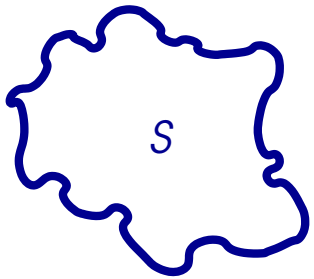
Consistency for Parametric Interval Markov Chains

Benoît Delahaye

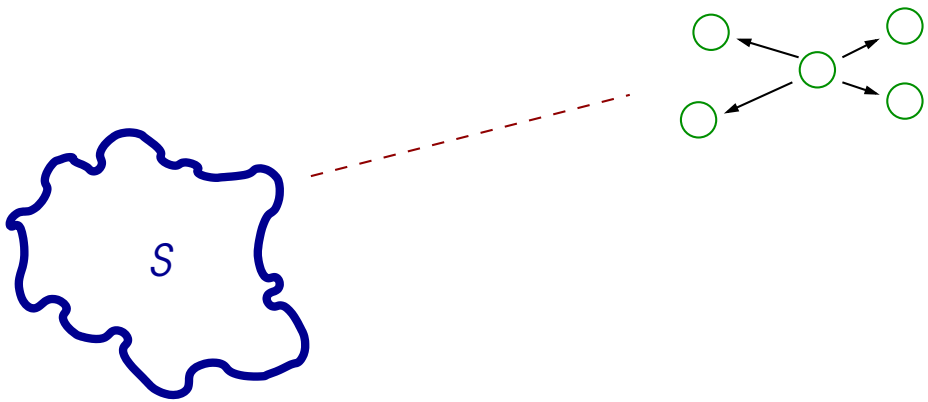
Université de Nantes

SynCoP 2015

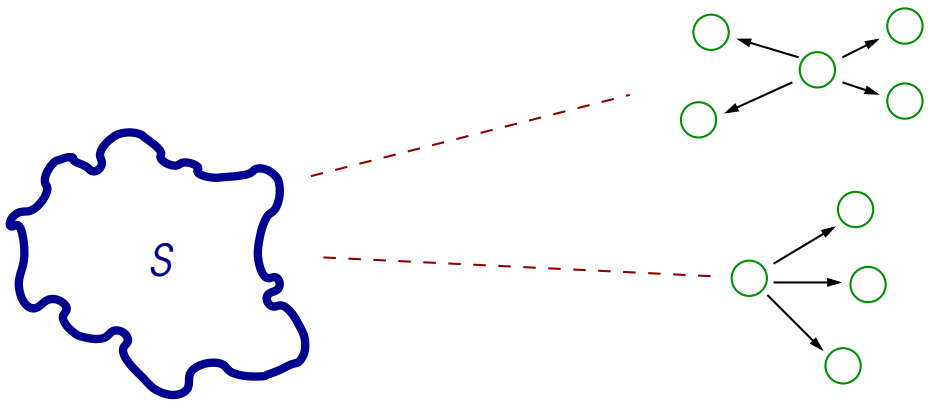
Context (1/2): Probabilistic Specifications



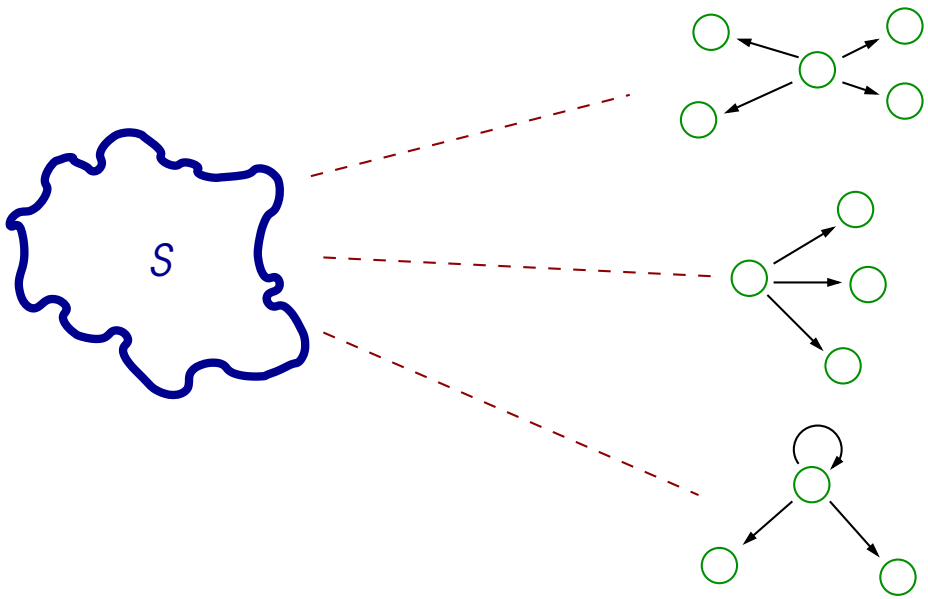
Context (1/2): Probabilistic Specifications



