Soutenance de thèse : Synthesis for Parameterized Systems

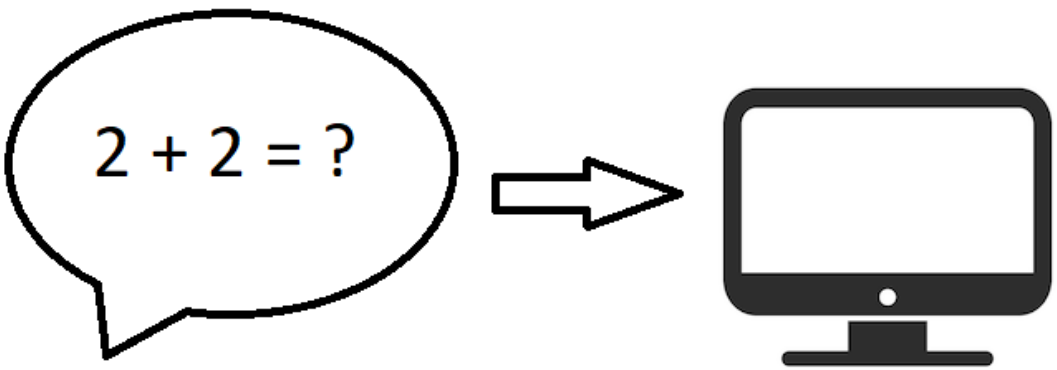
Mathieu Lehaut

Sorbonne Université, LIP6

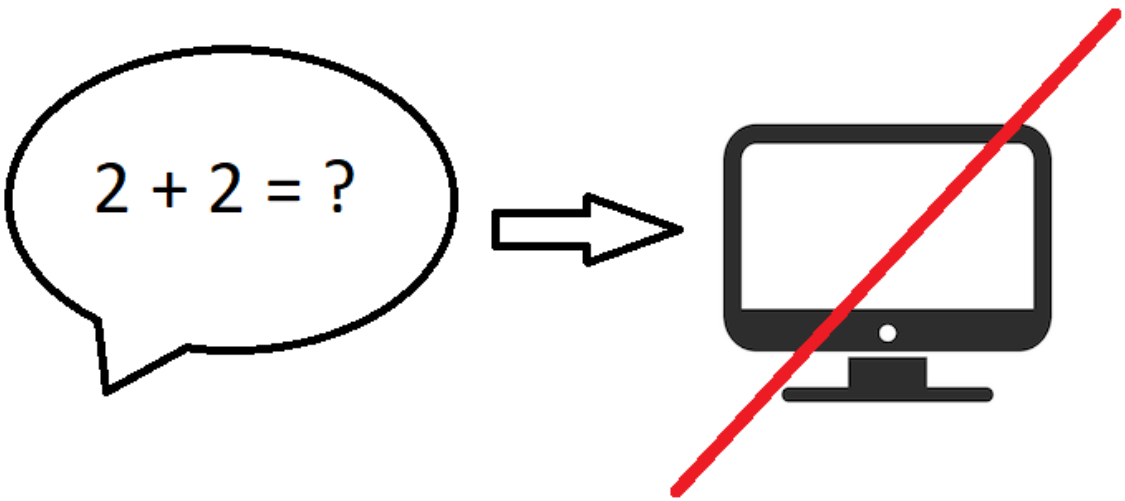
17/12/2020

Outline

Parameterized systems



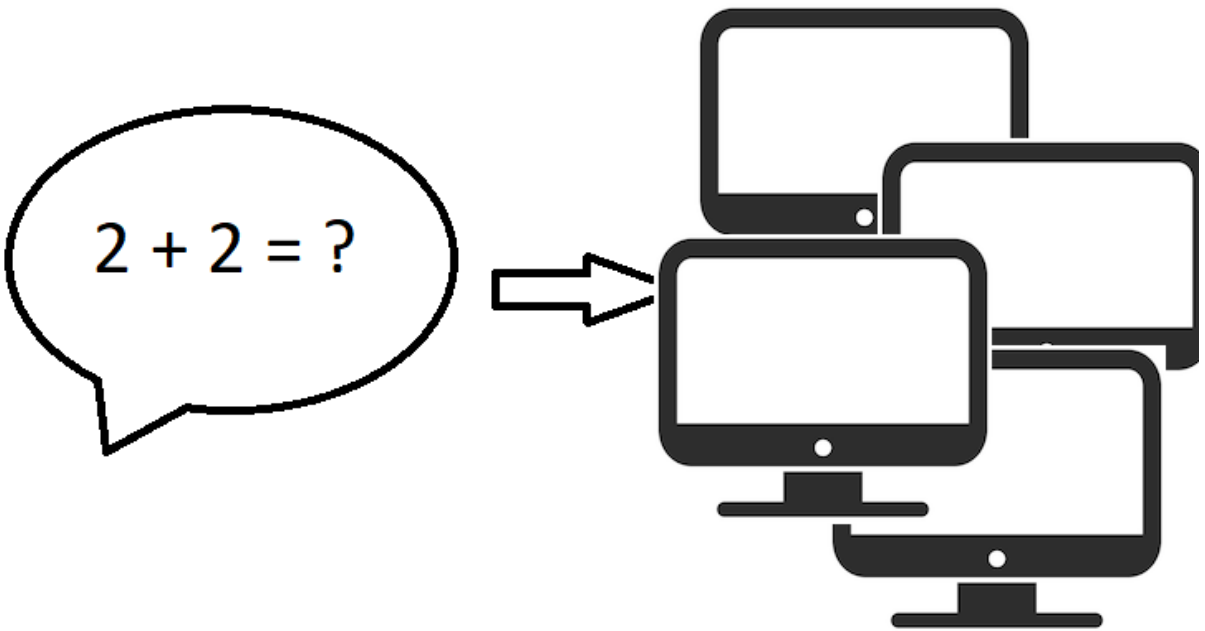
Parameterized systems



Parameterized systems

Distributed systems

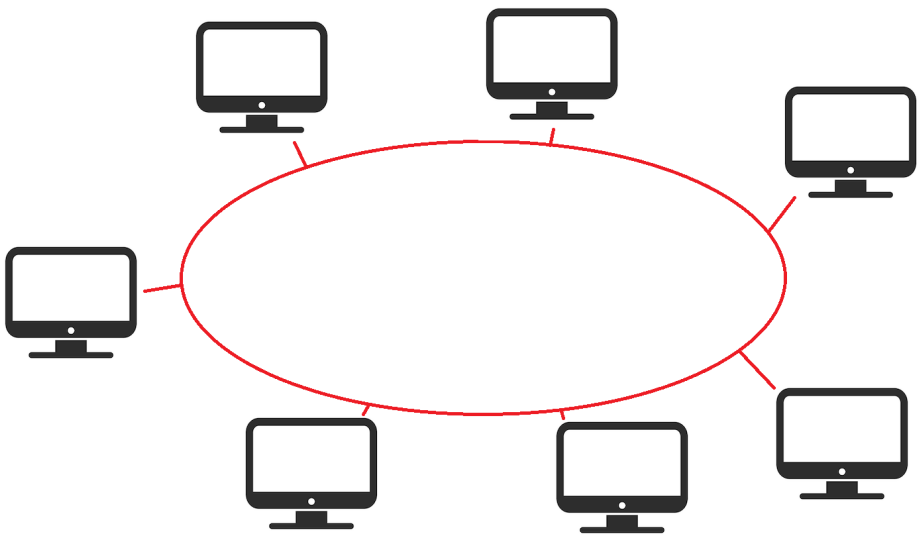
Systems with multiple components collaborating towards one goal



Parameterized systems

Examples of real-life distributed systems

- Distributed computing



Parameterized systems

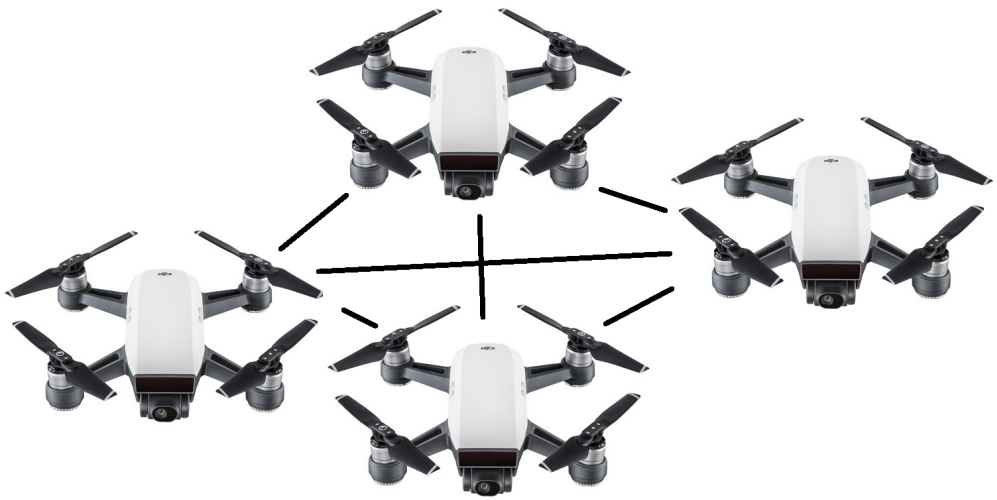
Examples of real-life distributed systems

- Distributed computing
- Communication networks

Parameterized systems

Examples of real-life distributed systems

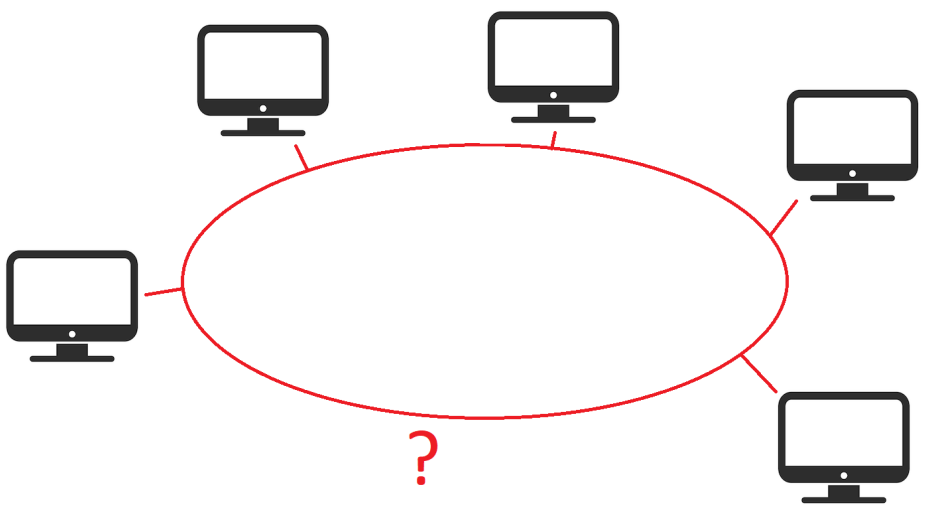
- Distributed computing
- Communication networks
- Drone fleets



Parameterized systems

Parameterized systems

Number of components not known in advance



Open systems

- Systems interact with an uncontrollable environment!

Open systems

- Systems interact with an uncontrollable environment!

Drone example

Open systems

- Systems interact with an uncontrollable environment!

Drone example

- Sensors (camera, thermometers, ...)

Open systems

- Systems interact with an uncontrollable environment!

Drone example

- Sensors (camera, thermometers, ...)
- Operator inputs

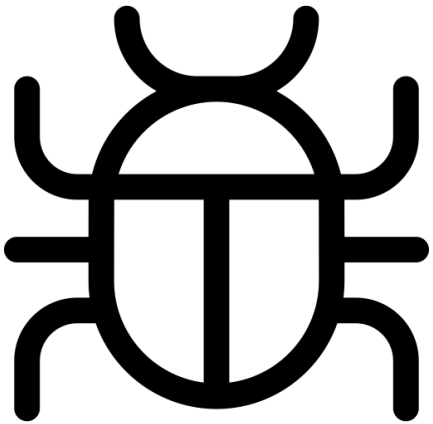
Open systems

- Systems interact with an uncontrollable environment!

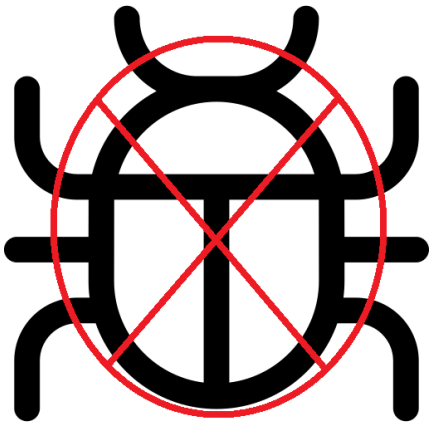
Drone example

- Sensors (camera, thermometers, ...)
- Operator inputs
- Environmental conditions (wind, rain, ...)

Verification



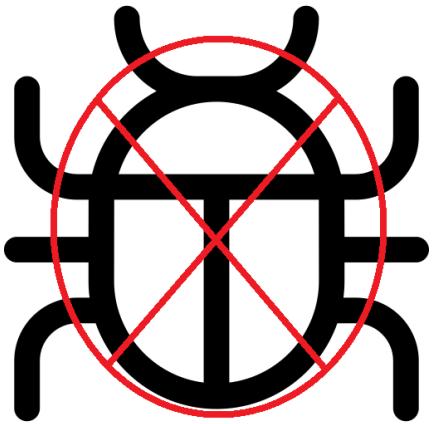
Verification



Bugs

Differences between:

Verification

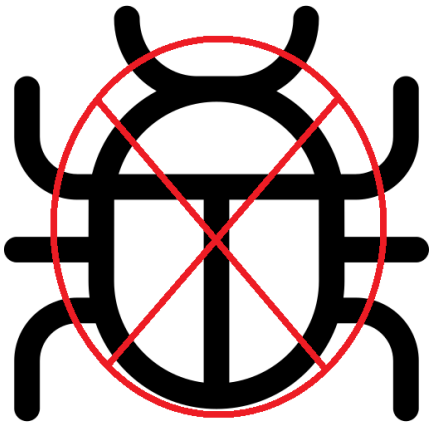


Bugs

Differences between:

1. What the system does (behaviors)

Verification

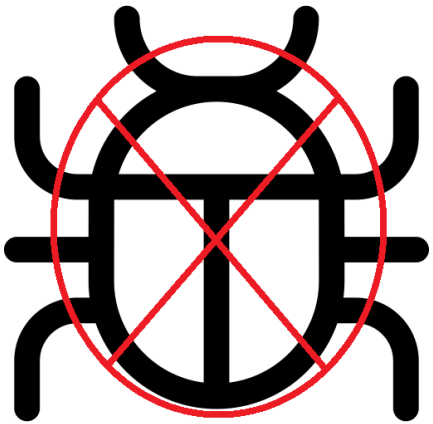


Bugs

Differences between:

1. What the system does (behaviors)
2. What the system is supposed to do (specification)

Verification



Bugs

Differences between:

1. What the system does (behaviors)
2. What the system is supposed to do (specification)

- Goal: all behaviors of the system must satisfy the specification

A technique of verification

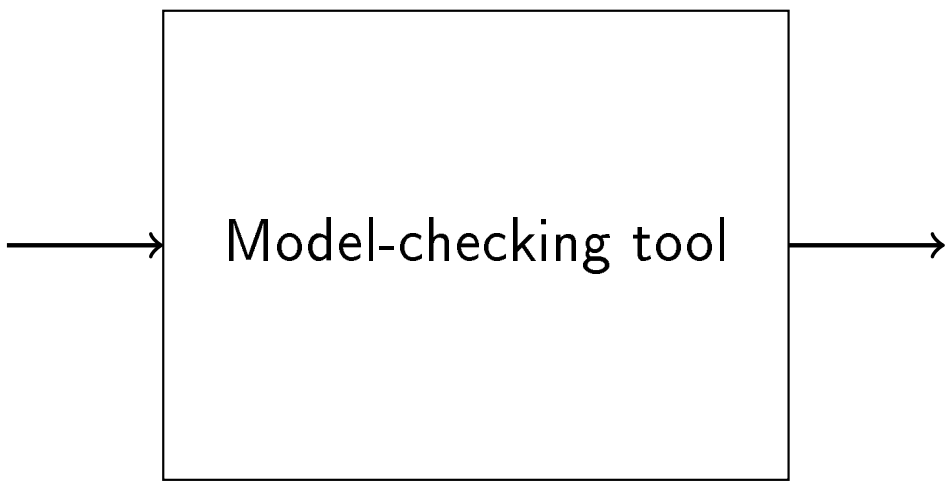
Model checking [Clarke, Emerson, Sifakis]

I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?

A technique of verification

Model checking [Clarke, Emerson, Sifakis]

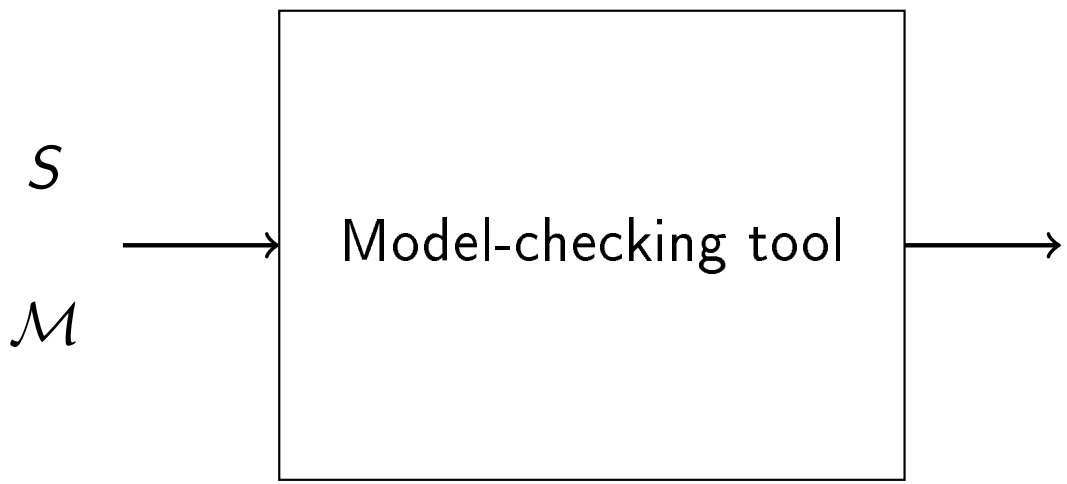
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

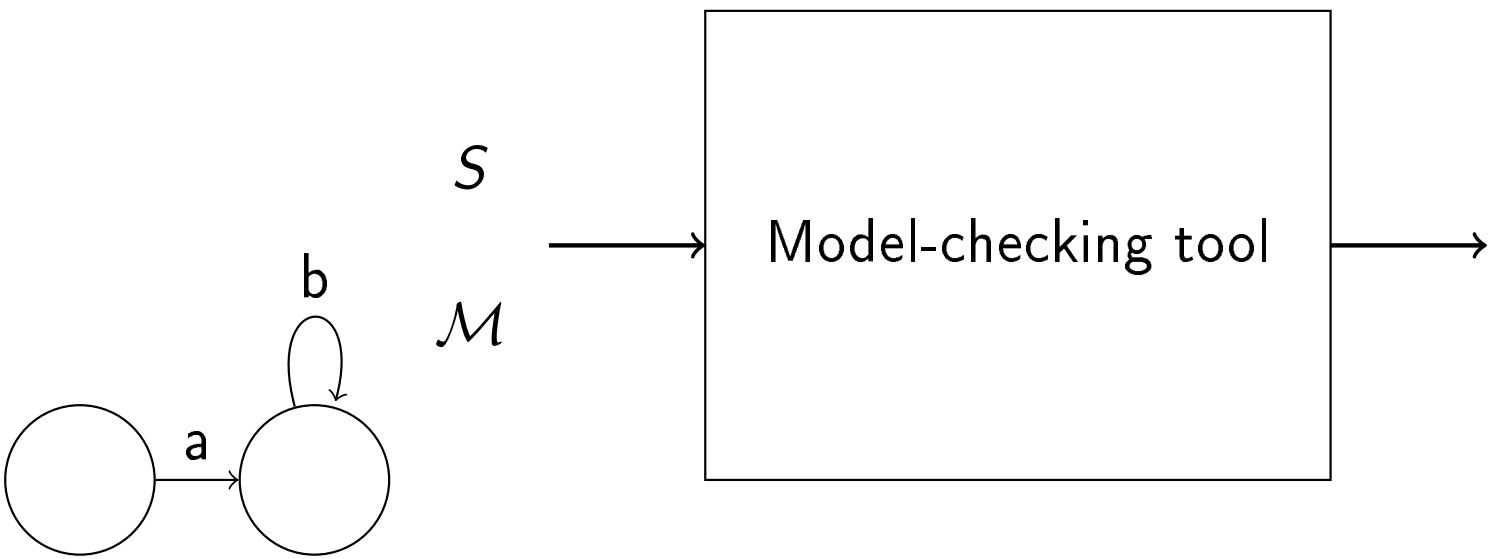
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

