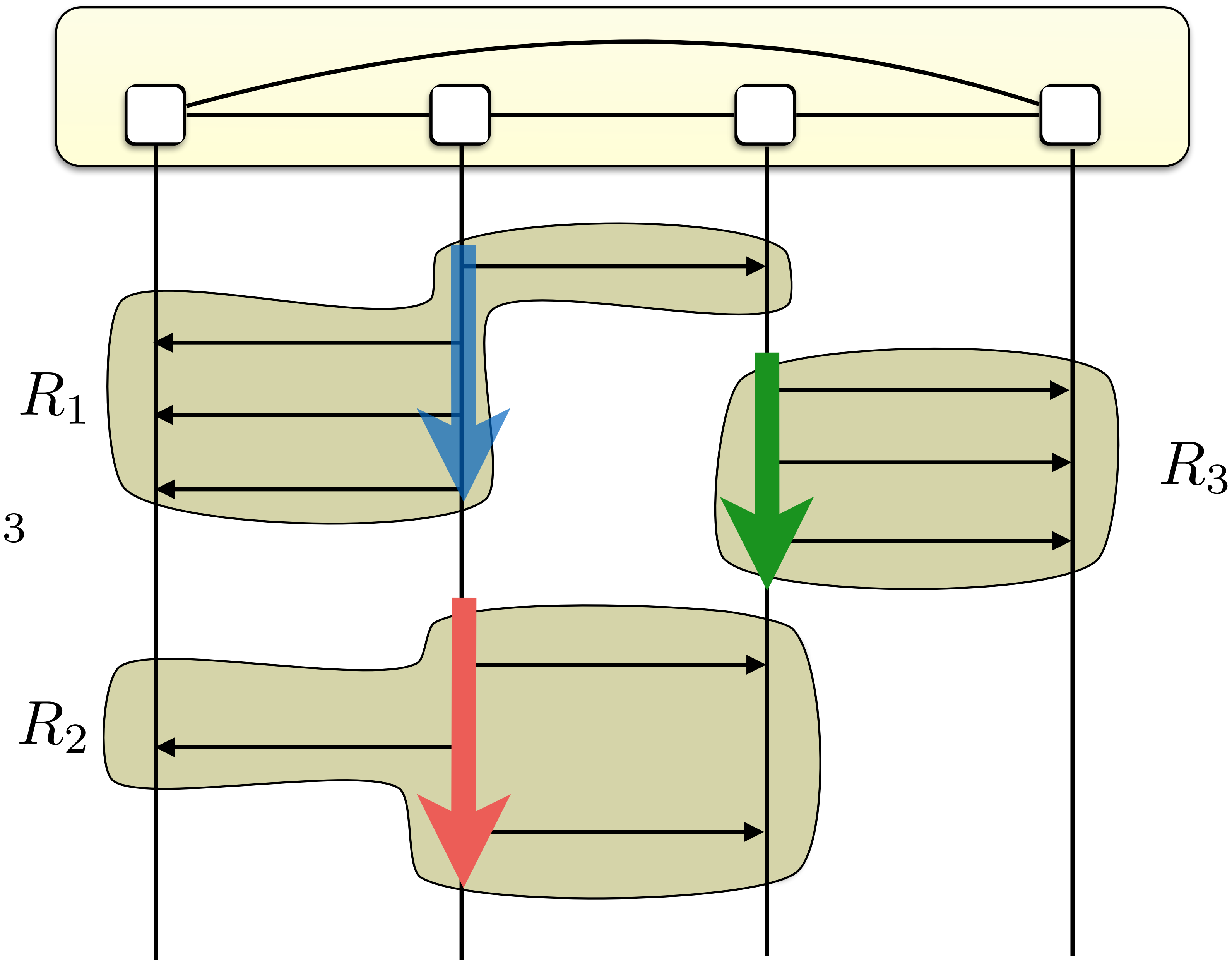
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously by a PCA.

Disambiguation of context-bounded PCAs



$R_i \subseteq S^3 \times S^3$

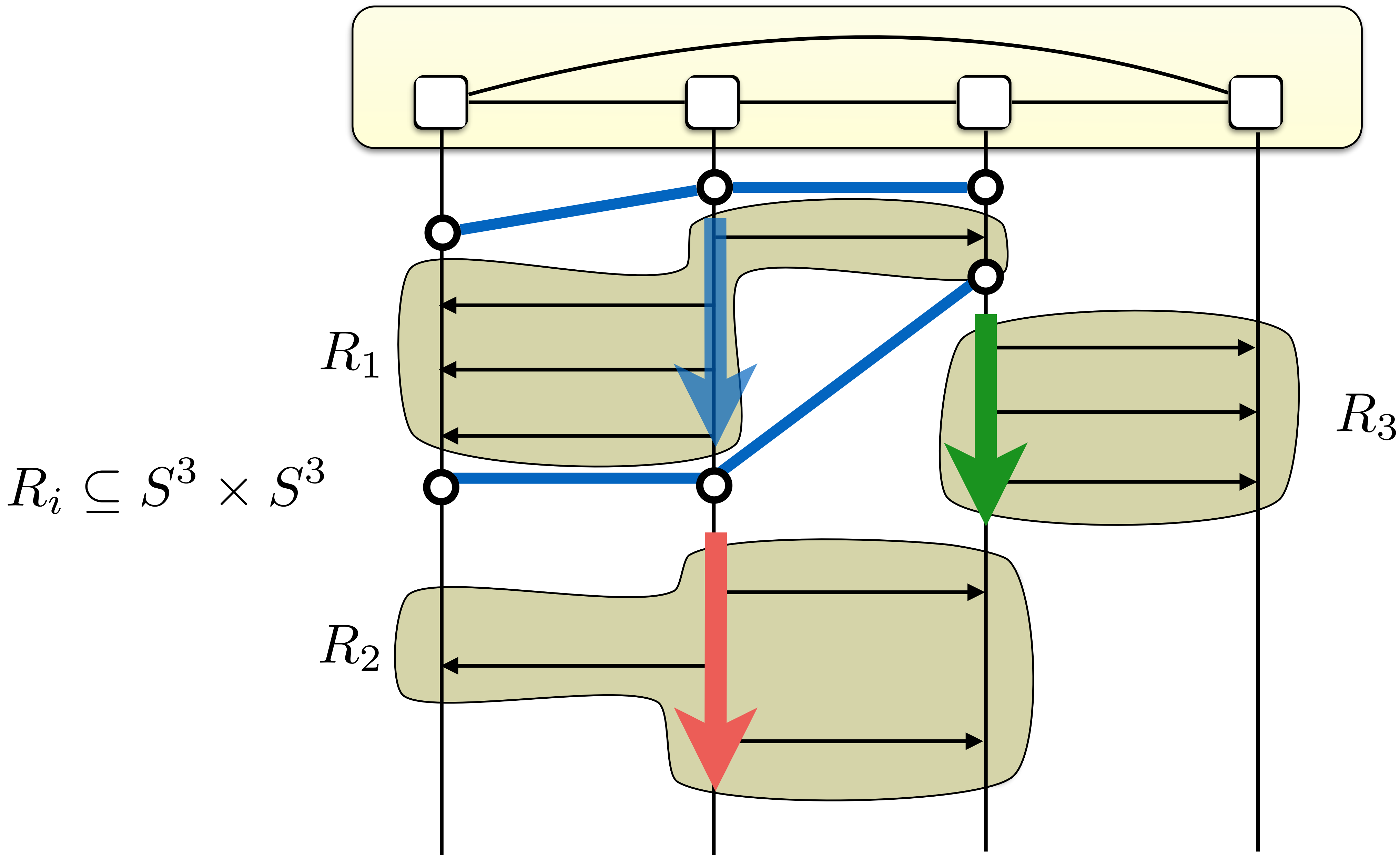
Disambiguation of context-bounded PCAs



$R_i \subseteq S^3 \times S^3$

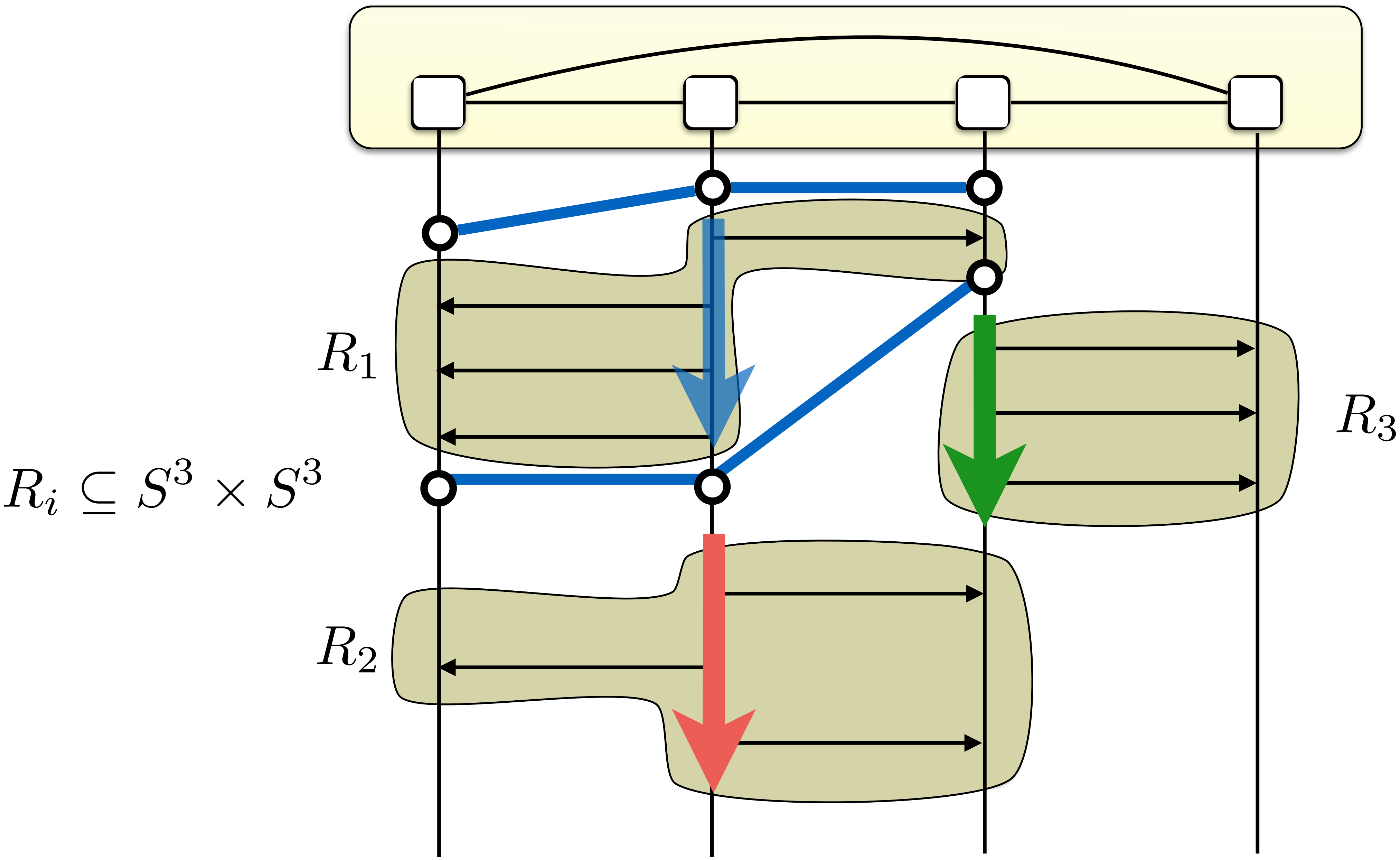
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.

Disambiguation of context-bounded PCAs



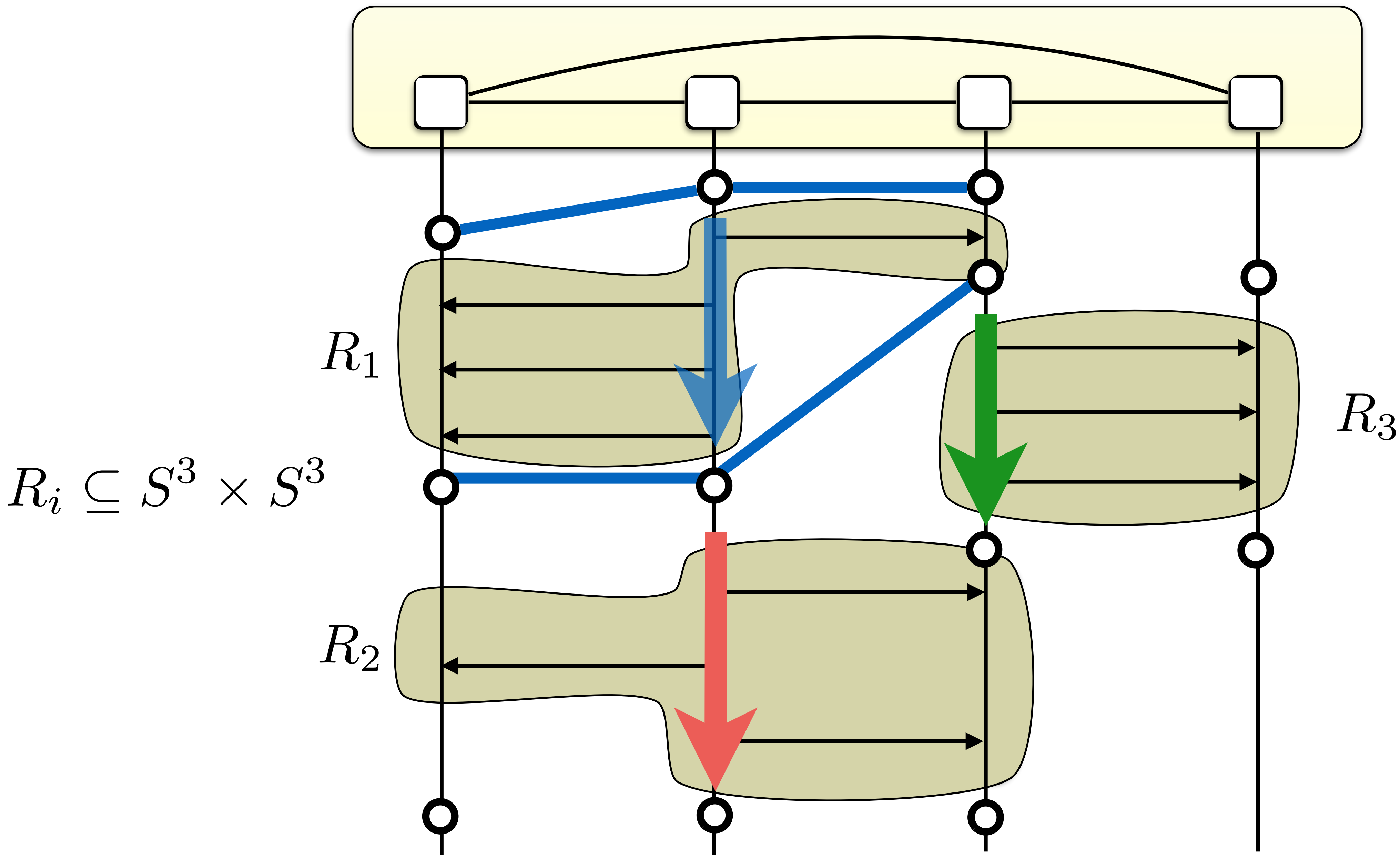
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.

Disambiguation of context-bounded PCAs



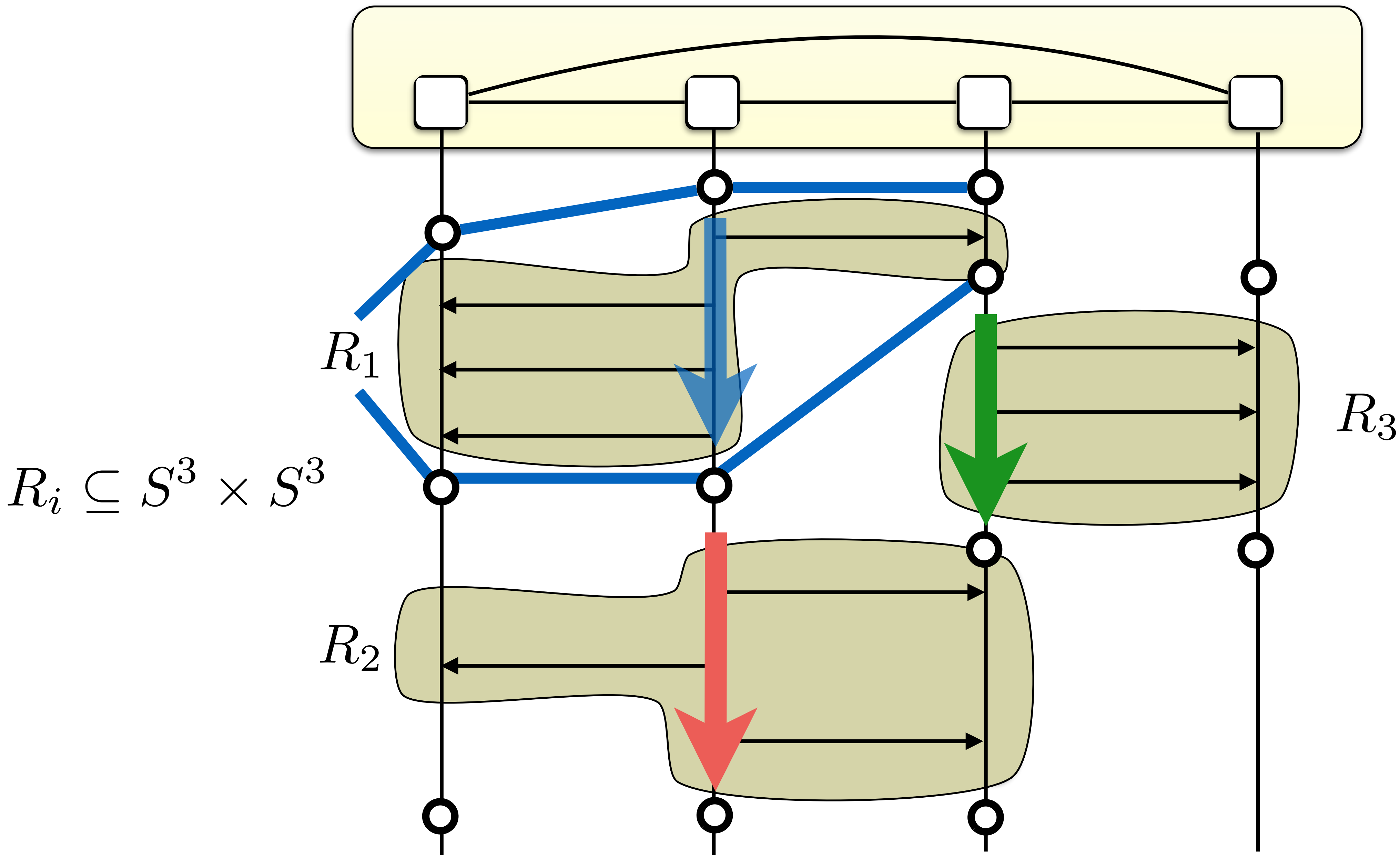
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.

Disambiguation of context-bounded PCAs



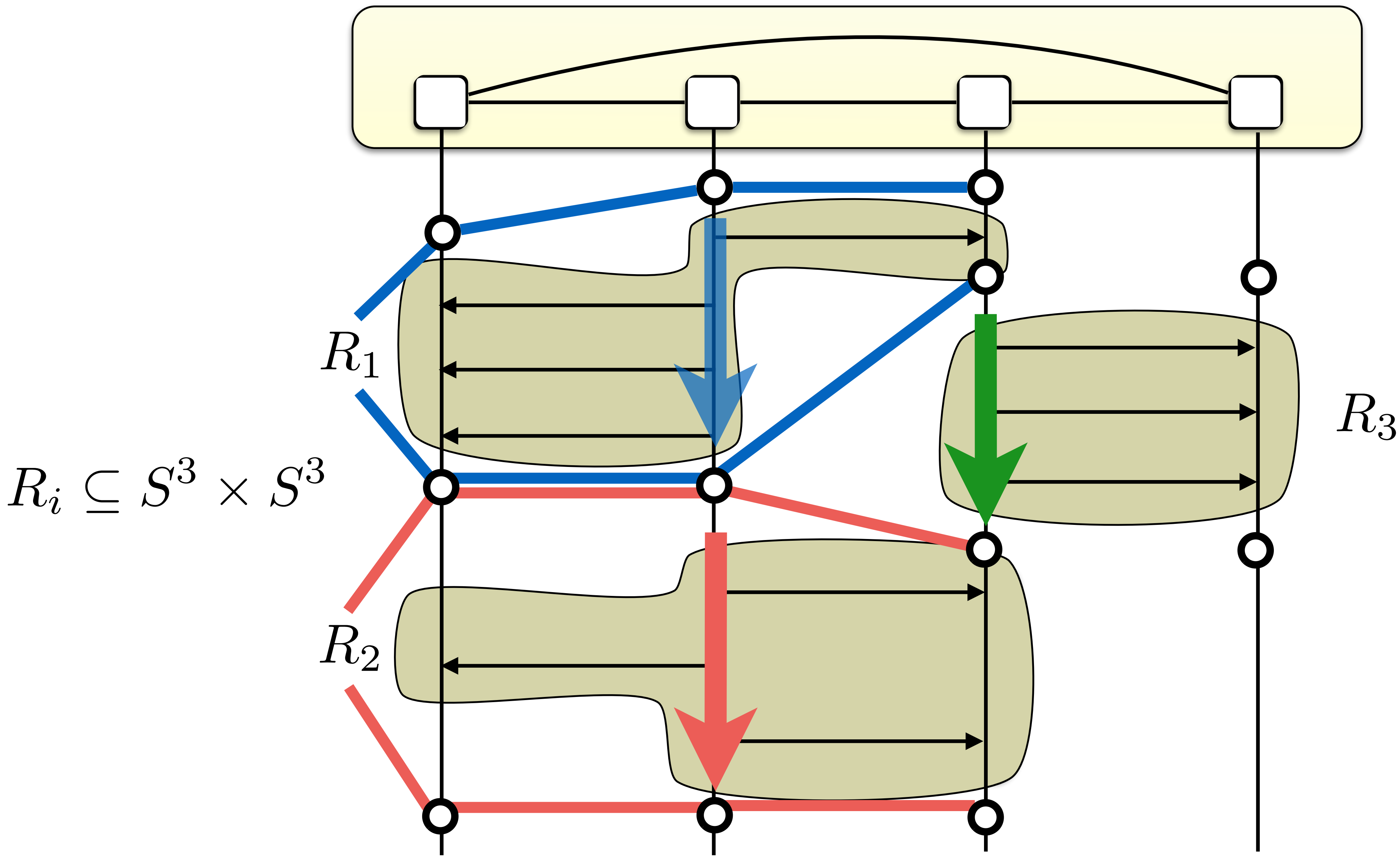
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.

Disambiguation of context-bounded PCAs



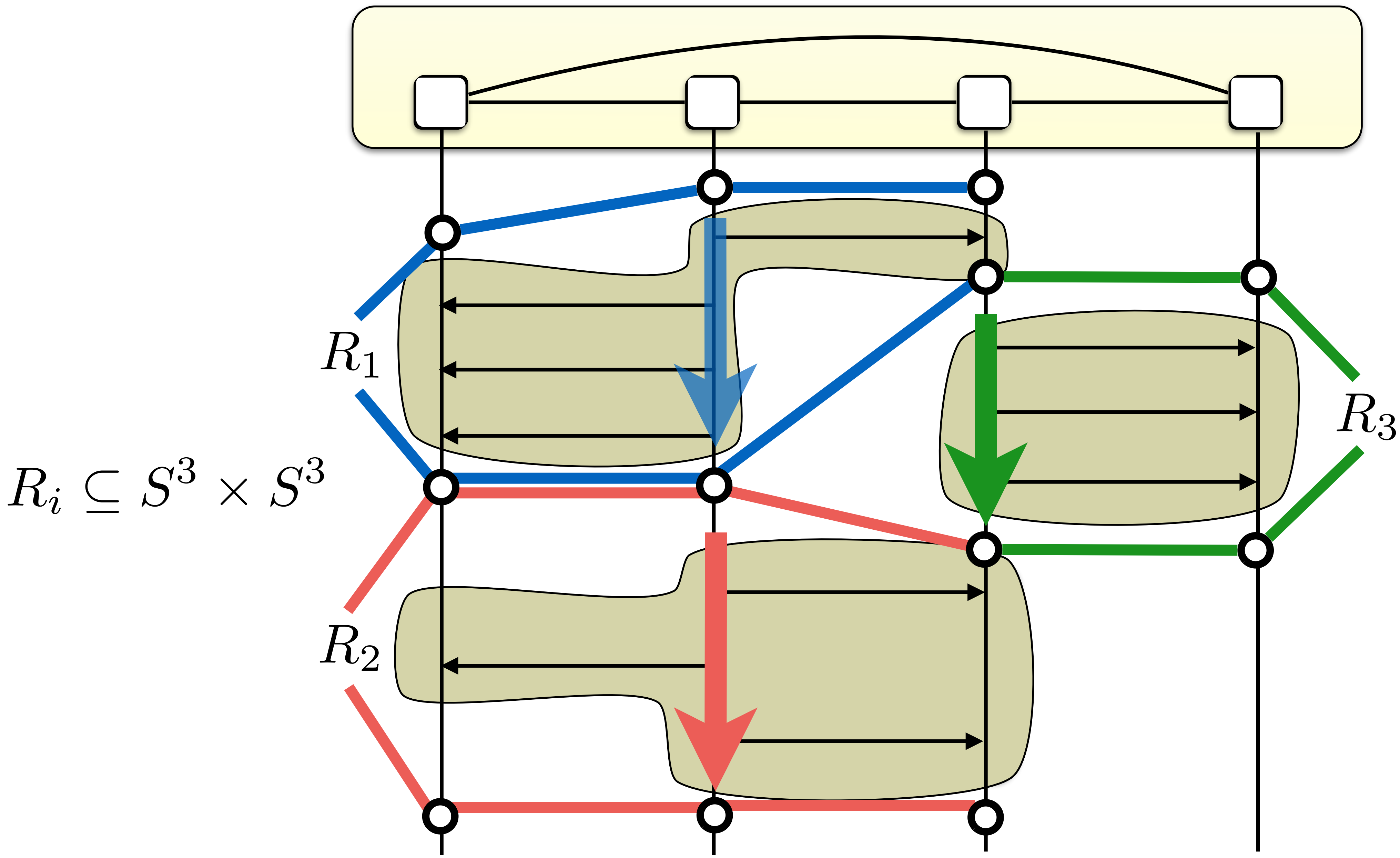
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.

Disambiguation of context-bounded PCAs



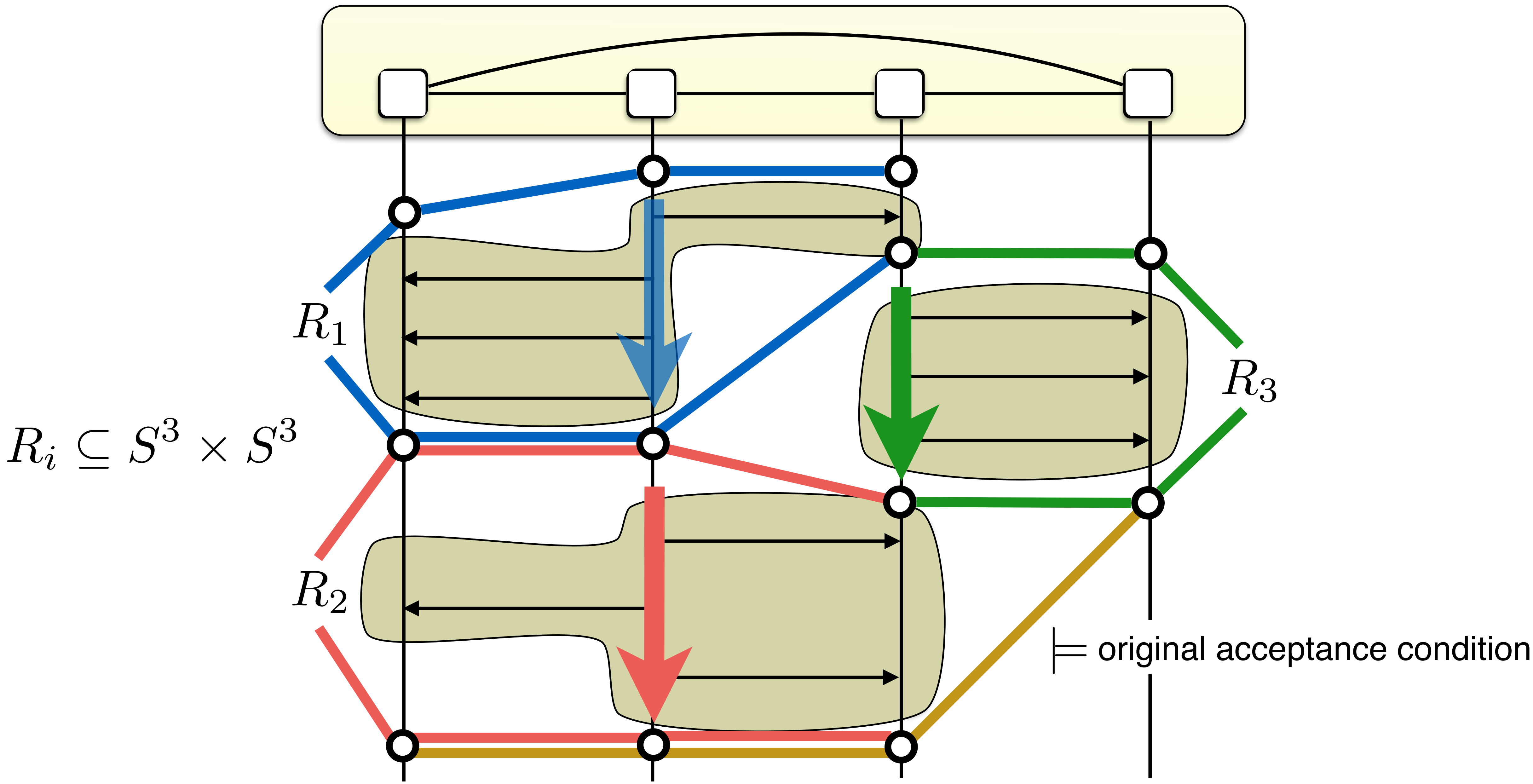
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.