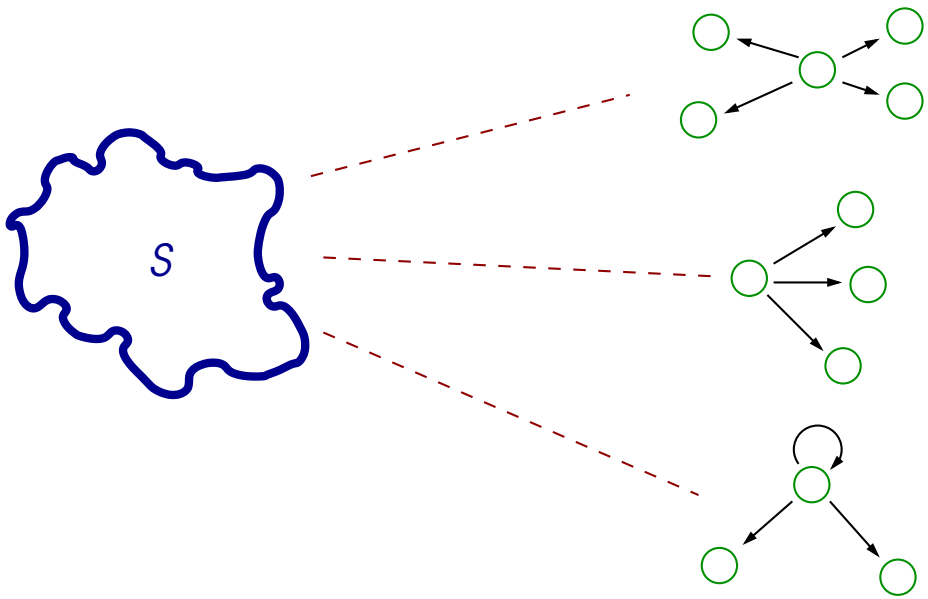
Context (1/2): Probabilistic Specifications