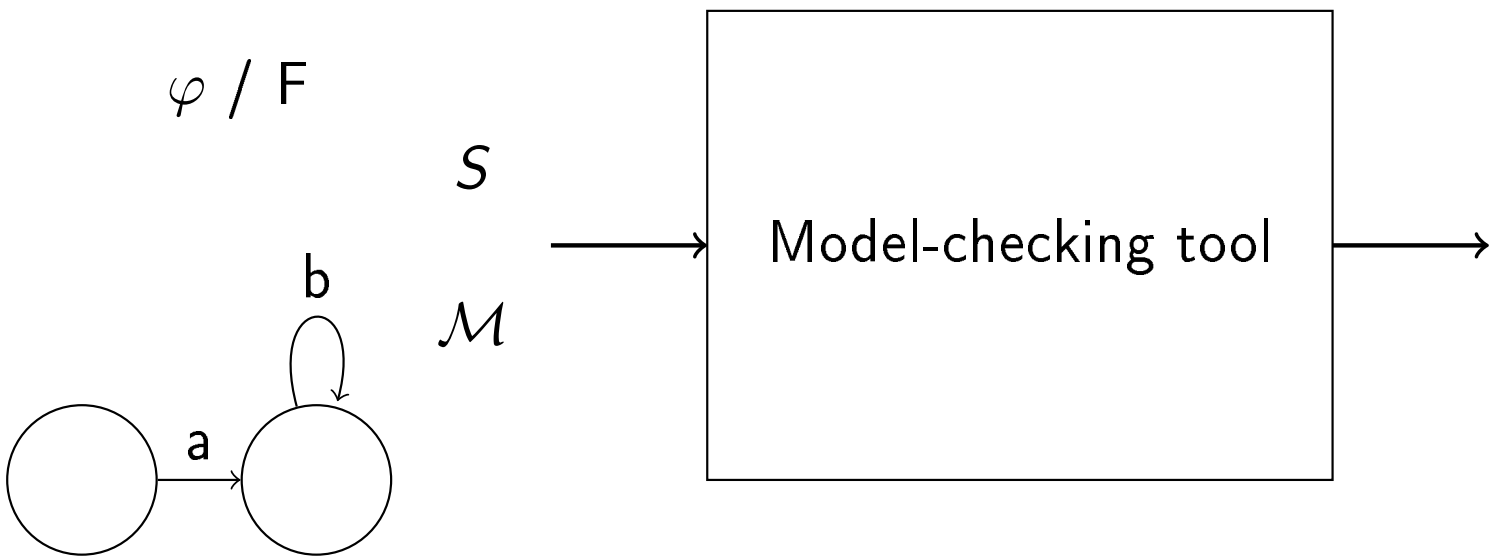
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

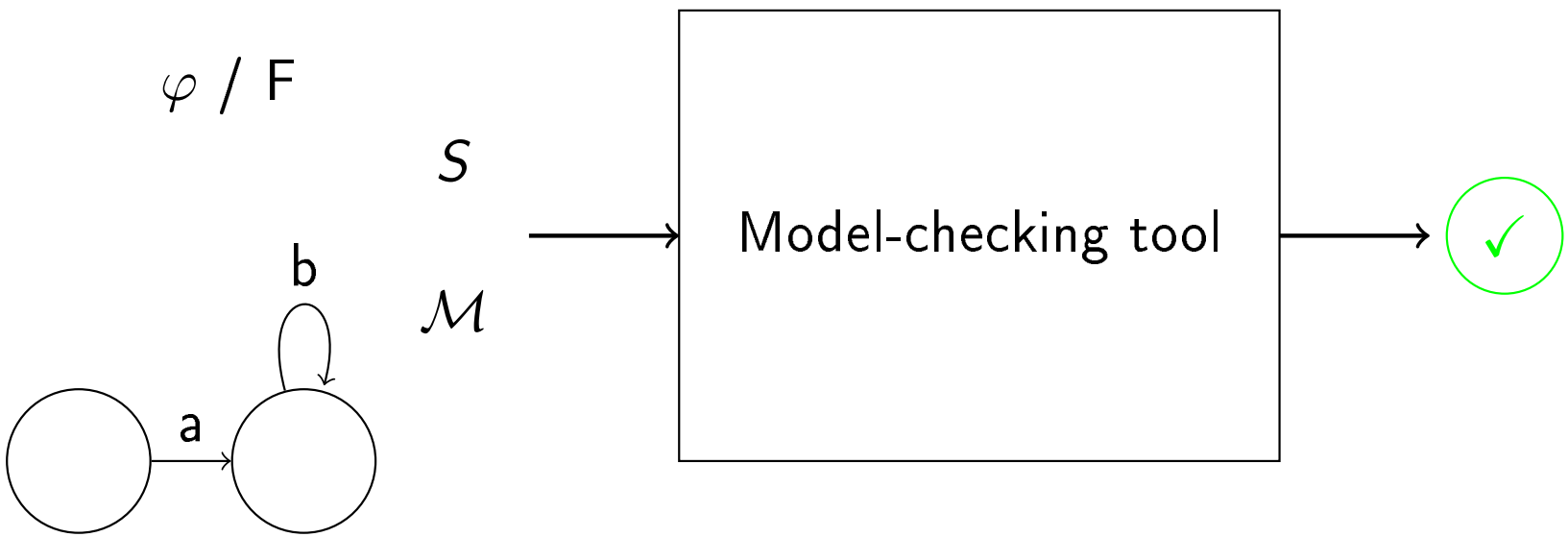
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

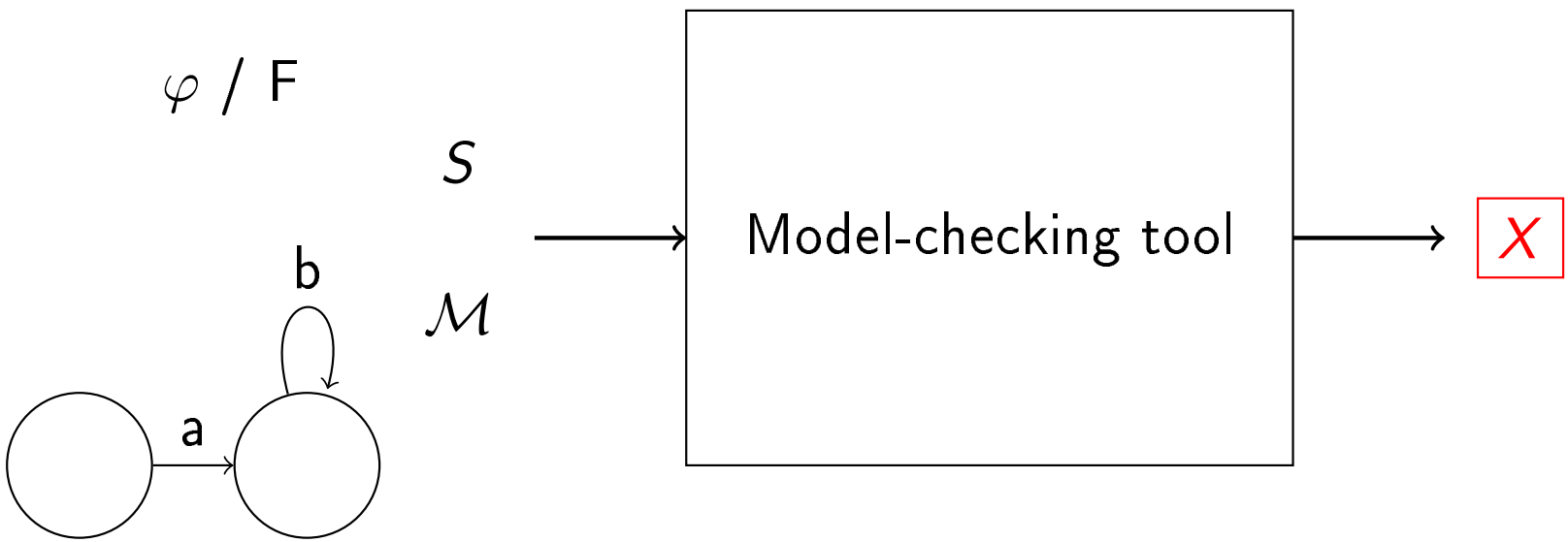
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

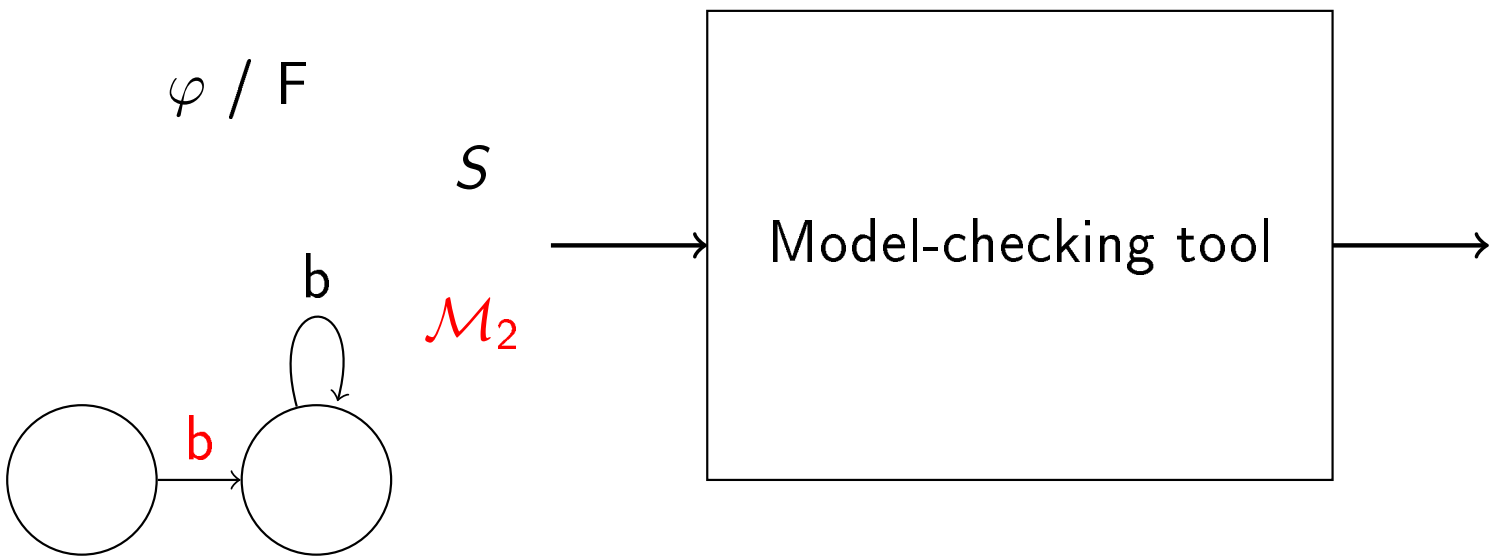
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

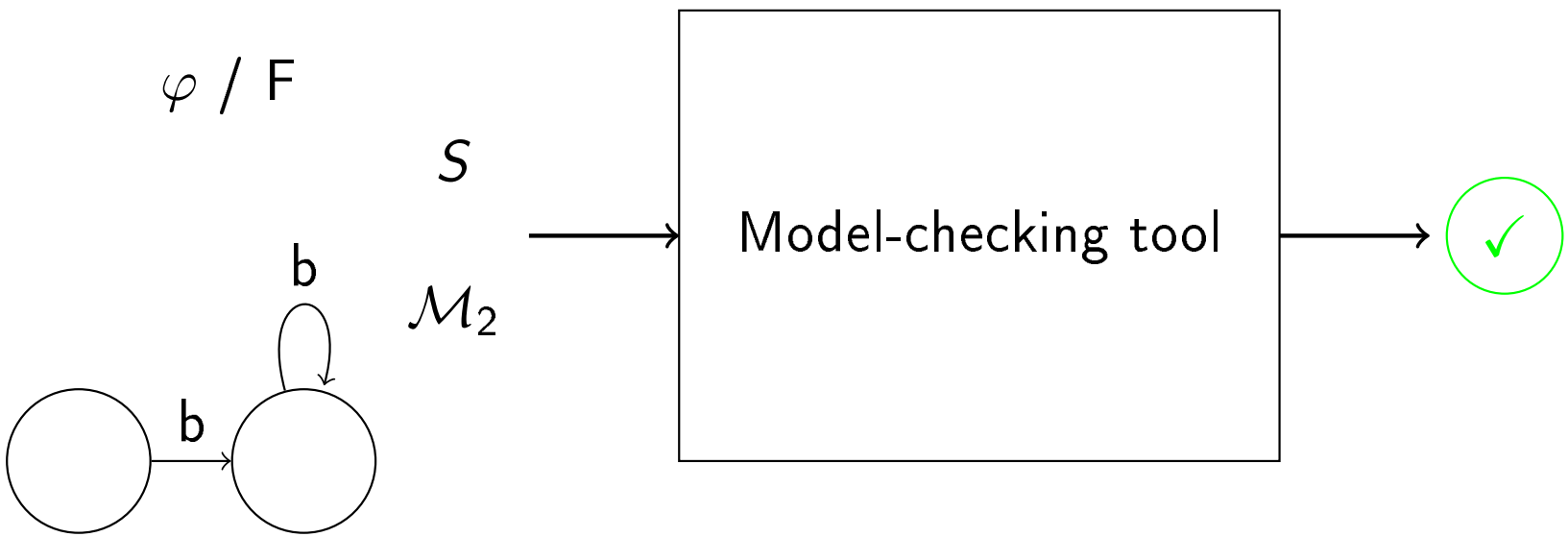
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

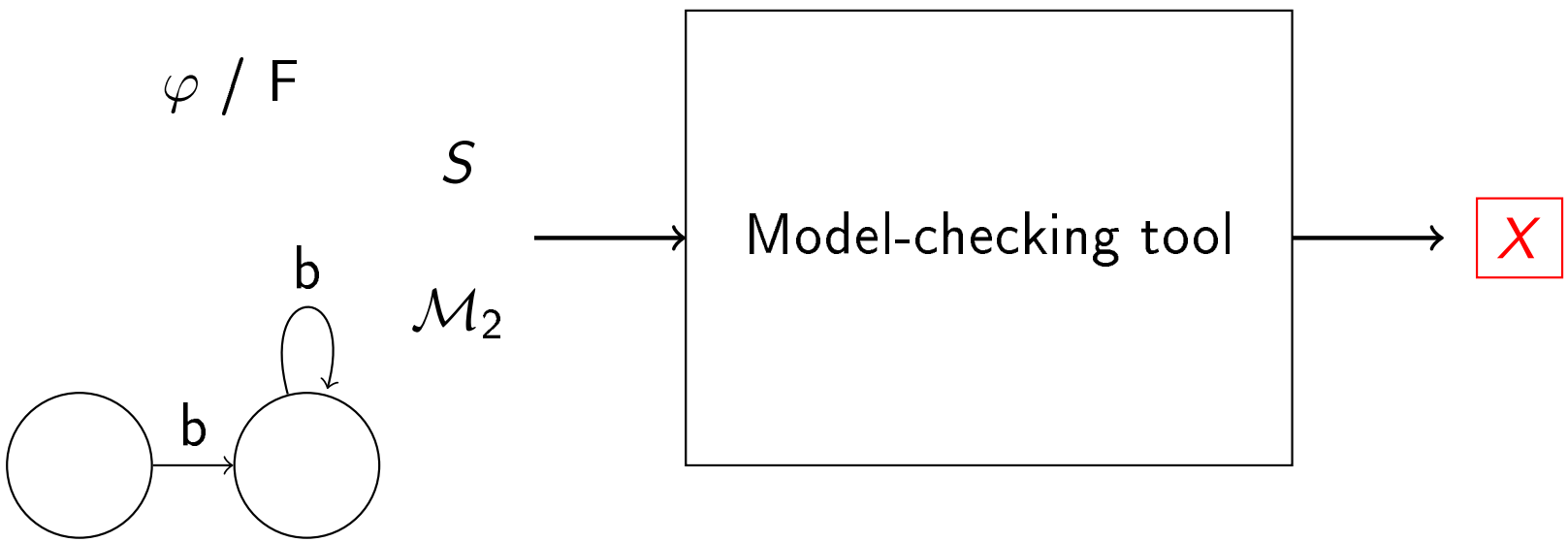
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

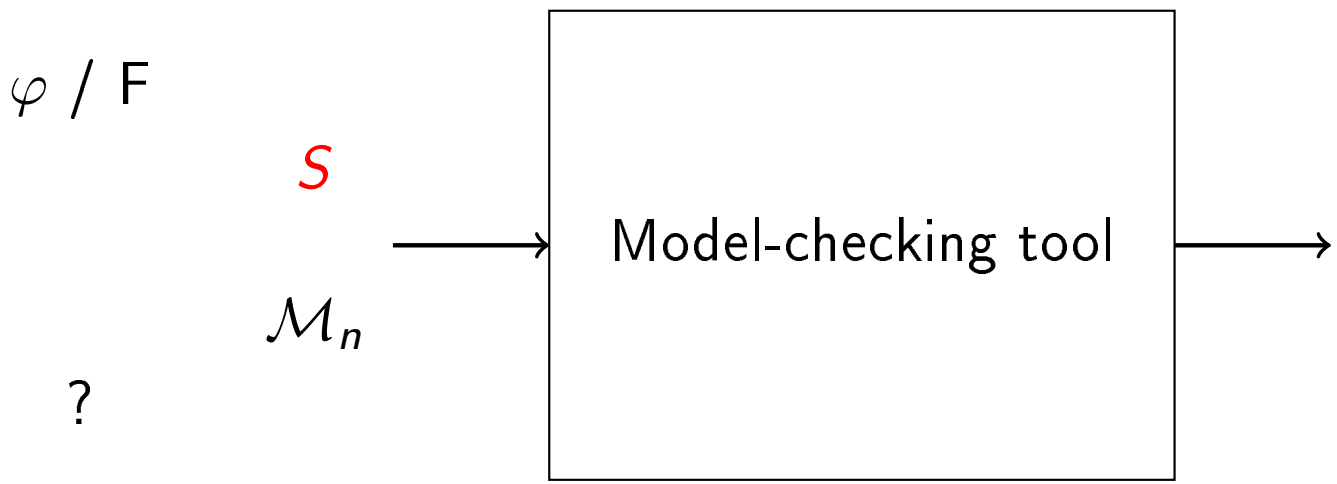
I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



A technique of verification

Model checking [Clarke, Emerson, Sifakis]

I: A specification S, a model $\mathcal{M}$
O: $\mathcal{M} \models S$?



Synthesis and control

Synthesis

I: A specification S

Synthesis and control

Synthesis

I: A specification S
O: $\mathcal{M}$ s.t. $\mathcal{M} \models S$ if it exists

Synthesis and control

Synthesis

I: A specification S
O: $\mathcal{M}$ s.t. $\mathcal{M} \models S$ if it exists

Control

I: A specification S, a partially-defined model $\mathcal{M}$

Synthesis and control

Synthesis

I: A specification S
O: $\mathcal{M}$ s.t. $\mathcal{M} \models S$ if it exists

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \,||\, \mathcal{C} \models S$

Outline

Model

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \mid\mid \mathcal{C} \models S$

Model

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \,||\, \mathcal{C} \models S$

- How to model dynamic parameterized systems?

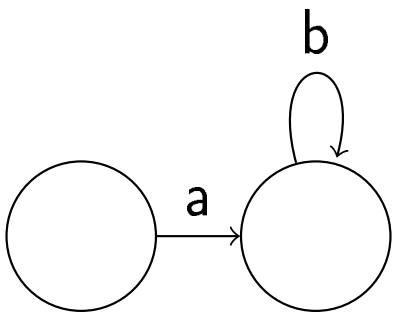
Model

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \,||\, \mathcal{C} \models S$

- How to model dynamic parameterized systems?