Disambiguation of context-bounded PCAs



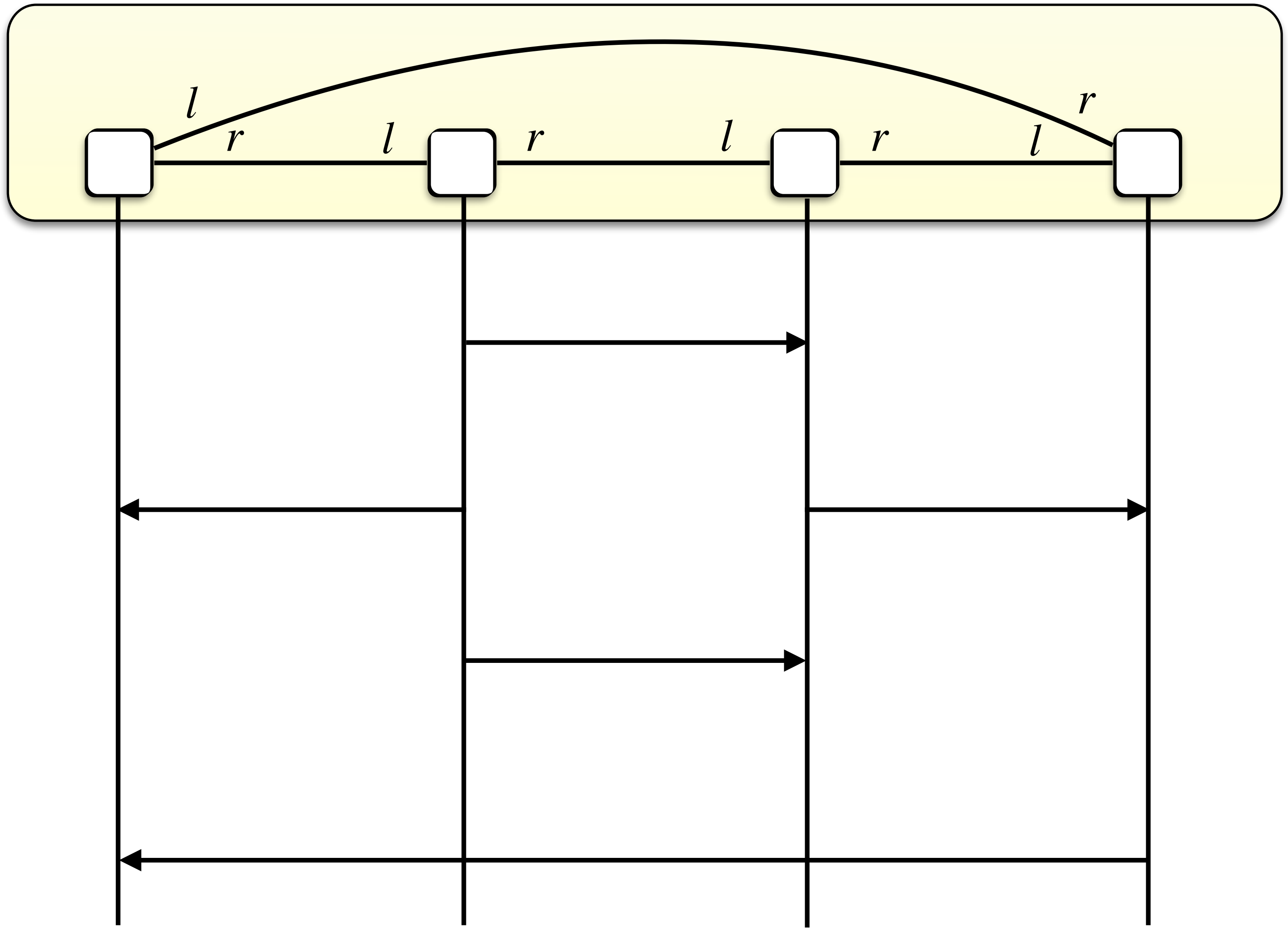
- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.

Disambiguation of context-bounded PCAs



- Every process traverses a bounded number of zones.
- Zone numbers can be computed unambiguously.
- Sending processes deterministically compute summaries for zones.
- Acceptance condition checks if summaries correspond to accepting run.

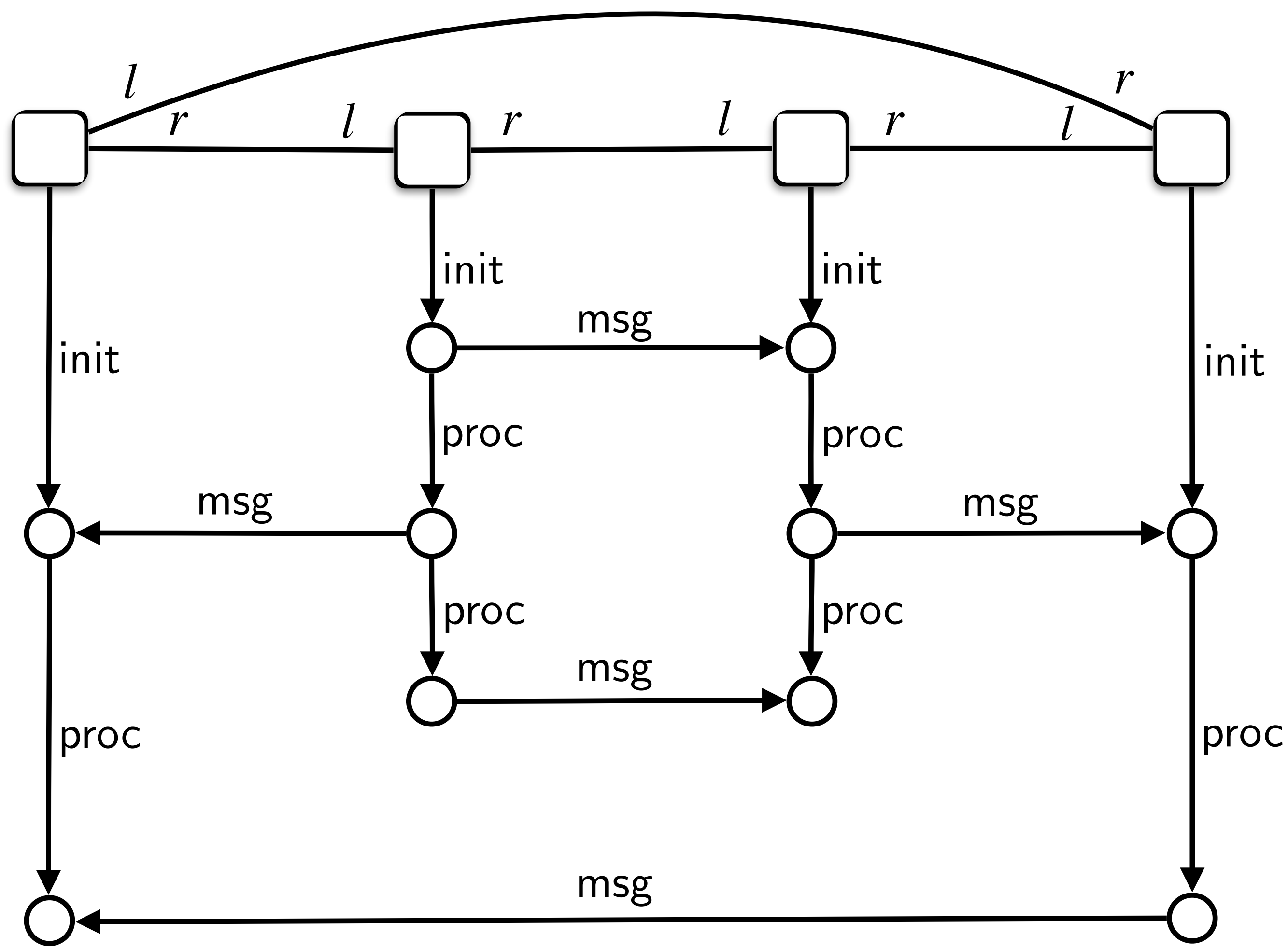
Logical Characterization of Context-Bounded PCAs



The Logic:

MSO logic over graphs, including process nodes and event nodes.

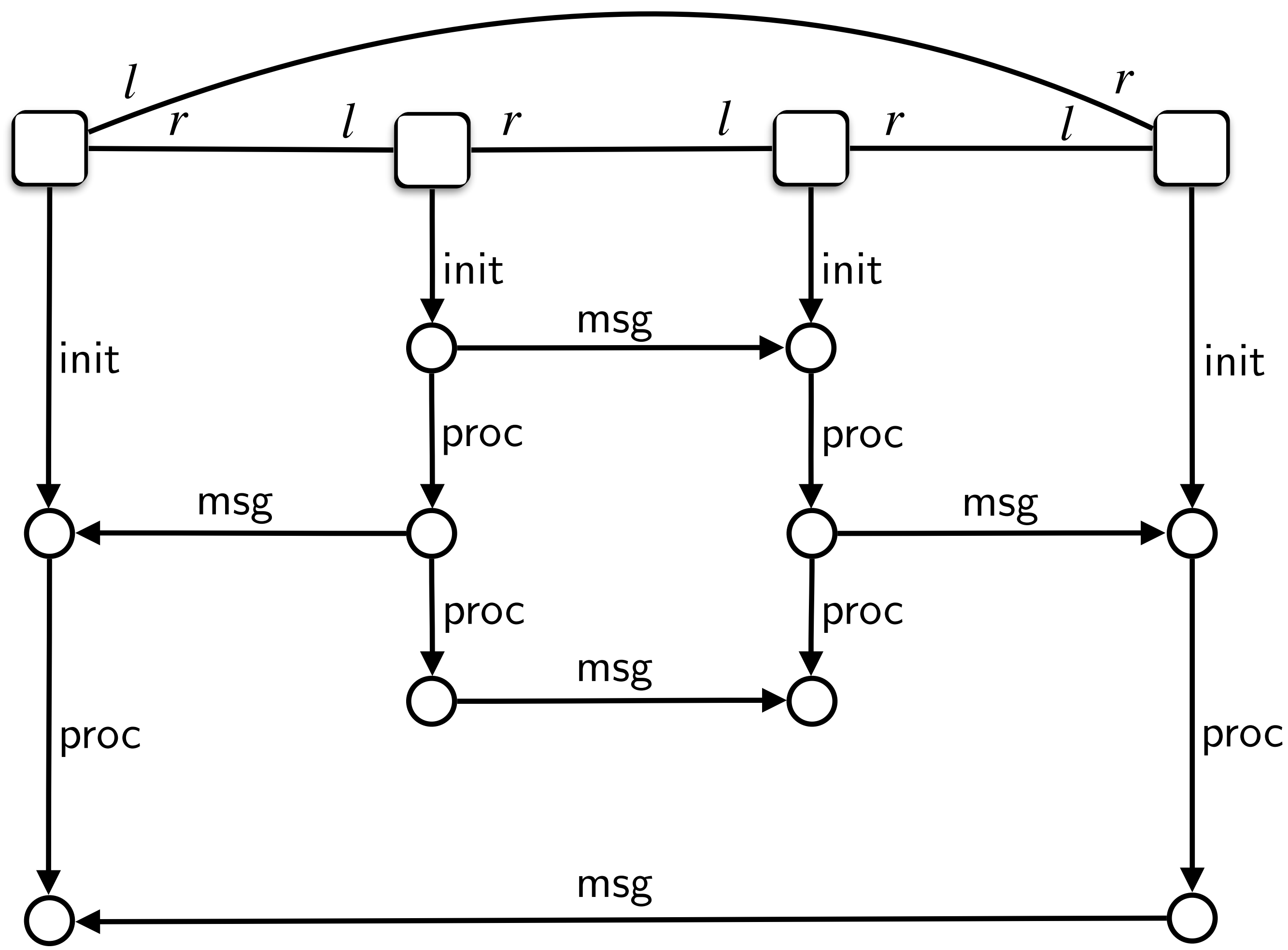
Logical Characterization of Context-Bounded PCAs



The Logic:

MSO logic over graphs, including process nodes and event nodes.

Logical Characterization of Context-Bounded PCAs



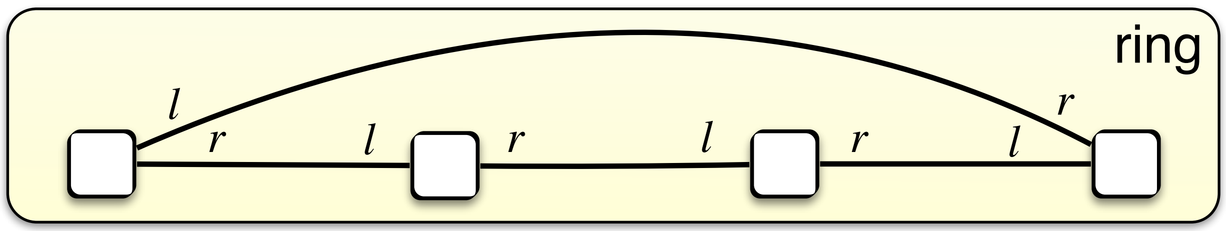
The Logic:
MSO logic over graphs, including process nodes and event nodes.

Corollary [B.-Gastin-Kumar; FSTTCS 2014]:
For every *bounded* set L of behaviors, the following are equivalent:

- L is recognized by some PCA.
- L is definable in MSO logic.

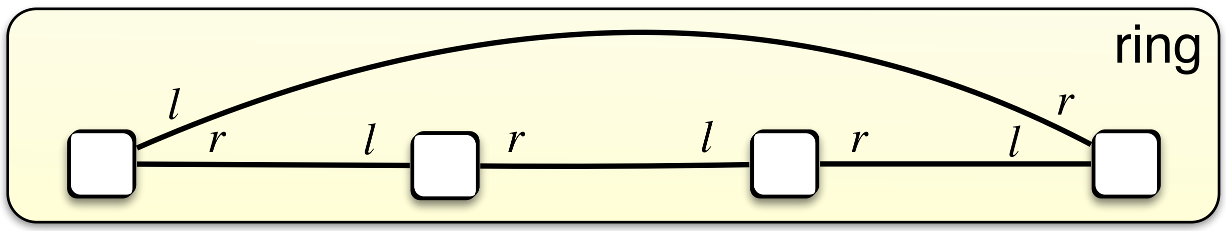
Topologies of Bounded Degree

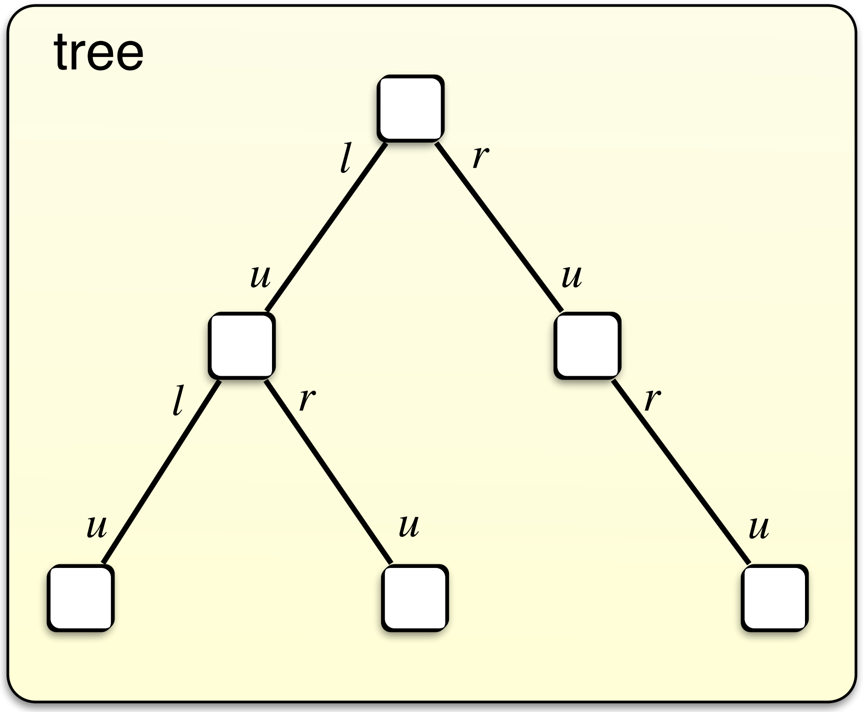
Complementation and MSO characterization hold wrt. the class of **all topologies** over a fixed set of ports. With 4 ports, this captures **rings, binary trees, and grids**.



Topologies of Bounded Degree

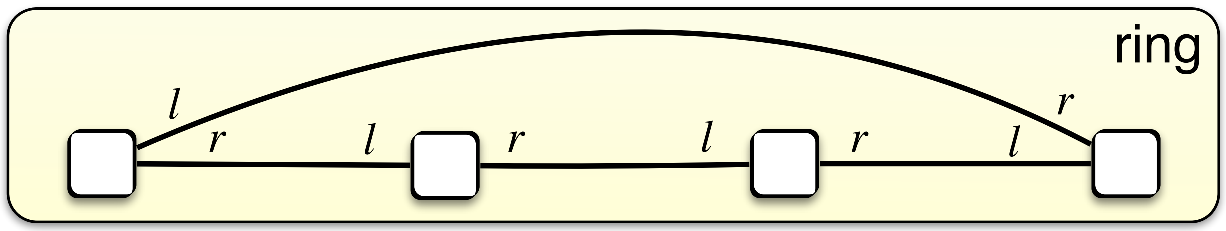
Complementation and MSO characterization hold wrt. the class of **all topologies** over a fixed set of ports. With 4 ports, this captures **rings, binary trees, and grids**.

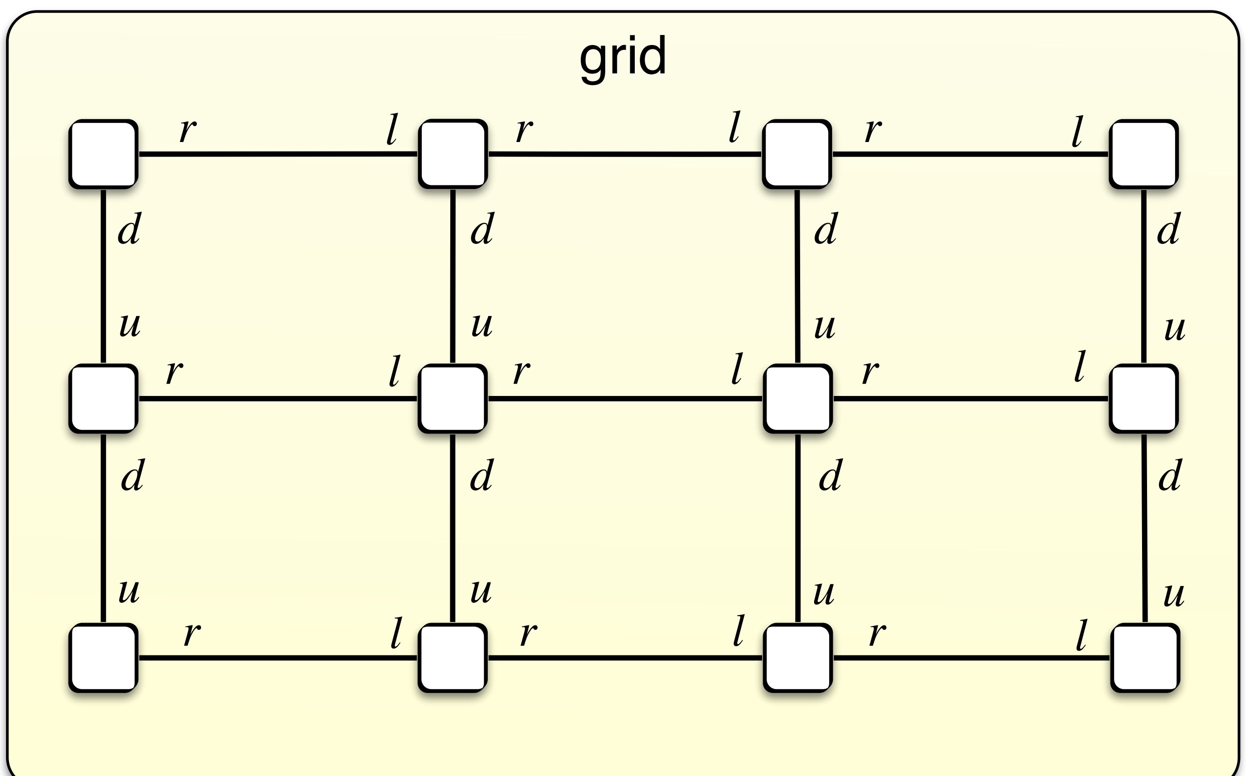


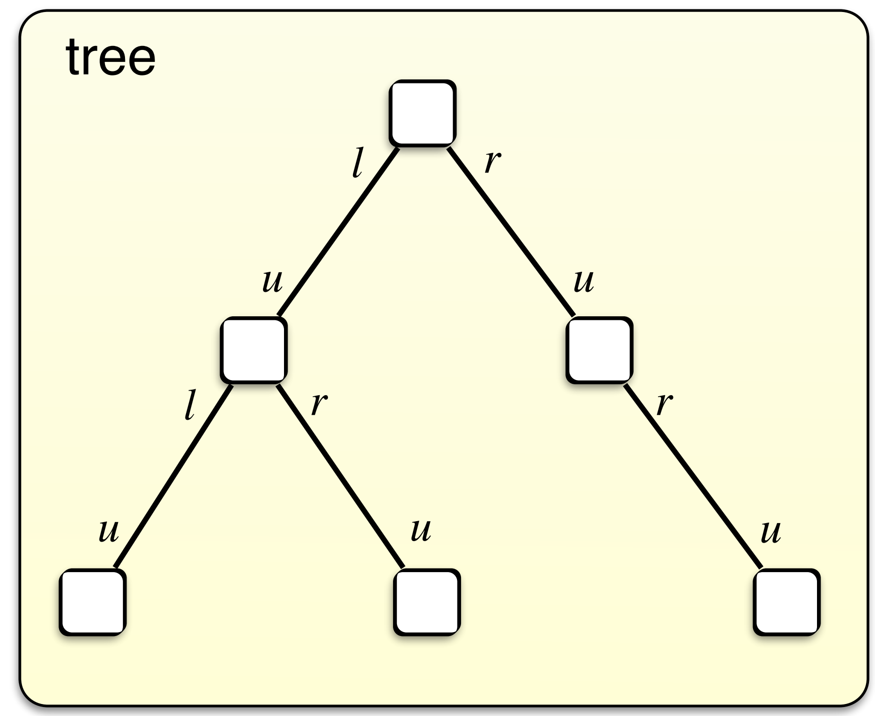


Topologies of Bounded Degree

Complementation and MSO characterization hold wrt. the class of **all topologies** over a fixed set of ports. With 4 ports, this captures **rings, binary trees, and grids**.







Context-Bounded Model Checking

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
Context-bounded MSO model checking is decidable over rings.

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$; MSO formula φ

Question: $M \models \varphi$ for all k-bounded $M \in L(\mathcal{A})$?

Context-Bounded Model Checking

Theorem [B.-Gastin-Kumar; FSTTCS 2014]:
Context-bounded MSO model checking is decidable over rings.

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$; MSO formula φ

Question: $M \models \varphi$ for all k-bounded $M \in L(\mathcal{A})$?

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete when the acceptance condition is presented as a finite automaton.

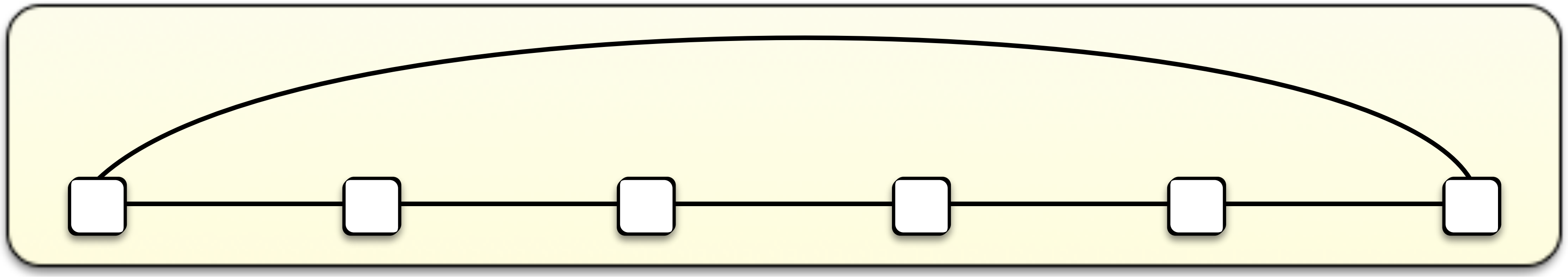
Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$

Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$

Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?



- Finite automaton guesses local states & checks membership in summaries.

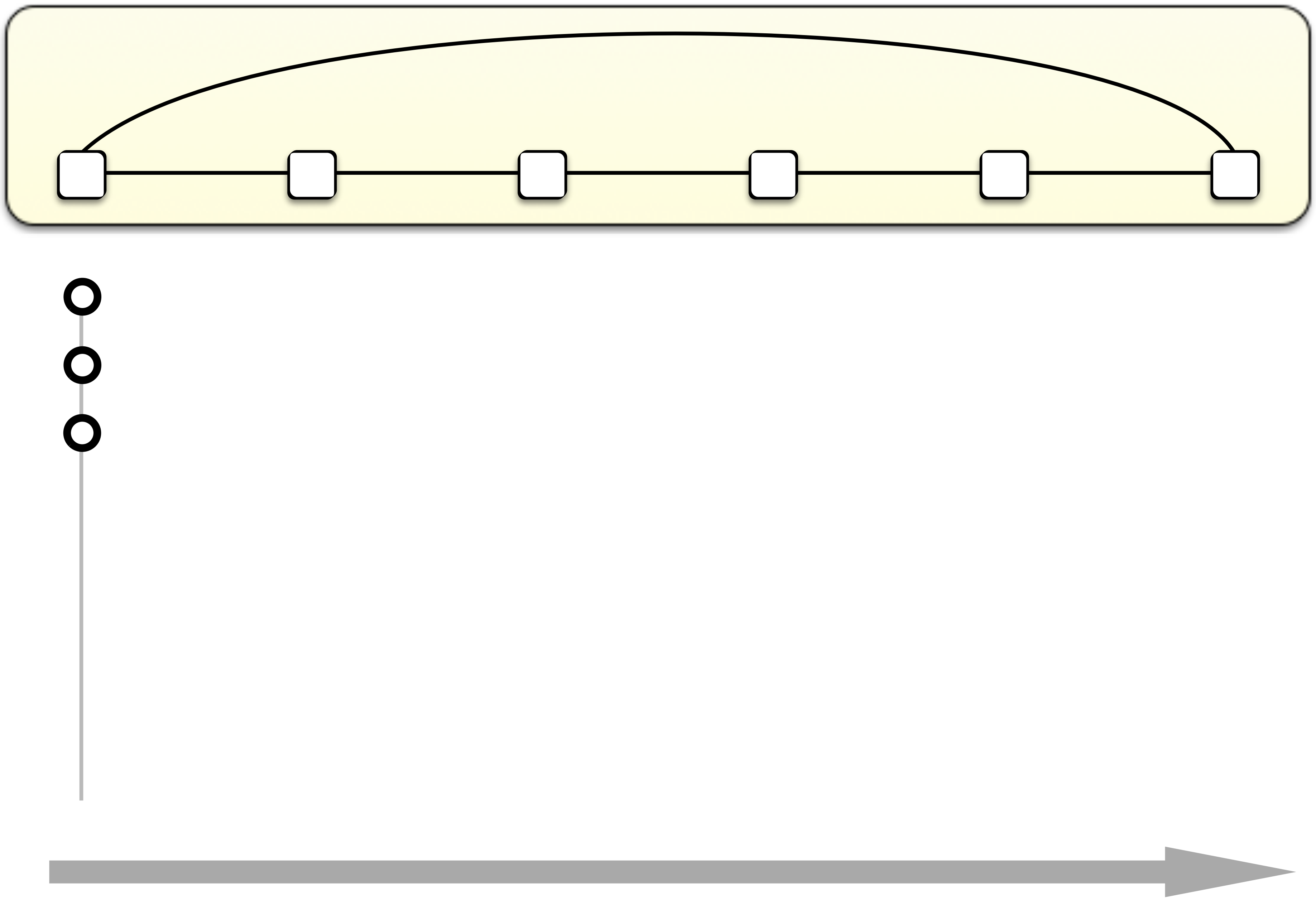
Theorem [B.-Gastin-Schubert; RP 2014]:

Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$

Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?



- Finite automaton guesses local states & checks membership in summaries.

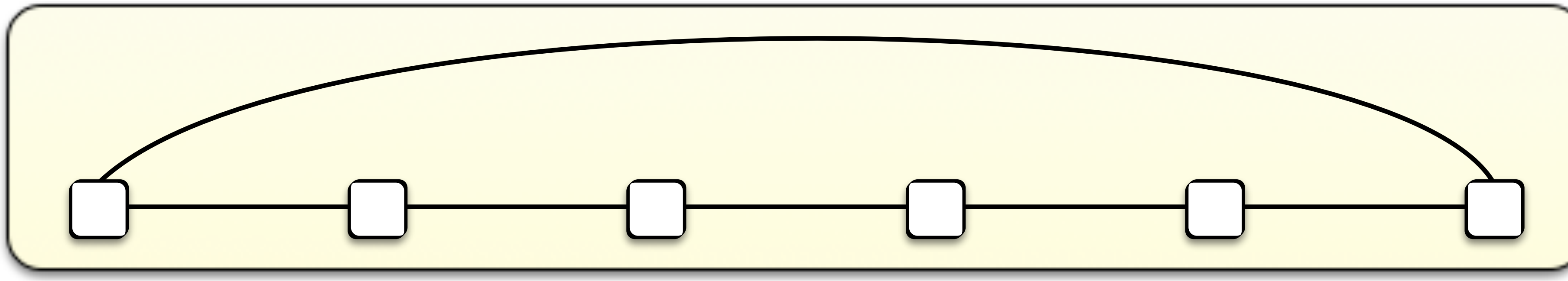
Theorem [B.-Gastin-Schubert; RP 2014]:

Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$

Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?