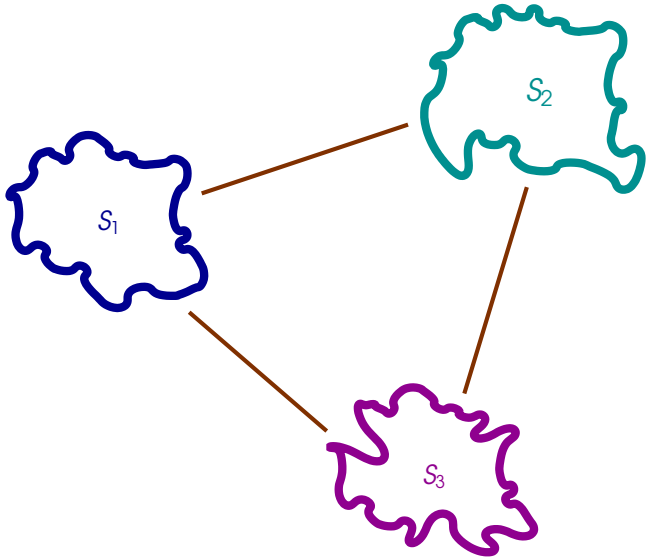


Context (1/2): Probabilistic Specifications

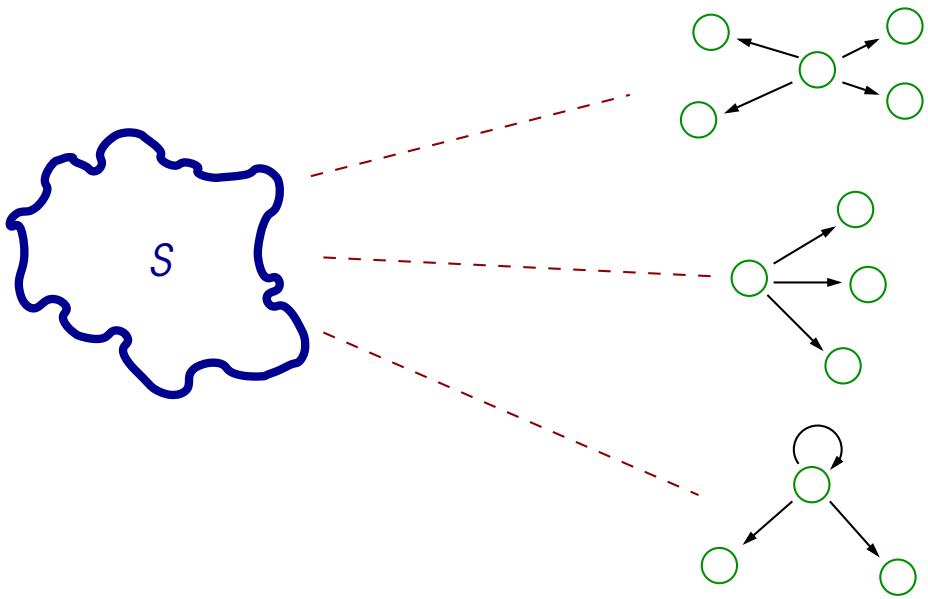


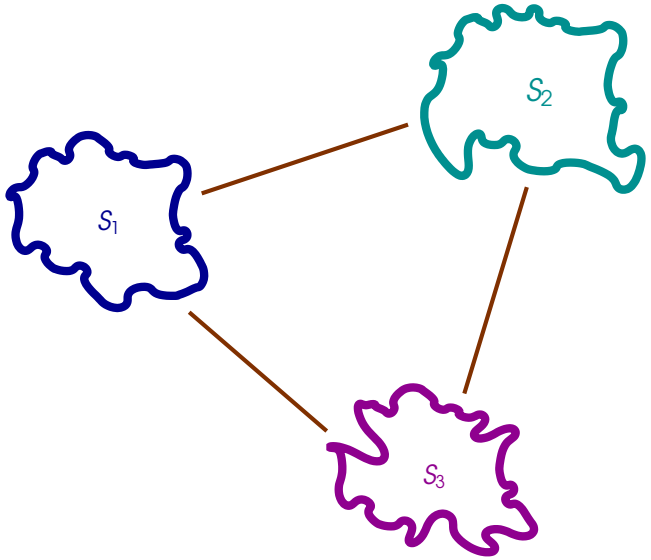
Context (1/2): Probabilistic Specifications





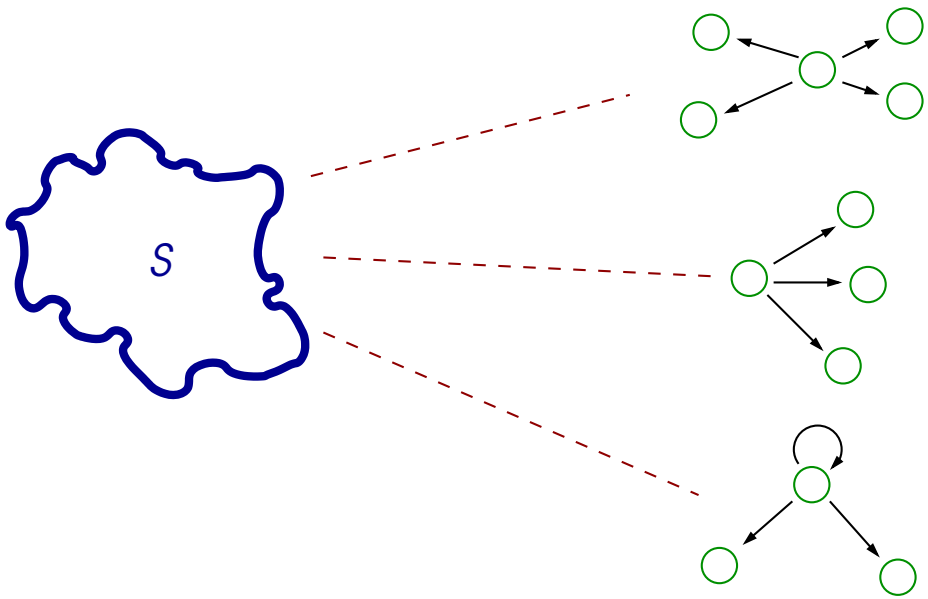
Context (1/2): Probabilistic Specifications