1 process



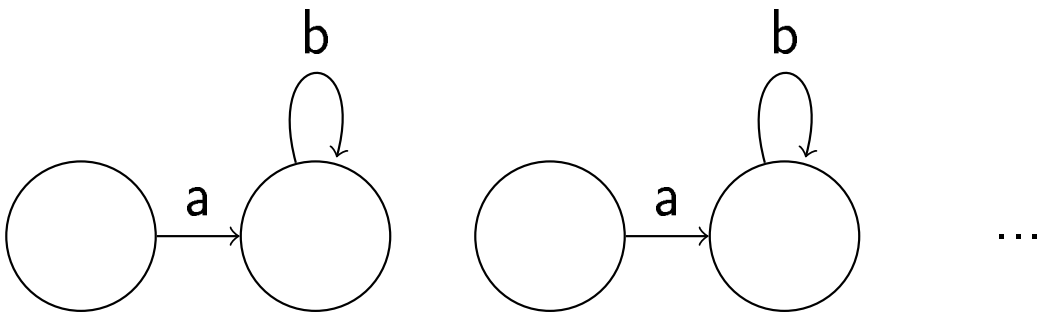
Model

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \,||\, \mathcal{C} \models S$

- How to model dynamic parameterized systems?

n processes



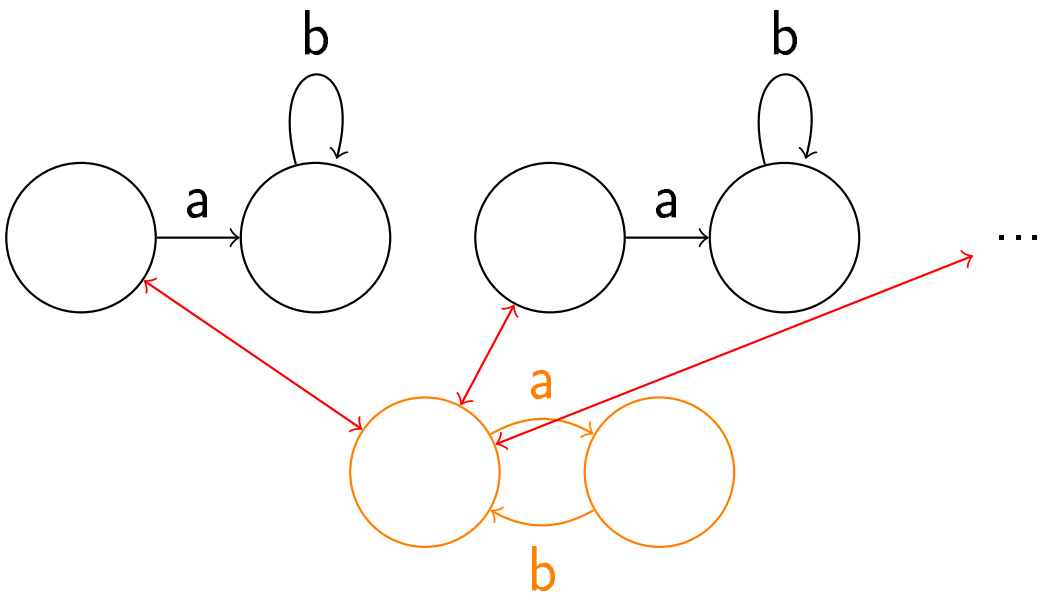
Model

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \,||\, \mathcal{C} \models S$

- How to model dynamic parameterized systems?

n communicating processes



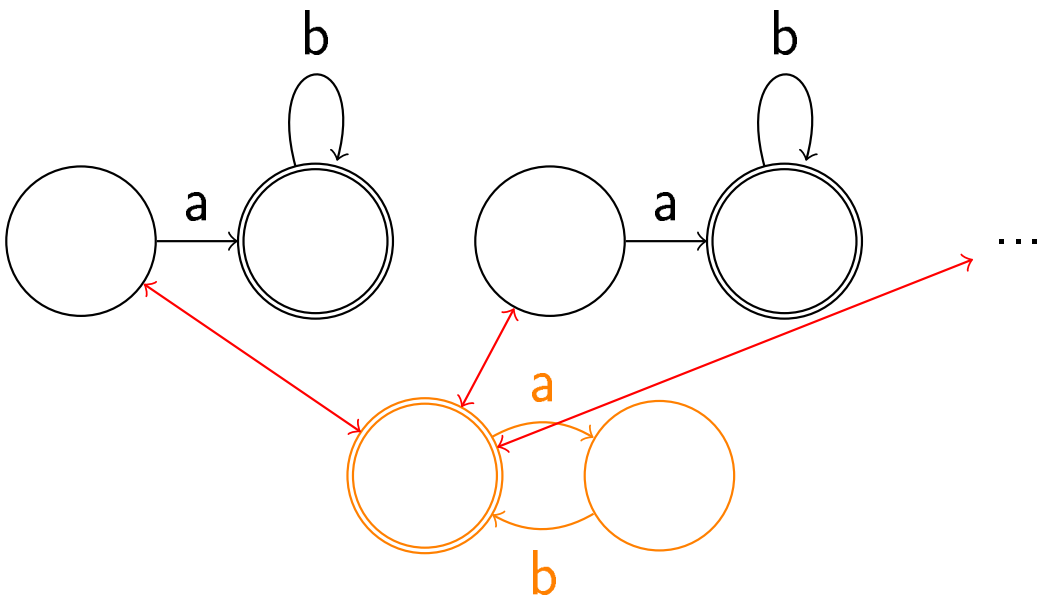
Model

Control

I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \,||\, \mathcal{C} \models S$

- How to model dynamic parameterized systems?

Specification : reachability objective



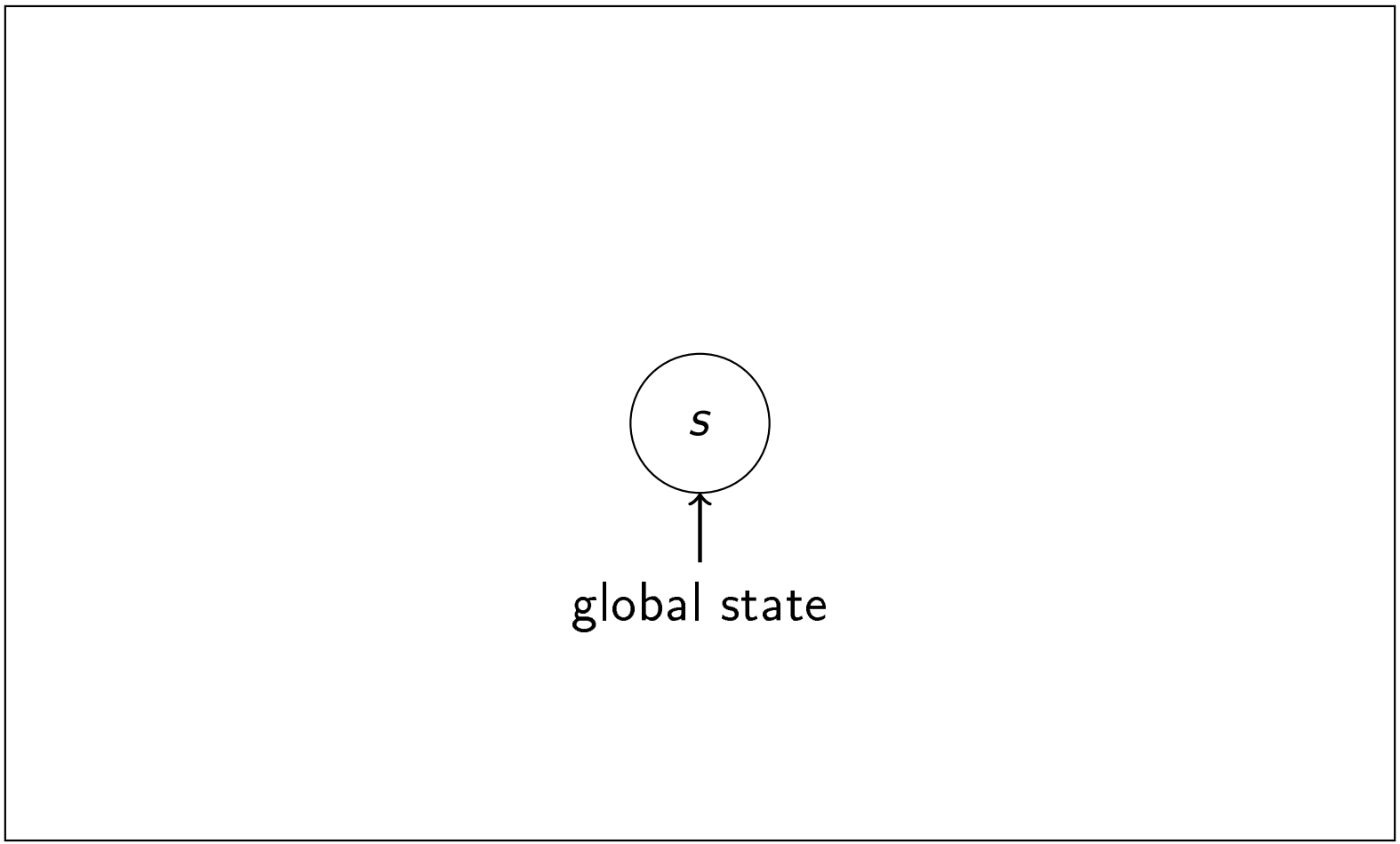
Model

Control

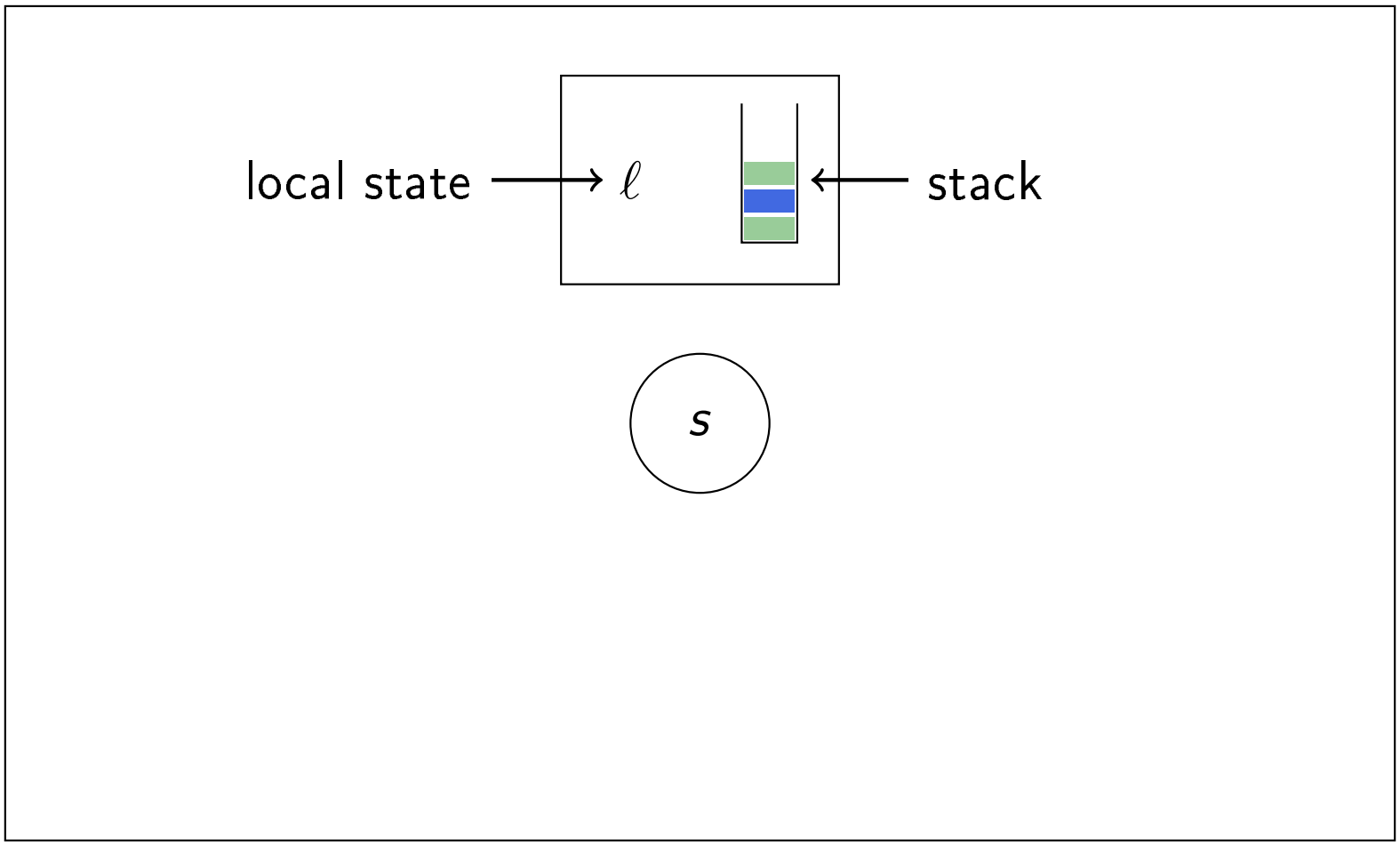
I: A specification S, a partially-defined model $\mathcal{M}$
O: A Controller $\mathcal{C}$ s.t. $\mathcal{M} \mid\mid \mathcal{C} \models S$

- How to model dynamic parameterized systems?
  - Data automata [Bojanczyk et al., 2011]
  - Asynchronous dynamic pushdown networks [Bouajjani et al., 2005]