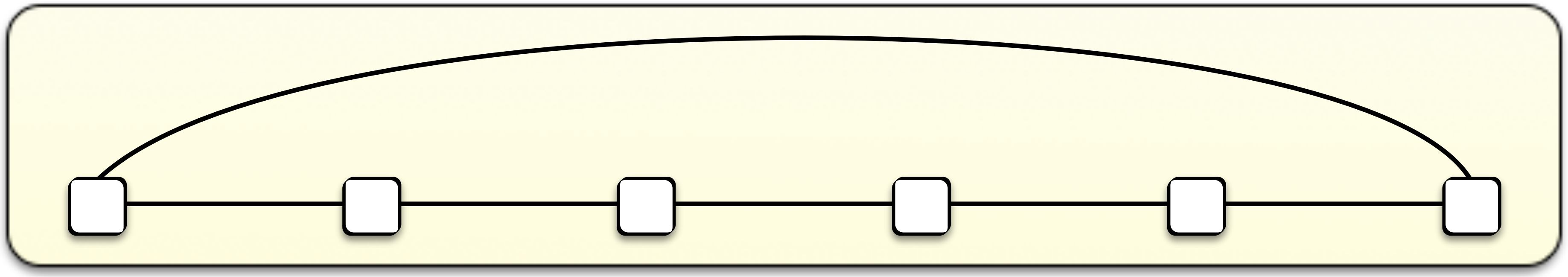
- Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:

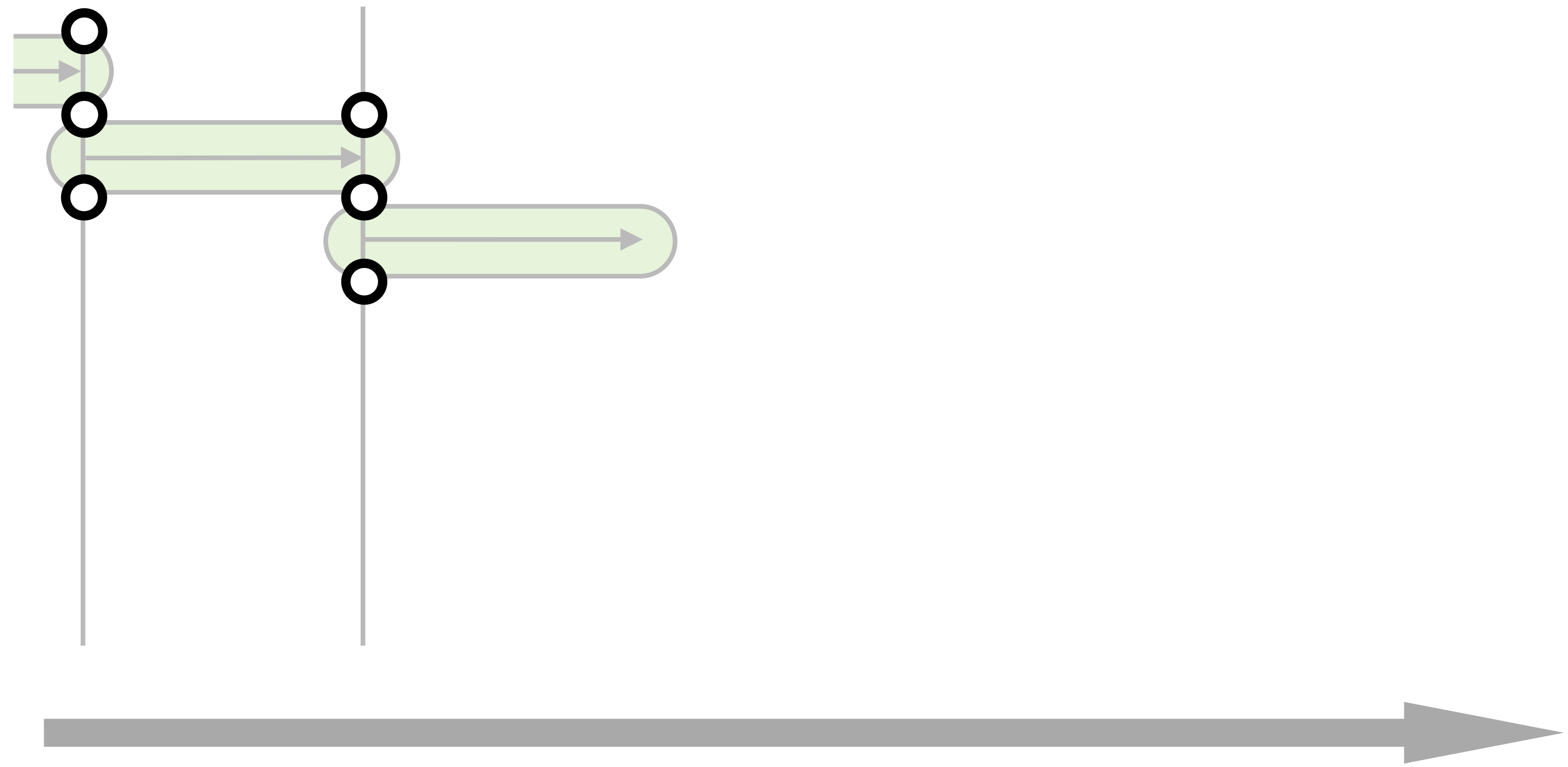
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$
Question: Does $L(\mathcal{A})$ contain some k-bounded behavior?



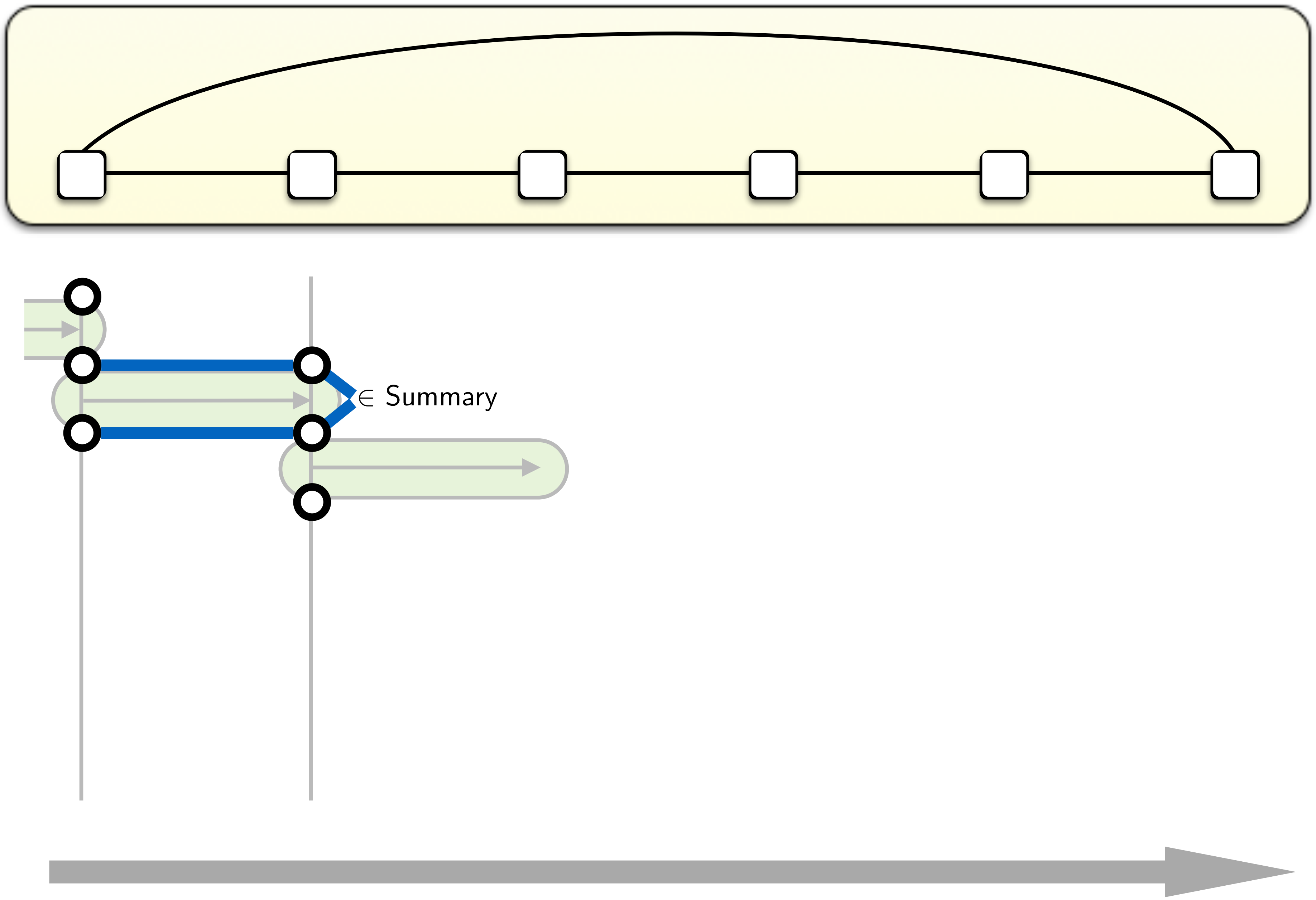
- Finite automaton guesses local states & checks membership in summaries.



Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$
Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?

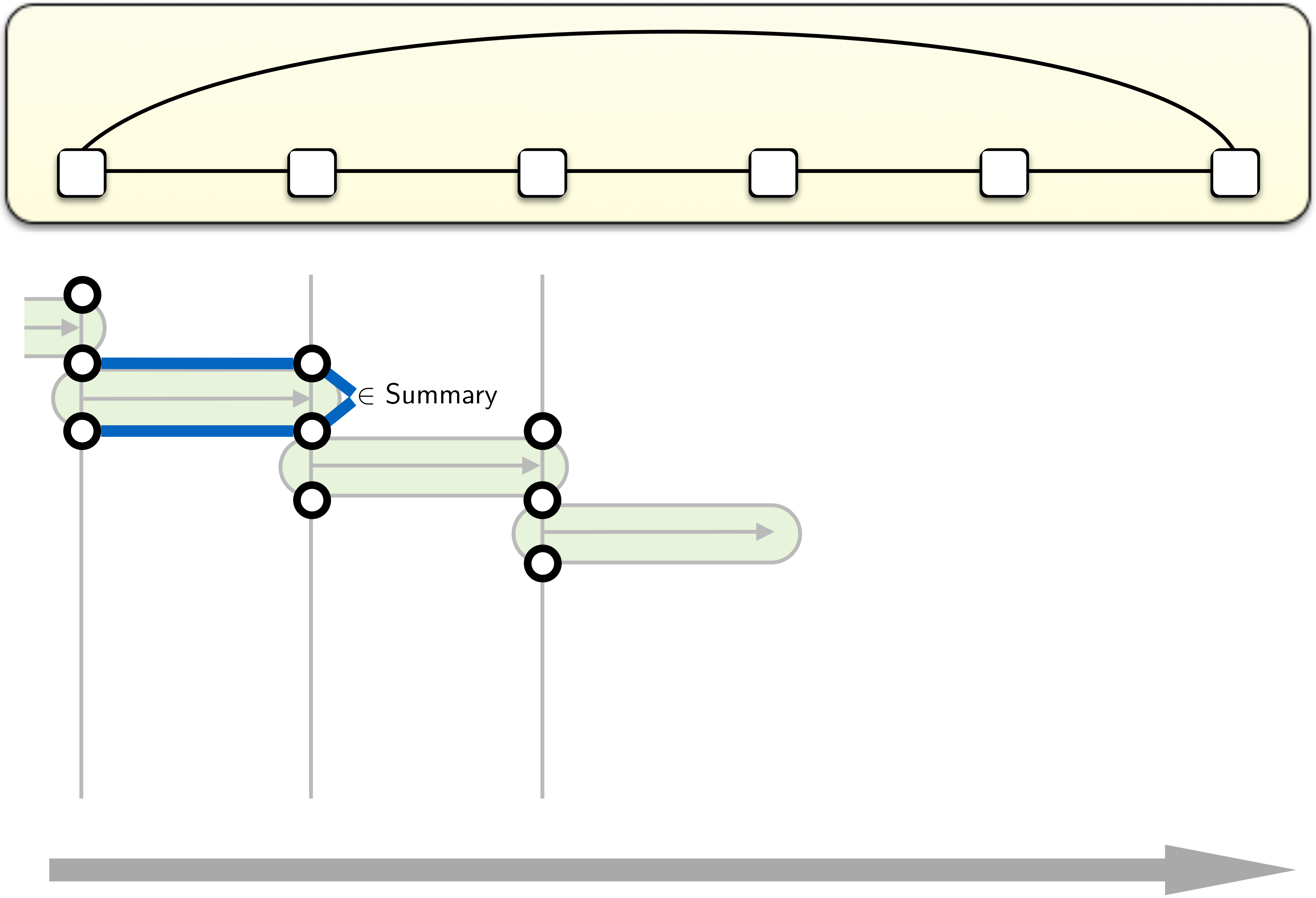


- Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input:	PCA $\mathcal{A}$; $k \in \mathbb{N}$
Question:	Does $L(\mathcal{A})$ contain some k-bounded behavior ?

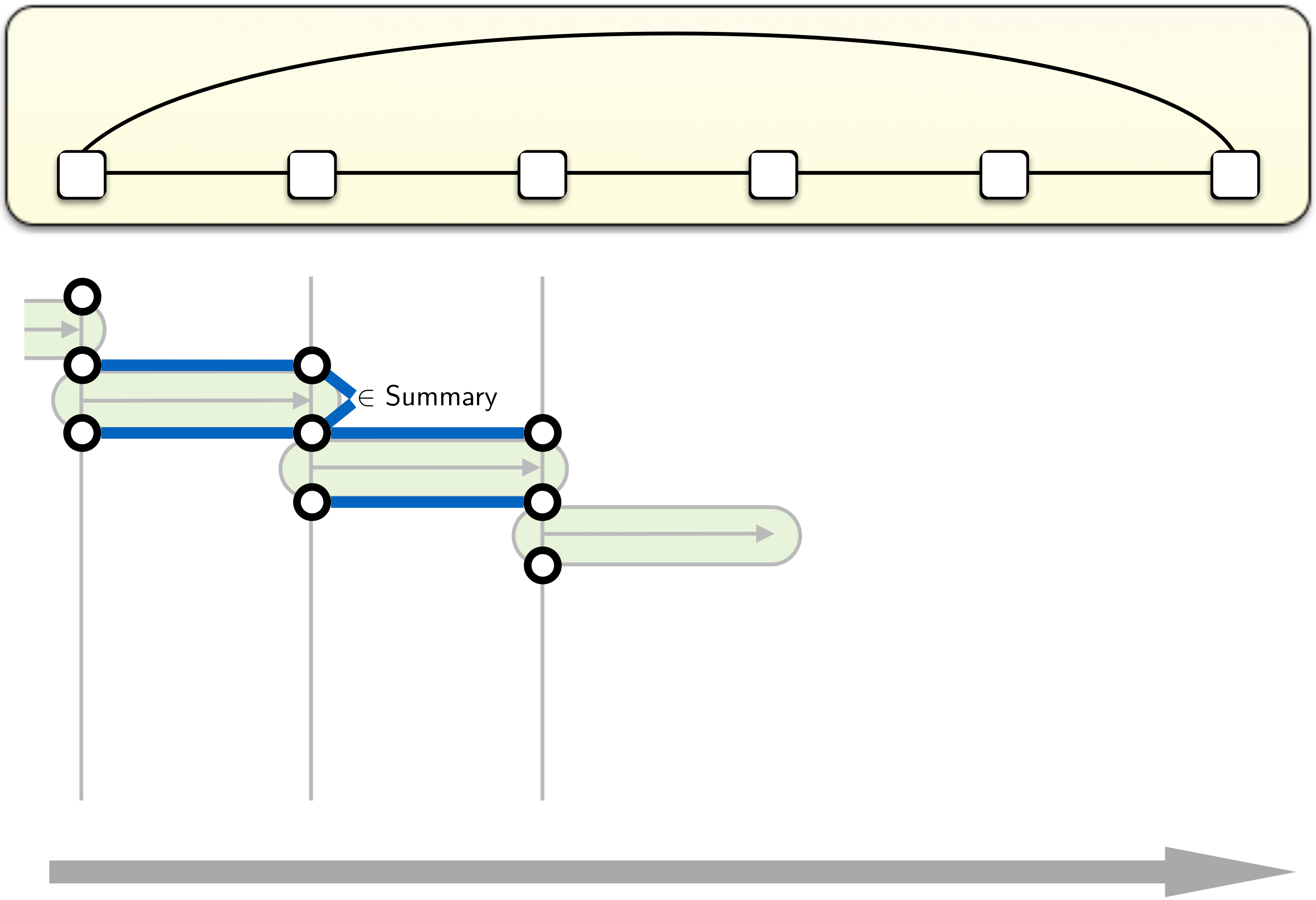


- Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$
Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?



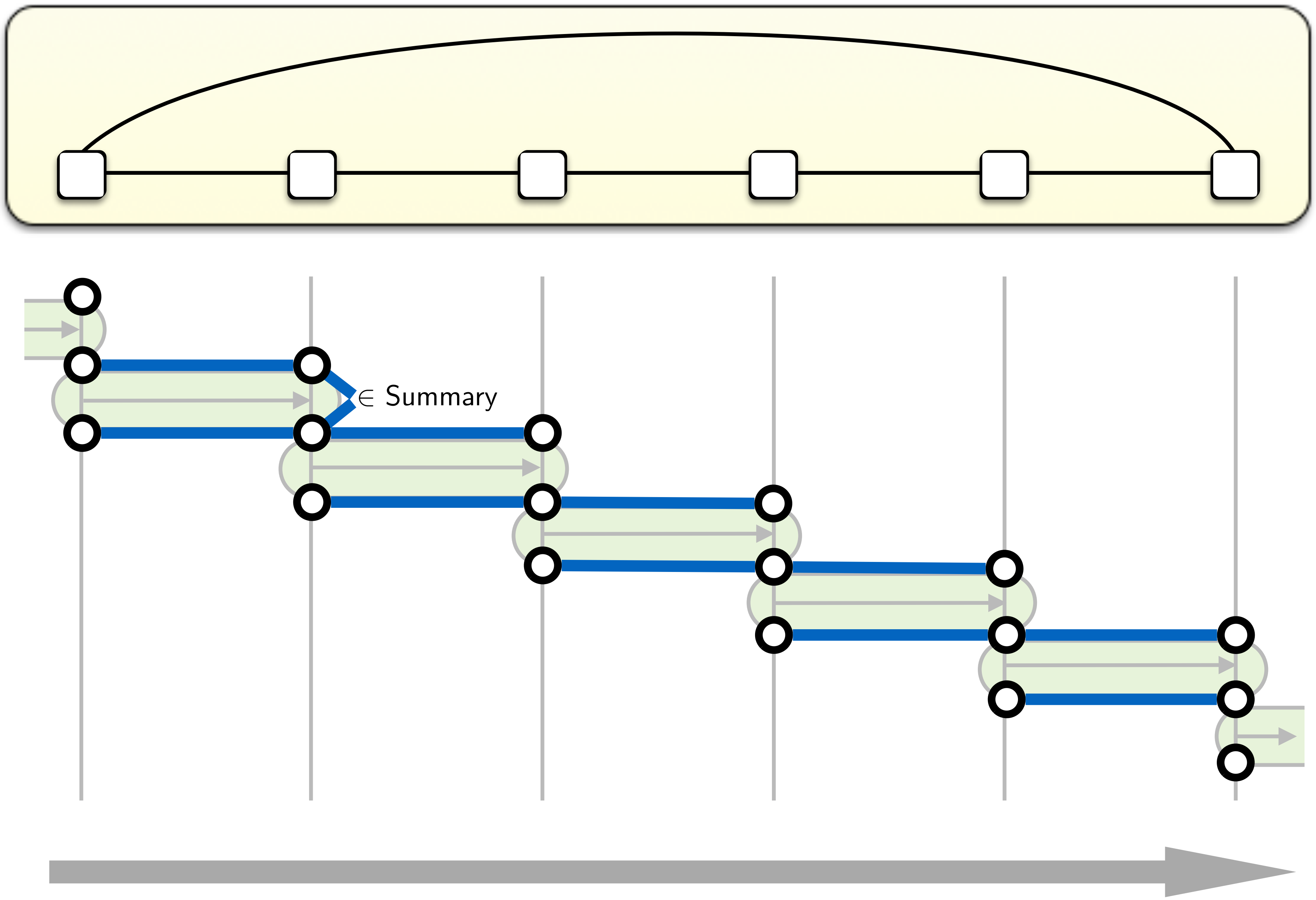
- Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$

Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?



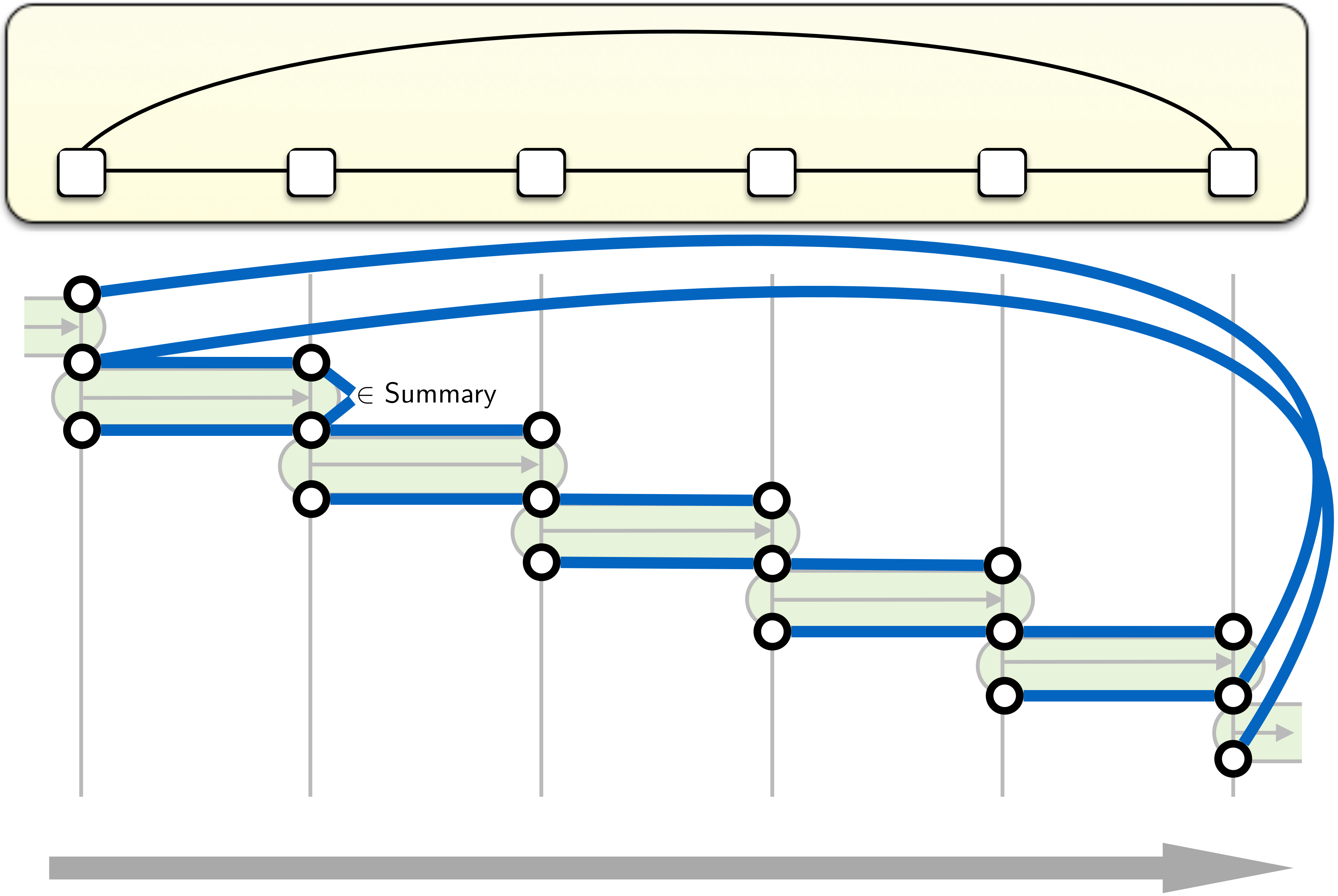
- Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

Input: PCA $\mathcal{A}$; $k \in \mathbb{N}$

Question: Does $L(\mathcal{A})$ contain some k-bounded behavior ?



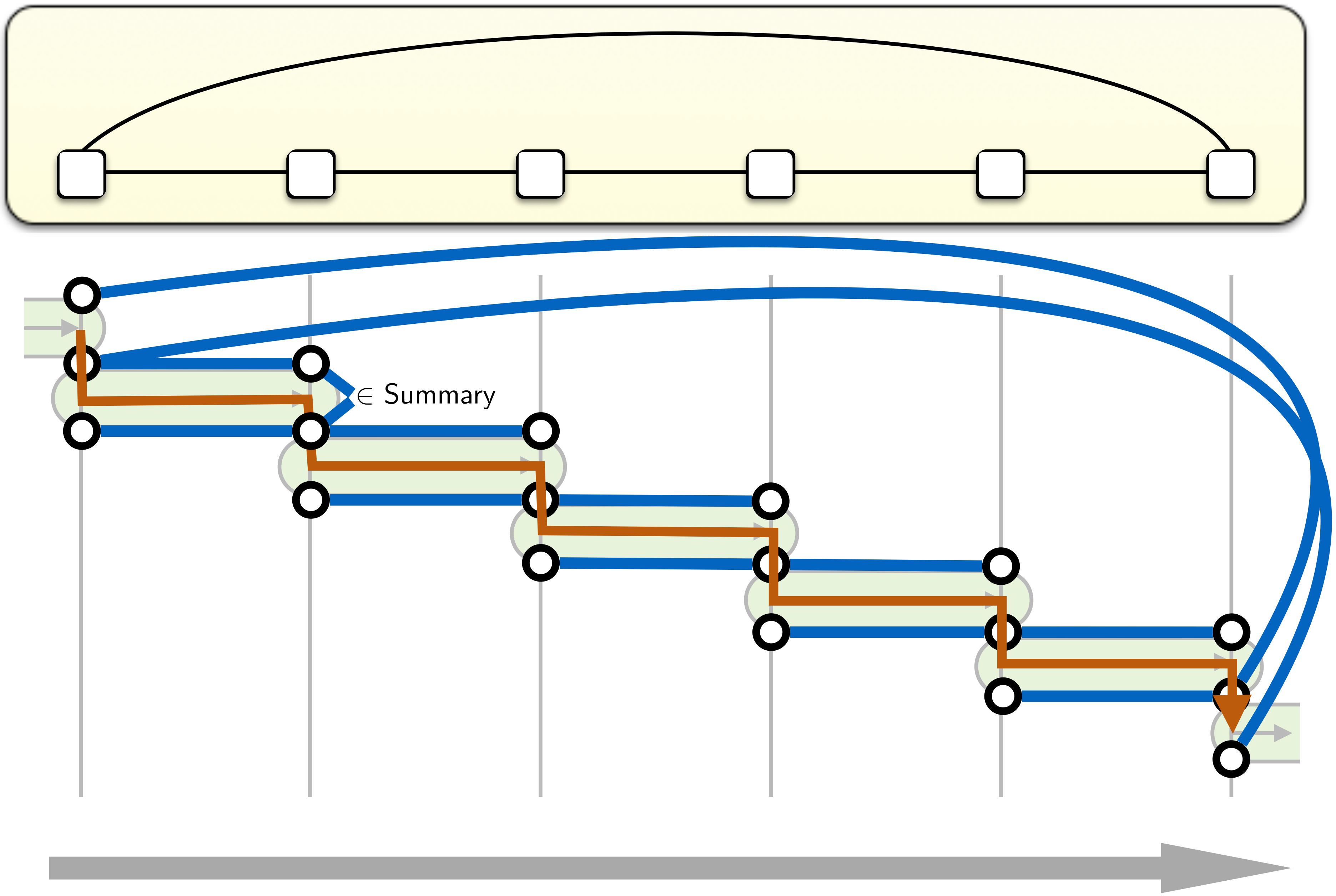
- Finite automaton guesses local states & checks membership in summaries.

Theorem [B.-Gastin-Schubert; RP 2014]:

Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

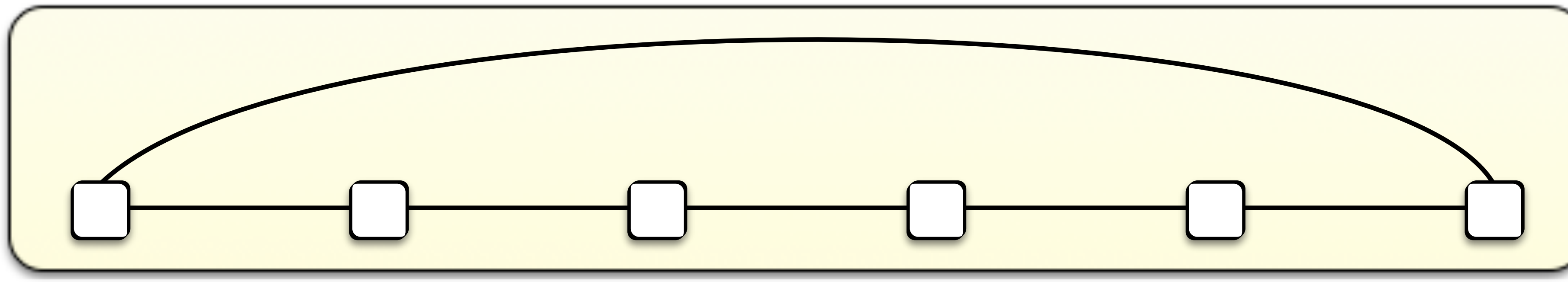
Input:	PCA $\mathcal{A}$; $k \in \mathbb{N}$
Question:	Does $L(\mathcal{A})$ contain some k-bounded behavior ?

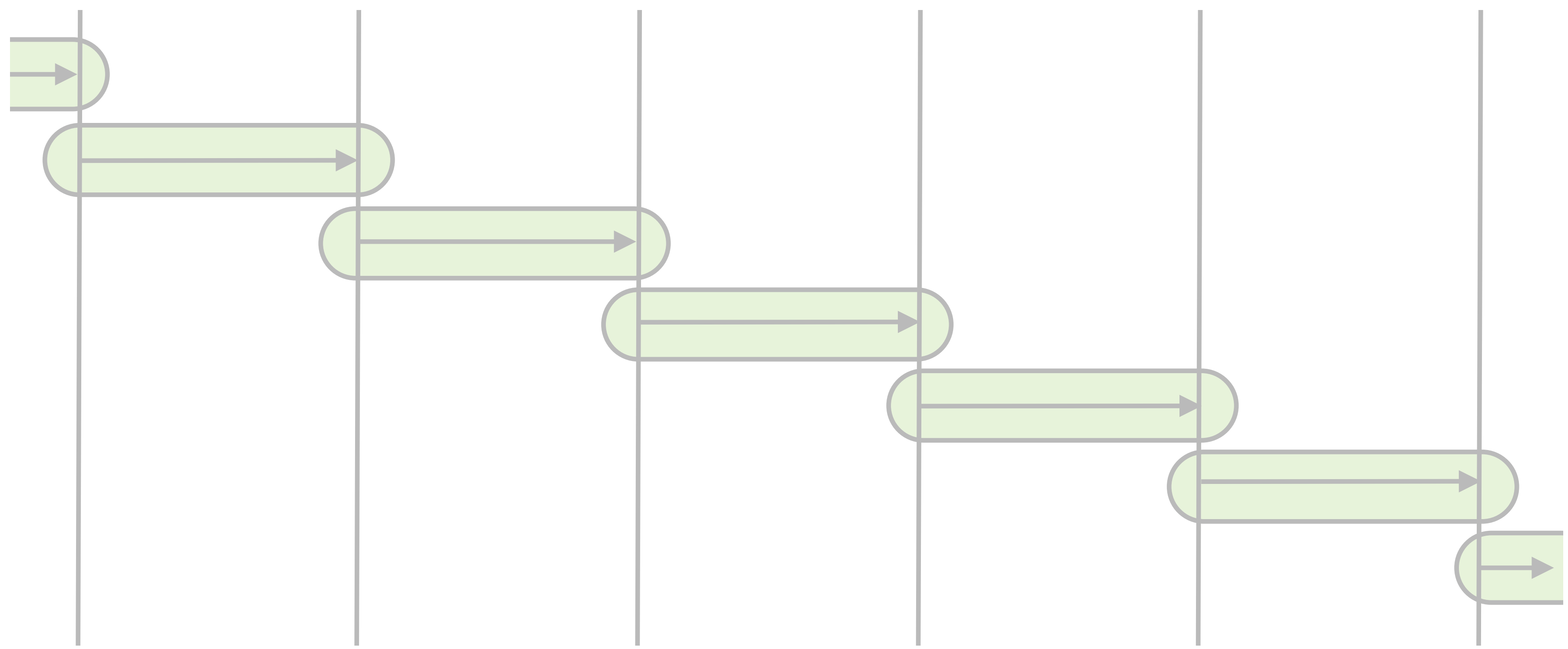


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!