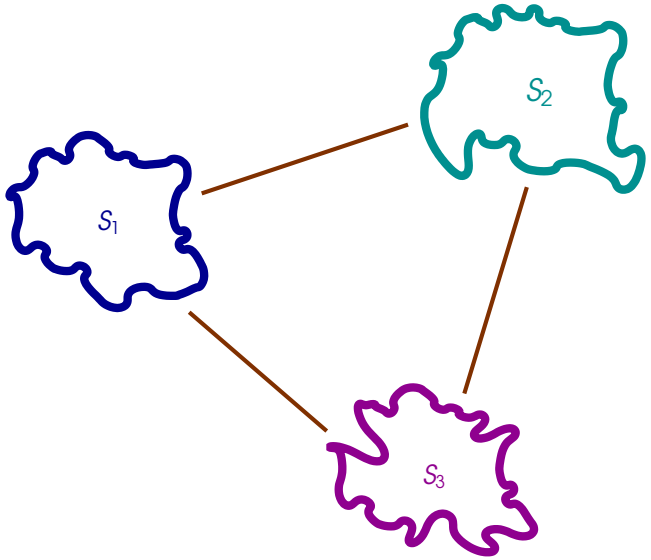




Compositional Design

Context (1/2): Probabilistic Specifications

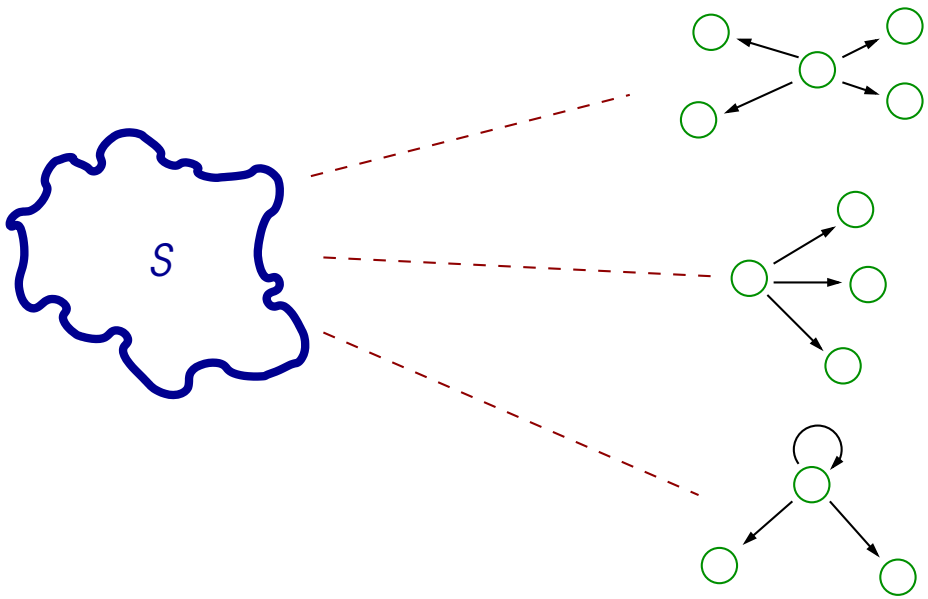


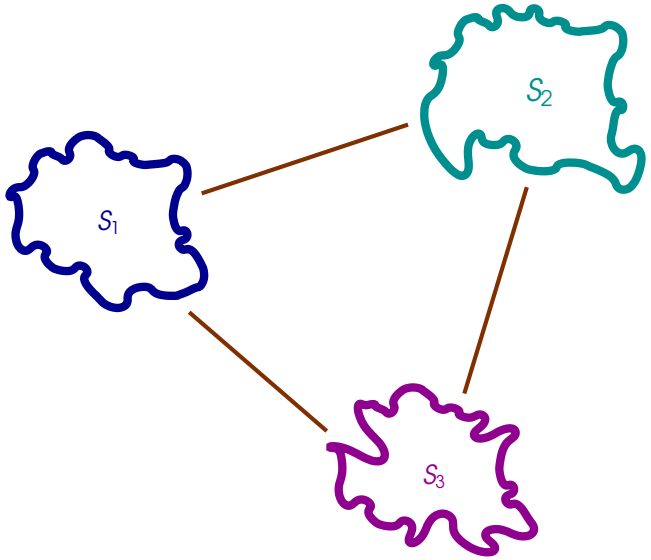


Compositional Design

Generic Results