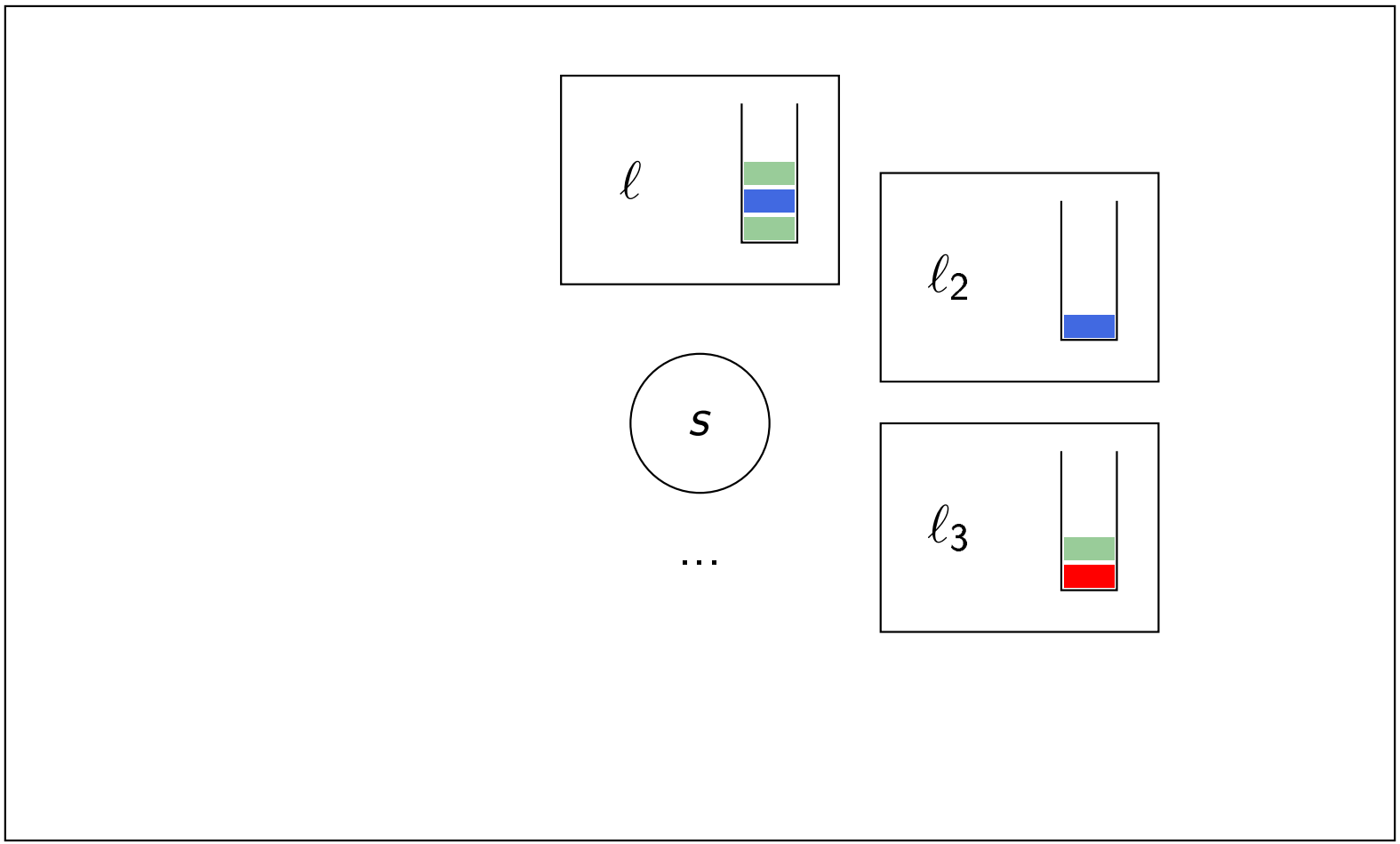
Parameterized Pushdown Systems



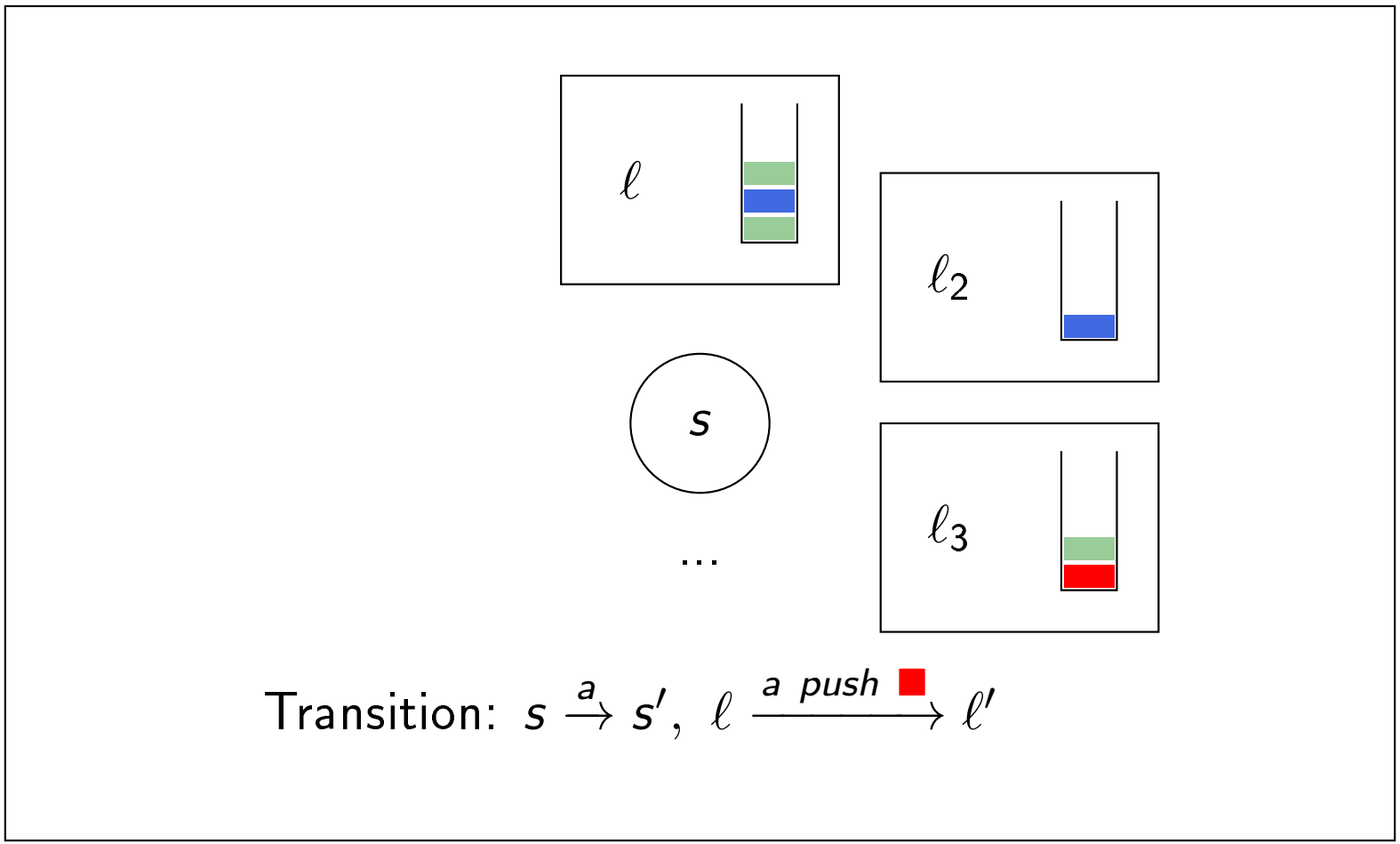
Parameterized Pushdown Systems



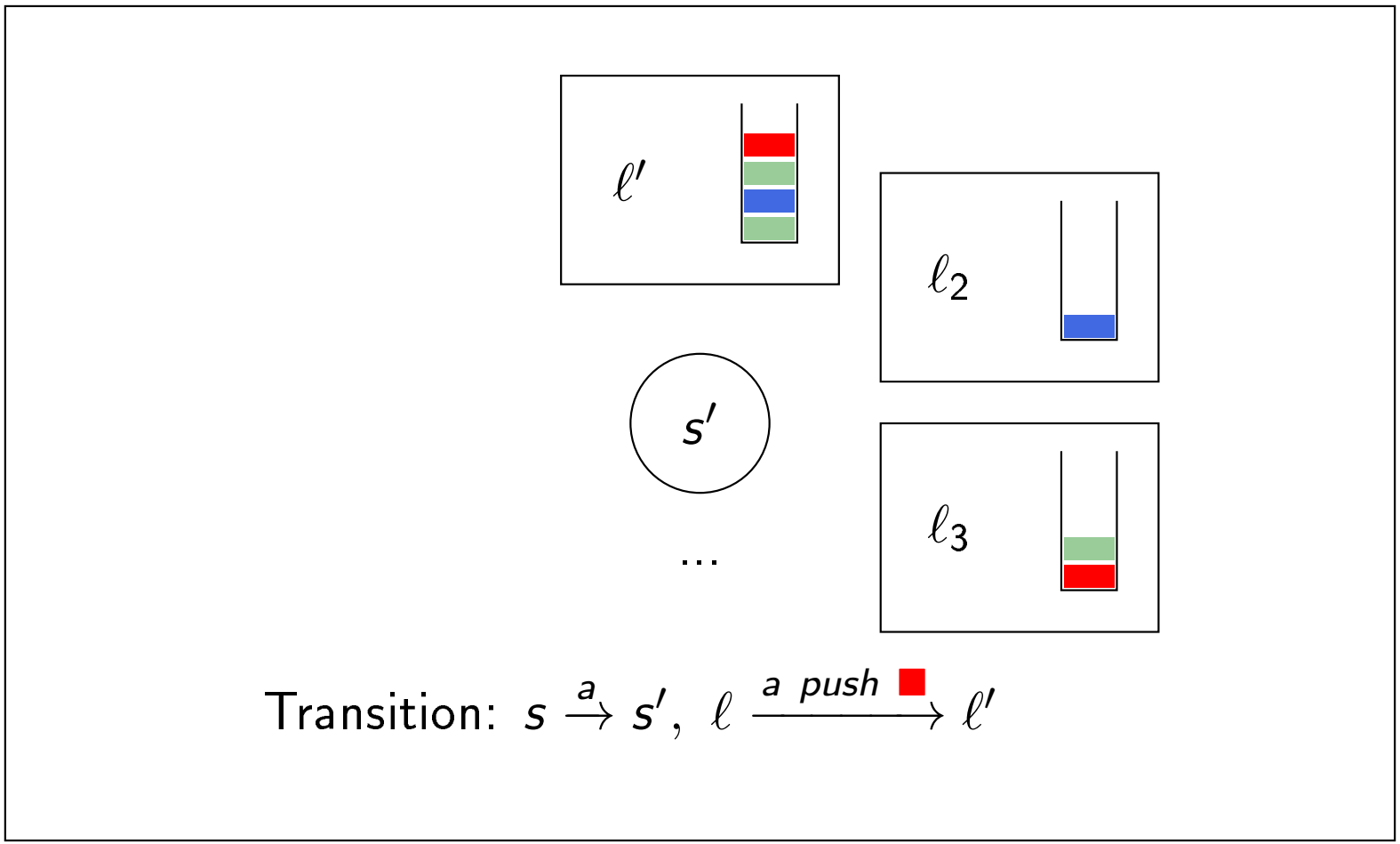
Parameterized Pushdown Systems



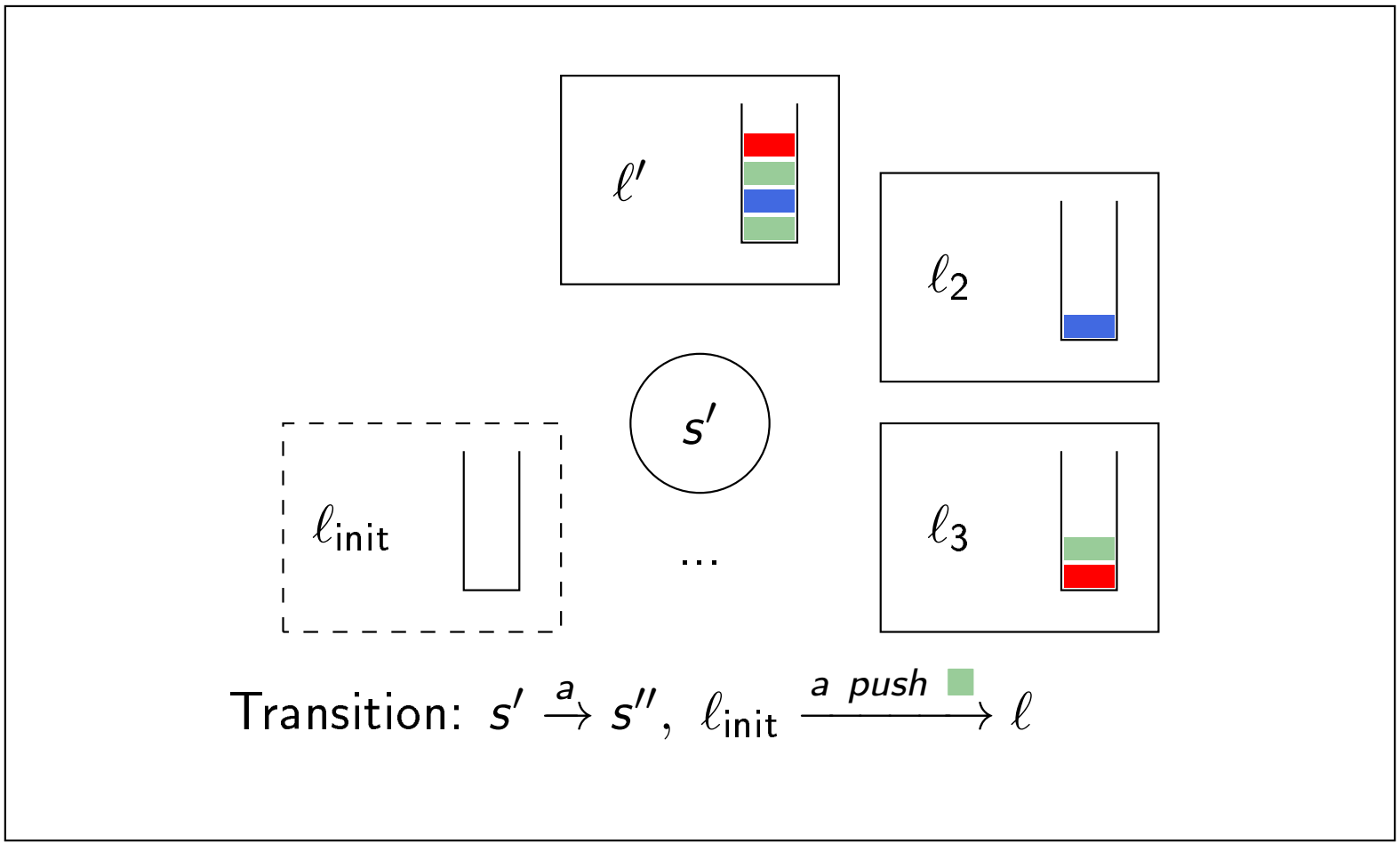
Parameterized Pushdown Systems



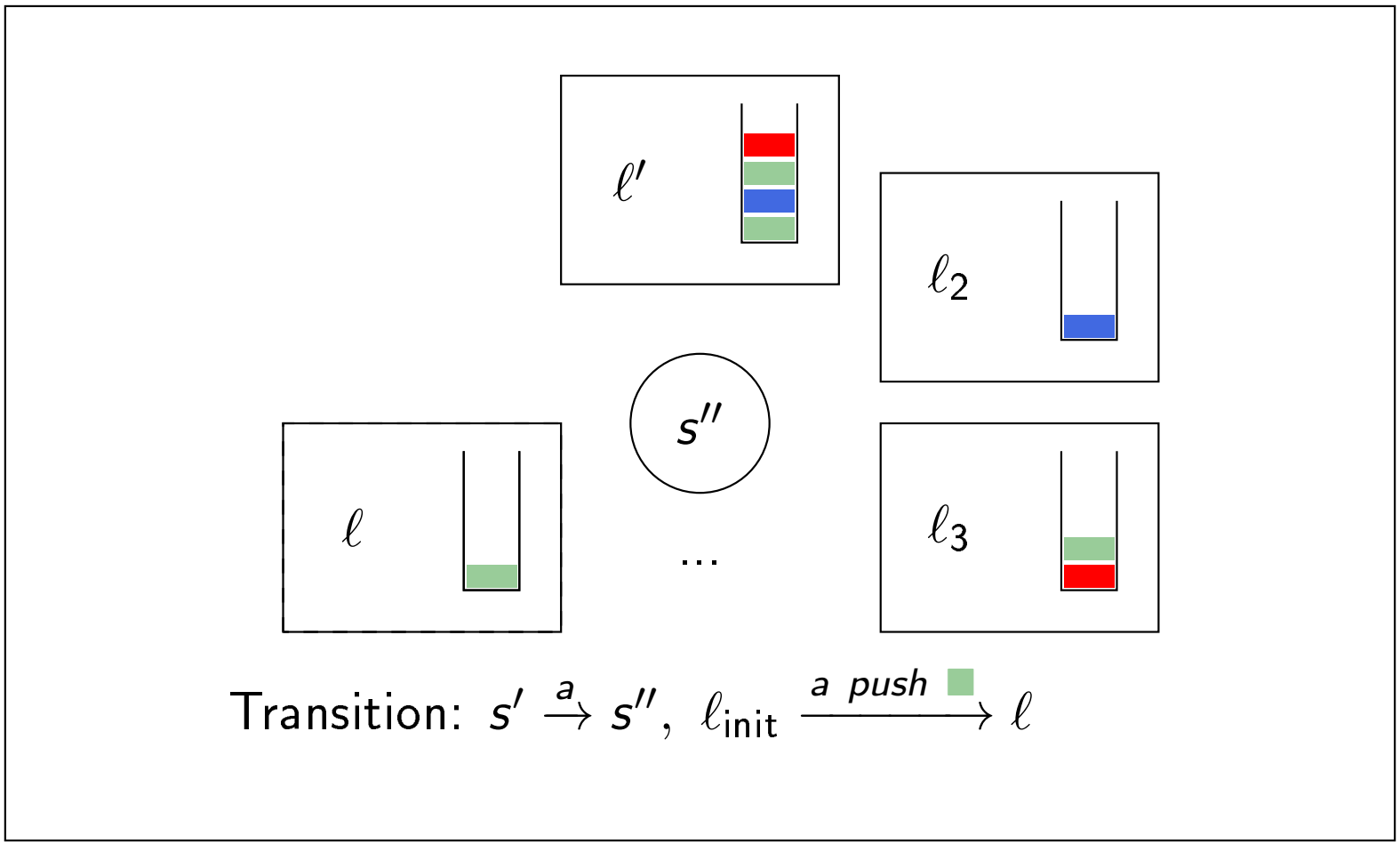
Parameterized Pushdown Systems



Parameterized Pushdown Systems



Parameterized Pushdown Systems

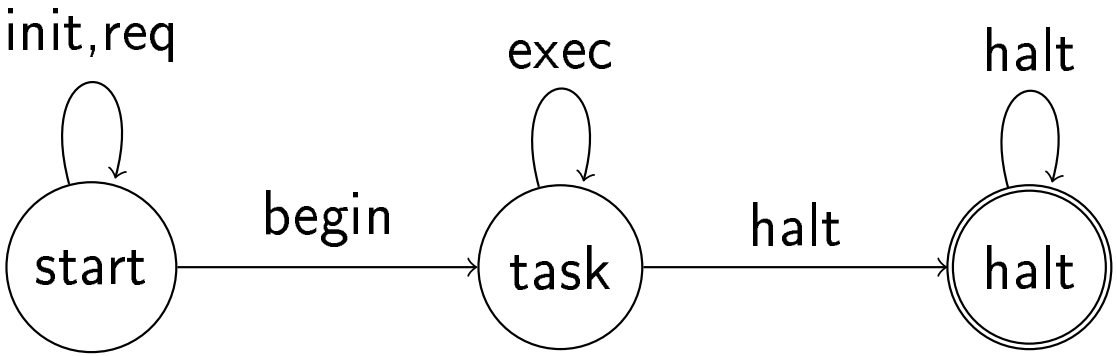


Example: Scheduler model (1/3)

Behavior we want to model:

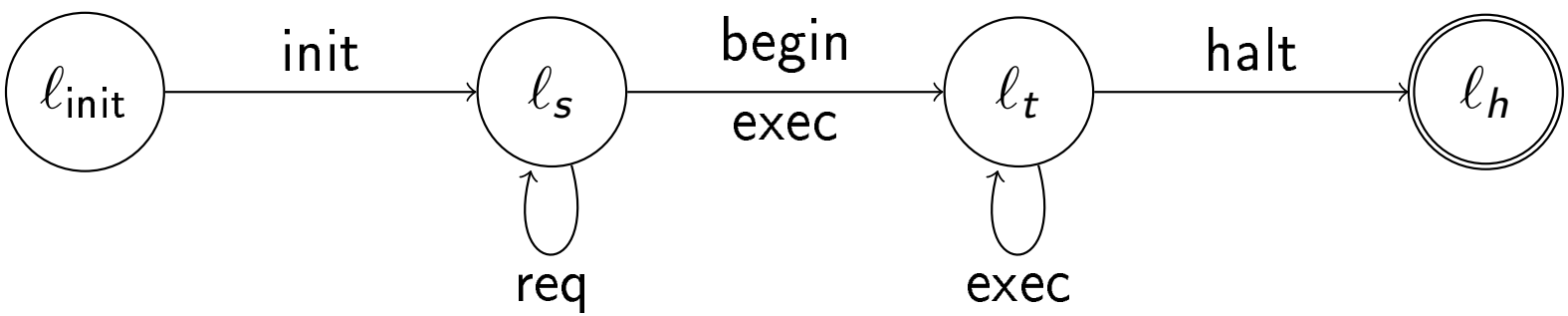
1. Scheduler sends requests and starts processes,
2. Then each process performs the requested tasks,
3. Then each process stops.

Example: Scheduler model (2/3)



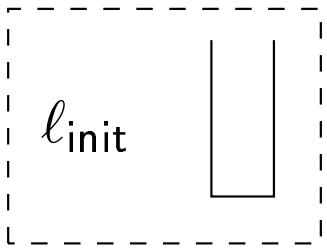
Global automaton

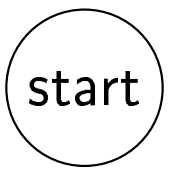
Local automaton



Example: Scheduler model (3/3)

1. Scheduler sends requests and starts processes

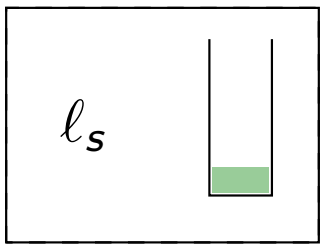


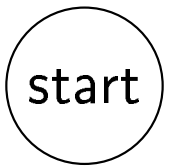


$$\text{start} \xrightarrow{\mathit{init}} \text{start},\ \ell_{\text{init}} \xrightarrow{\mathit{init}\ \mathit{push}\ \blacksquare} \ell_s$$

Example: Scheduler model (3/3)

1. Scheduler sends requests and starts processes

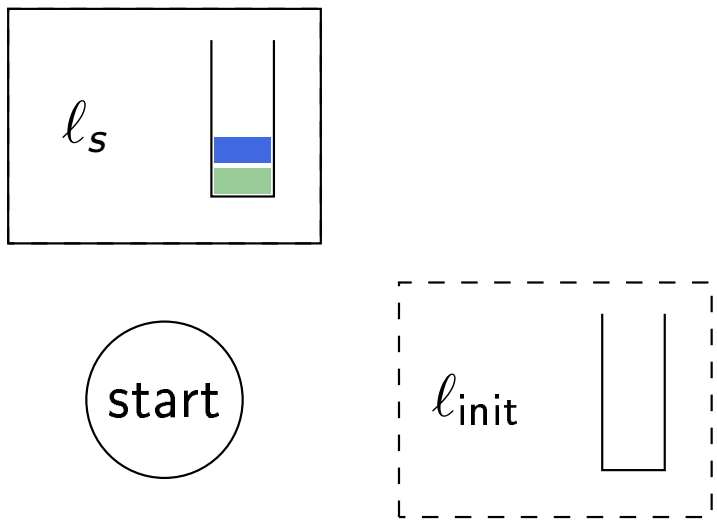




$$\text{start} \xrightarrow{req} \text{start},\ \ell_s \xrightarrow{req\ \ push\ \blacksquare} \ell_s$$

Example: Scheduler model (3/3)

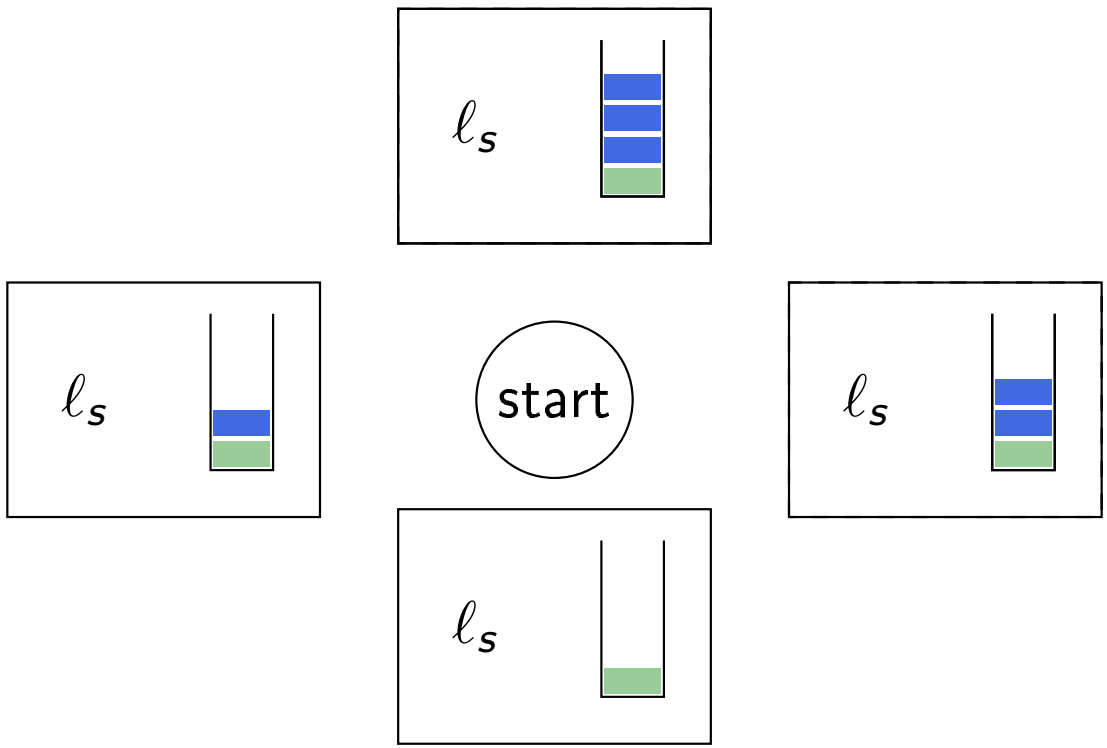
1. Scheduler sends requests and starts processes



$$\text{start} \xrightarrow{\textit{init}} \text{start}, \ \ell_{\text{init}} \xrightarrow{\textit{init push}\ \blacksquare} \ell_s$$
$$\dots$$

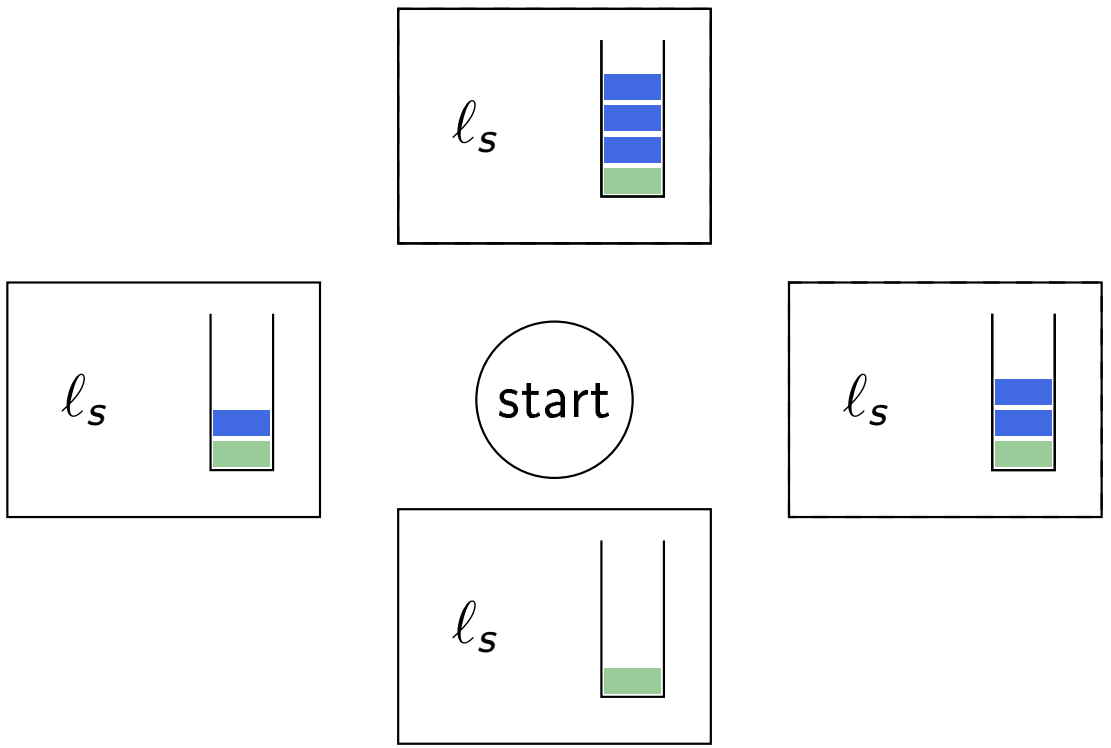
Example: Scheduler model (3/3)

1. Scheduler sends requests and starts processes



Example: Scheduler model (3/3)

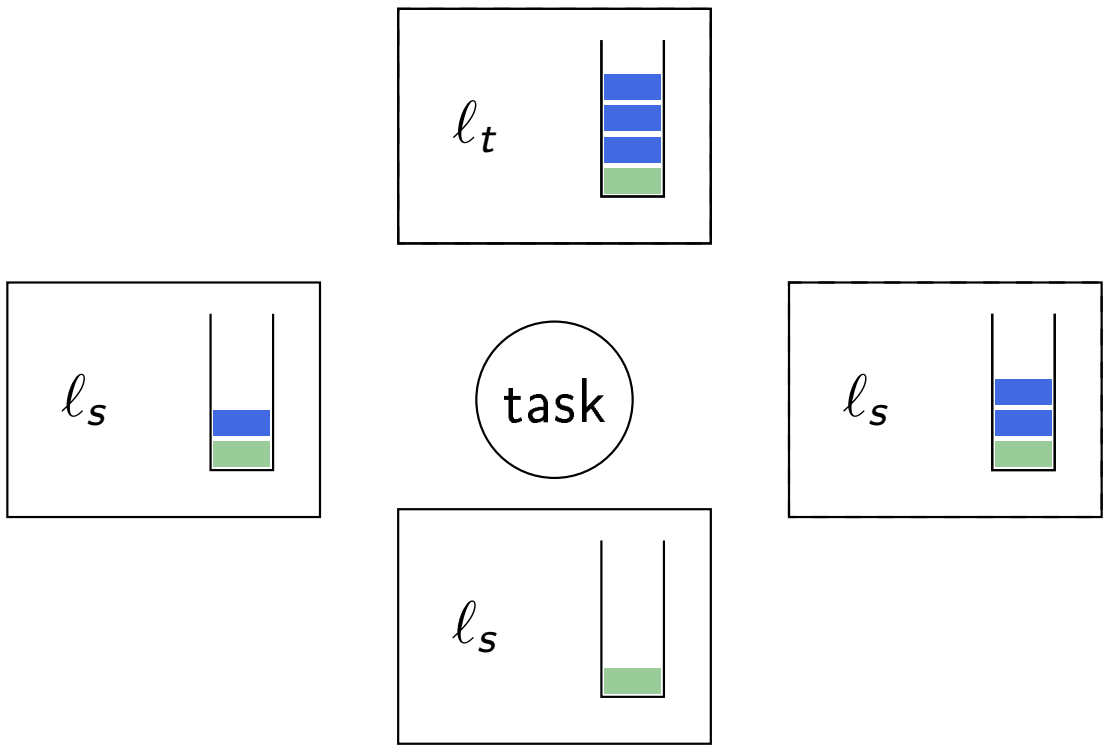
2. Each process performs the requested tasks



$$\text{start} \xrightarrow{\textit{begin}} \text{task},\ \ell_s \xrightarrow{\textit{begin int} -} \ell_t$$