Theorem [B.-Gastin-Schubert; RP 2014]:
Context-bounded nonemptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

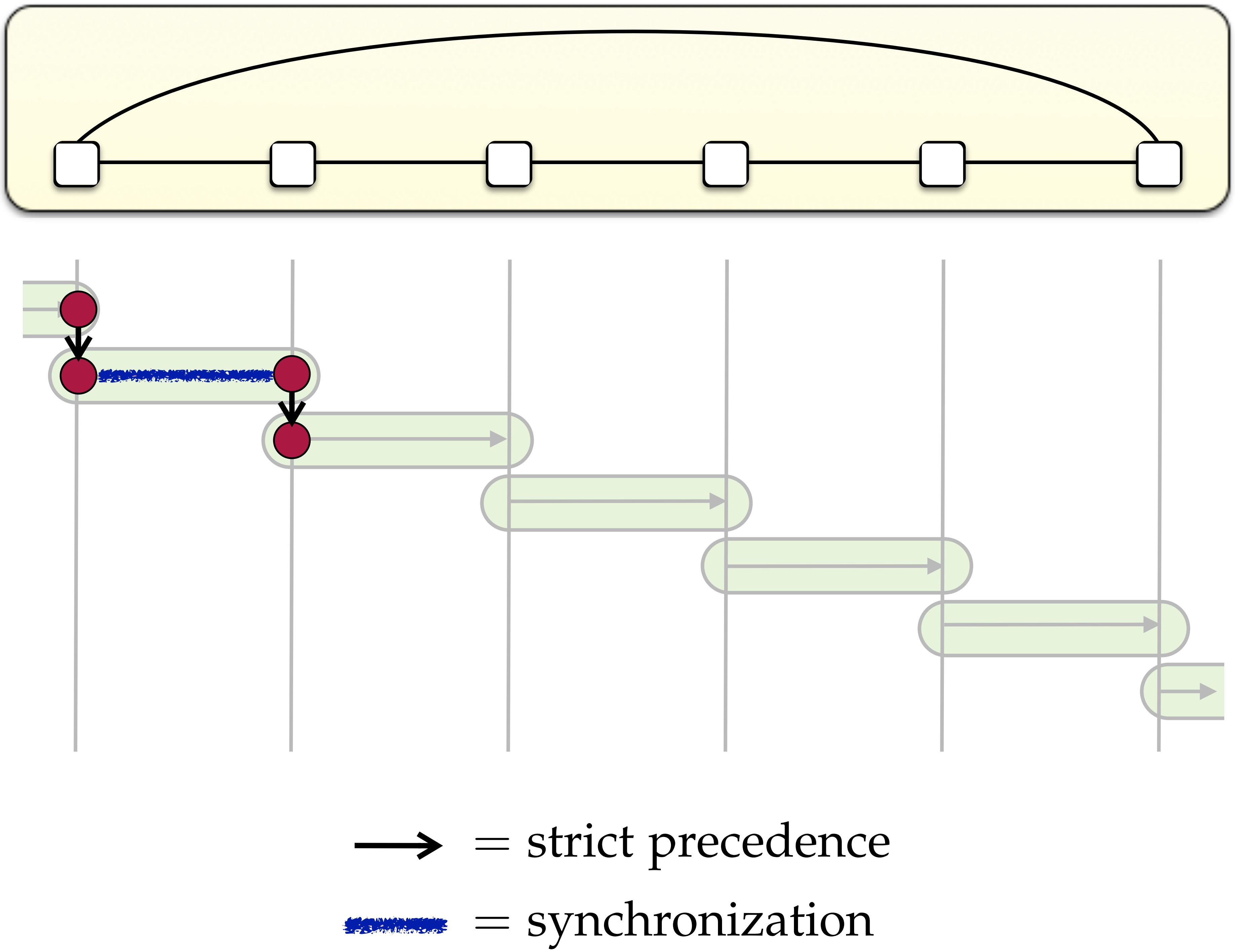




- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

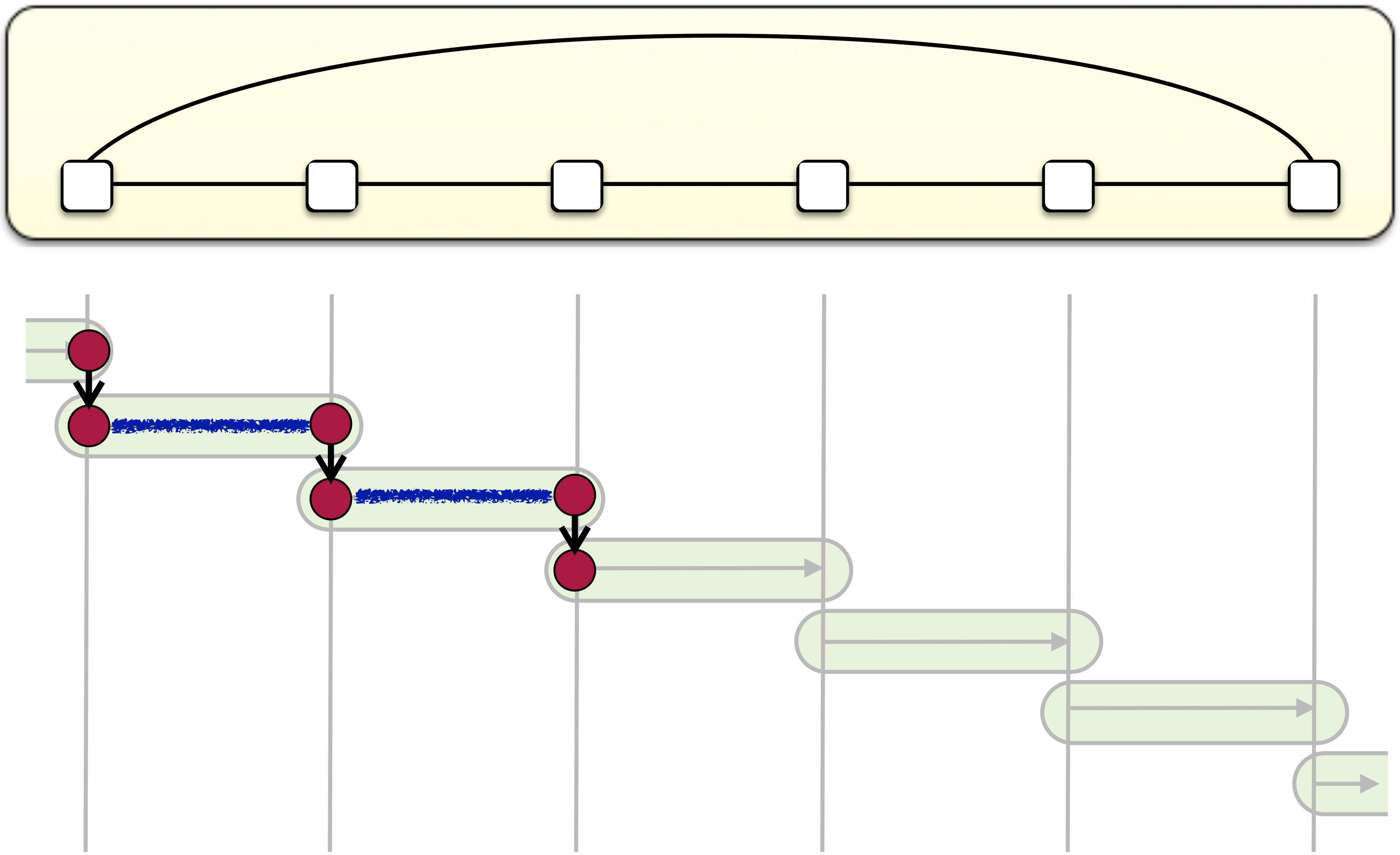
Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem



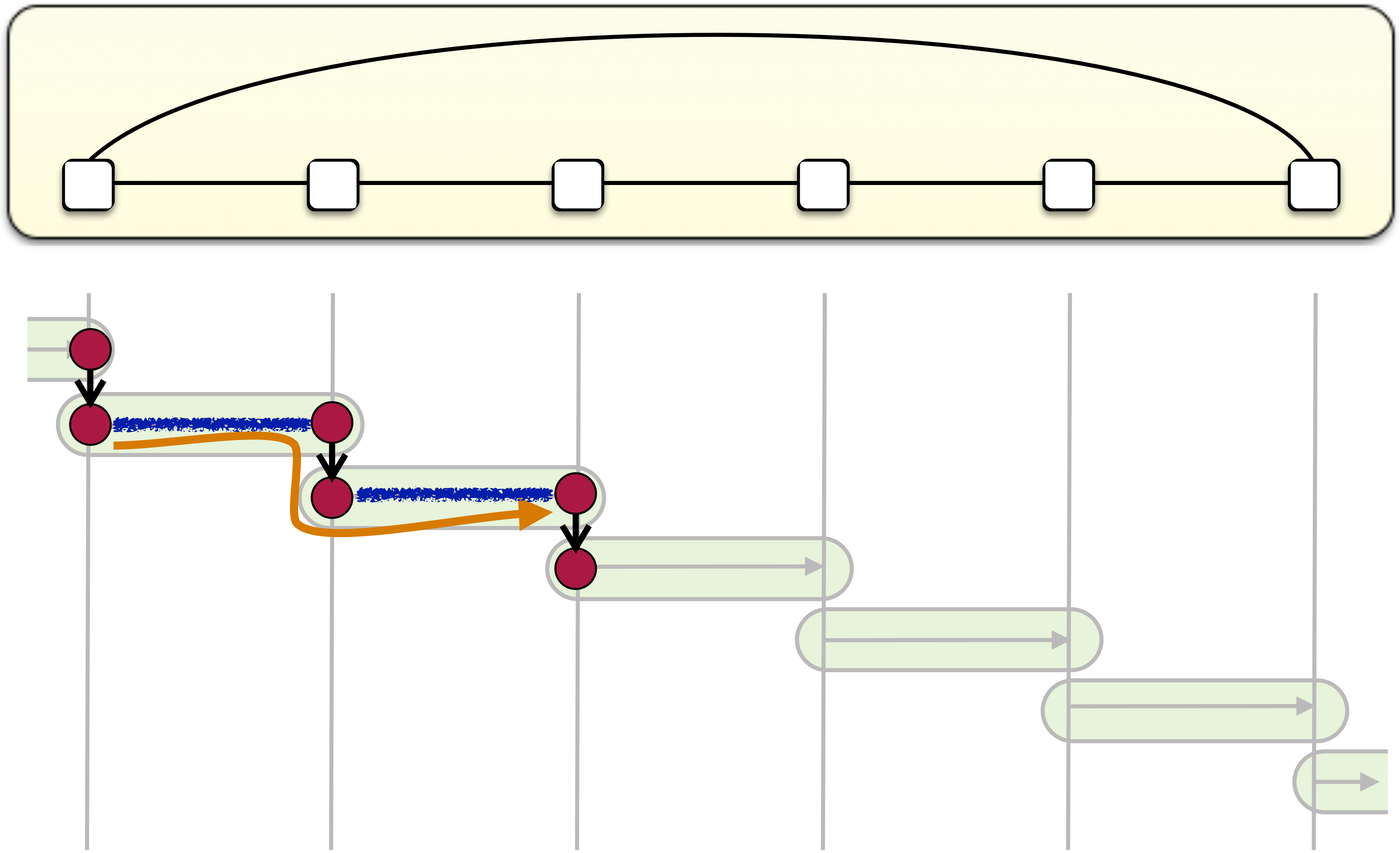
- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

$\longrightarrow$ = strict precedence

= synchronization

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem



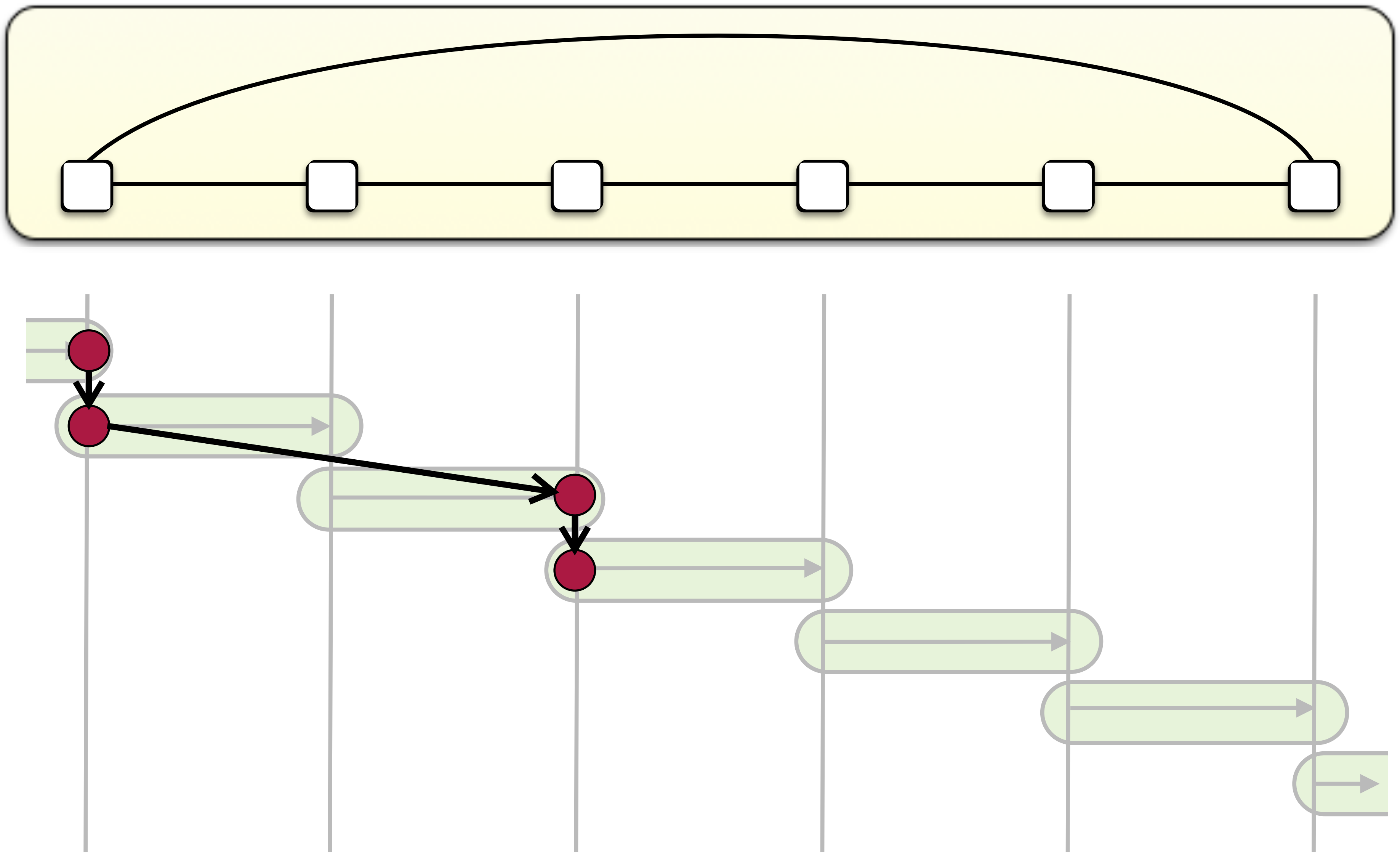
- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

$\longrightarrow$ = strict precedence

= synchronization

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem



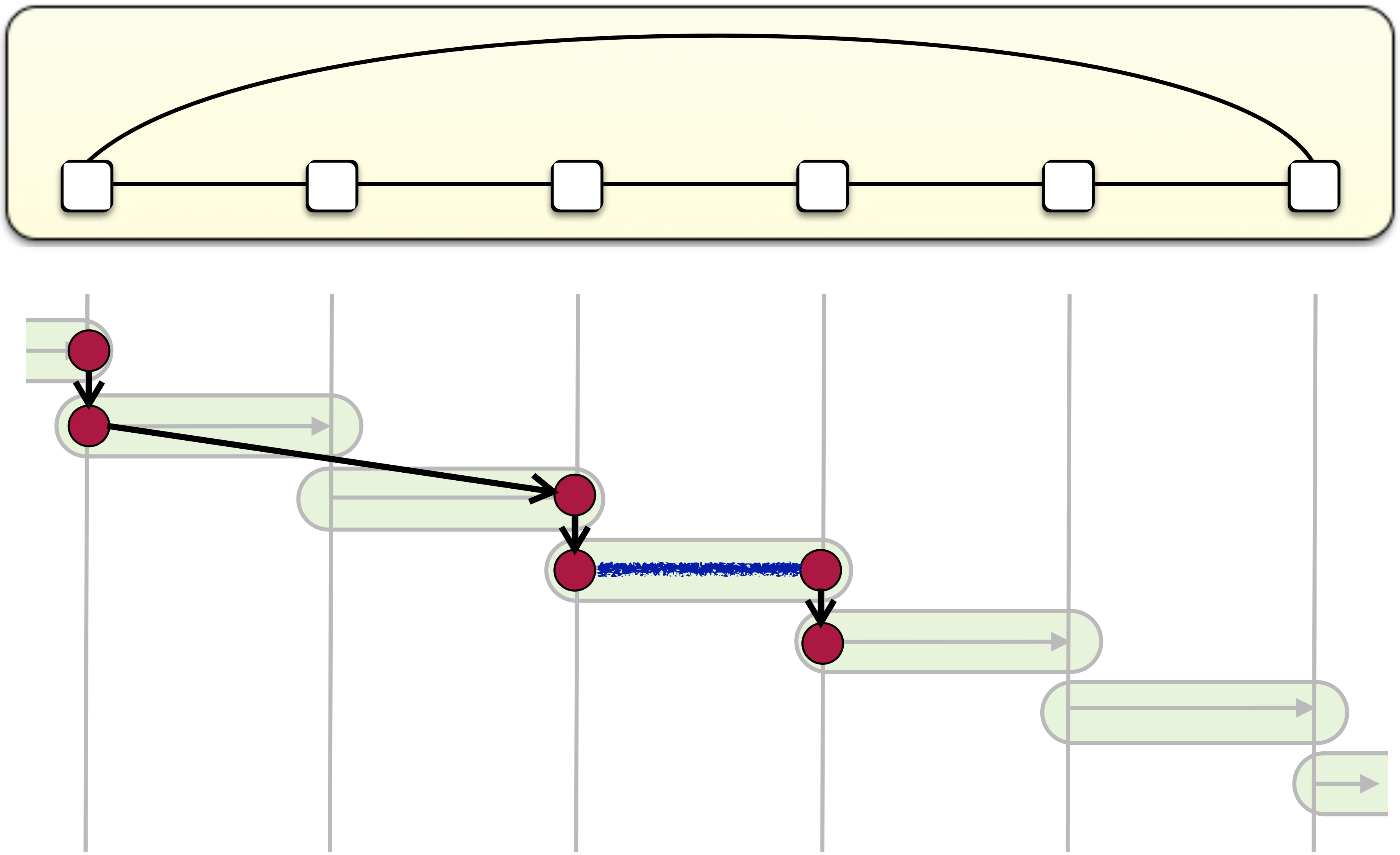
- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

$\longrightarrow$ = strict precedence

 = synchronization

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem



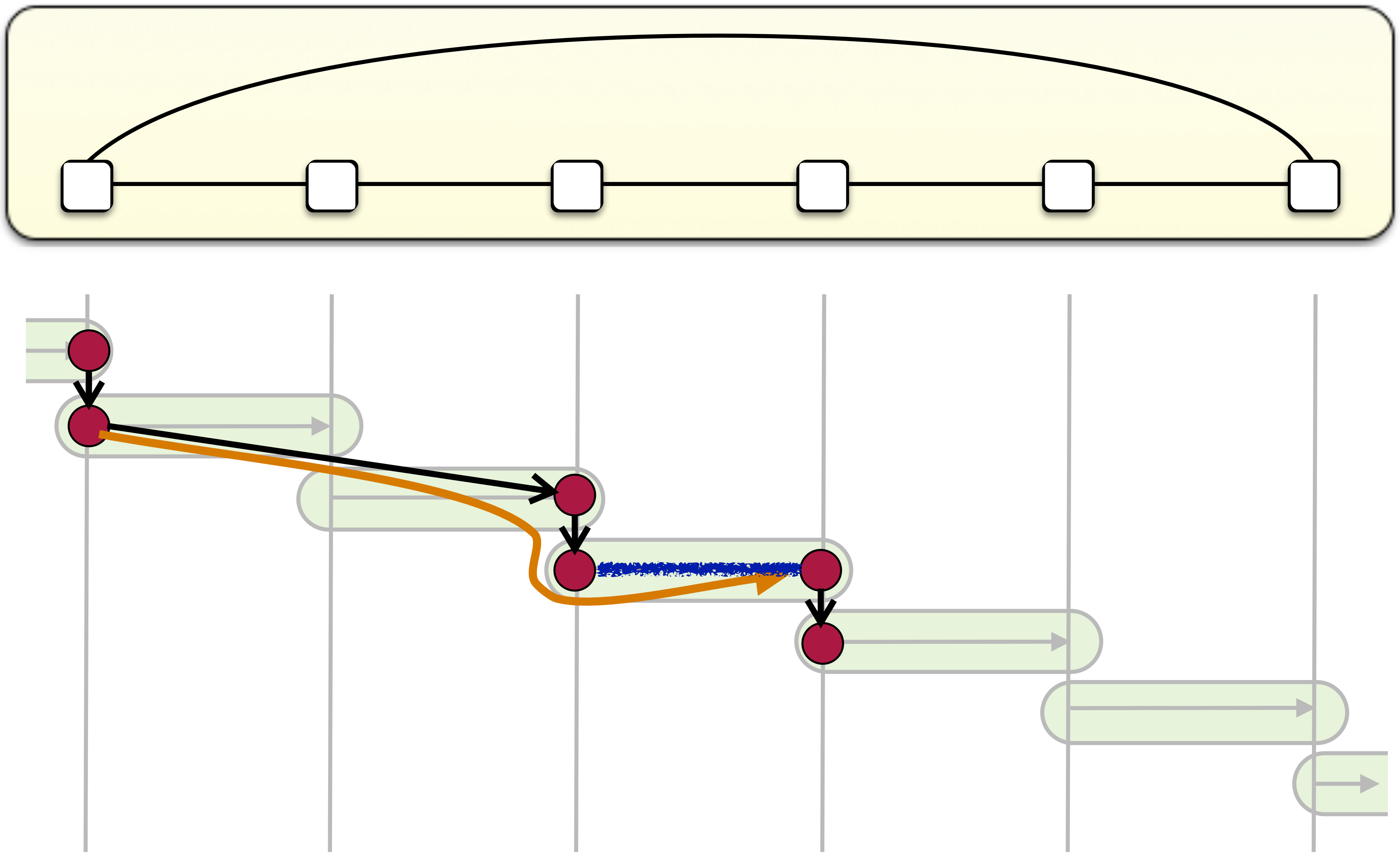
- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

$\longrightarrow$ = strict precedence

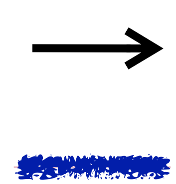
= synchronization

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

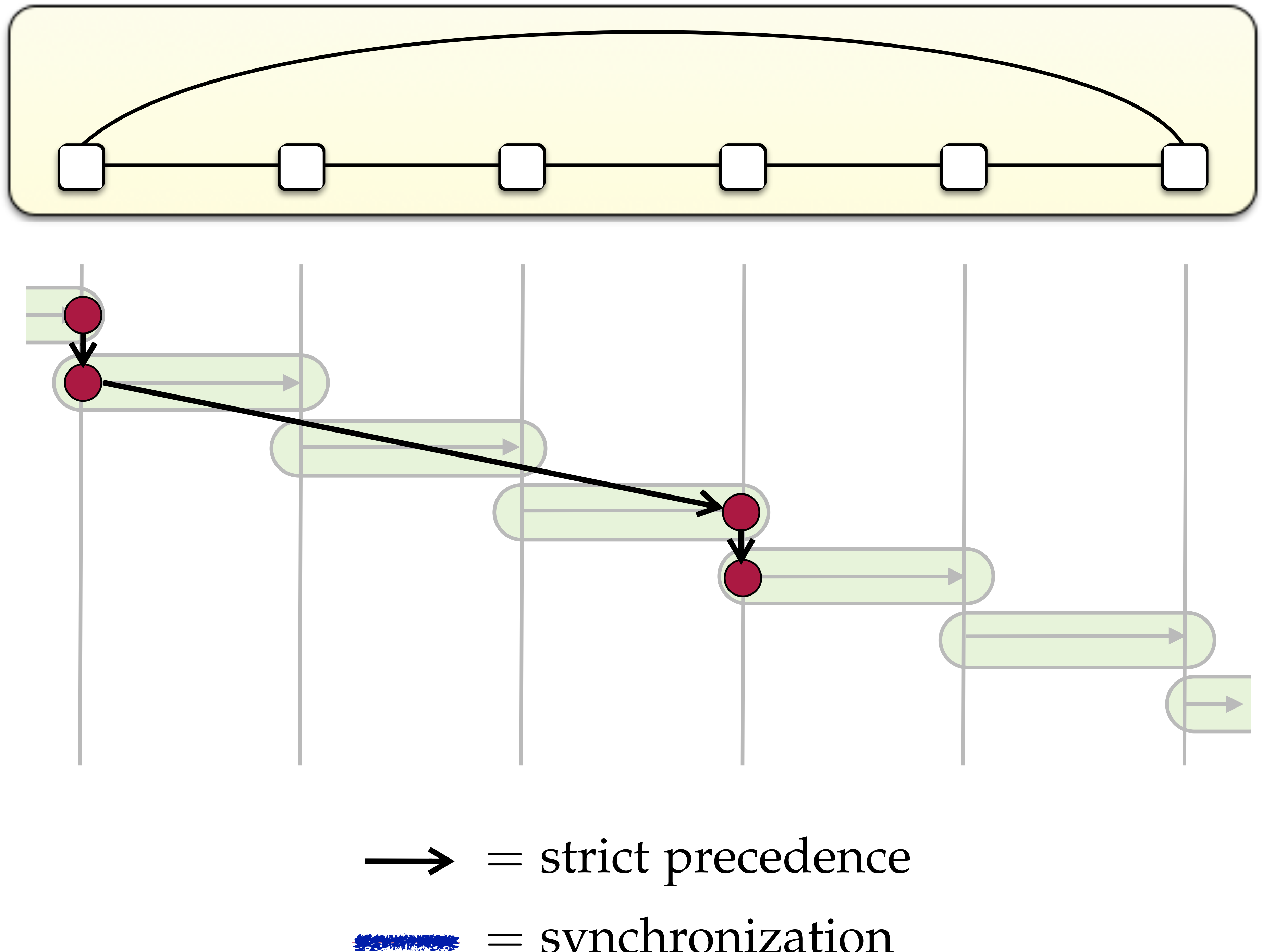


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.



Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

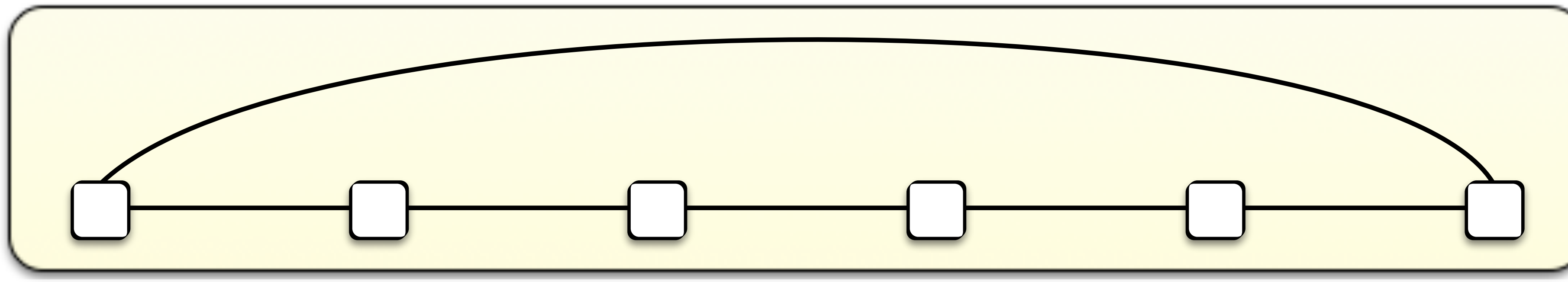
Context-Bounded Nonemptiness Problem



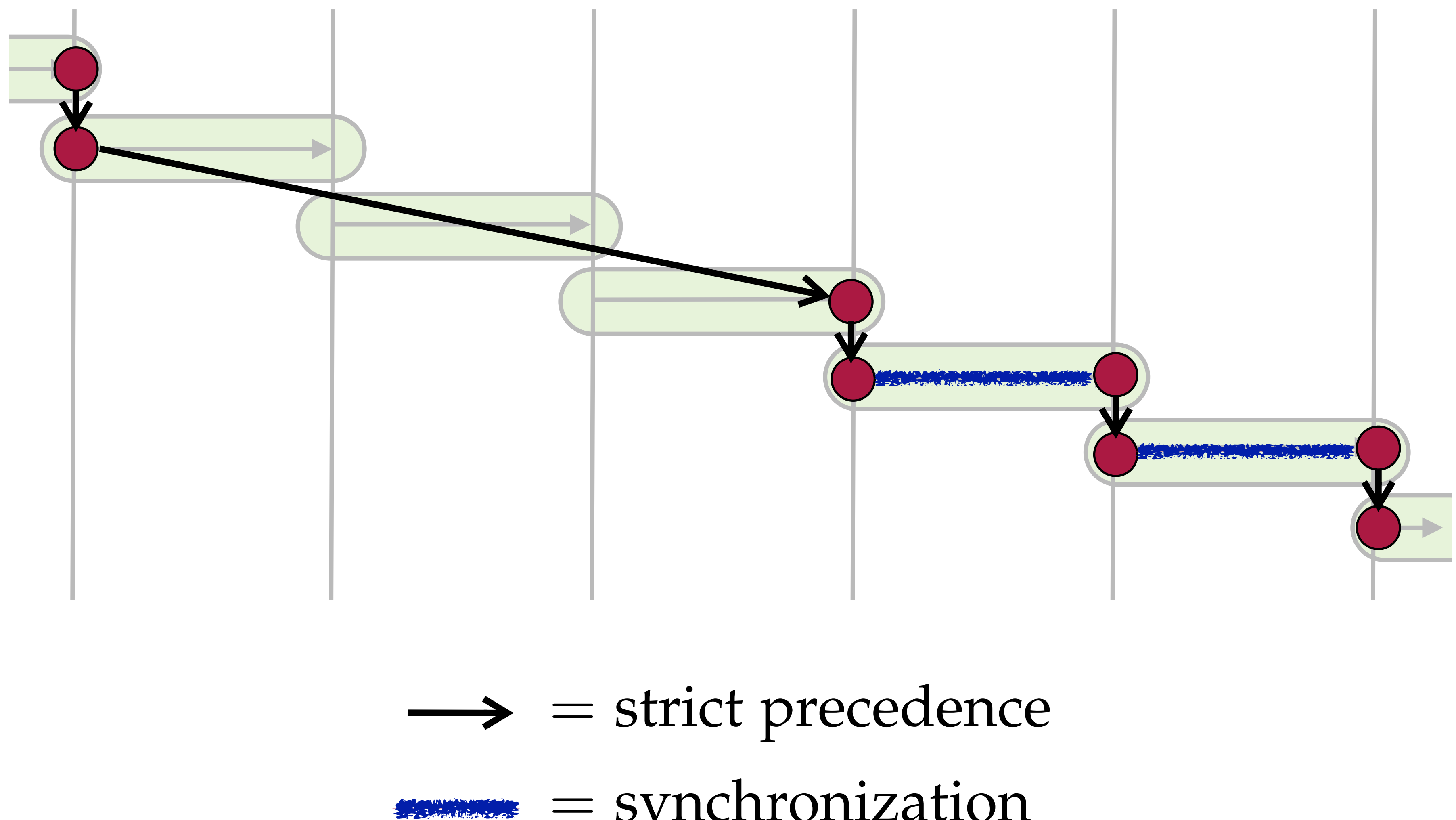
- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