Example: Scheduler model (3/3)

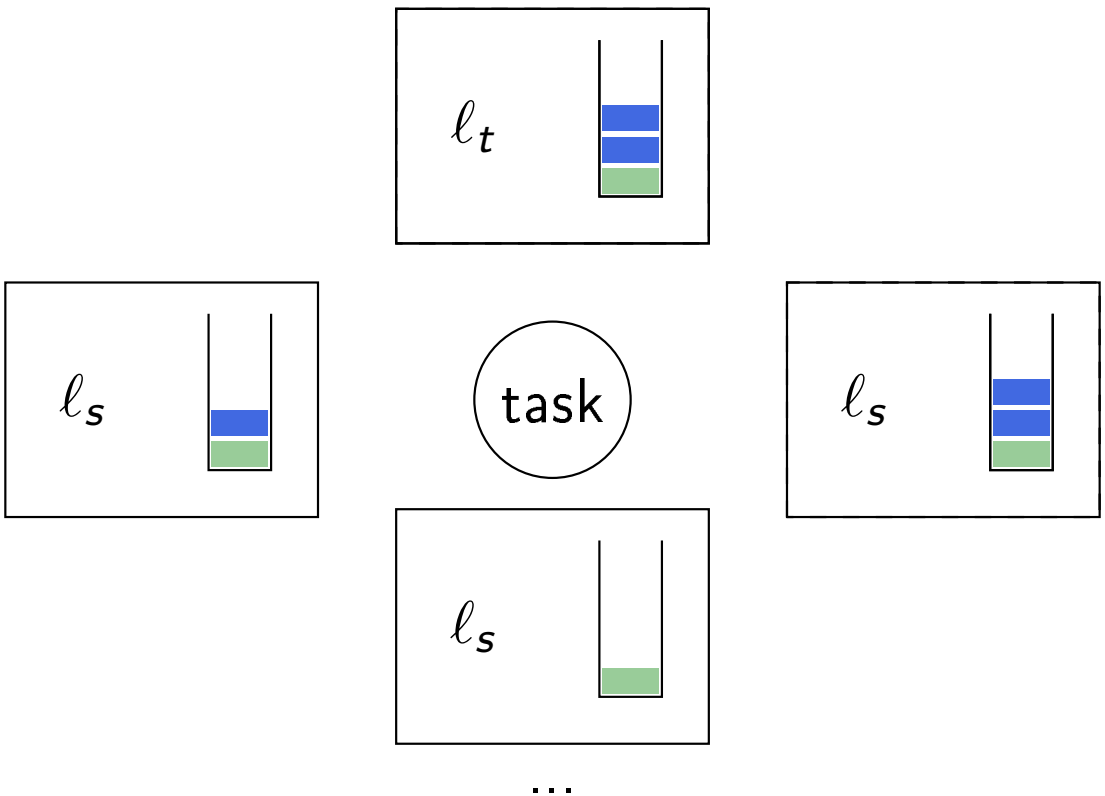
2. Each process performs the requested tasks



$$\text{task} \xrightarrow{exec} \text{task},\ \ell_t \xrightarrow{exec\ pop\ \blacksquare} \ell_t$$

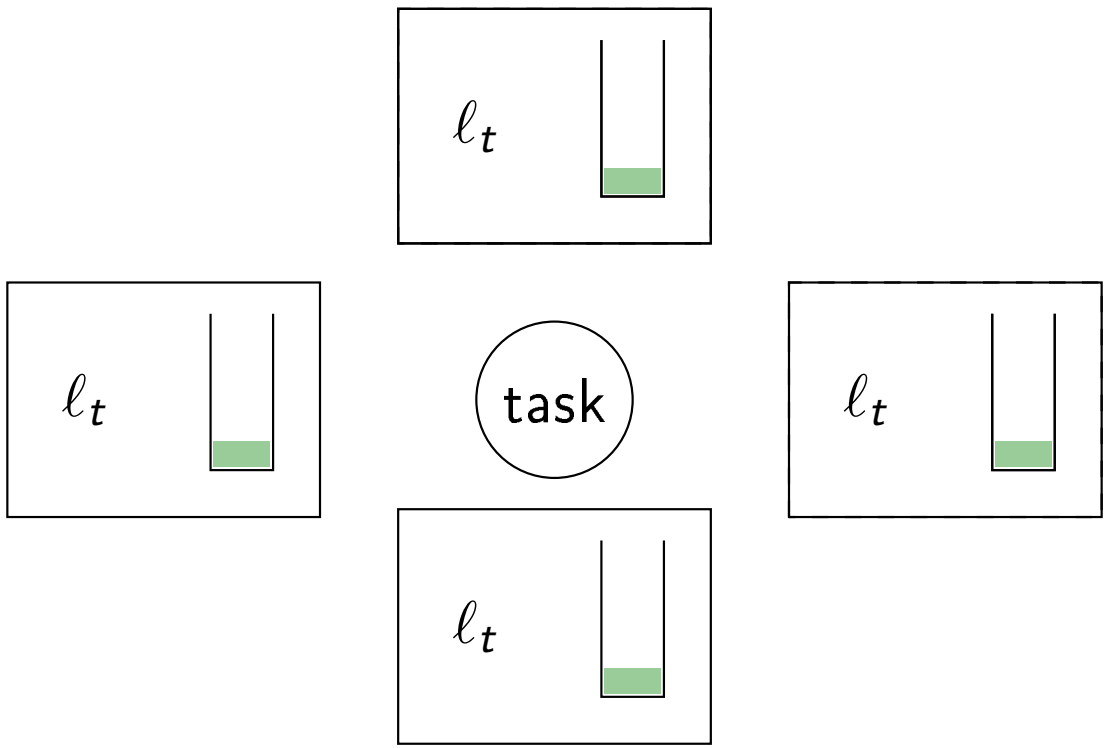
Example: Scheduler model (3/3)

2. Each process performs the requested tasks



Example: Scheduler model (3/3)

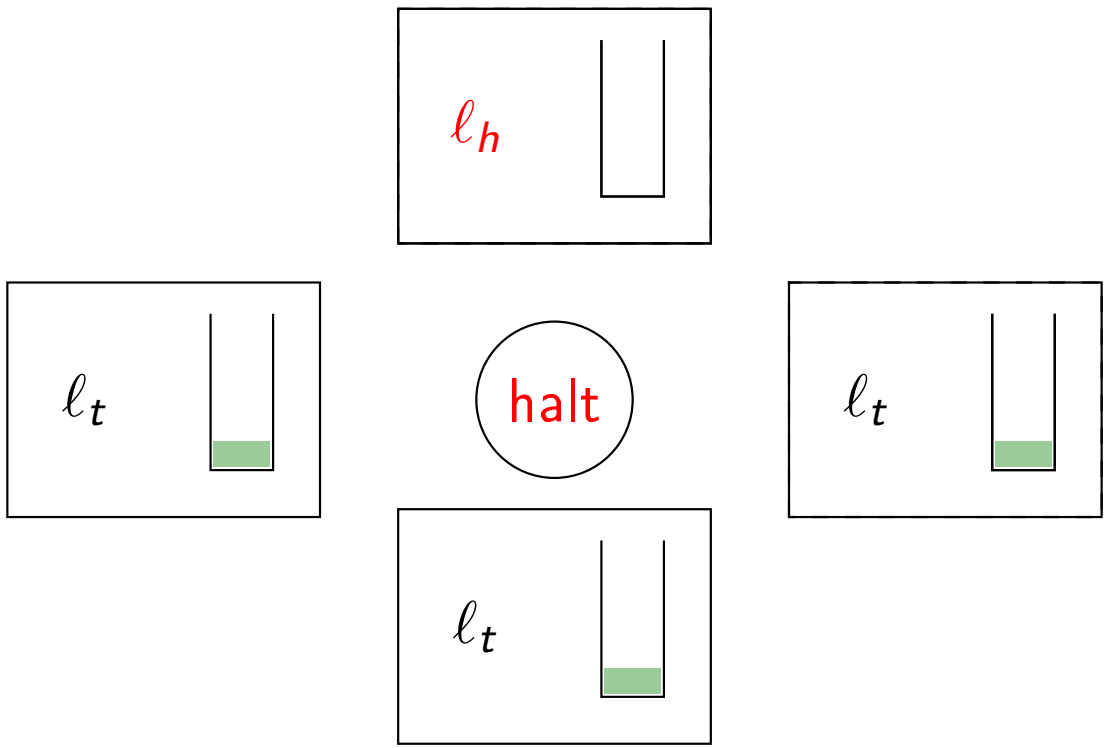
3. Each process stops



$$\text{task} \xrightarrow{halt} \text{halt},\ \ell_t \xrightarrow{halt\ \ pop\ \blacksquare} \ell_h$$

Example: Scheduler model (3/3)

3. Each process stops



Round-bounded behaviors

Even Reachability is undecidable!

Round-bounded behaviors

Even Reachability is undecidable!

- Need underapproximations techniques:

Round-bounded behaviors

Even Reachability is undecidable!

- Need underapproximations techniques:
  - Context-bounded behaviors [Qadeer, Rehof, 2005]

Round-bounded behaviors

Even Reachability is undecidable!

- Need underapproximations techniques:
  - Context-bounded behaviors [Qadeer, Rehof, 2005]
  - Round-bounded behaviors [La Torre et al., 2010]

Round-bounded behaviors

Restriction: rounds

One round:

- ✓ 1 1 1 1 2 2 2 3 3 3 3 3 4 5 5 5 ...
- ✓ 1 1 1 1 2 2 2 2 2 5 5 5 8 8 8 ...
- X 1 1 1 2 2 3 3 1 4 4 6 ...

Round-bounded behaviors

Restriction: rounds

One round:

✓ 1 1 1 1 2 2 2 3 3 3 3 3 4 5 5 5 ...

✓ 1 1 1 1 2 2 2 2 2 5 5 5 8 8 8 ...

X 1 1 1 2 2 3 3 1 4 4 6 ...

N rounds:

1 1 1 2 3 4 4 | 1 3 3 4 5 | 3 3 5 6 6 6 7 | ...

Round-bounded behaviors

Restriction: rounds

One round:

✓ 1 1 1 1 2 2 2 3 3 3 3 3 4 5 5 5 ...

✓ 1 1 1 1 2 2 2 2 2 5 5 5 8 8 8 ...

X 1 1 1 2 2 3 3 1 4 4 6 ...

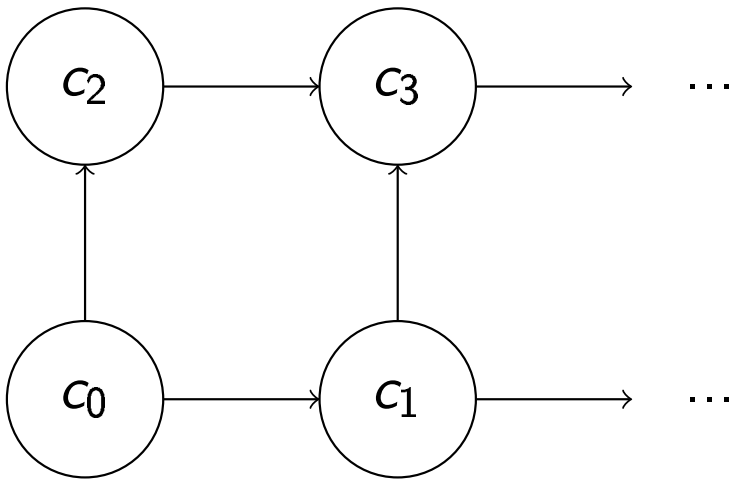
N rounds:

1 1 1 2 3 4 4 | 1 3 3 4 5 | 3 3 5 6 6 6 7 | ...

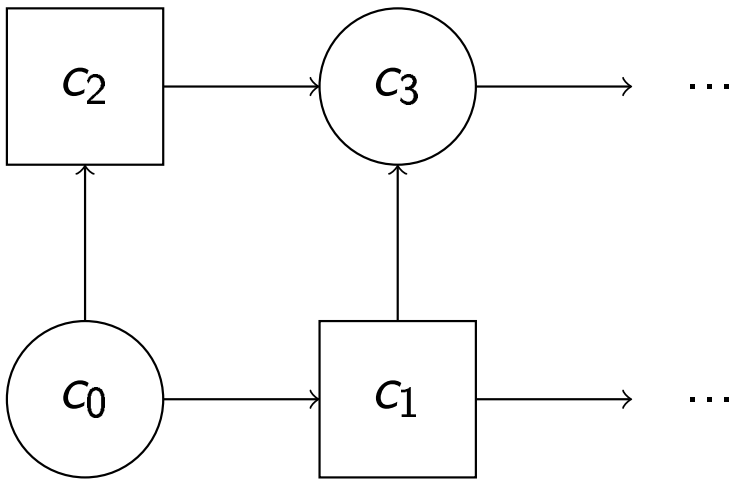
Theorem [La Torre et al., 2010] [ATVA 2018]

Round-bounded Reachability is PSPACE-complete.

Control: Parameterized Pushdown Games

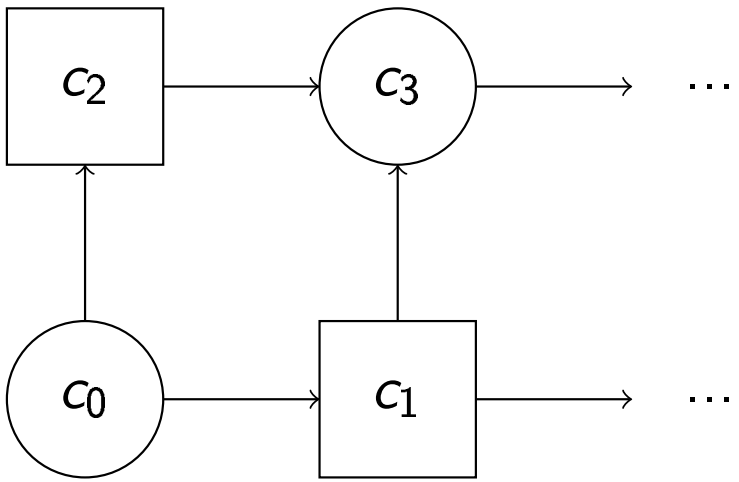


Control: Parameterized Pushdown Games



- Partition of global states.

Control: Parameterized Pushdown Games



- Partition of global states.

Control problem

Is there a winning strategy for the Controller?

Our result

Theorem [ATVA 2018]

The round-bounded control problem is decidable, but inherently non-elementary.

Our result

Theorem [ATVA 2018]

The round-bounded control problem is decidable, but inherently non-elementary.

Proof idea

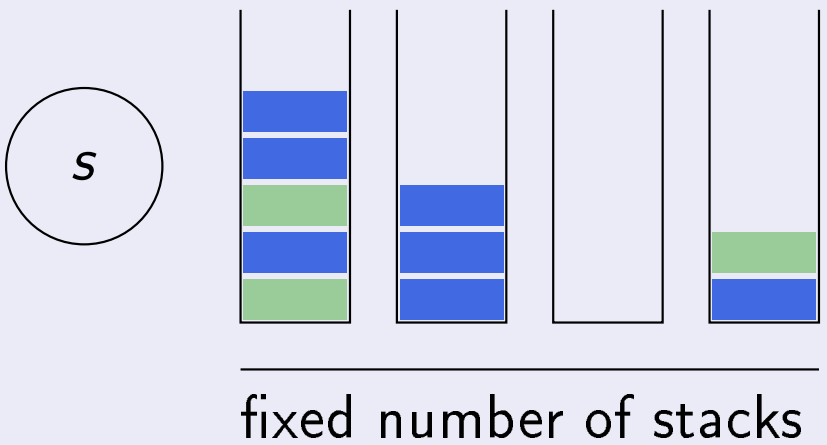
- Decidability: Reduction to *phase-bounded multi-pushdown games* [Atig et al., 2017]
- Hardness: Reduction of satisfiability of $\mathrm{FO}[<]$ on finite words [Stockmeyer, 1970]

Our result

Theorem [ATVA 2018]

The round-bounded control problem is decidable, but inherently non-elementary.

Phase-bounded multi-pushdown systems

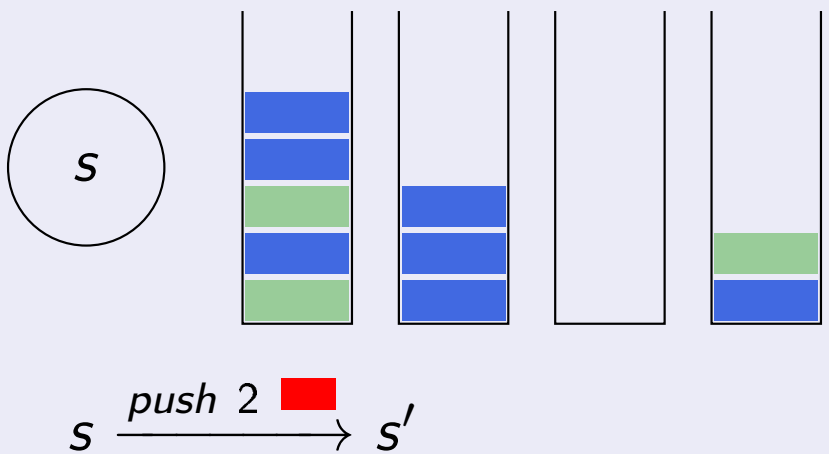


Our result

Theorem [ATVA 2018]

The round-bounded control problem is decidable, but inherently non-elementary.

Phase-bounded multi-pushdown systems

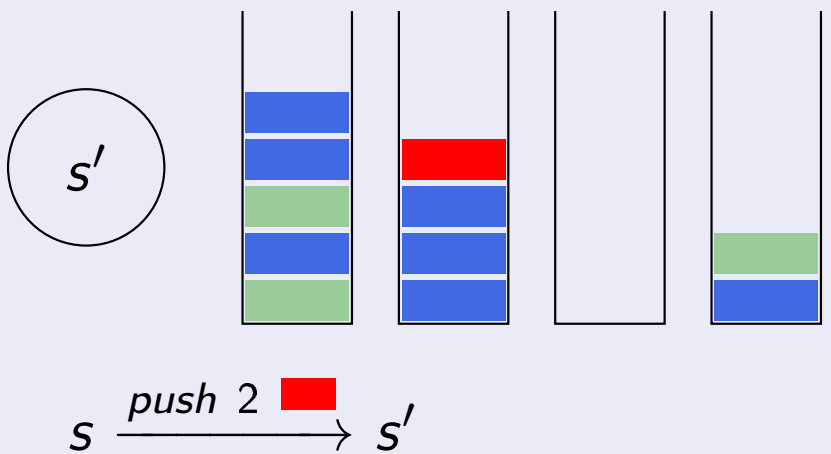


Our result

Theorem [ATVA 2018]

The round-bounded control problem is decidable, but inherently non-elementary.