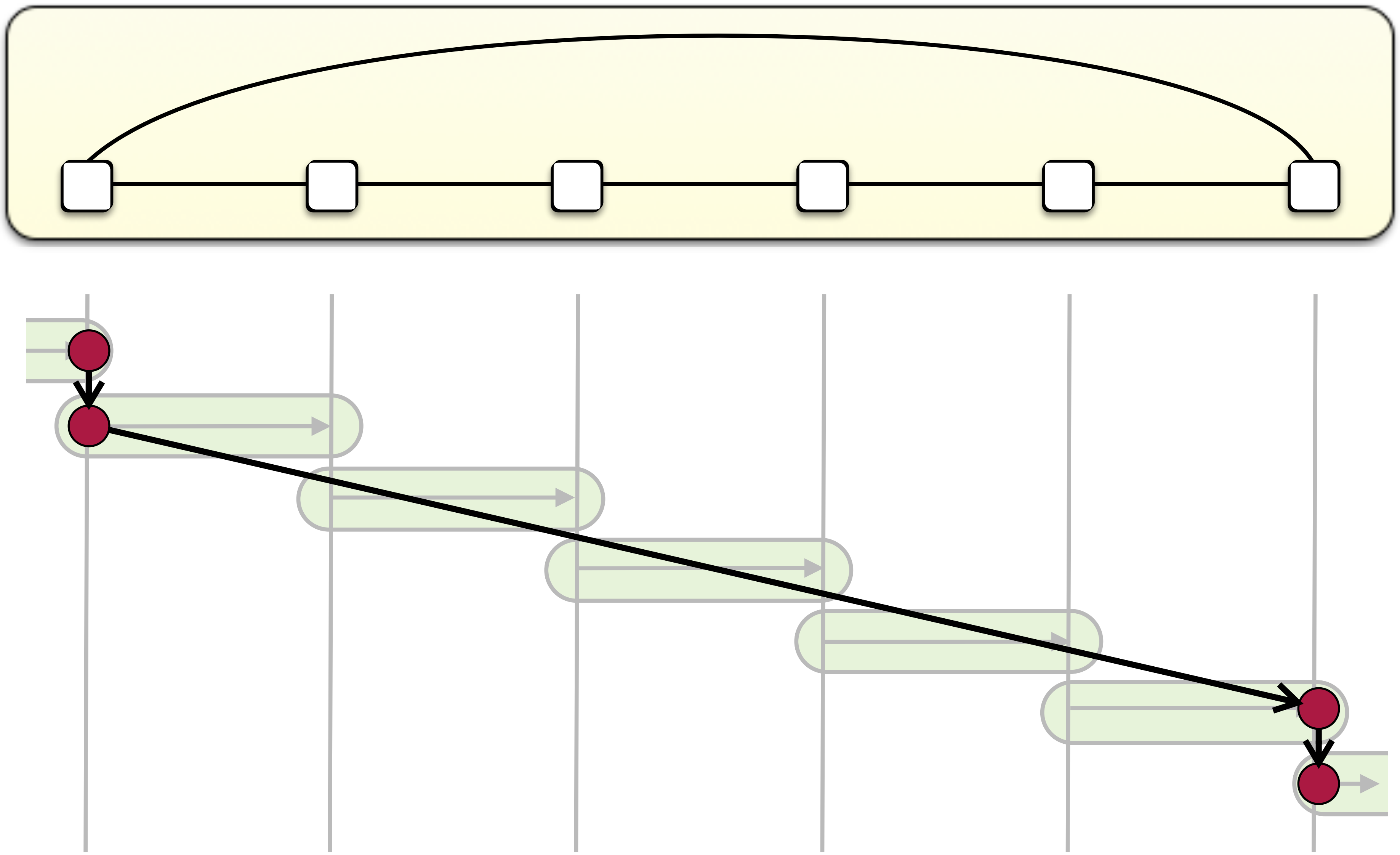


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

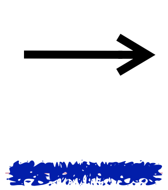


Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

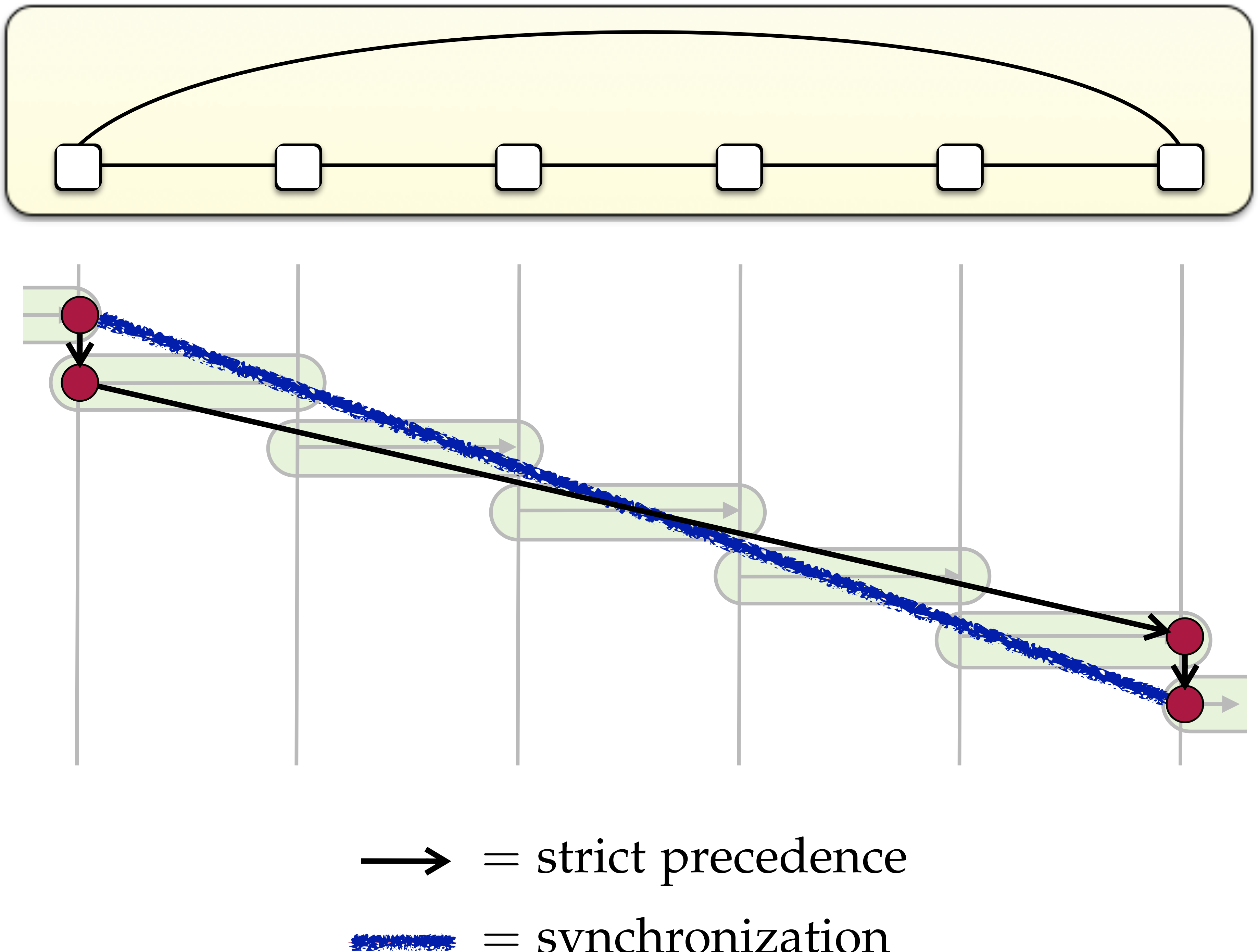


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.



Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

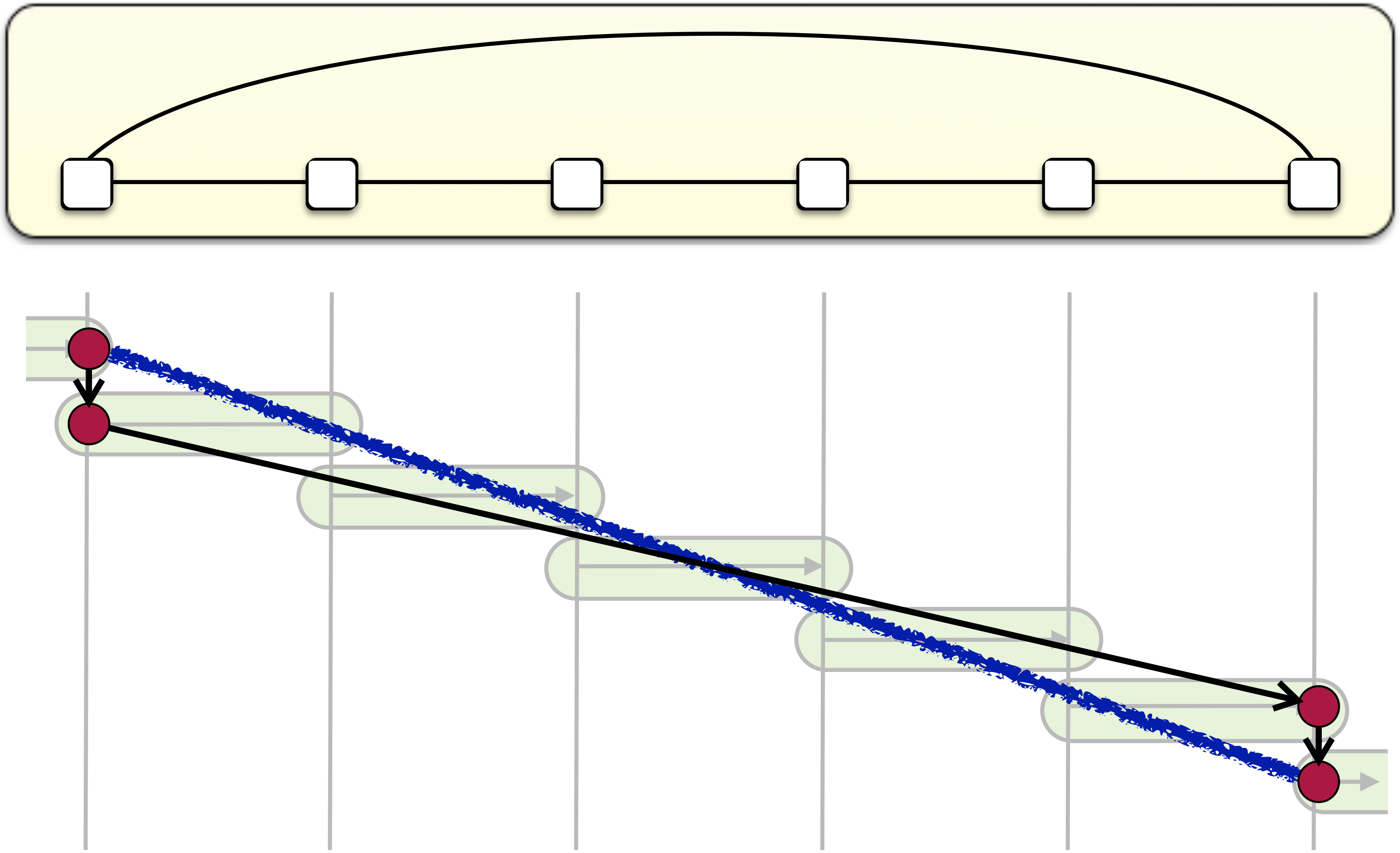
Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

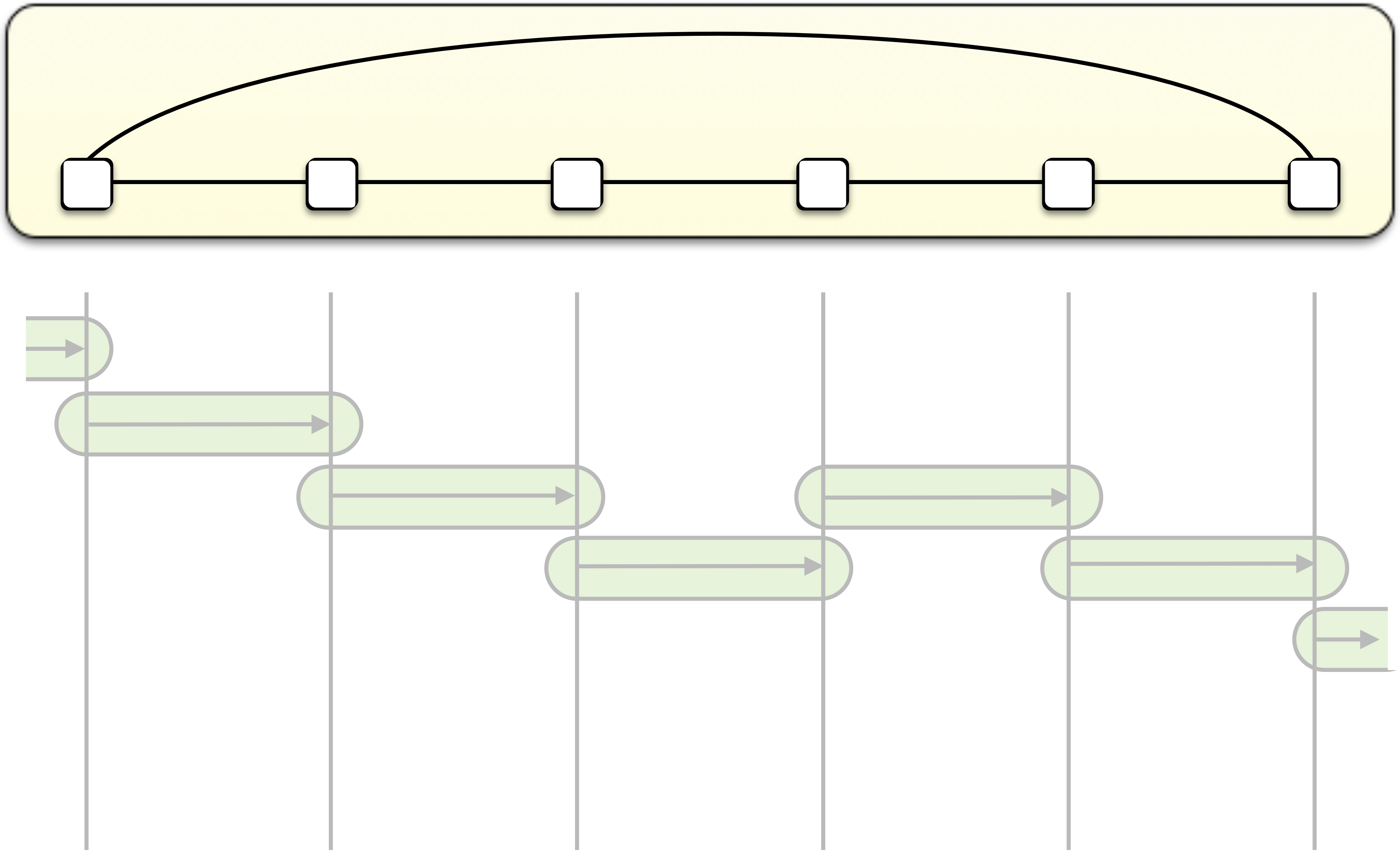


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

strict cycle $\Longrightarrow$ run is not accepting

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

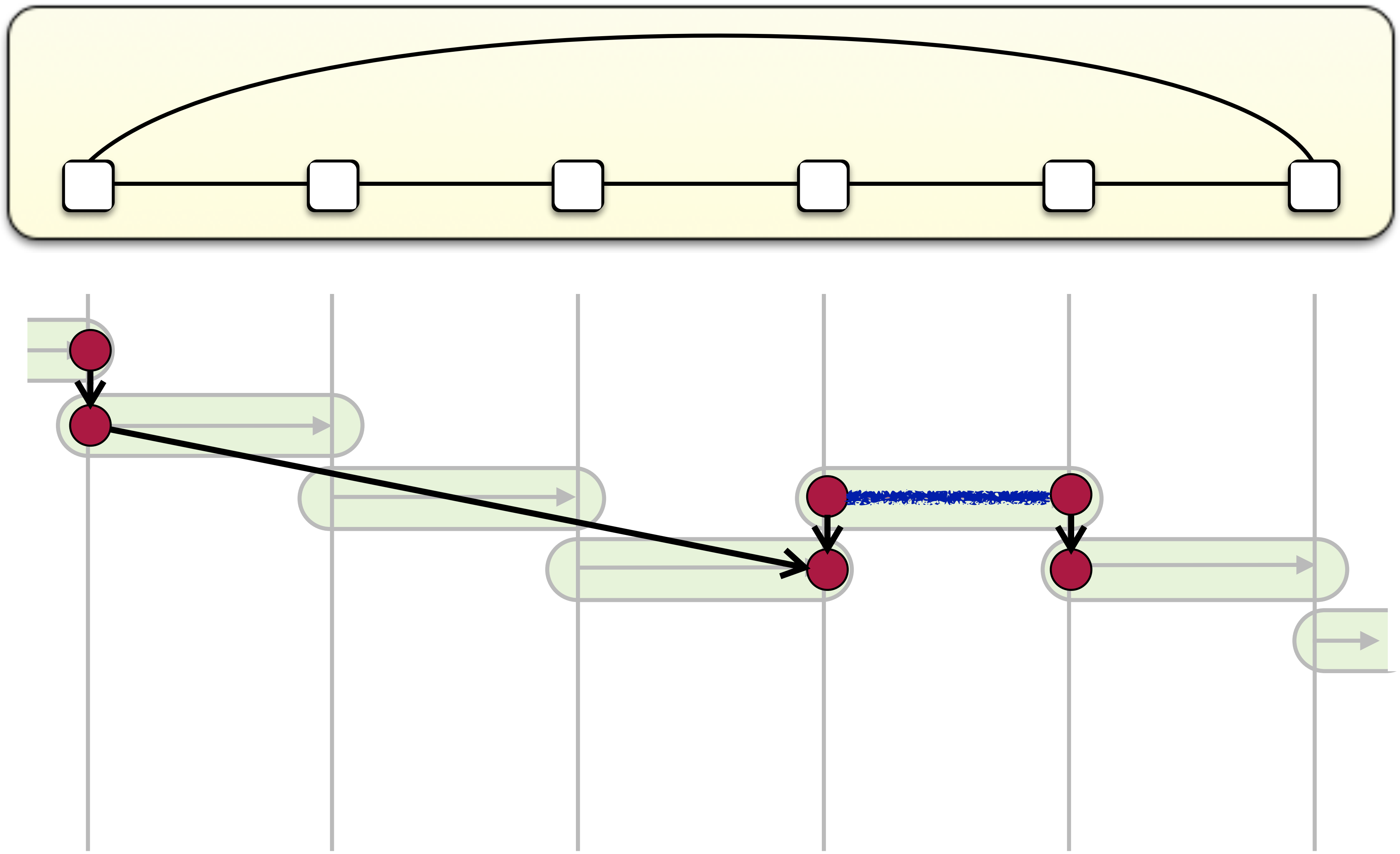
Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

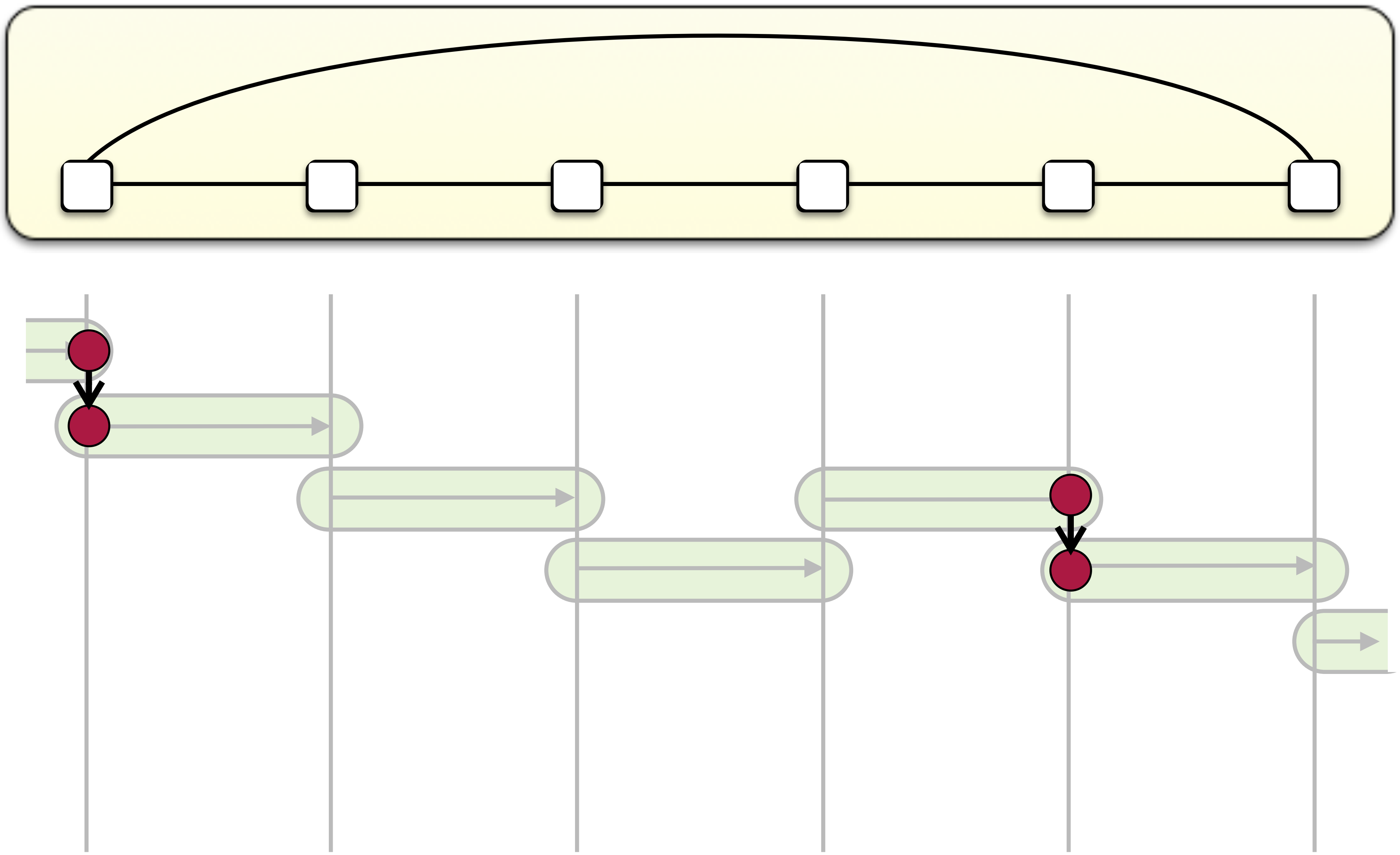
Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

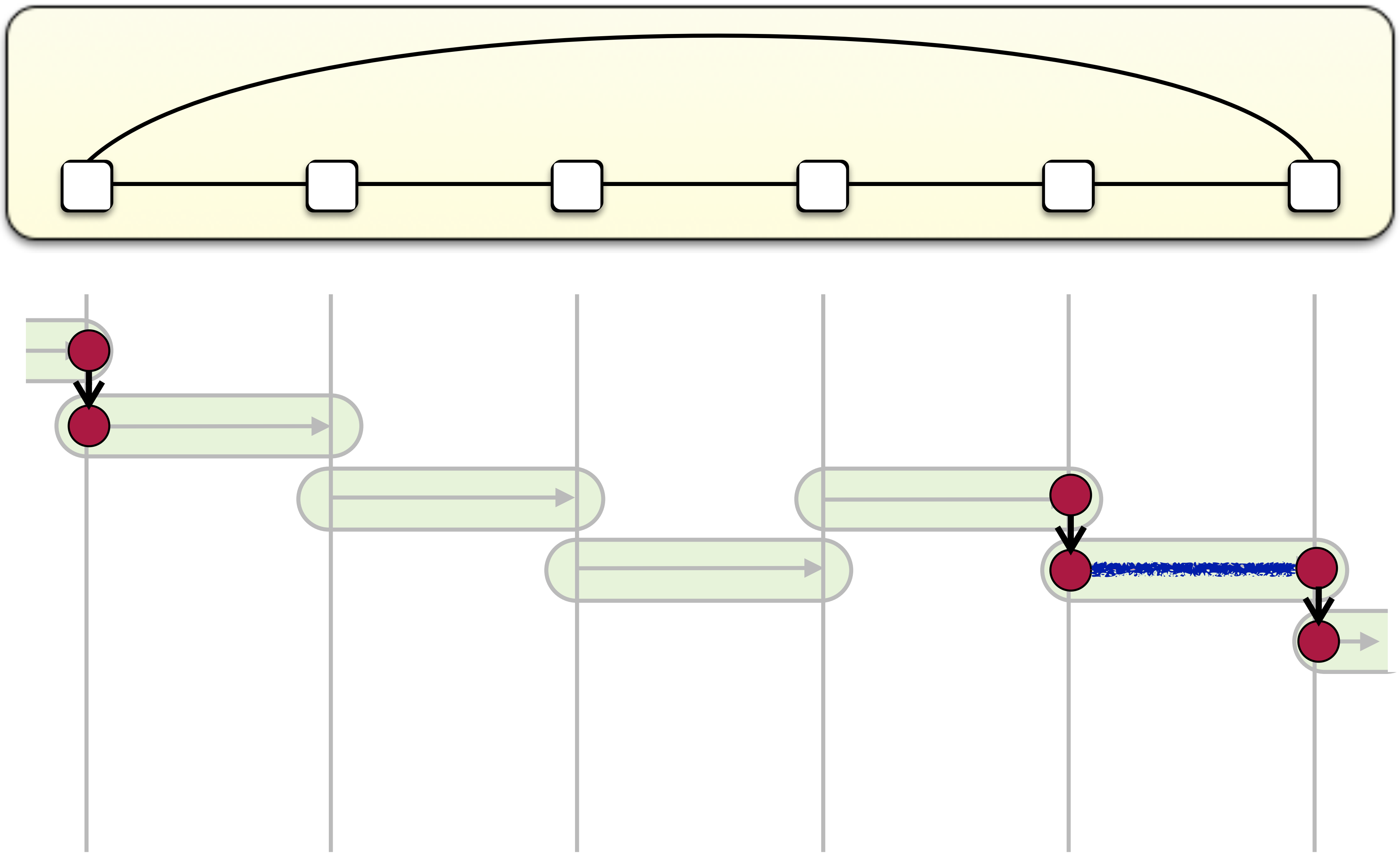
Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- → Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

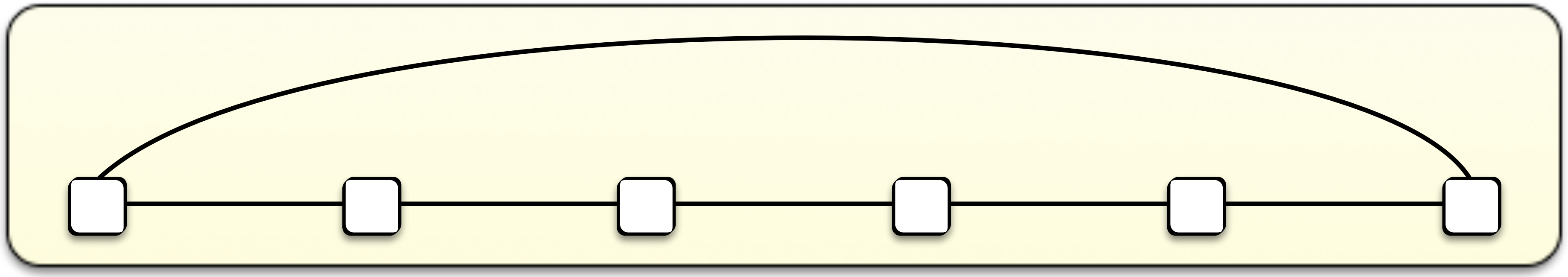
Context-Bounded Nonemptiness Problem

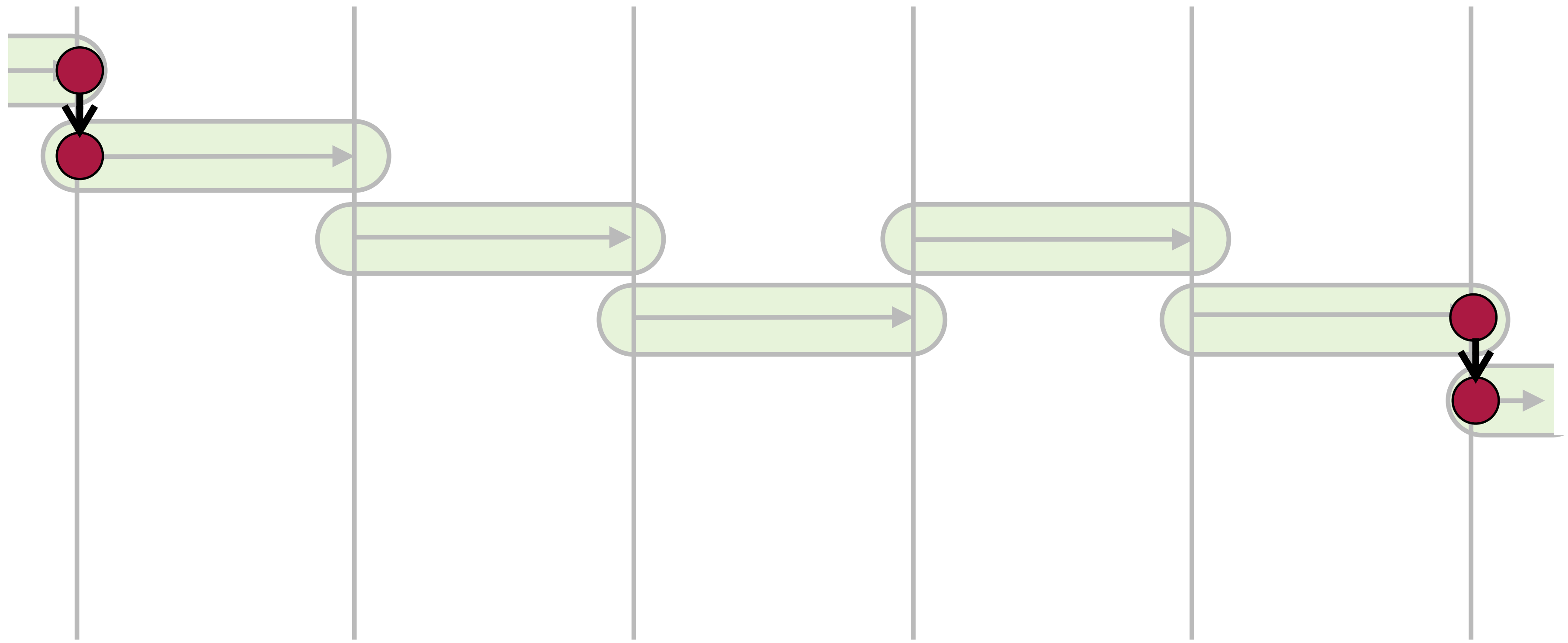


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ⇒ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