Phase-bounded multi-pushdown systems



Outline

Parameterized Synthesis

Synthesis

I: A specification S

O: $\mathcal{M}$ s.t. $\mathcal{M} \models S$ if it exists

Parameterized Synthesis

Synthesis

I: A specification S
O: $\mathcal{M}$ s.t. $\mathcal{M} \models S$ if it exists

- Fixed number of processes

Executions I: Data words

Behaviors for $A = \{req, ack\}$

- 1 process: $w = req\ ack\ req\ ack$

Executions I: Data words

Behaviors for $A = \{req, ack\}$

- 1 process: $w = req\ ack\ req\ ack$
- fixed number of processes: $w = req_1\, req_3\, ack_1\, ack_3$

Executions I: Data words

Behaviors for $A = \{req, ack\}$

- 1 process: $w = req\ ack\ req\ ack$
- fixed number of processes: $w = req_1\, req_3\, ack_1\, ack_3$
- unknown (not bounded) number of processes:
  $w = (req, 1)(req, 3)(ack, 1)(req, 6)(ack, 6)(ack, 3)$

Executions I: Data words

Behaviors for $A = \{req, ack\}$

- 1 process: $w = req\ ack\ req\ ack$
- fixed number of processes: $w = req_1\, req_3\, ack_1\, ack_3$
- unknown (not bounded) number of processes:
  $w = (req, 1)(req, 3)(ack, 1)(req, 6)(ack, 6)(ack, 3)$

Data words [Bojanczyk et al., 2006]

- A: finite alphabet (*actions*),
- $\mathcal{D}$: infinite set of data values (*process identities*)

Data word: (in)finite word over $A \times \mathcal{D}$

Executions II: System vs Environment

System actions and Environment actions

- $A = A_{sys} \uplus A_{env}$

Executions II: System vs Environment

System actions and Environment actions

- $A = A_{sys} \uplus A_{env}$

System and Environment processes

- $\mathbb{P} = (\mathbb{P}_{sys}, \mathbb{P}_{env}, \mathbb{P}_{se})$

with $\mathbb{P}_\theta$ finite set of processes

Executions II: System vs Environment

System actions and Environment actions

- $A = A_{sys} \uplus A_{env}$

System and Environment processes

- $\mathbb{P} = (\mathbb{P}_{sys}, \mathbb{P}_{env}, \mathbb{P}_{se})$

with $\mathbb{P}_\theta$ finite set of processes

$$\Sigma_{sys} = A_{sys} \times (\mathbb{P}_{sys} \cup \mathbb{P}_{se})$$

Executions II: System vs Environment

System actions and Environment actions

- $A = A_{sys} \uplus A_{env}$

System and Environment processes

- $\mathbb{P} = (\mathbb{P}_{sys}, \mathbb{P}_{env}, \mathbb{P}_{se})$

with $\mathbb{P}_\theta$ finite set of processes

$$\Sigma_{sys} = A_{sys} \times (\mathbb{P}_{sys} \cup \mathbb{P}_{se})$$
$$\Sigma_{env} = A_{env} \times (\mathbb{P}_{env} \cup \mathbb{P}_{se})$$

Executions II: System vs Environment

System actions and Environment actions

- $A = A_{sys} \uplus A_{env}$

System and Environment processes

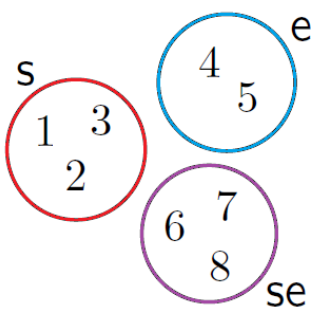
- $\mathbb{P} = (\mathbb{P}_{sys}, \mathbb{P}_{env}, \mathbb{P}_{se})$

with $\mathbb{P}_\theta$ finite set of processes

$$\Sigma_{sys} = A_{sys} \times (\mathbb{P}_{sys} \cup \mathbb{P}_{se})$$
$$\Sigma_{env} = A_{env} \times (\mathbb{P}_{env} \cup \mathbb{P}_{se})$$

$\mathbb{P}$-Execution

Execution = word over $\Sigma_{sys} \cup \Sigma_{env}$



$A_{sys} = \{a, b\}$ $A_{env} = \{c, d\}$

$$\underset{1}{a} \to \underset{8}{b} \to \underset{7}{d} \to \underset{4}{c} \to \underset{6}{a} \to \underset{6}{c} \to \underset{7}{a} \to \underset{6}{d} \to \underset{2}{b} \to \underset{7}{d} \to \underset{7}{a}$$

Executions III: Strategies

- *Asynchronous* synthesis problem

Strategy for System

$f : \Sigma^* \to \Sigma_{sys} \cup \{\varepsilon\}$

Executions III: Strategies

- *Asynchronous* synthesis problem

Strategy for System

$f : \Sigma^* \to \Sigma_{sys} \cup \{\varepsilon\}$

An execution is

- f-compatible if System actions follow f
- f-fair if Environment does not always block System

Executions III: Strategies

- *Asynchronous* synthesis problem

Strategy for System

$f : \Sigma^* \to \Sigma_{sys} \cup \{\varepsilon\}$

An execution is

- f-compatible if System actions follow f
- f-fair if Environment does not always block System

Winning strategy

f is winning for a set S of executions if all f-compatible, f-fair executions are in S

Executions III: Strategies

- *Asynchronous* synthesis problem

Strategy for System

$f : \Sigma^* \to \Sigma_{sys} \cup \{\varepsilon\}$

An execution is

- f-compatible if System actions follow f
- f-fair if Environment does not always block System

Winning strategy

f is winning for a set S of executions if all f-compatible, f-fair executions are in S?

First order logic I: Definition

Example on words

$\varphi = \forall x.\,(req(x) \Rightarrow \exists y.(y > x \wedge ack(y)))$

First order logic I: Definition

Example on words

$\varphi = \forall x.\,(req(x) \Rightarrow \exists y.(y > x \wedge ack(y)))$

"every *req* is eventually followed by an *ack*"

First order logic I: Definition

Example on words

$\varphi = \forall x.\,(req(x) \Rightarrow \exists y.(y > x \wedge ack(y)))$
"every *req* is eventually followed by an *ack*"

- $req\ ack\ req\ req\ ack \models \varphi$
- $req\ req\ ack\ req \not\models \varphi$

First order logic I: Definition

Example on words

$\varphi = \forall x.\,(req(x) \Rightarrow \exists y.(y > x \wedge ack(y)))$
"every *req* is eventually followed by an *ack*"

- $req\ ack\ req\ req\ ack \models \varphi$
- $req\ req\ ack\ req \not\models \varphi$

Syntax for FO on words

Basic formulas: $a(x) \mid x = y \mid x < y \mid \mathrm{succ}(x, y)$
$a \in A$
Connectors and quantifiers: $\neg, \vee, \wedge, \Rightarrow, \exists, \forall$

First order logic I: Definition

Examples on data words

$\varphi = \forall x.(req(x) \Rightarrow \exists y.(y \sim x \wedge y > x \wedge ack(y)))$

"every *req* is eventually followed by an *ack* on the same process"

First order logic I: Definition

Examples on data words

$\varphi = \forall x.(req(x) \Rightarrow \exists y.(y \sim x \wedge y > x \wedge ack(y)))$

"every *req* is eventually followed by an *ack* on the same process"

- $(req, 1)(req, 3)(ack, 1)(req, 6)(ack, 6)(ack, 3) \models \varphi$
- $(req, 1)(ack, 2)(req, 1)(ack, 2) \cdots \not\models \varphi$

First order logic I: Definition

Examples on data words

$\varphi = \forall x.(req(x) \Rightarrow \exists y.(y \sim x \wedge y > x \wedge ack(y)))$
"every *req* is eventually followed by an *ack* on the same process"

- $(req, 1)(req, 3)(ack, 1)(req, 6)(ack, 6)(ack, 3) \models \varphi$
- $(req, 1)(ack, 2)(req, 1)(ack, 2) \cdots \not\models \varphi$

Syntax for FO on datawords

Basic formulas: $a(x) \mid x = y \mid x < y \mid \mathrm{succ}(x, y) \mid \theta(x) \mid x \sim y$
$a \in A$, $\theta \in \{sys, env, se\}$
Connectors and quantifiers: $\neg, \vee, \wedge, \Rightarrow, \exists, \forall$

First order logic II: Satisfiability

- Specification $S_\varphi = \{w \mid w \models \varphi\}$

First order logic II: Satisfiability

- Specification $S_\varphi = \{w \mid w \models \varphi\}$

Satisfiability

I: A first order formula φ
O: $S_\varphi \neq \emptyset$?

First order logic II: Satisfiability

- Specification $S_\varphi = \{w \mid w \models \varphi\}$

Satisfiability

I: A first order formula φ
O: $S_\varphi \neq \emptyset$?

- Decidable for words (but non-elementary) [Büchi, 60]

First order logic II: Satisfiability

- Specification $S_\varphi = \{w \mid w \models \varphi\}$

Satisfiability

I: A first order formula φ
O: $S_\varphi \neq \emptyset$?

- Decidable for words (but non-elementary) [Büchi, 60]
- Undecidable for data words [Neven et al., 04]

Winning triples

- Only important point for synthesis is number of processes, not concrete identities!

Winning triples

- Only important point for synthesis is number of processes, not concrete identities!

Winning triples for φ

$(n_{sys}, n_{env}, n_{se}) \in \mathbb{N}^3$ is a winning triple if there is a winning strategy for data words limited to $(n_{sys}, n_{env}, n_{se})$ processes

Winning triples

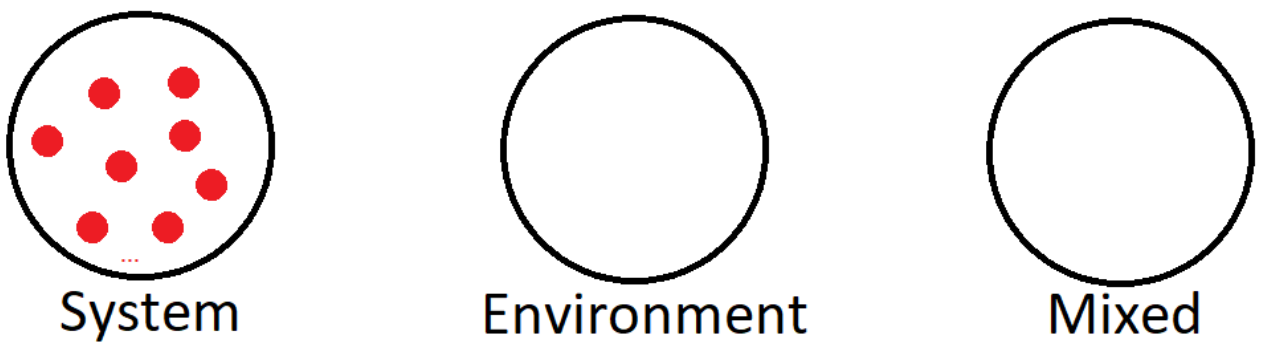
- Only important point for synthesis is number of processes, not concrete identities!

Winning triples for φ

$(n_{sys}, n_{env}, n_{se}) \in \mathbb{N}^3$ is a winning triple if there is a winning strategy for data words limited to $(n_{sys}, n_{env}, n_{se})$ processes

Intersection of set of winning triples $Win(\varphi)$ with:

$\mathbb{N} \times \{0\} \times \{0\}$: only System processes (satisfiability)



Winning triples

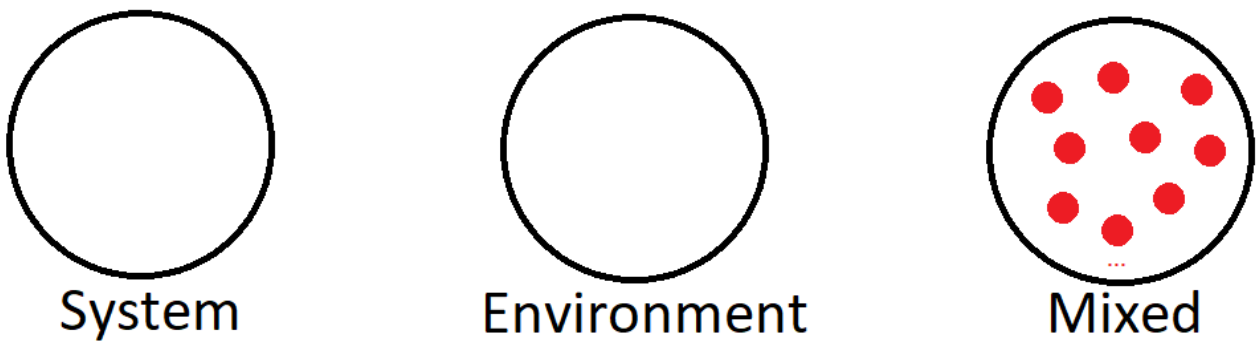
- Only important point for synthesis is number of processes, not concrete identities!

Winning triples for φ

$(n_{sys}, n_{env}, n_{se}) \in \mathbb{N}^3$ is a winning triple if there is a winning strategy for data words limited to $(n_{sys}, n_{env}, n_{se})$ processes

Intersection of set of winning triples $Win(\varphi)$ with:

$\{0\} \times \{0\} \times \mathbb{N}$: each process controlled by both System and Environment



Winning triples

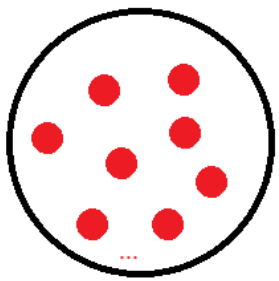
- Only important point for synthesis is number of processes, not concrete identities!

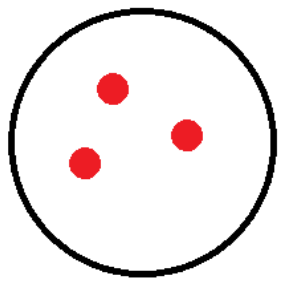
Winning triples for φ

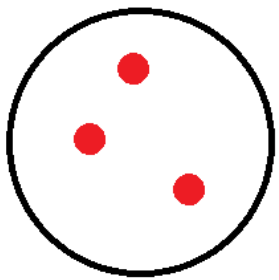
$(n_{sys}, n_{env}, n_{se}) \in \mathbb{N}^3$ is a winning triple if there is a winning strategy for data words limited to $(n_{sys}, n_{env}, n_{se})$ processes

Intersection of set of winning triples $Win(\varphi)$ with:

$\mathbb{N} \times \{k_{env}\} \times \{k_{se}\}$: constant number of Environment and mixed processes, but unboundedly many System processes







Parameterized synthesis problem

SYNTH($\mathcal{F}, (\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se})$)

I: Alphabet $A = A_{sys} \uplus A_{env}$, formula $\varphi \in \mathcal{F}$ over A

O: $Win(\varphi) \cap (\mathcal{N}_{sys} \times \mathcal{N}_{env} \times \mathcal{N}_{se}) \neq \emptyset$?

Parameterized synthesis problem

Example 1

$\varphi_1 = \forall x.(req(x) \Rightarrow \exists y.(y \sim x \wedge y > x \wedge ack(y)))$

- $A_{sys} = \{ack\}$,
- $A_{env} = \{req\}$,
- $(\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se}) = (\{0\}, \{0\}, \mathbb{N})$

Parameterized synthesis problem

Example 1

$\varphi_1 = \forall x.(req(x) \Rightarrow \exists y.(y \sim x \wedge y > x \wedge ack(y)))$

- $A_{sys} = \{ack\}$,
- $A_{env} = \{req\}$,
- $(\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se}) = (\{0\}, \{0\}, \mathbb{N})$

▶ $(0, 0, k)$ is a winning triple for φ_1 for all $k \in \mathbb{N}$:

Winning strategy

$f(w) = (ack, i)$ s.t. $\sigma = (req, i)$ is the first pending req of w

Parameterized synthesis problem

Example 2

$\varphi_2 = (\neg\exists x.a(x)) \Leftrightarrow (\forall y.sys(y) \Rightarrow \exists z.z \sim y \wedge b(z))$

- $A_{sys} = \{b\}$,
- $A_{env} = \{a\}$,
- $(\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se}) = (\mathbb{N}, \{k_{env}\}, \{k_{se}\})$

Parameterized synthesis problem

Example 2

$\varphi_2 = (\neg\exists x.a(x)) \Leftrightarrow (\forall y.sys(y) \Rightarrow \exists z.z \sim y \wedge b(z))$

- $A_{sys} = \{b\}$,
- $A_{env} = \{a\}$,
- $(\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se}) = (\mathbb{N}, \{k_{env}\}, \{k_{se}\})$

▶ No winning triple unless $k_{env} = k_{se} = 0$!

Two-variable first-order logic: FO^2

- FO^2: restrict to two variable names

Two-variable first-order logic: FO^2

- FO^2: restrict to two variable names

Examples

- $\exists x, y, z. \neg(x \sim y) \wedge \neg(y \sim z) \wedge \neg(x \sim z) \notin \mathrm{FO}^2$

Two-variable first-order logic: FO^2

- FO^2: restrict to two variable names

Examples

- $\exists x, y, z. \neg(x \sim y) \wedge \neg(y \sim z) \wedge \neg(x \sim z) \notin \mathrm{FO}^2$
- $\exists x. a(x) \wedge (\exists y. x < y \wedge a(y) \wedge (\exists x. y < x \wedge a(x))) \in \mathrm{FO}^2$

Two-variable first-order logic: FO^2

- FO^2: restrict to two variable names

Examples

- $\exists x, y, z. \neg(x \sim y) \wedge \neg(y \sim z) \wedge \neg(x \sim z) \notin \mathrm{FO}^2$
- $\exists x. a(x) \wedge (\exists y. x < y \wedge a(y) \wedge (\exists x. y < x \wedge a(x))) \in \mathrm{FO}^2$

- Satisfiability is decidable! [Bojanczyk et al., 06]

Results for FO^2

Theorem [FoSSaCS 20]

SYNTH($\mathrm{FO}^2, (\{0\}, \{0\}, \mathbb{N})$) is undecidable

Results for FO^2

Theorem [FoSSaCS 20]

SYNTH($\mathrm{FO}^2, (\{0\}, \{0\}, \mathbb{N})$) is undecidable

Proof

Adapt proof of [Figueira and Praveen, 18] to reduce halting problem for D2CM:

- Counters value encoded by number of processes with an action from System but not Environment (and vice versa)
- FO^2 formula to enforce simulation of a run

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.\mathit{bcast}(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge \mathit{rcv}(z)))$$

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.bcast(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge rcv(z)))$$

Some remarks

- No way to specify an order

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.\mathit{bcast}(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge \mathit{rcv}(z)))$$

Some remarks

- No way to specify an order
- Can count letters on a given class up to some bound B

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.\mathit{bcast}(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge \mathit{rcv}(z)))$$

Some remarks

- No way to specify an order
- Can count letters on a given class up to some bound B
- Can count such classes up to some number

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.bcast(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge rcv(z)))$$

Some remarks

- No way to specify an order
- Can count letters on a given class up to some bound B
- Can count such classes up to some number

Roadmap

1. Establish normal form for FO[∼]

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.bcast(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge rcv(z)))$$

Some remarks

- No way to specify an order
- Can count letters on a given class up to some bound B
- Can count such classes up to some number

Roadmap

1. Establish normal form for FO[∼]
2. Translate to game formalism

FO[∼]

- FO[∼] = FO without $<$ and succ

$$\exists x.bcast(x) \wedge \forall y.(y \not\sim x \Rightarrow \exists z.(z \sim y \wedge rcv(z)))$$

Some remarks

- No way to specify an order
- Can count letters on a given class up to some bound B
- Can count such classes up to some number

Roadmap

1. Establish normal form for FO[∼]
2. Translate to game formalism
3. Use games to prove results

Normal form

Normal form [FoSSaCS 20]

There is a bound $\mathrm{B} \in \mathbb{N}$ s.t. φ is equivalent to a disjunction of conjunctions of formulas of the form

$$\exists^{\bowtie m} y.(\theta(y) \wedge \psi_{\mathrm{B},\ell}(y))$$

Normal form

Normal form [FoSSaCS 20]

There is a bound $\mathrm{B} \in \mathbb{N}$ s.t. φ is equivalent to a disjunction of conjunctions of formulas of the form

$$\exists^{\bowtie m} y.(\theta(y) \wedge \psi_{\mathrm{B},\ell}(y))$$

$\equiv$ "There are $\bowtie$ m processes of type θ with local state ℓ."

Normal form

Normal form [FoSSaCS 20]

There is a bound $\mathrm{B} \in \mathbb{N}$ s.t. φ is equivalent to a disjunction of conjunctions of formulas of the form

$$\exists^{\bowtie m} y.(\theta(y) \wedge \psi_{\mathrm{B},\ell}(y))$$

$\equiv$ "There are $\bowtie m$ processes of type θ with local state ℓ."

Normal form

Normal form [FoSSaCS 20]

There is a bound $\mathrm{B} \in \mathbb{N}$ s.t. φ is equivalent to a disjunction of conjunctions of formulas of the form

$$\exists^{\bowtie m} y.(\theta(y) \wedge \psi_{\mathrm{B},\ell}(y))$$

$\equiv$ "There are $\bowtie m$ processes of type θ with local state ℓ."