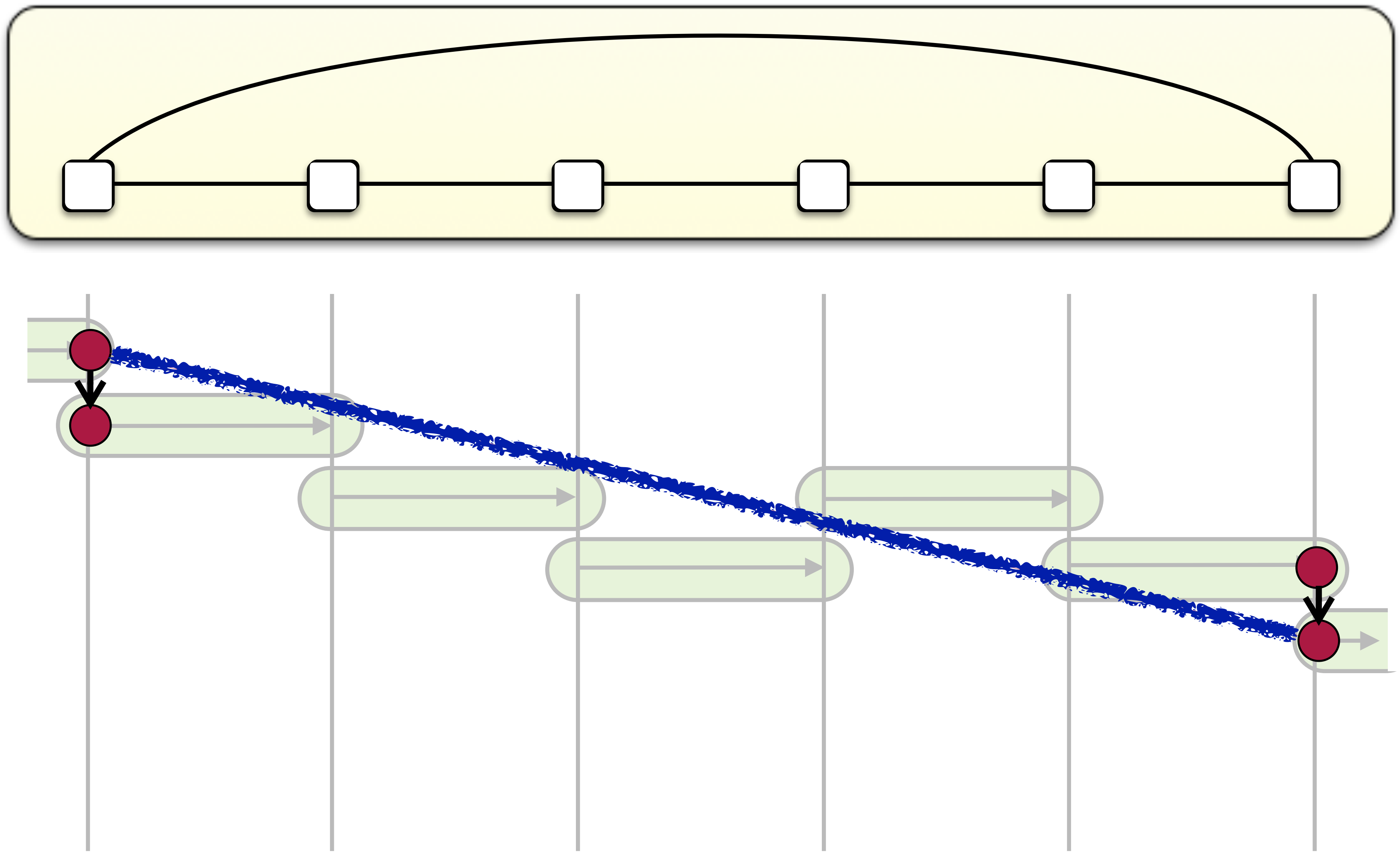




- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

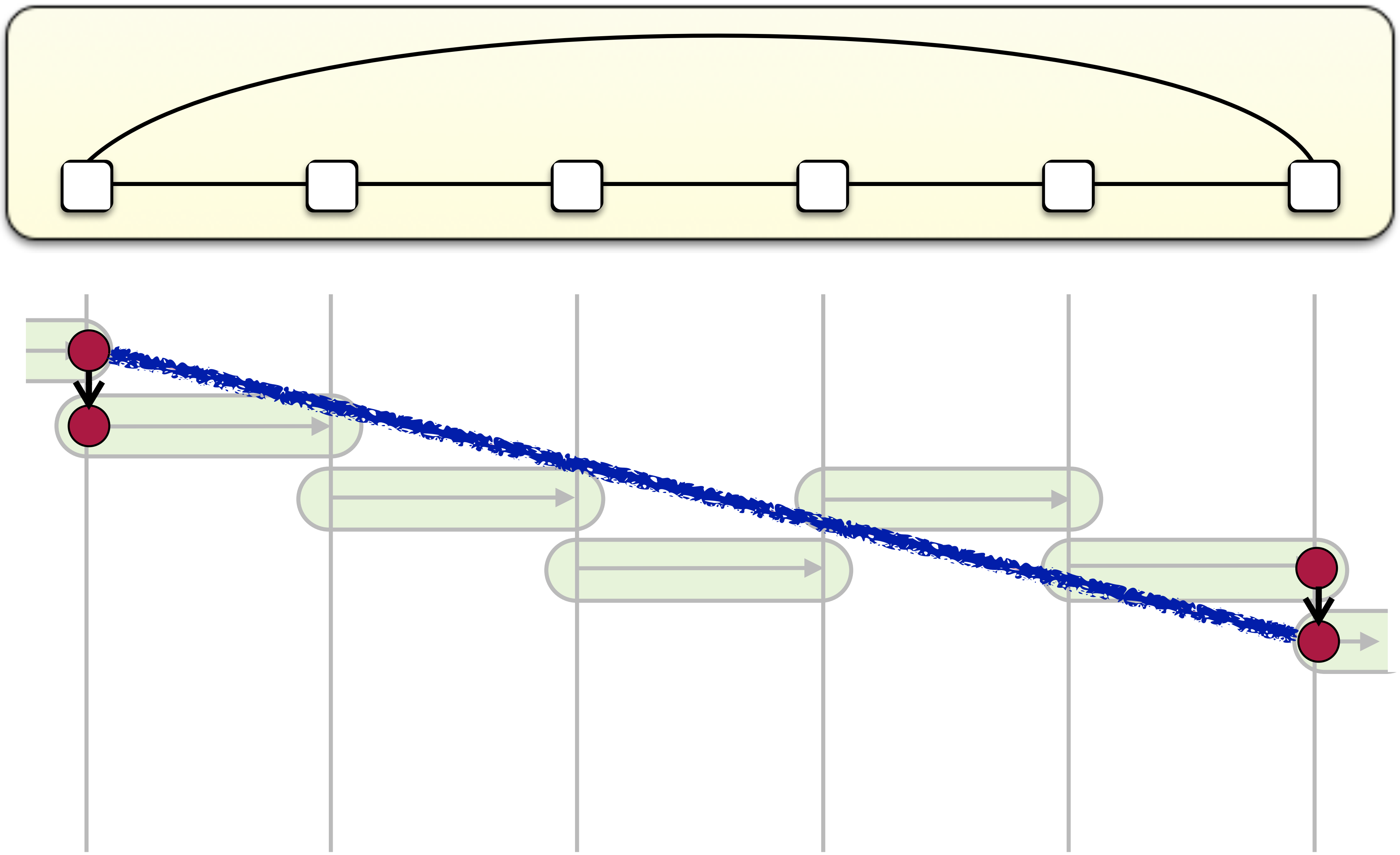
Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem

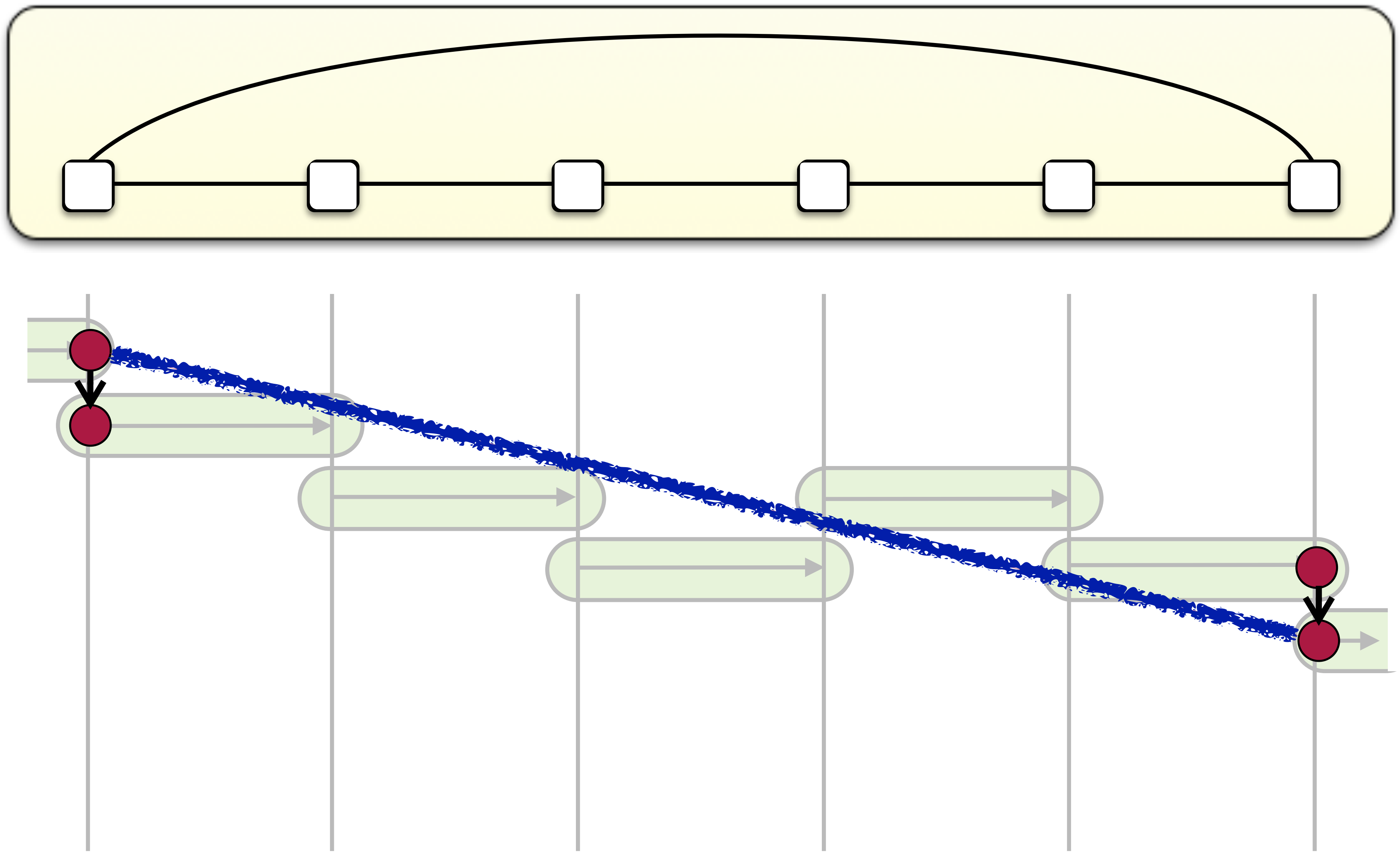


- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- ➡ Check causal dependencies.

no strict cycle $\implies$ run is accepting

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Context-Bounded Nonemptiness Problem



- Finite automaton guesses local states & checks membership in summaries.
- However, summaries may match locally, but not give rise to an accepting run!
- Check causal dependencies.
- Gives PSPACE procedure.

no strict cycle $\Longrightarrow$ run is accepting

Theorem [B.-Gastin-Schubert 2014]:
Context-bounded emptiness checking over rings is PSPACE-complete.

Summary of Results

Theorem:
Context-bounded PCAs are **complementable** and expressively equivalent to **MSO logic**.

Summary of Results

Theorem:
Context-bounded PCAs are **complementable** and expressively equivalent to **MSO logic**.

Theorem:
Context-bounded **nonemptiness checking is decidable** over rings and trees.

Summary of Results

Theorem:
Context-bounded PCAs are **complementable** and expressively equivalent to **MSO logic**.

Theorem:
Context-bounded **nonemptiness checking is decidable** over rings and trees.

Corollary:
Context-bounded **MSO model checking is decidable** over rings and trees.

Summary of Results

Theorem:
Context-bounded PCAs are **complementable** and expressively equivalent to **MSO logic**.

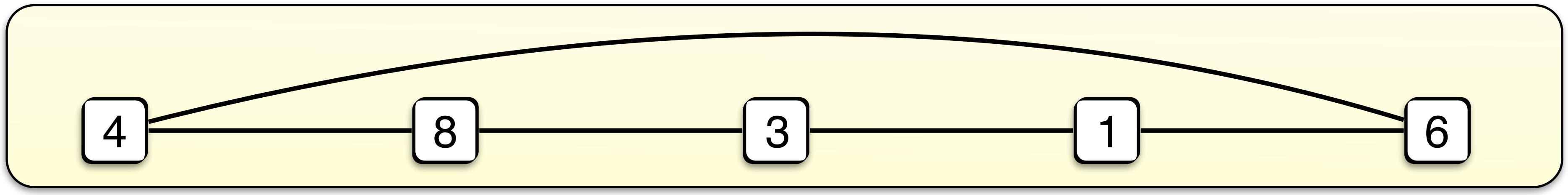
Theorem:
Context-bounded **nonemptiness checking is decidable** over rings and trees.

Corollary:
Context-bounded **MSO model checking is decidable** over rings and trees.

→ Context-bounded PCAs form a robust automata model.

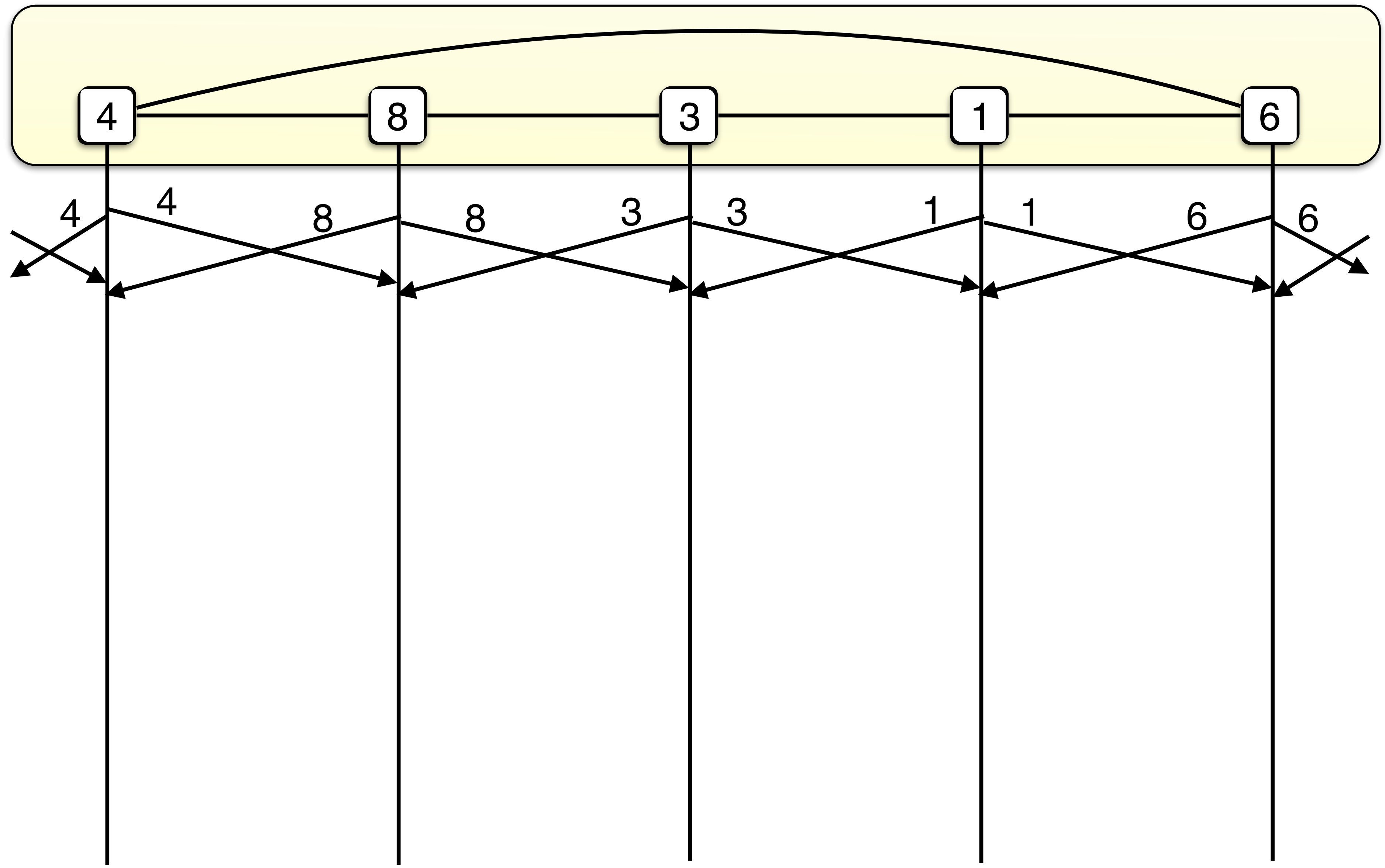
Application to Verification of Distributed Algorithms

Franklin's leader-election protocol (1982)



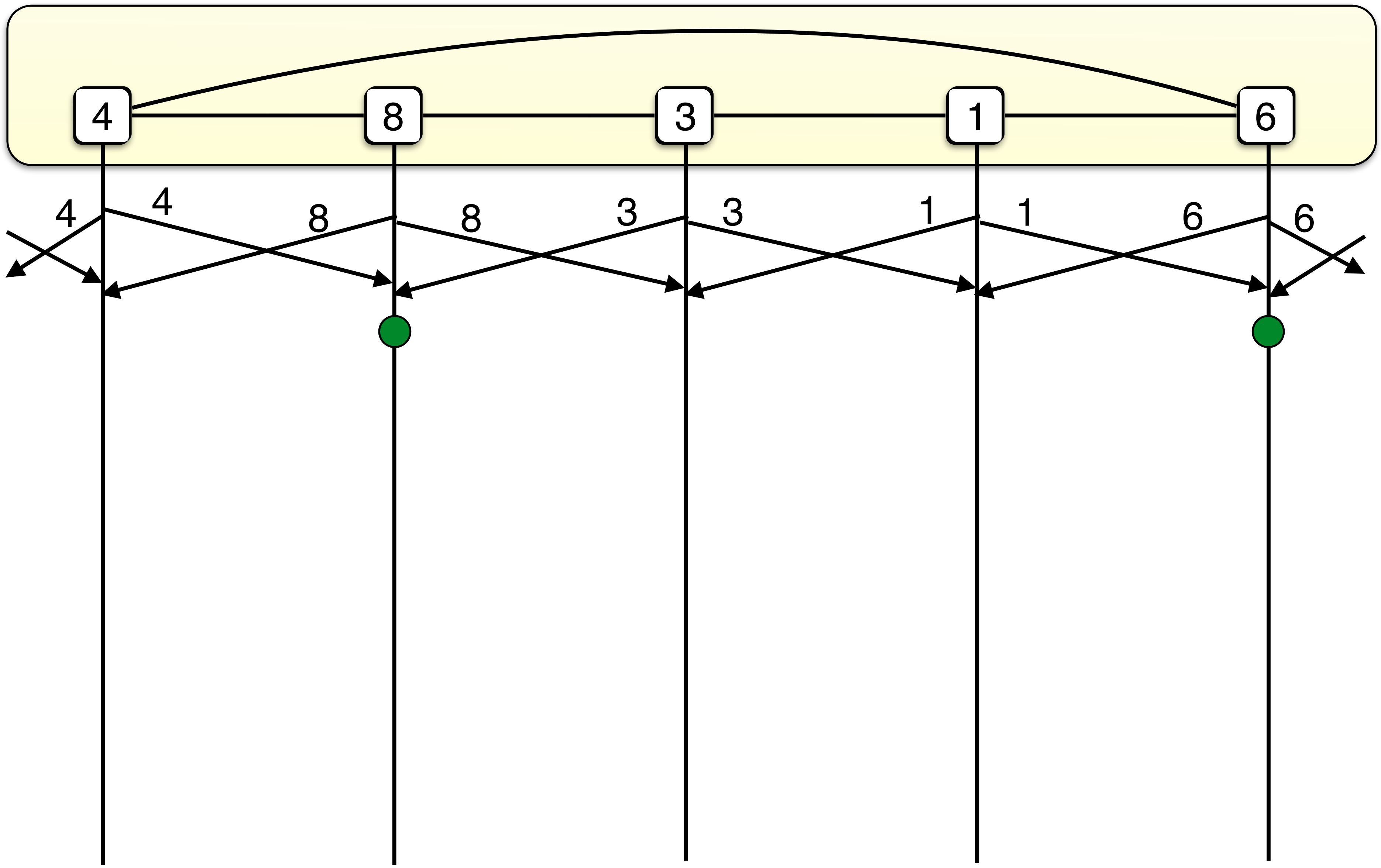
Application to Verification of Distributed Algorithms

Franklin's leader-election protocol (1982)



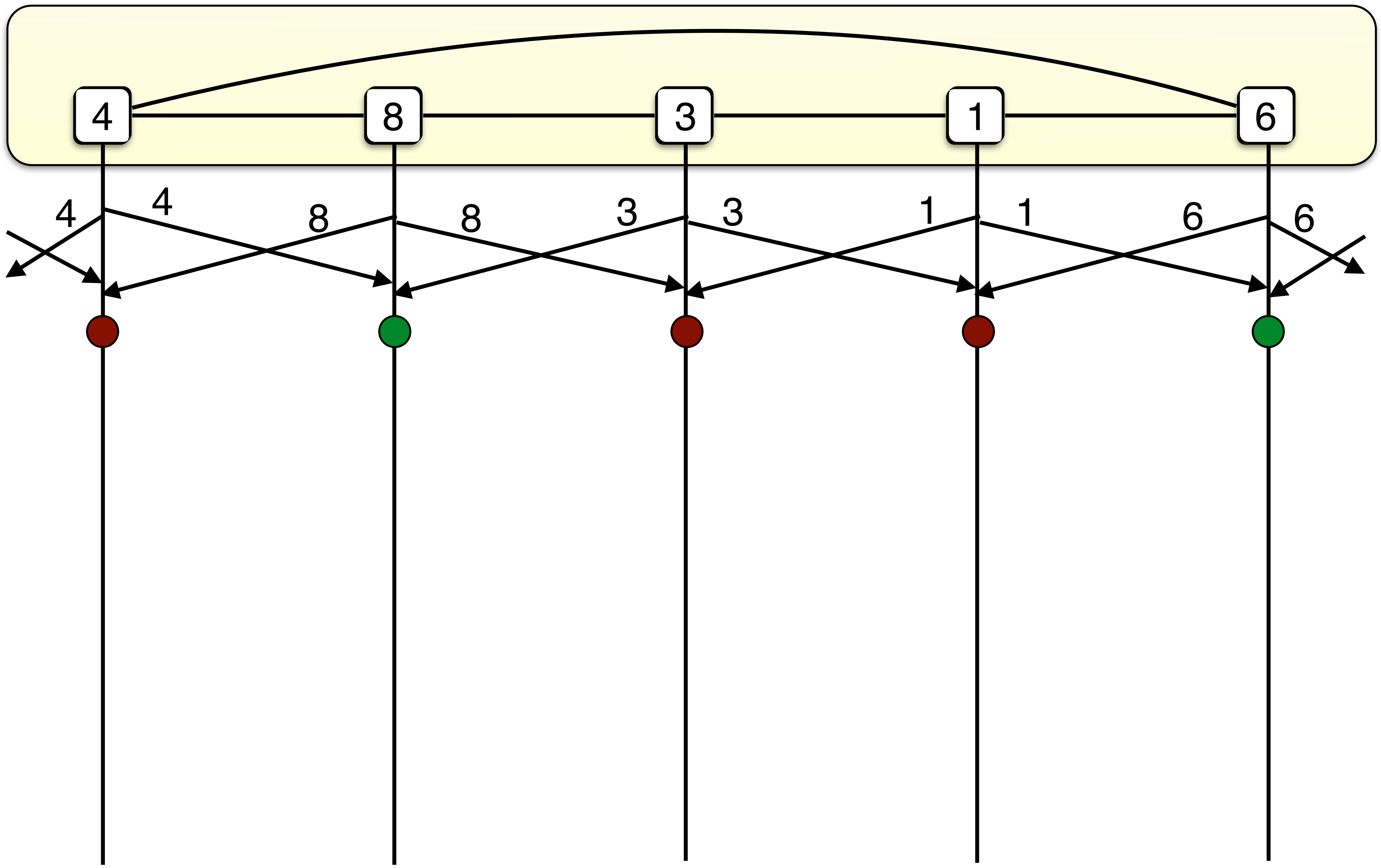
Application to Verification of Distributed Algorithms

Franklin's leader-election protocol (1982)



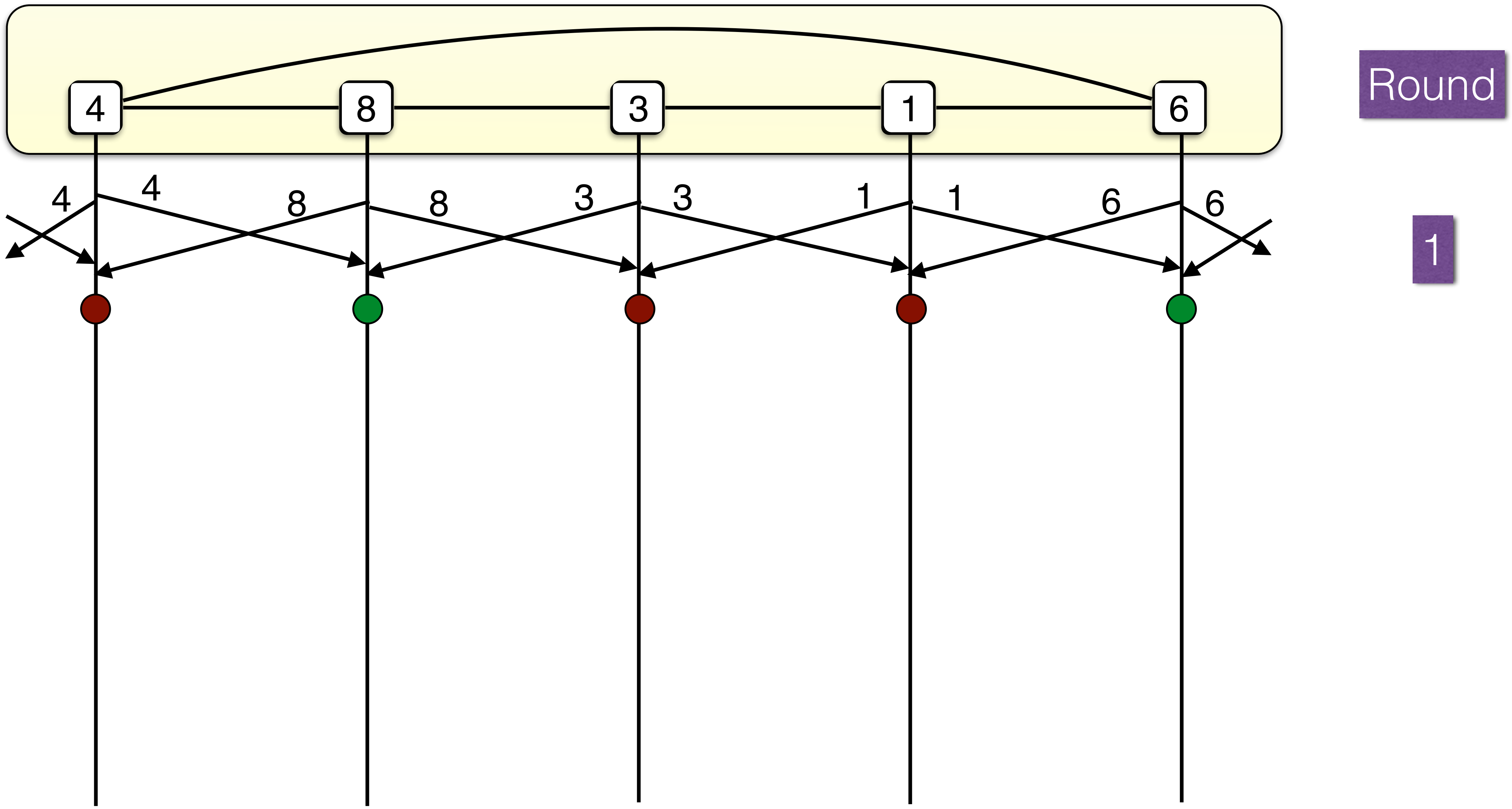
Application to Verification of Distributed Algorithms

Franklin’s leader-election protocol (1982)



Application to Verification of Distributed Algorithms

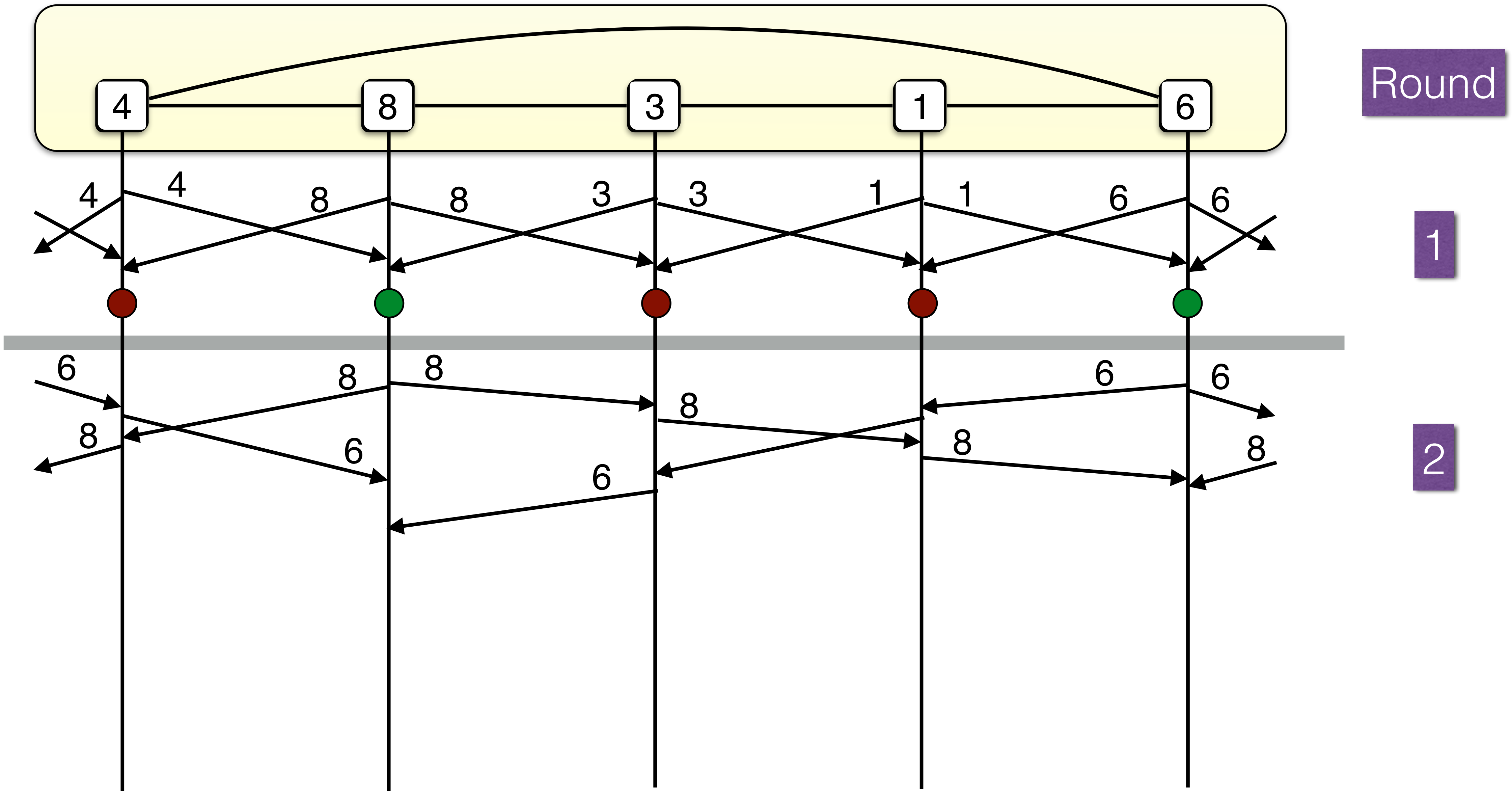
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.

Application to Verification of Distributed Algorithms

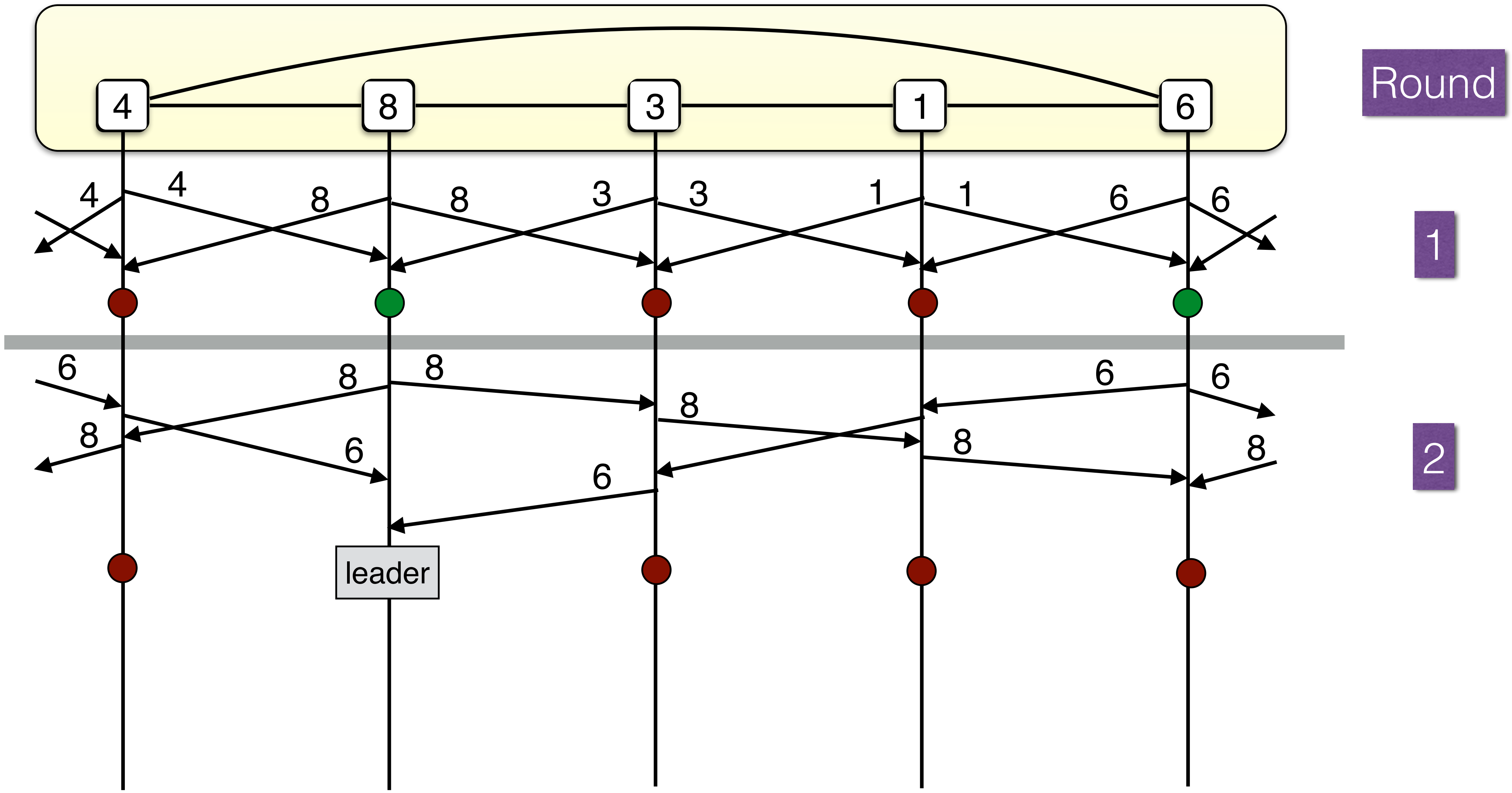
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.

Application to Verification of Distributed Algorithms

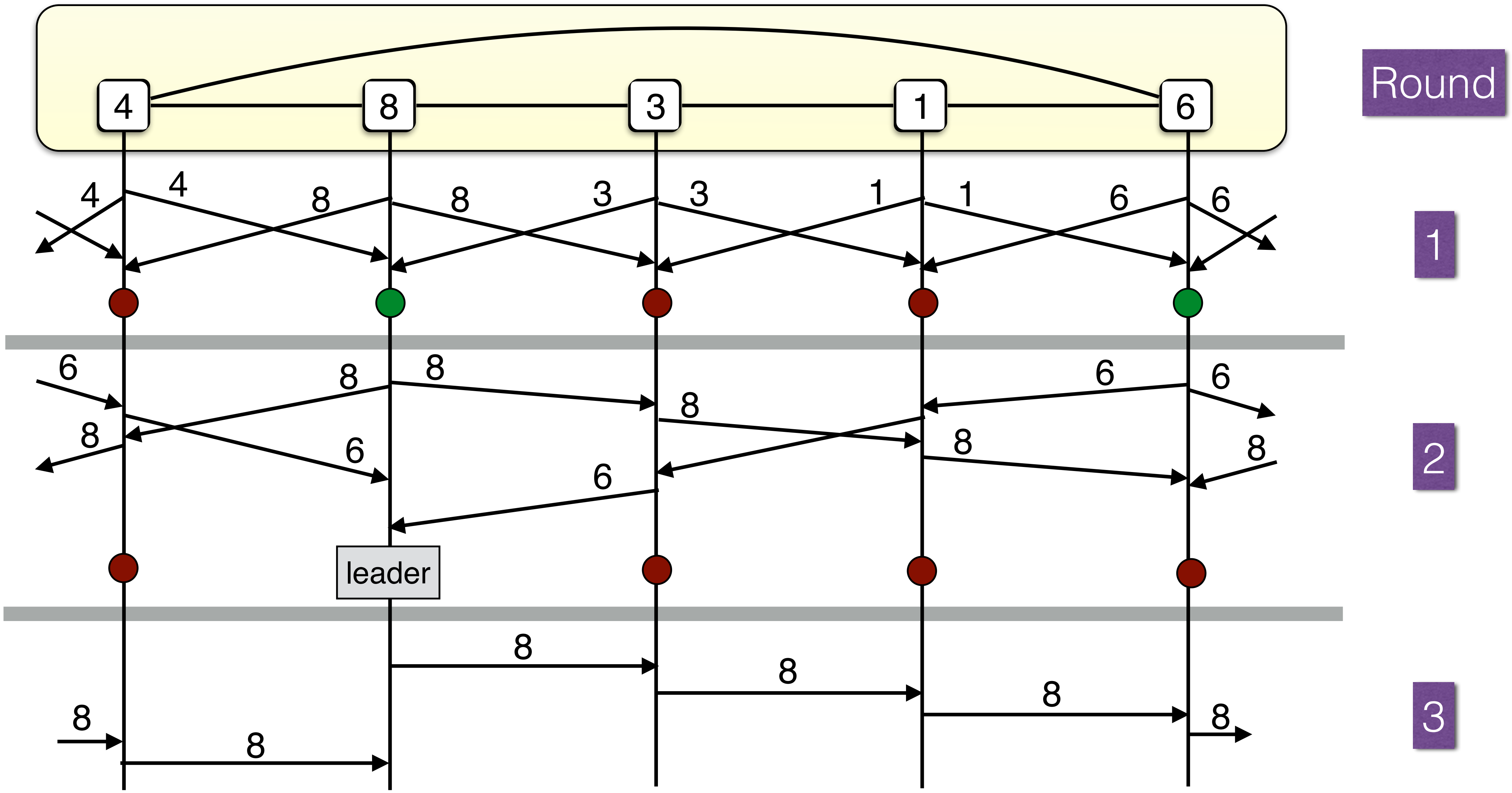
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.

Application to Verification of Distributed Algorithms

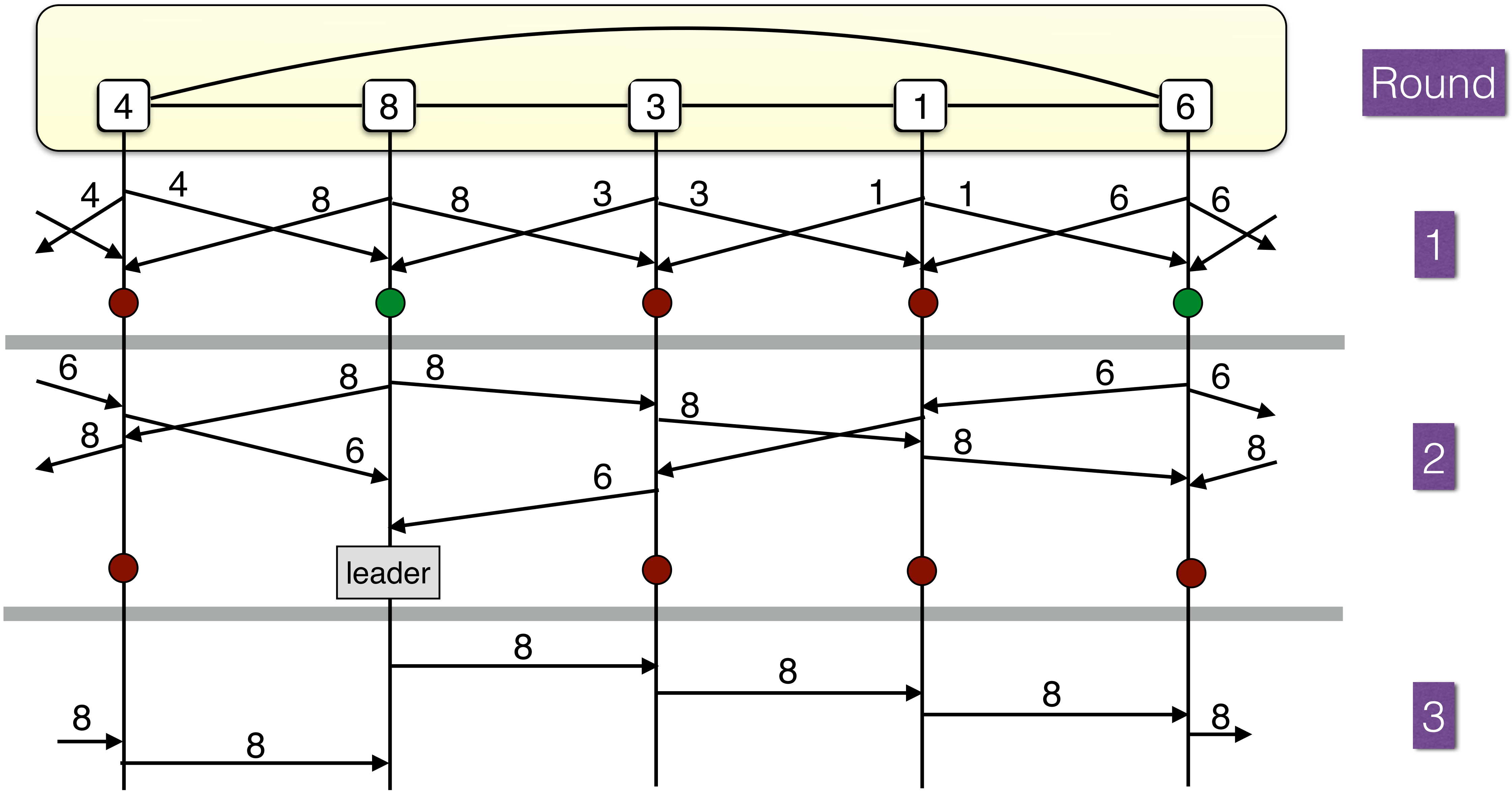
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.

Application to Verification of Distributed Algorithms

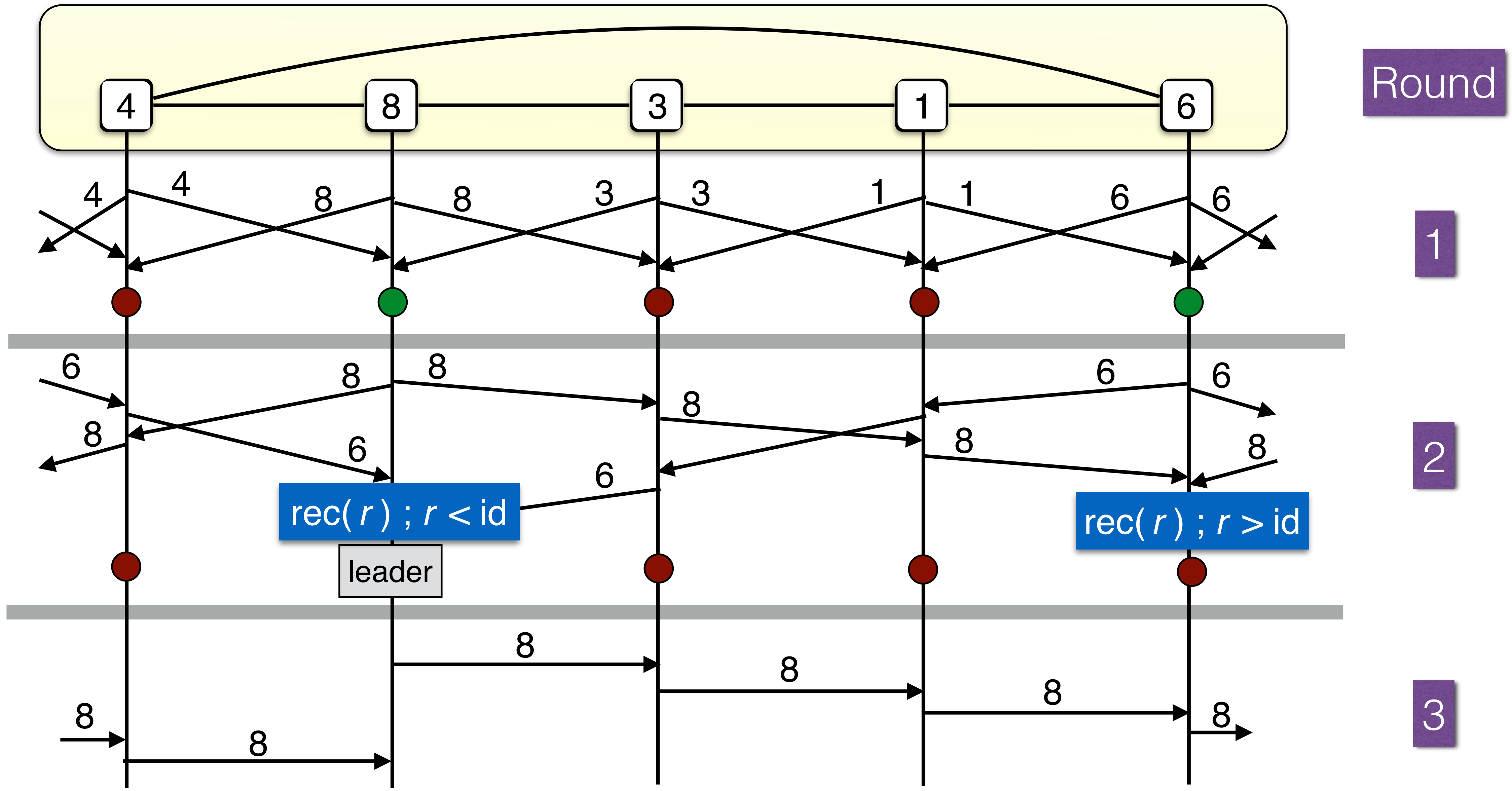
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.
- Number of rounds is sometimes logarithmic in the number of processes.

Application to Verification of Distributed Algorithms

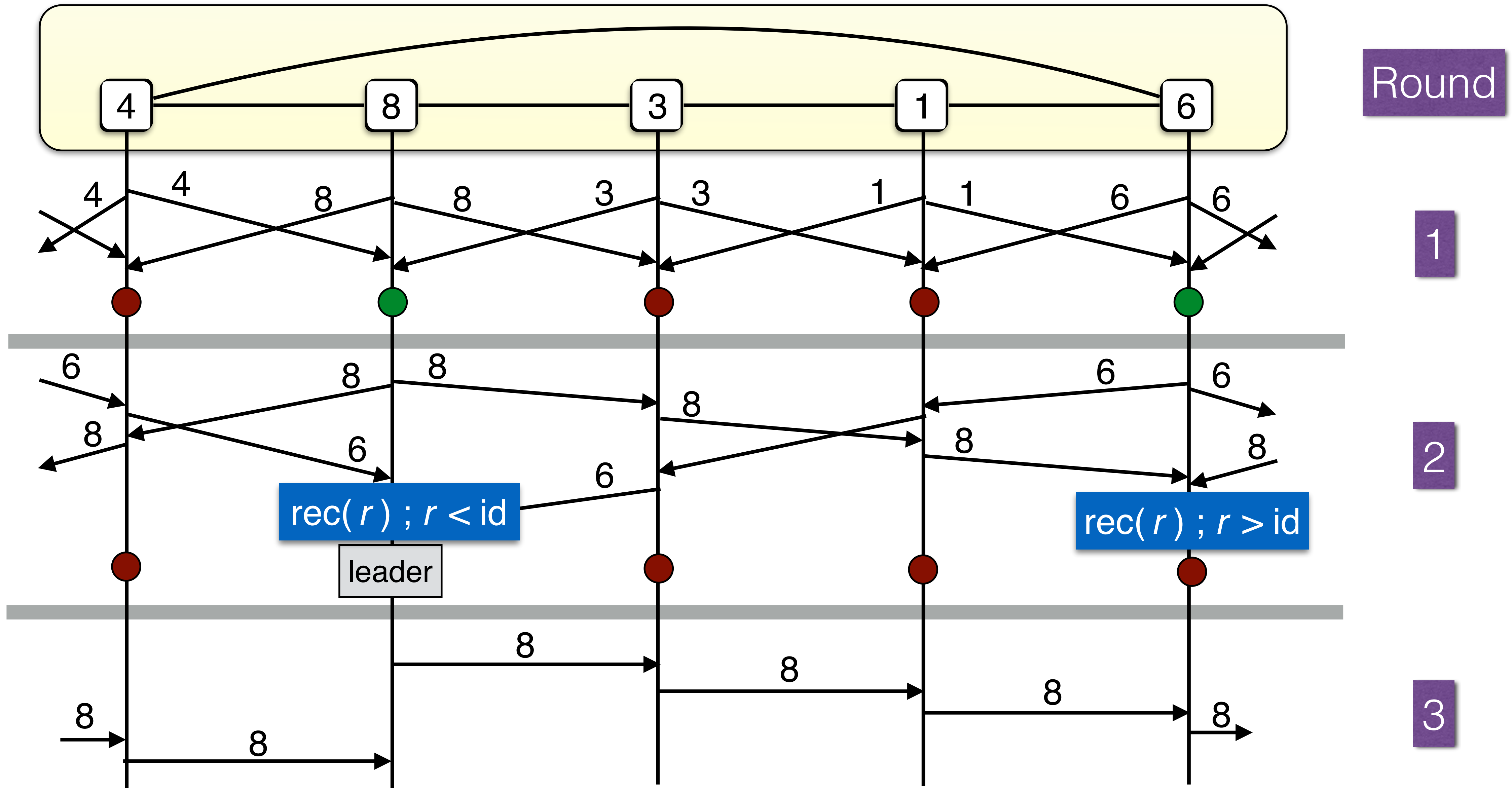
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.
- Number of rounds is sometimes logarithmic in the number of processes.

Application to Verification of Distributed Algorithms

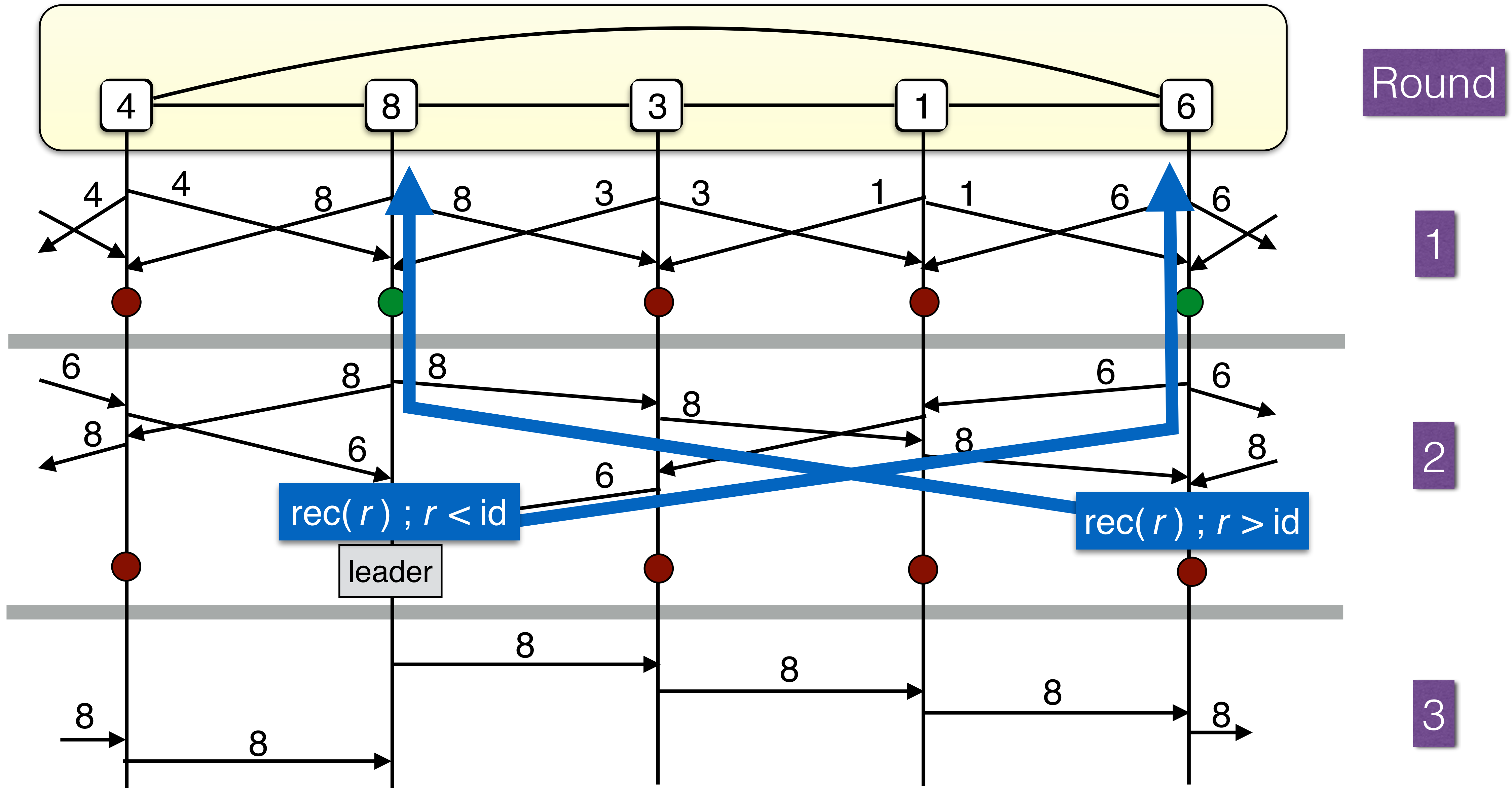
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.
- Number of rounds is sometimes logarithmic in the number of processes.
- MSO can trace back origin of unique process identifiers (pids).

Application to Verification of Distributed Algorithms

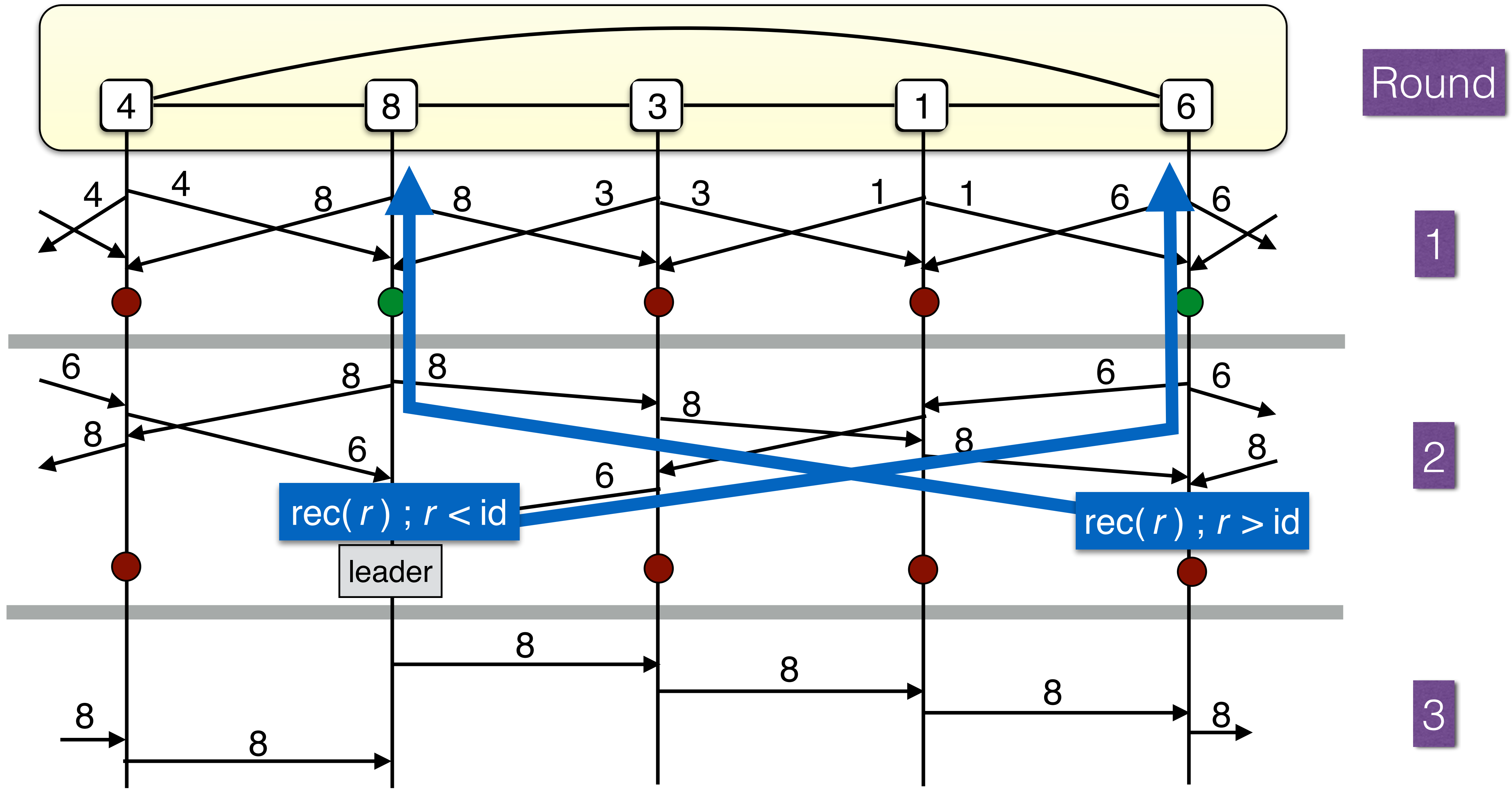
Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.
- Number of rounds is sometimes logarithmic in the number of processes.
- MSO can trace back origin of unique process identifiers (pids).

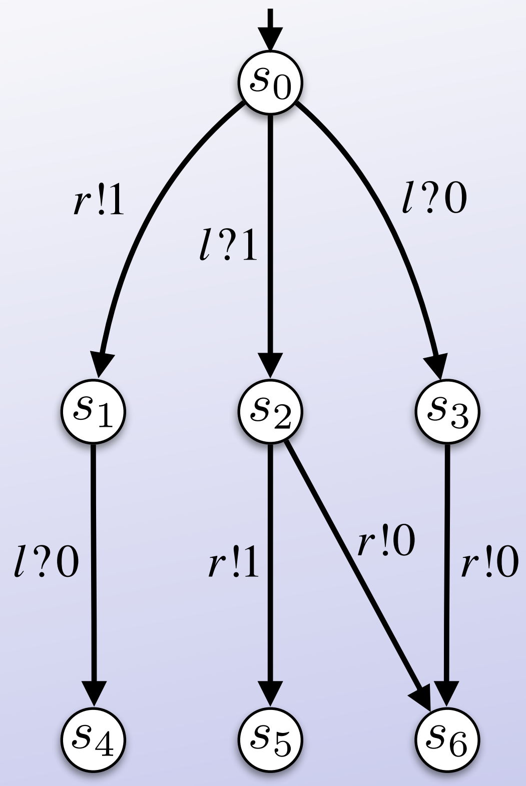
Application to Verification of Distributed Algorithms

Franklin's leader-election protocol (1982)



- Distributed algorithms often proceed in rounds/contexts.
- Number of rounds is sometimes logarithmic in the number of processes.
- MSO can trace back origin of unique process identifiers (pids).
- Underapproximate verification of distributed algorithms that send and compare pids.