- Local state of a process $\ell : A \to \{0, \ldots, \mathrm{B}\}$

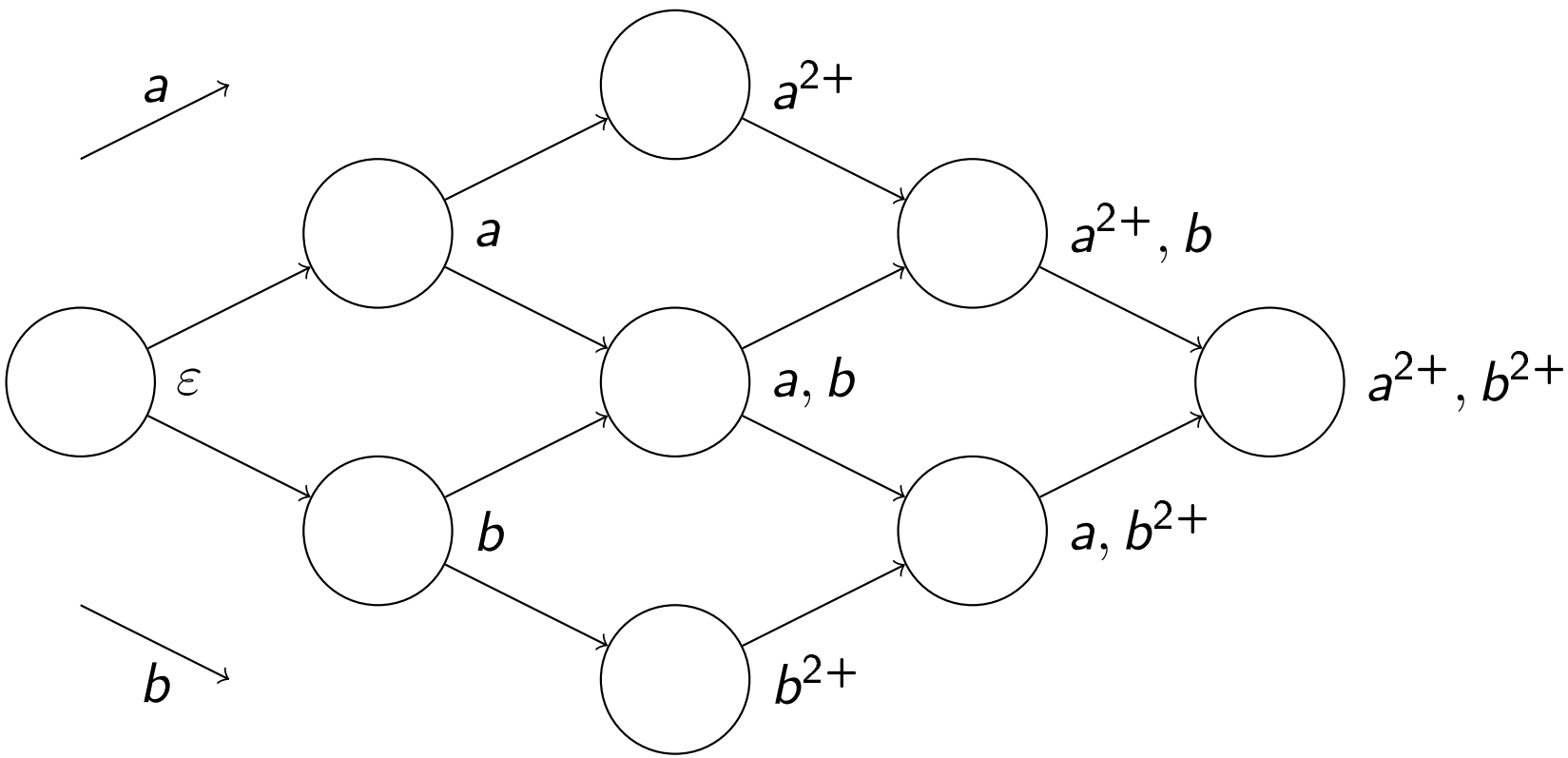
Parameterized Vector Games

Game framework for $\mathrm{FO}[\sim]$ formulas

$\mathcal{G} = (A, \mathrm{B}, \mathfrak{F})$ where $A = A_{sys} \uplus A_{env}$, $\mathrm{B} > 0$, and $\mathfrak{F}$ is the *acceptance condition*

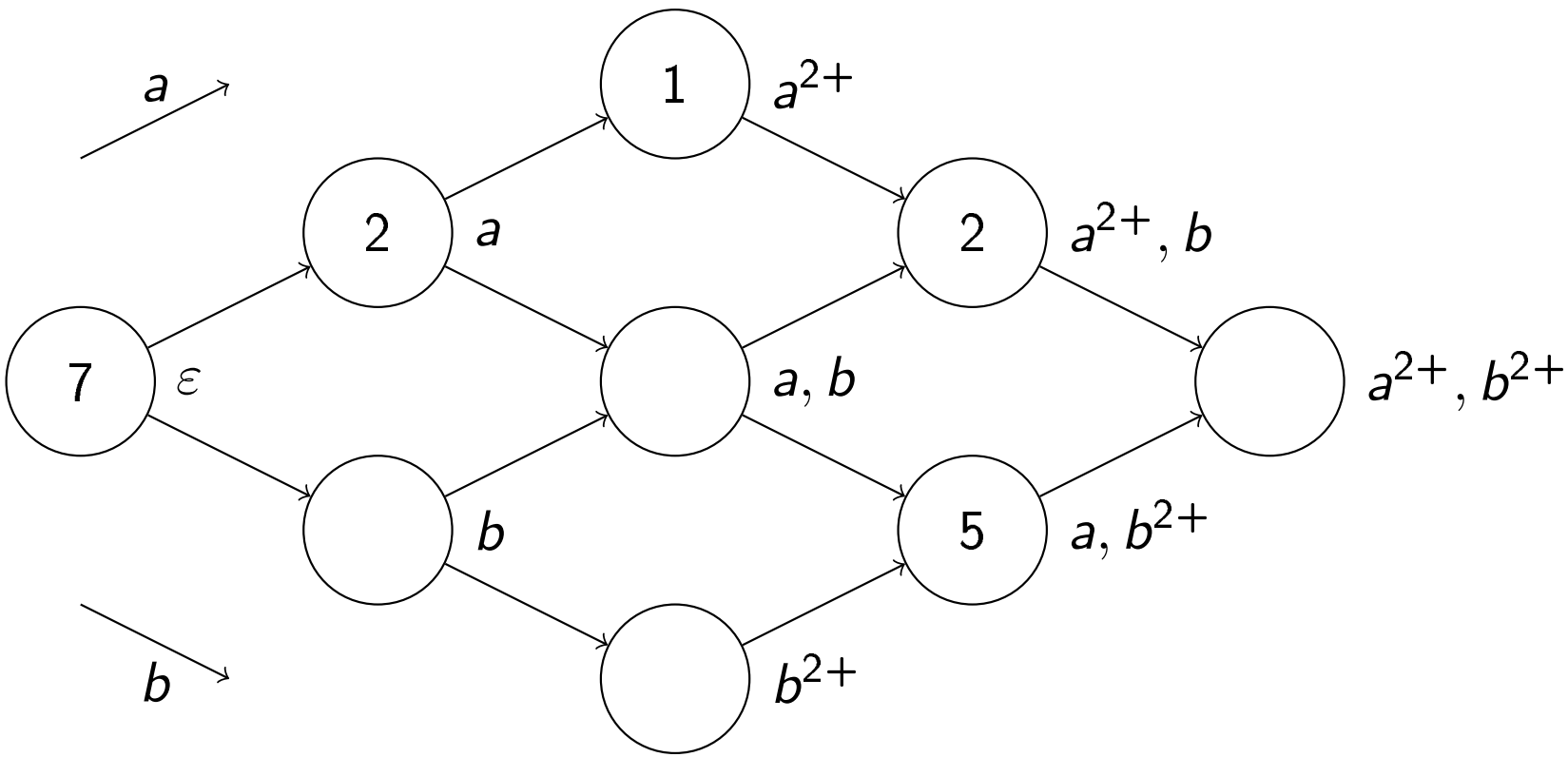
Parameterized Vector Games

Arena for $A_{sys} = \{a\}, A_{env} = \{b\}, \mathrm{B} = 2$: local states



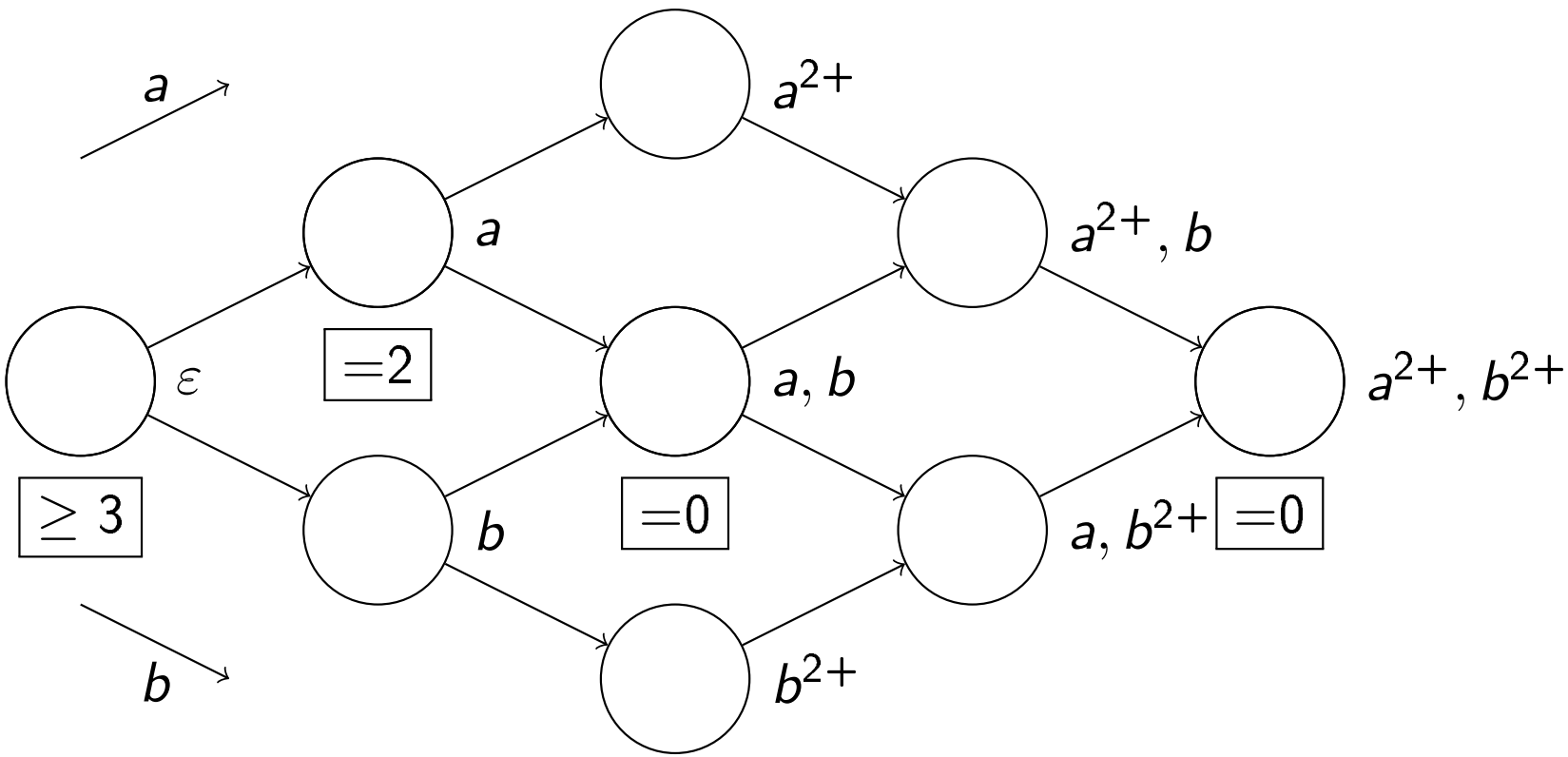
Parameterized Vector Games

Configuration c maps local states to number of tokens (default: 0)



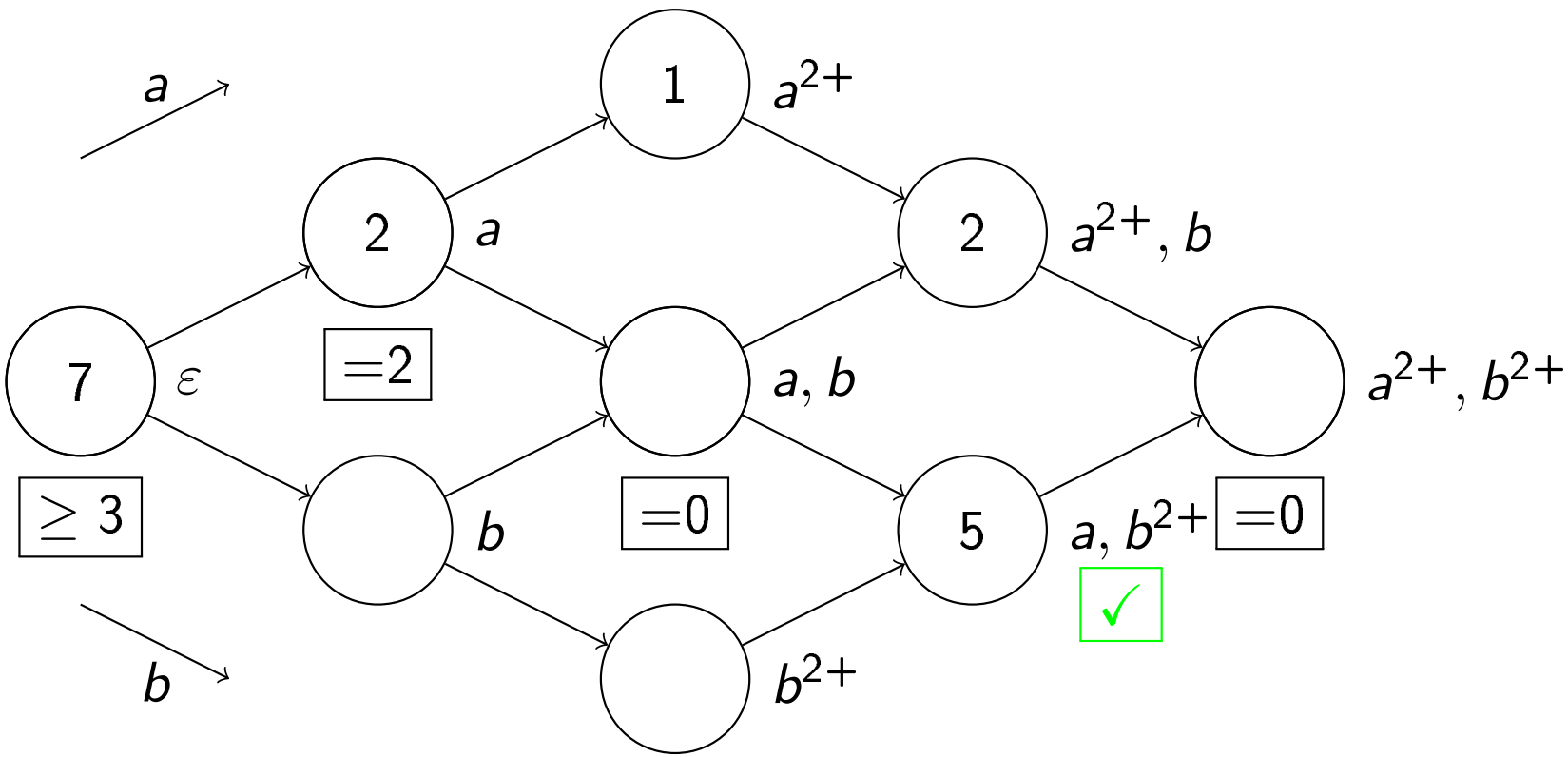
Parameterized Vector Games

Goal g = set of constraints for local states (default: ≥ 0)



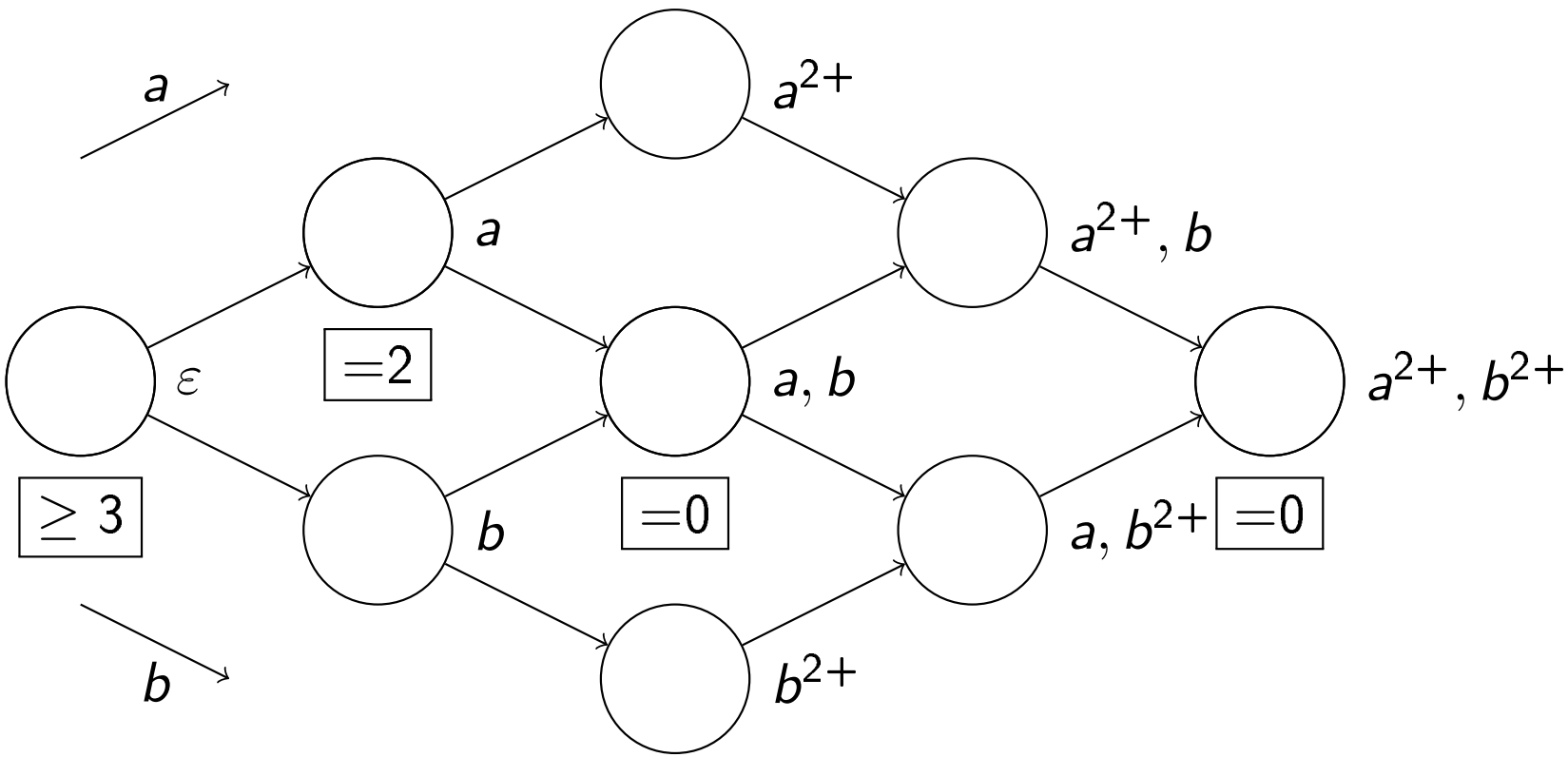
Parameterized Vector Games

Goal g = set of constraints for local states (default: ≥ 0)



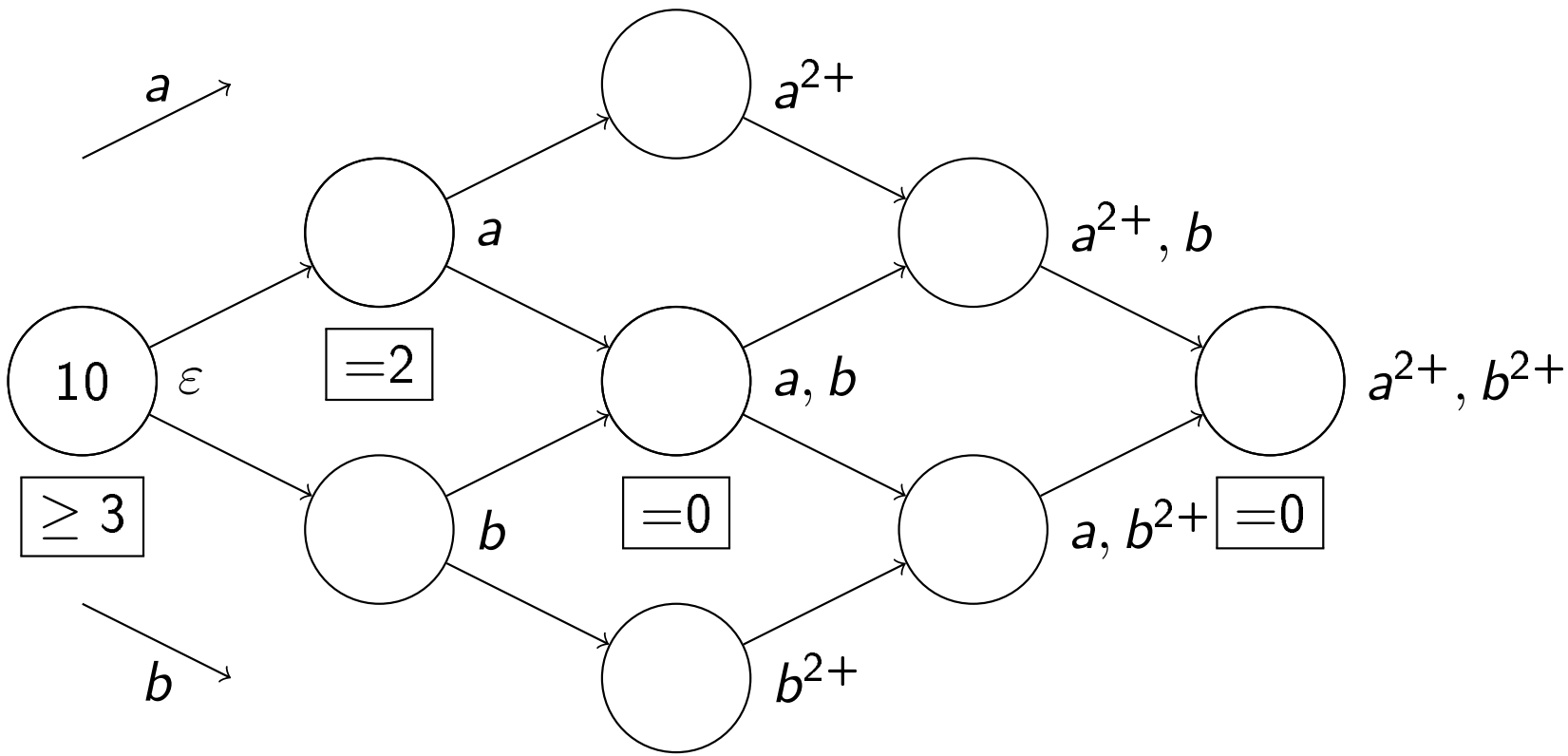
Parameterized Vector Games

Goal g = set of constraints for local states (default: ≥ 0)
Acceptance condition $\mathfrak{F}$ = disjunction of goals