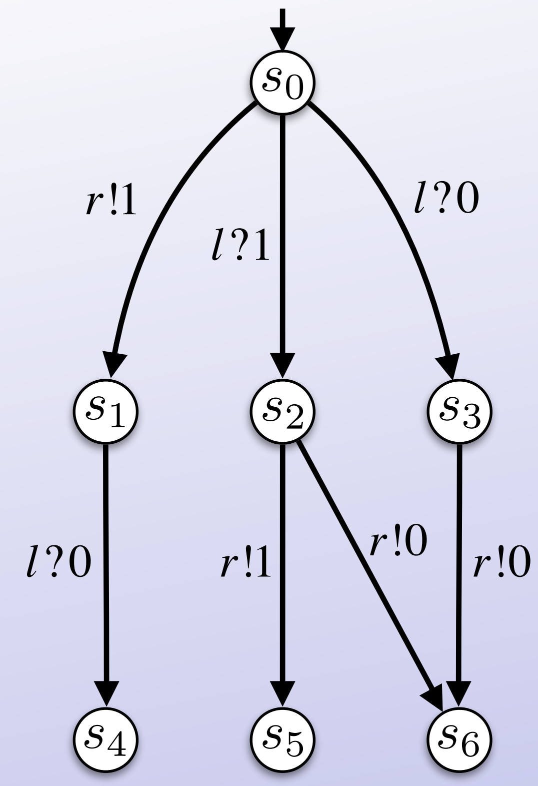
Beyond Context Bounds …



$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

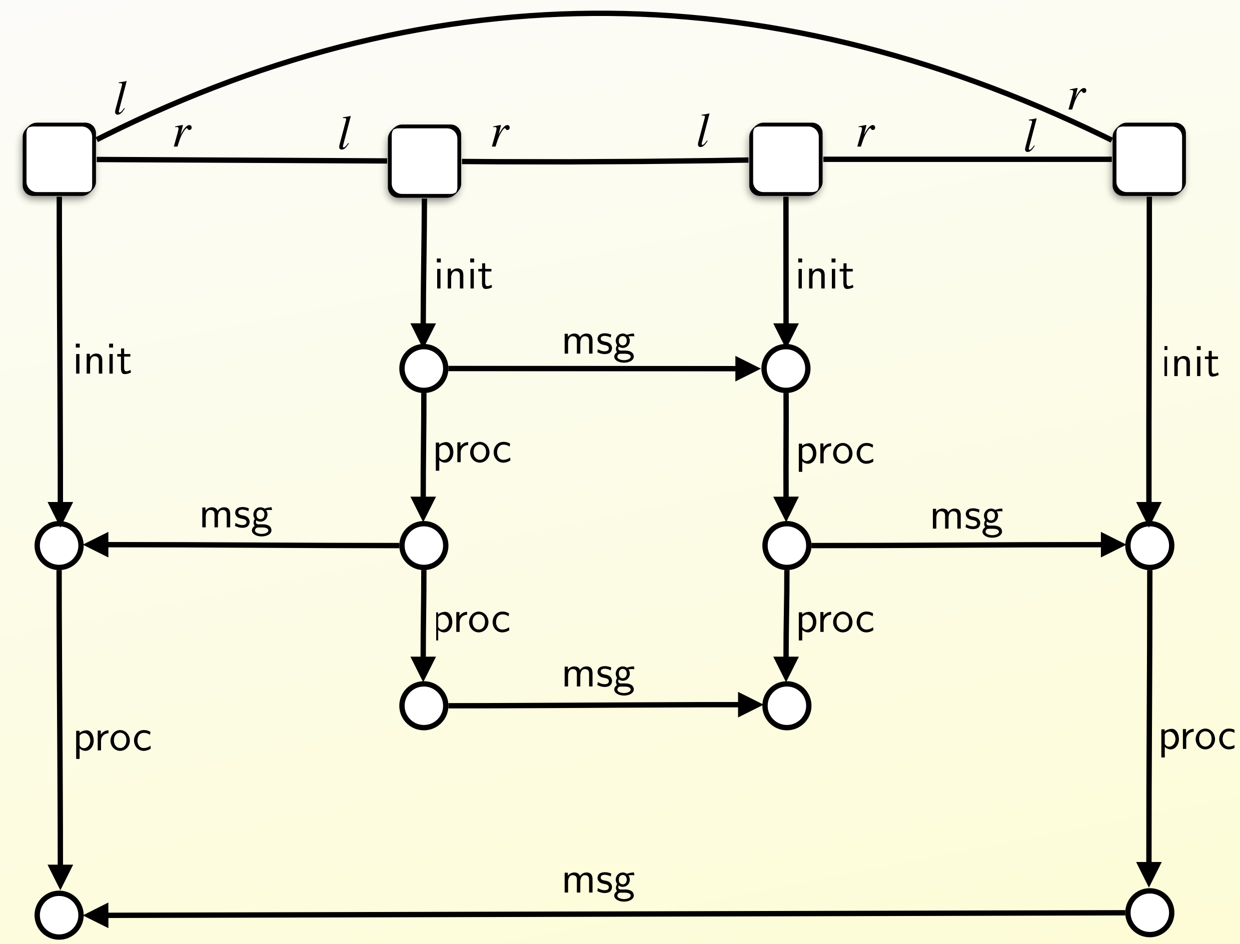
weak PCA

Beyond Context Bounds …



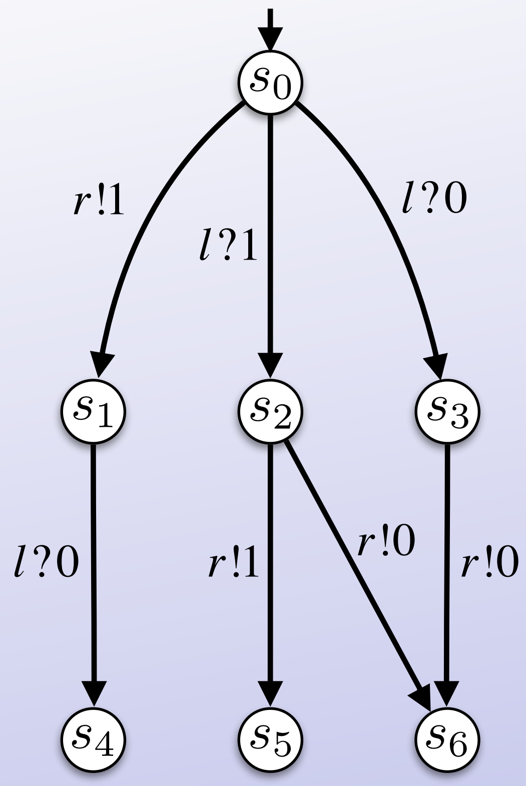
$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

weak PCA



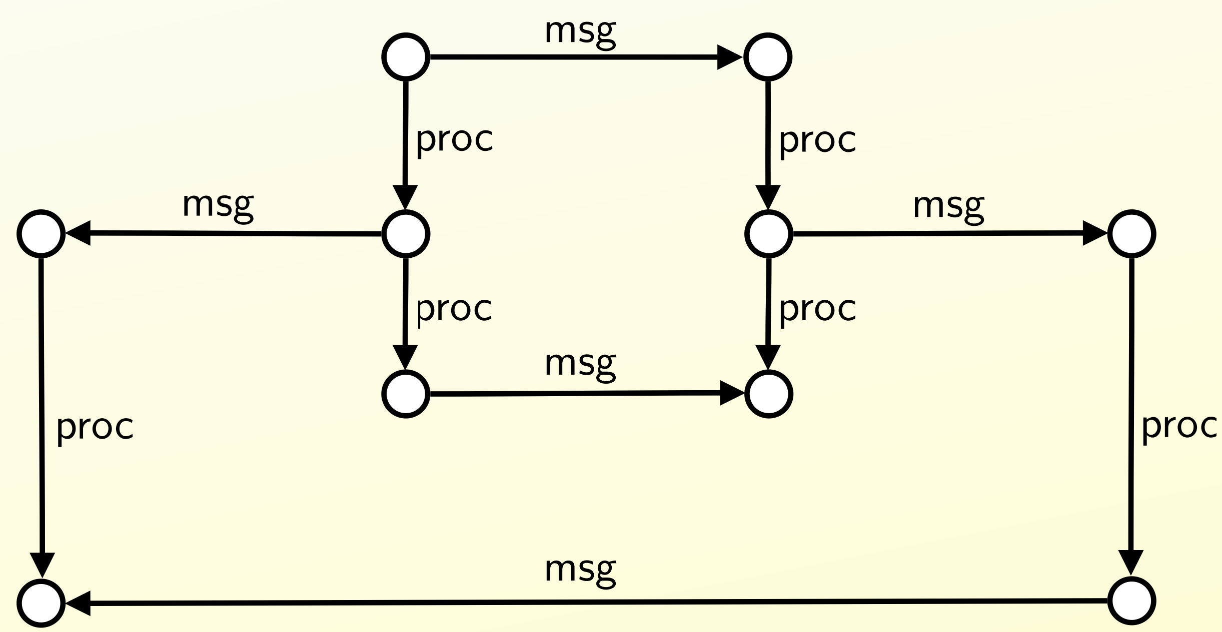
weak logic

Beyond Context Bounds …



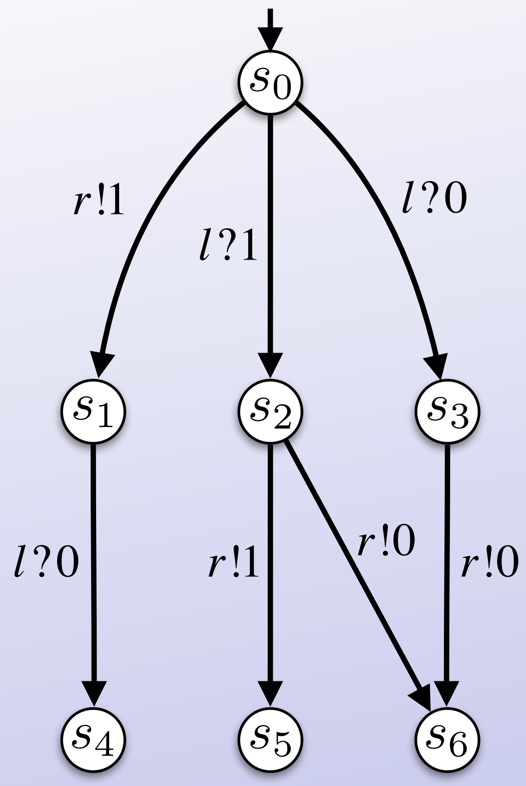
$$\exists x(s_4(x) \land \forall y(y \neq x \rightarrow s_5(y) \lor s_6(y)))$$

weak PCA



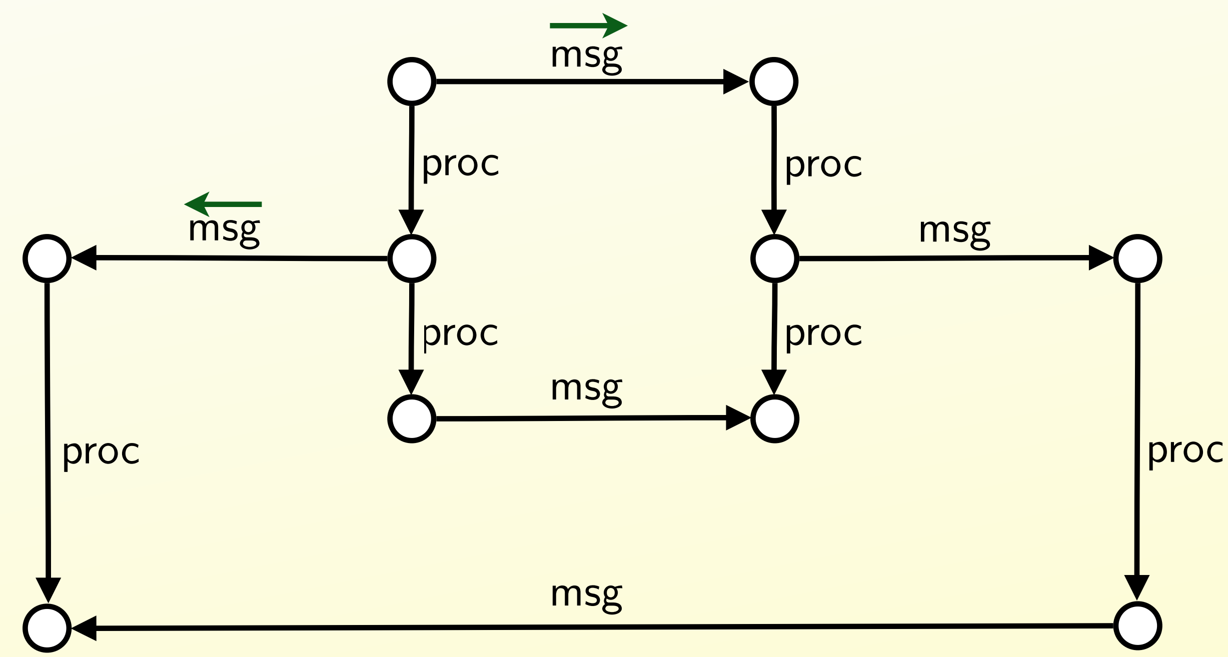
weak logic

Beyond Context Bounds …



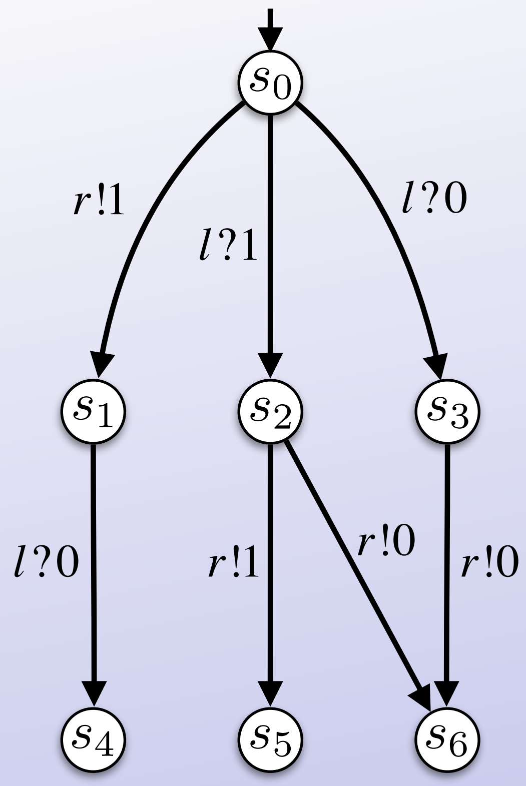
$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

weak PCA



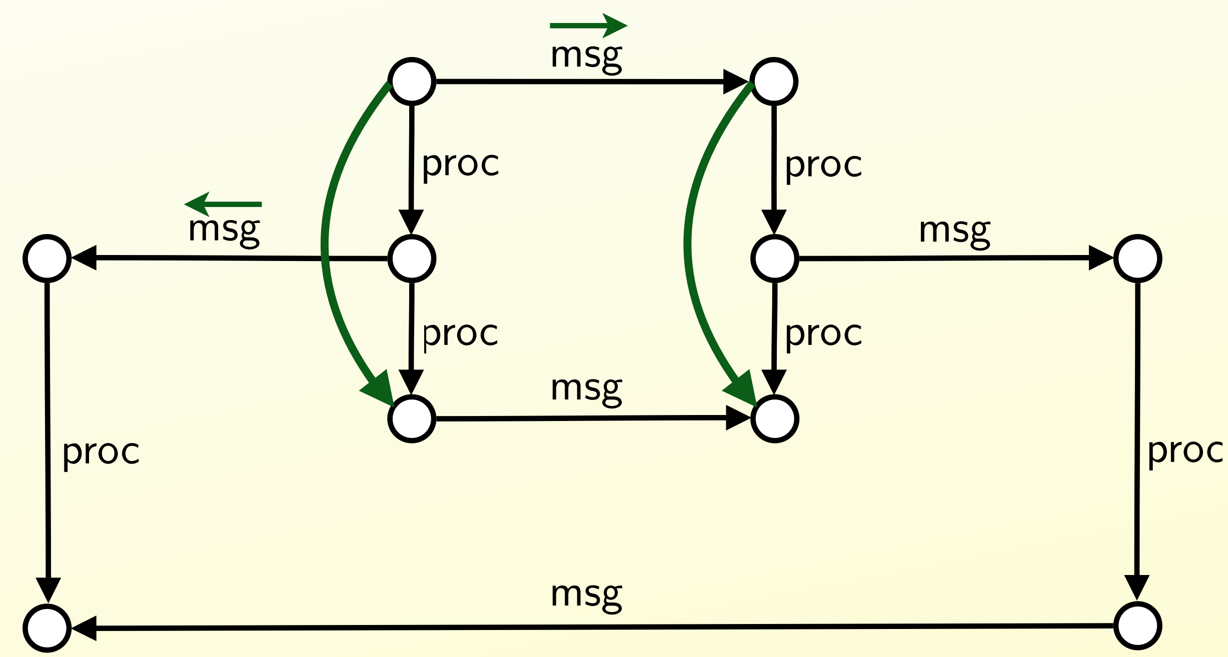
weak logic

Beyond Context Bounds …



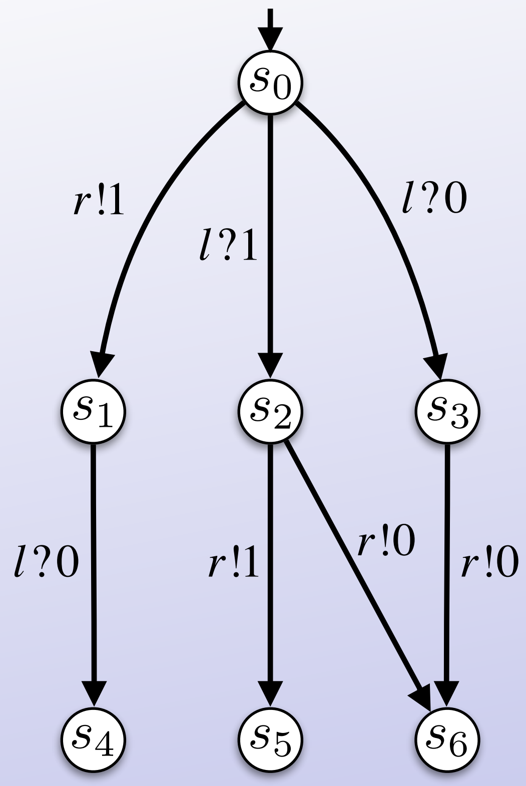
$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

weak PCA



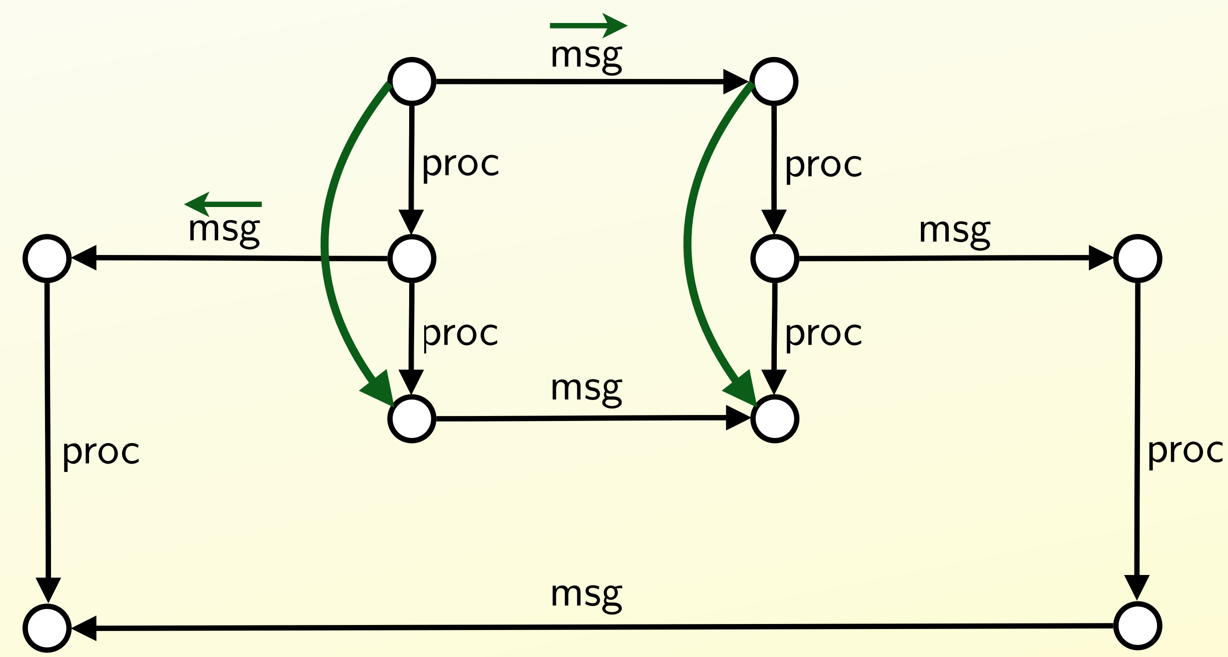
weak logic

Beyond Context Bounds …



$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

weak PCA

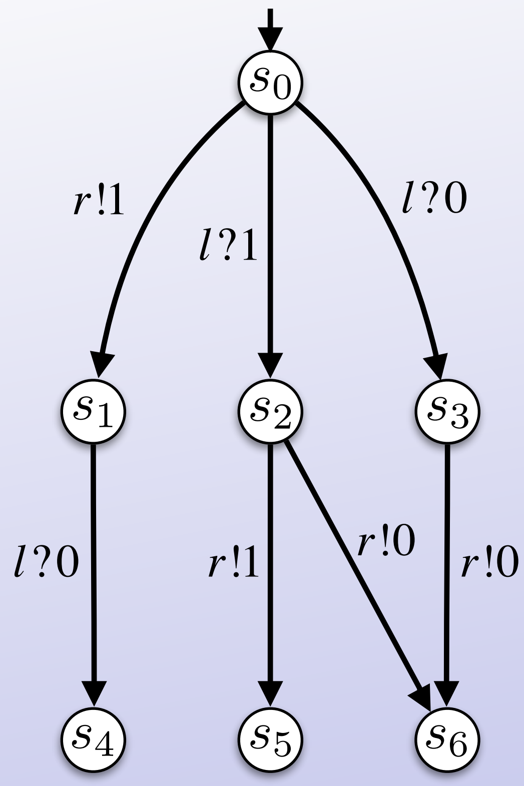


weak logic

Theorem [B.; CSL-LICS 2014]:

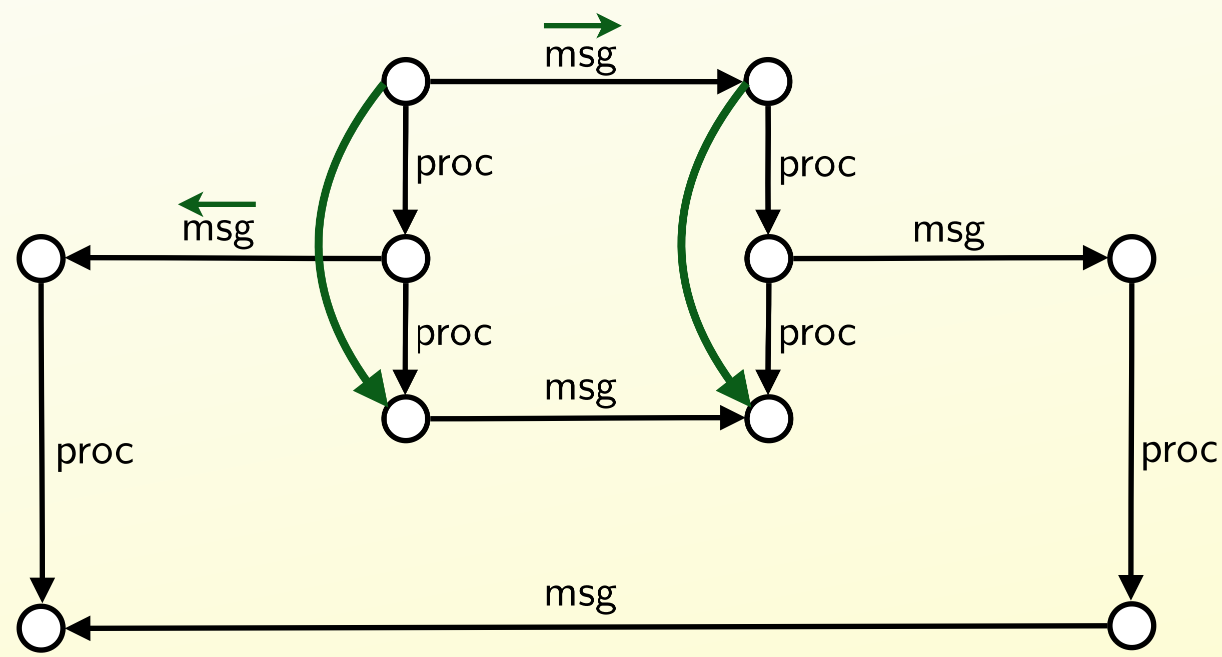
Let T be any of the following topology classes: rings, grids, binary trees.

Beyond Context Bounds …



$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

weak PCA



weak logic

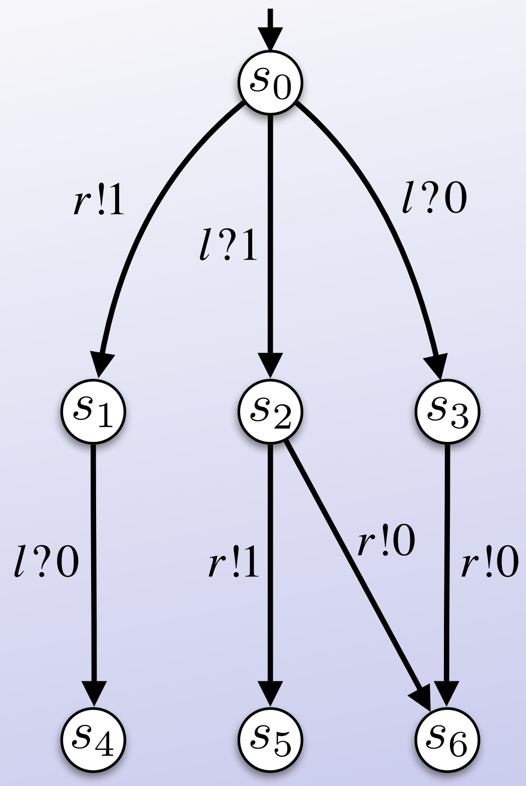
Theorem [B.; CSL-LICS 2014]:

Let T be any of the following topology classes: rings, grids, binary trees.

For every set L of behaviors over a topology from T the following are equivalent:

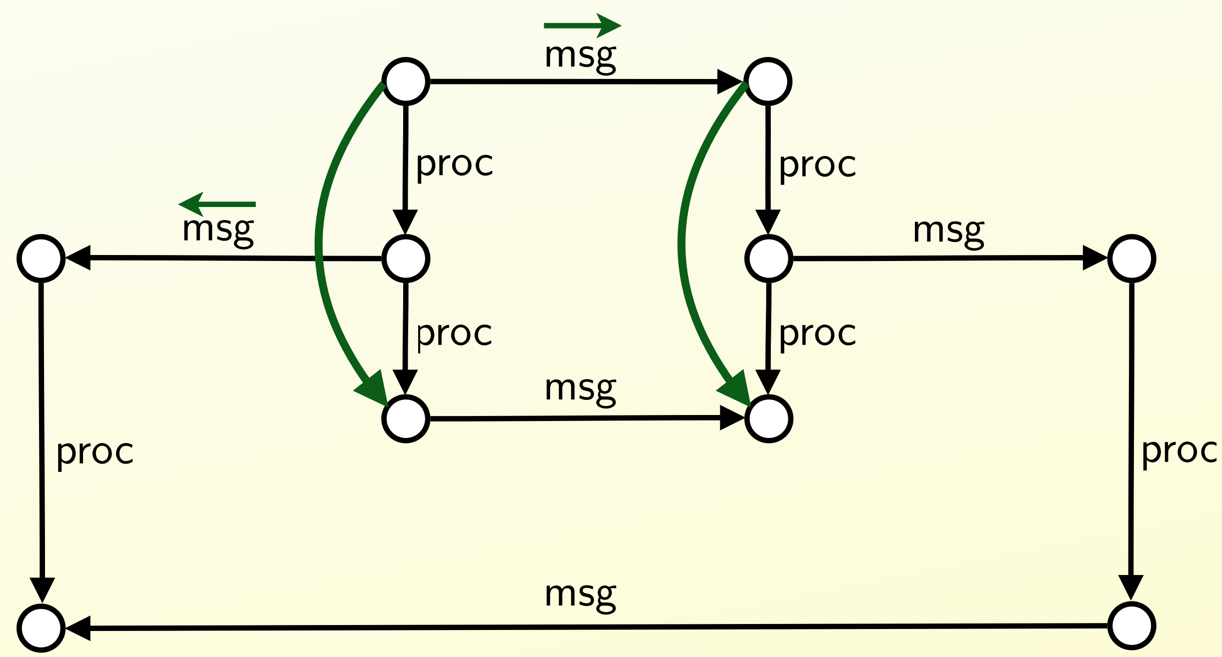
- L is recognized by some weak PCA.
- L is definable in weak EMSO logic (projection of weak-FO-definable language).

Beyond Context Bounds …



$$\exists x(s_4(x) \wedge \forall y(y \neq x \rightarrow s_5(y) \vee s_6(y)))$$

weak PCA



weak logic

Theorem [B.; CSL-LICS 2014]:

Let T be any of the following topology classes: rings, grids, binary trees.

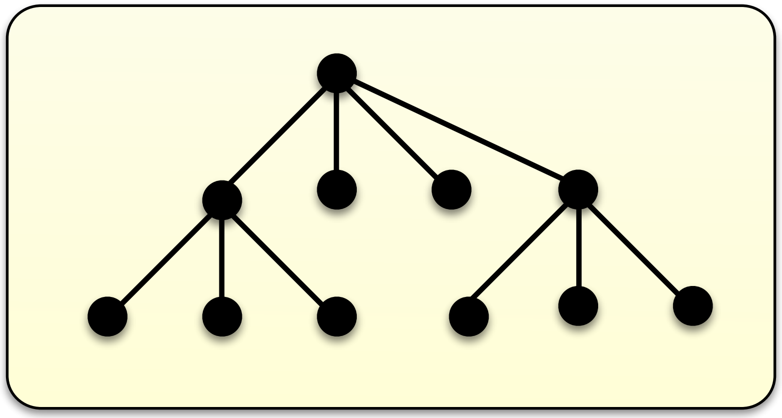
For every set L of behaviors over a topology from T the following are equivalent:

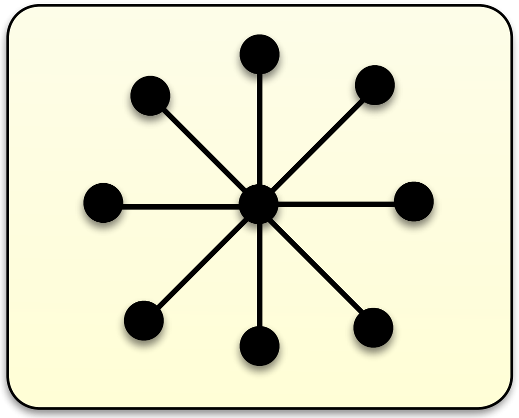
- L is recognized by some weak PCA.
- L is definable in weak EMSO logic (projection of weak-FO-definable language).

Proof uses [Schwentick-Barthelmann 1999] & [Genest-Kuske-Muscholl 2006].

Other Future Work

- Topologies of unbounded degree (unranked trees, stars, …)



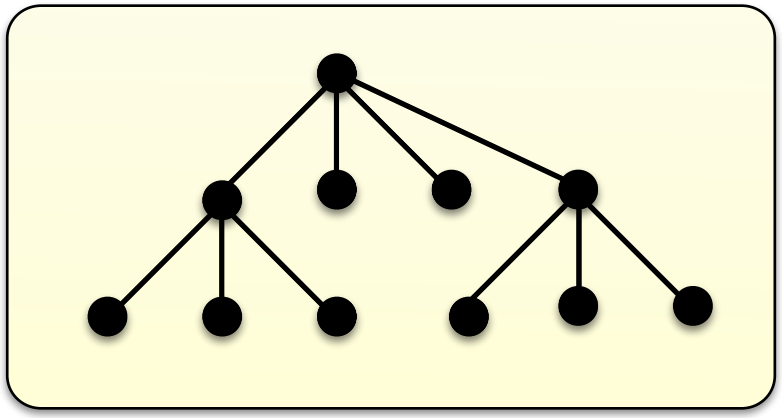


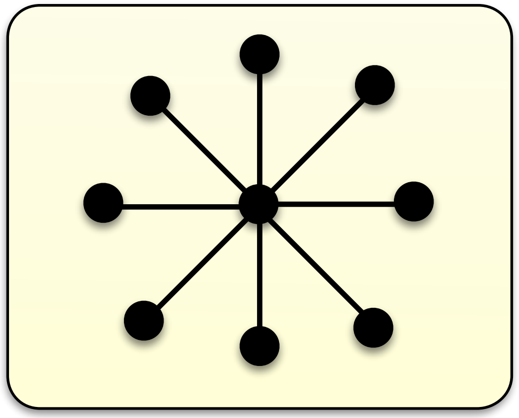
- Temporal logics and efficient model checking

- Split-width for parameterized systems
  [Aiswarya-Gastin-Narayan Kumar 2012]

Other Future Work

- Topologies of unbounded degree (unranked trees, stars, …)





- Temporal logics and efficient model checking

- Split-width for parameterized systems
  [Aiswarya-Gastin-Narayan Kumar 2012]

Thank You!