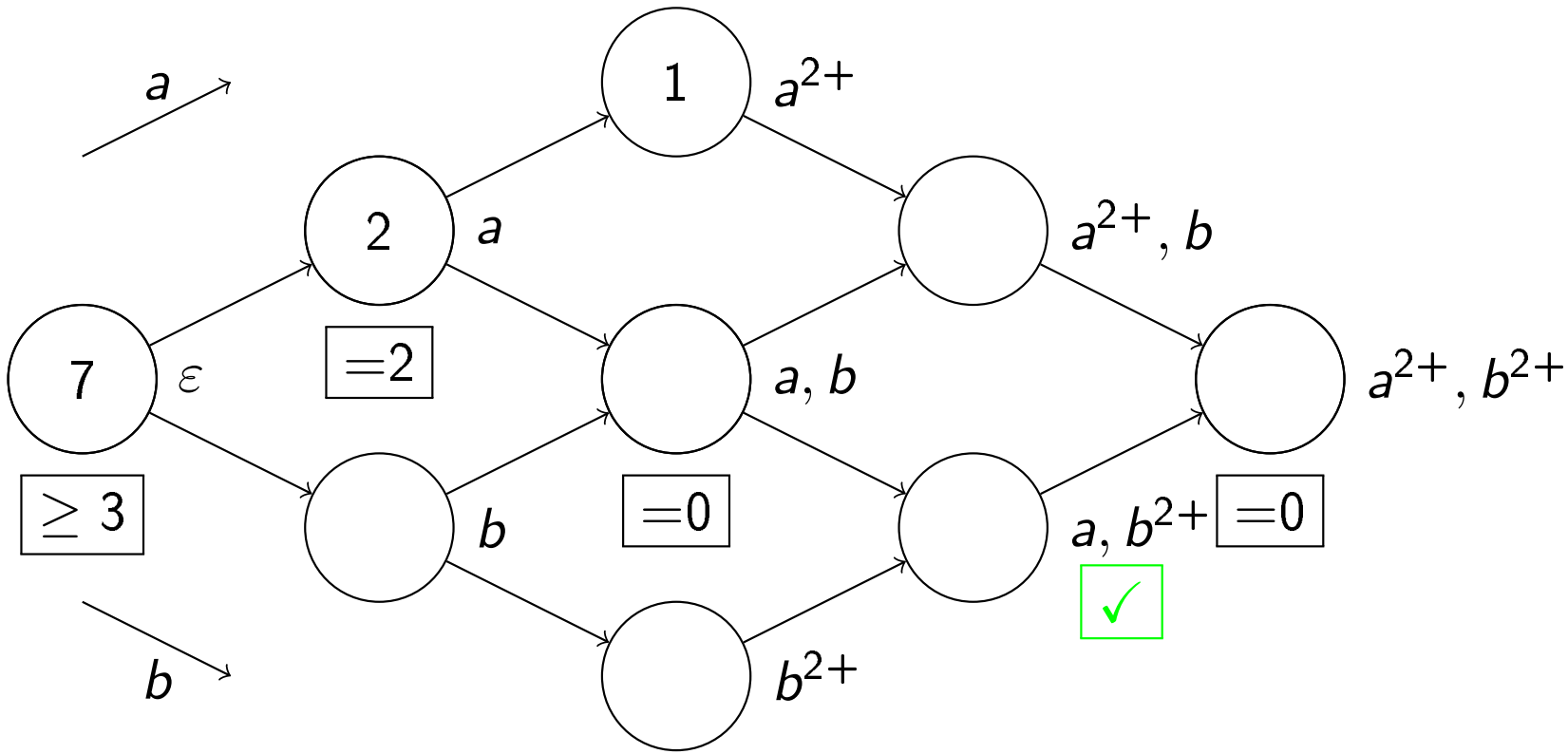
Parameterized Vector Games

Play on $\mathcal{G}$: System's turn



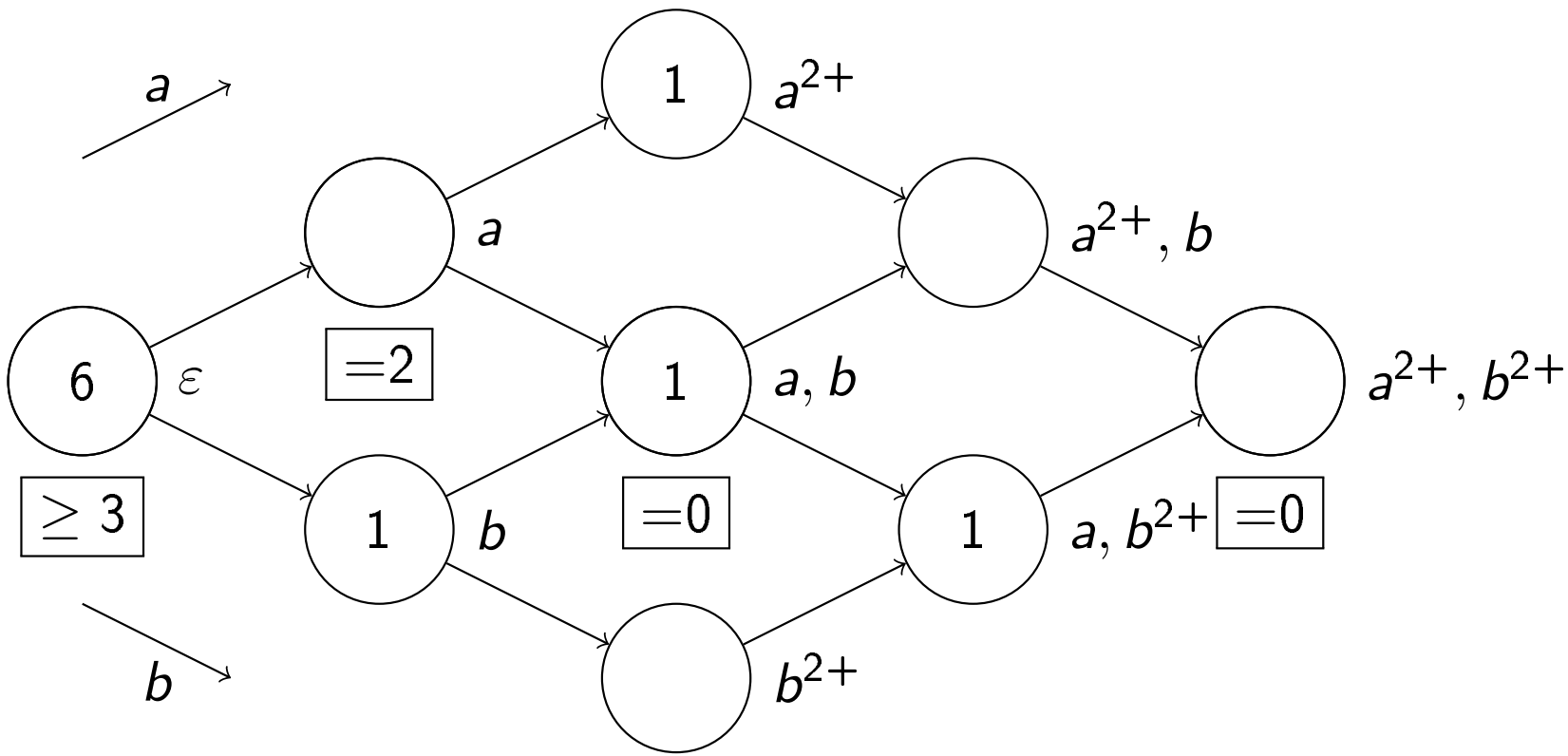
Parameterized Vector Games

Play on $\mathcal{G}$: Environment's turn



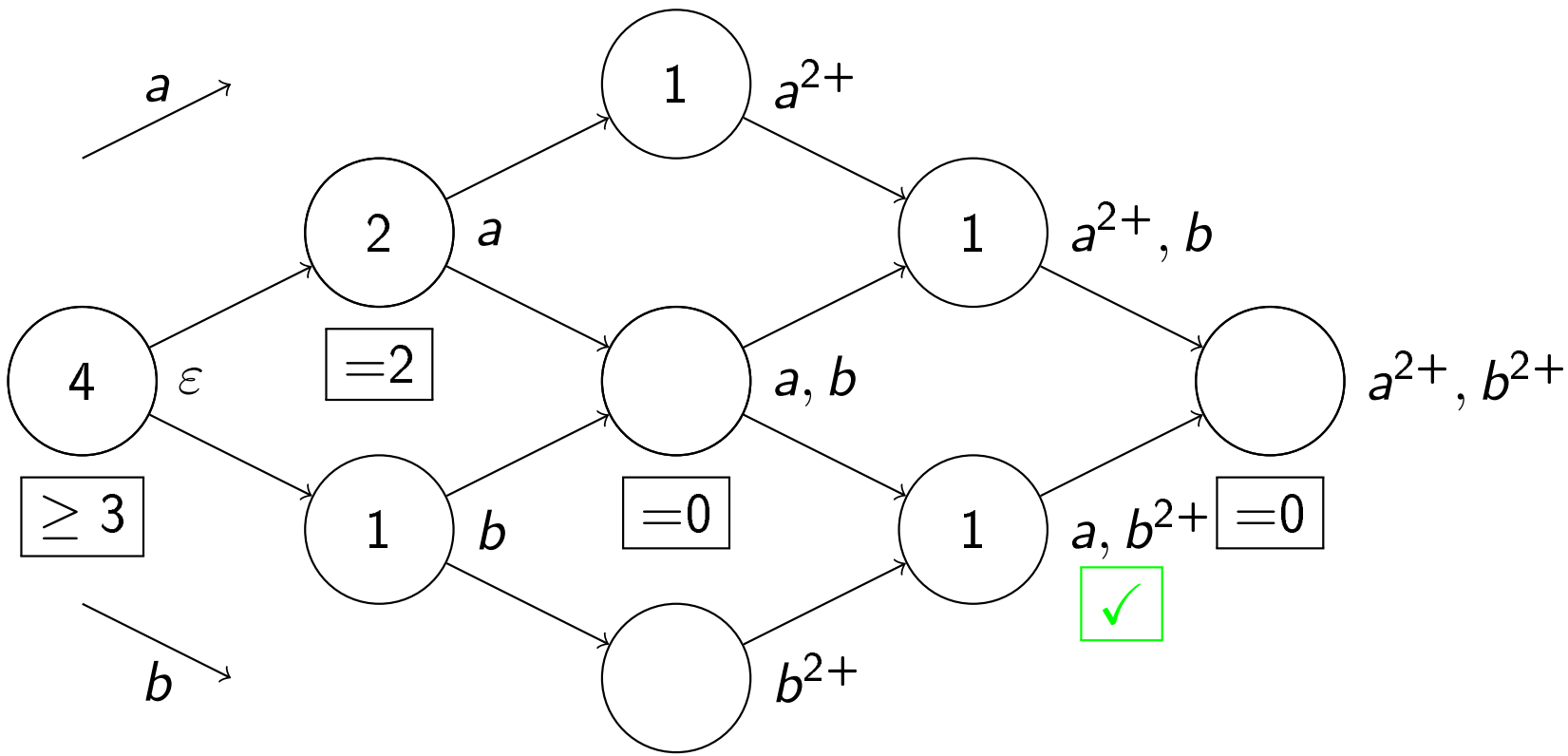
Parameterized Vector Games

Play on $\mathcal{G}$: System's turn



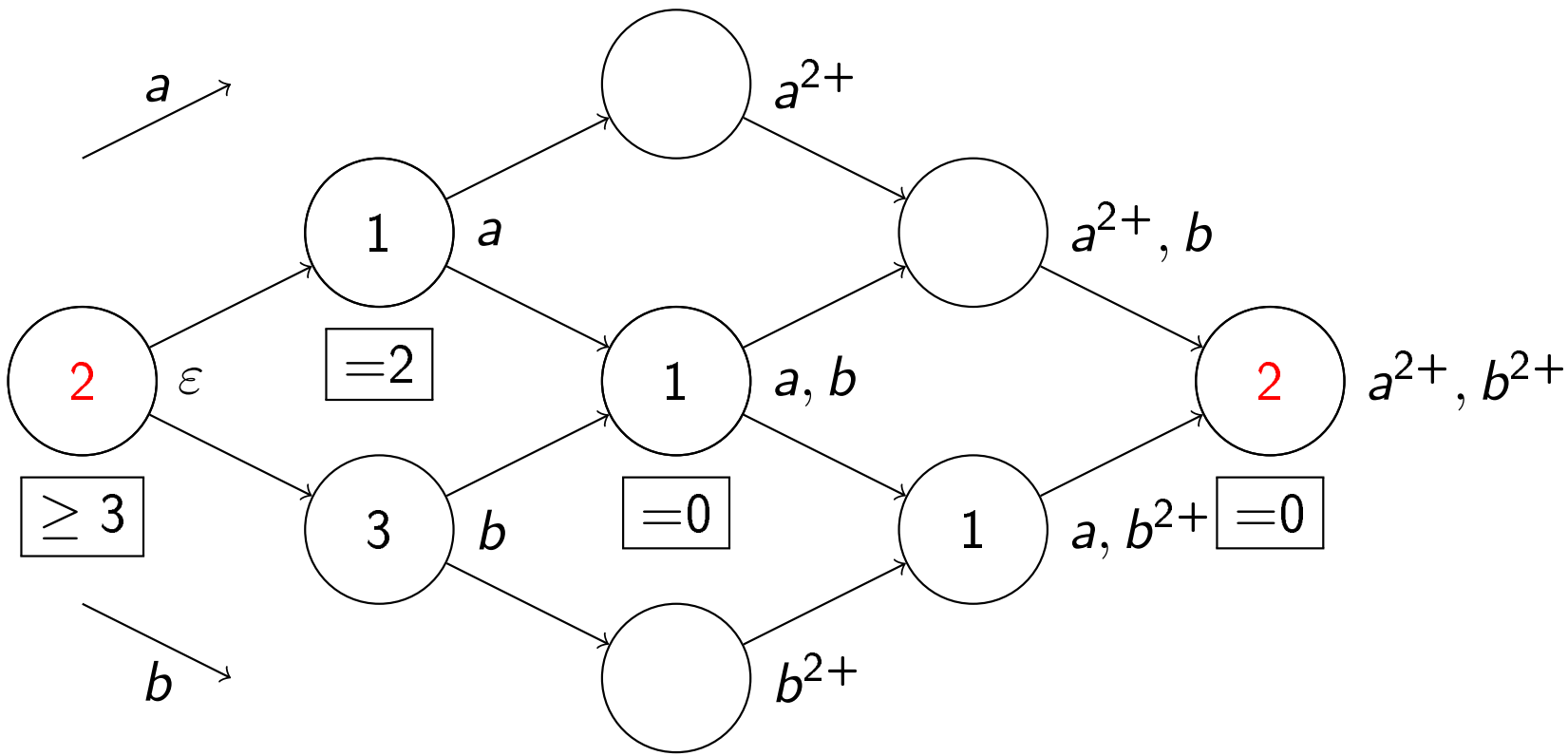
Parameterized Vector Games

Play on $\mathcal{G}$: Environment's turn



Parameterized Vector Games

Play on $\mathcal{G}$: System's turn



Equivalence with FO[∼]

- Asynchronous → turn-based game

No way to specify an order with FO[∼]

Equivalence with FO[∼]

- Asynchronous → turn-based game

No way to specify an order with FO[∼]

$(req, 1)(req, 2)(ack, 1)(req, 3)(ack, 2)(ack, 3)$

Equivalence with FO[∼]

- Asynchronous → turn-based game

No way to specify an order with FO[∼]

$$\begin{aligned} & (req,1)(req,2)(ack,1)(req,3)(ack,2)(ack,3) \\ \equiv\ & (req,1)(ack,1)(req,2)(ack,2)(req,3)(ack,3) \end{aligned}$$

Equivalence with FO[∼]

- Asynchronous → turn-based game

No way to specify an order with FO[∼]

$$(req, 1)(req, 2)(ack, 1)(req, 3)(ack, 2)(ack, 3)$$
$$\equiv (req, 1)(ack, 1)(req, 2)(ack, 2)(req, 3)(ack, 3)$$

- Normal form → Acceptance condition

There is a bound $\mathrm{B} \in \mathbb{N}$ s.t. φ is equivalent to a disjunction of conjunctions of formulas of the form

$$\exists^{\bowtie m} y.(\theta(y) \wedge \psi_{\mathrm{B},\ell}(y))$$

$\equiv$ "There are $\bowtie m$ processes of type θ with local state ℓ."

Results [FoSSaCS 20]

Undecidability

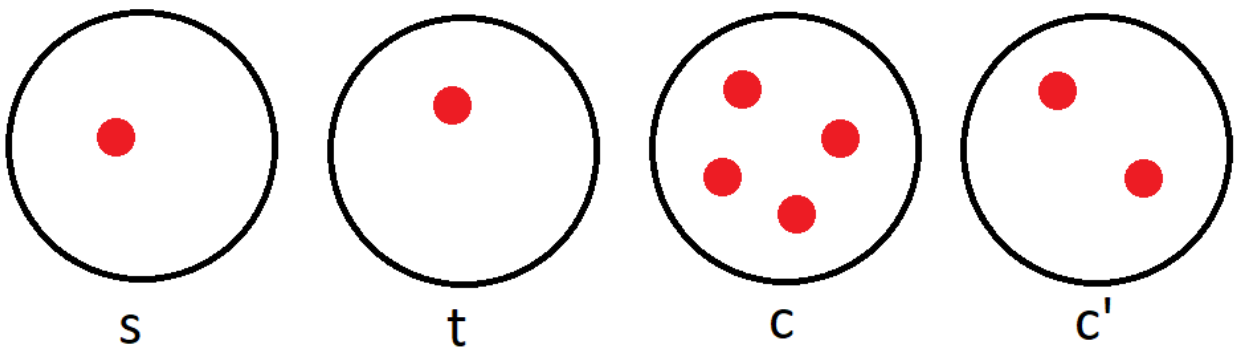
SYNTH($\mathrm{FO}[\sim], (\{0\}, \{0\}, \mathbb{N})$) is undecidable

Results [FoSSaCS 20]

Undecidability

SYNTH($\mathrm{FO}[\sim], (\{0\}, \{0\}, \mathbb{N})$) is undecidable

- Proof idea: encoding 2CM configuration



$$(s, c, c') \xrightarrow{t} \dots$$

Results [FoSSaCS 20]

Positive result

SYNTH($\mathrm{FO}[\sim], (\mathbb{N}, \{k_{env}\}, \{k_{se}\})$) is decidable

Results [FoSSaCS 20]

Positive result

SYNTH($\mathrm{FO}[\sim], (\mathbb{N}, \{k_{env}\}, \{k_{se}\})$) is decidable

Cutoff

$\mathrm{k} = (k_{sys}, k_{env}, k_{se})$ is a cutoff wrt $(\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se})$ for φ if either:

- for all $\mathrm{k}' \geq \mathrm{k}$, $\mathrm{k}' \in \mathit{Win}(\varphi)$
- for all $\mathrm{k}' \geq \mathrm{k}$, $\mathrm{k}' \notin \mathit{Win}(\varphi)$

Results [FoSSaCS 20]

Positive result

SYNTH($\mathrm{FO}[\sim], (\mathbb{N}, \{k_{env}\}, \{k_{se}\})$) is decidable

Cutoff

$\mathrm{k} = (k_{sys}, k_{env}, k_{se})$ is a cutoff wrt $(\mathcal{N}_{sys}, \mathcal{N}_{env}, \mathcal{N}_{se})$ for φ if either:

- for all $\mathrm{k}' \geq \mathrm{k}$, $\mathrm{k}' \in \mathit{Win}(\varphi)$
- for all $\mathrm{k}' \geq \mathrm{k}$, $\mathrm{k}' \notin \mathit{Win}(\varphi)$

▶ Existence of cutoff ⇒ Synthesis decidable!

Conclusion

Summary of main results:

Control of dynamic parameterized systems

Round-bounded control of parameterized pushdown systems is decidable

Conclusion

Summary of main results:

Control of dynamic parameterized systems

Round-bounded control of parameterized pushdown systems is decidable

Synthesis of fixed parameterized systems

- Synthesis of FO^2 and $\mathrm{FO}[\sim]$ undecidable for mixed processes
- Synthesis of $\mathrm{FO}[\sim]$ decidable when Environment affects a bounded number of processes

Perspectives

Future works on control

- Other under-approximations (context, ...)
- Specifications given not as a Reach objective

Perspectives

Future works on control

- Other under-approximations (context, ...)
- Specifications given not as a Reach objective

Future works on synthesis

- Cases left open ($\mathrm{FO}^2[\sim]$, ...)
- Less centralized synthesis

Perspectives

Future works on control

- Other under-approximations (context, ...)
- Specifications given not as a Reach objective

Future works on synthesis

- Cases left open ($\mathrm{FO}^2[\sim]$, ...)
- Less centralized synthesis

Thank you for your attention!