Context (1/2): Probabilistic Specifications



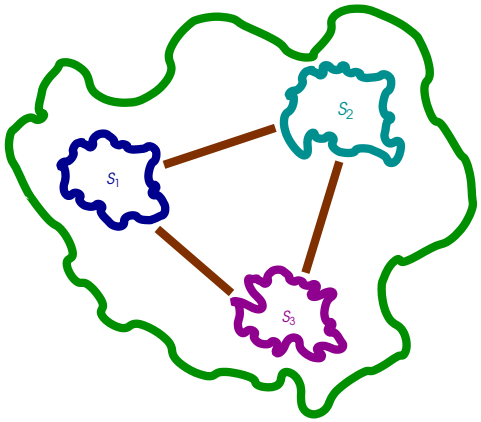


Compositional Design

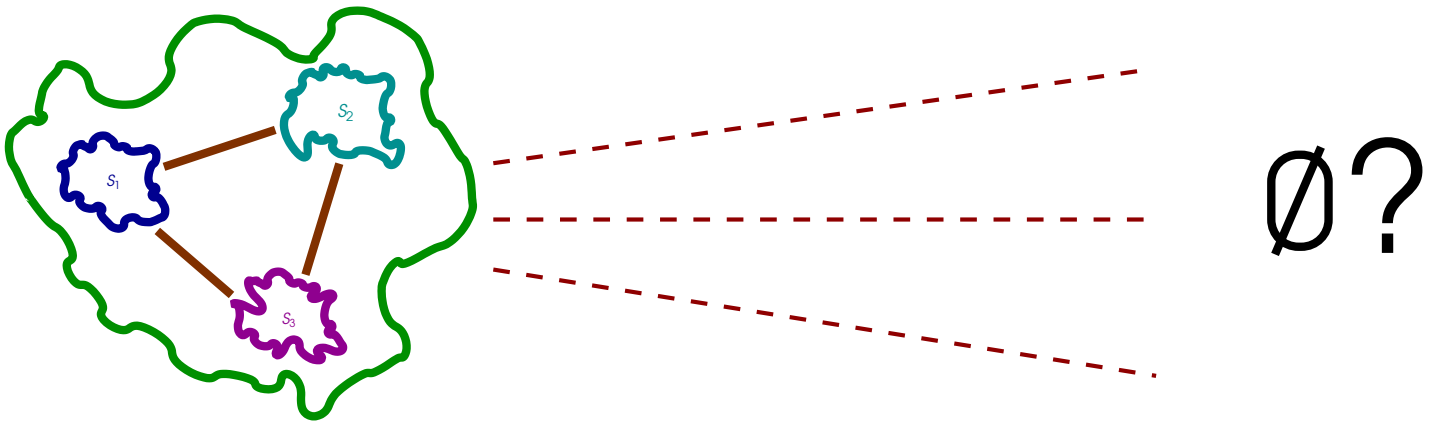
Generic Results

Independent Reasoning

Context (2/2): Consistency



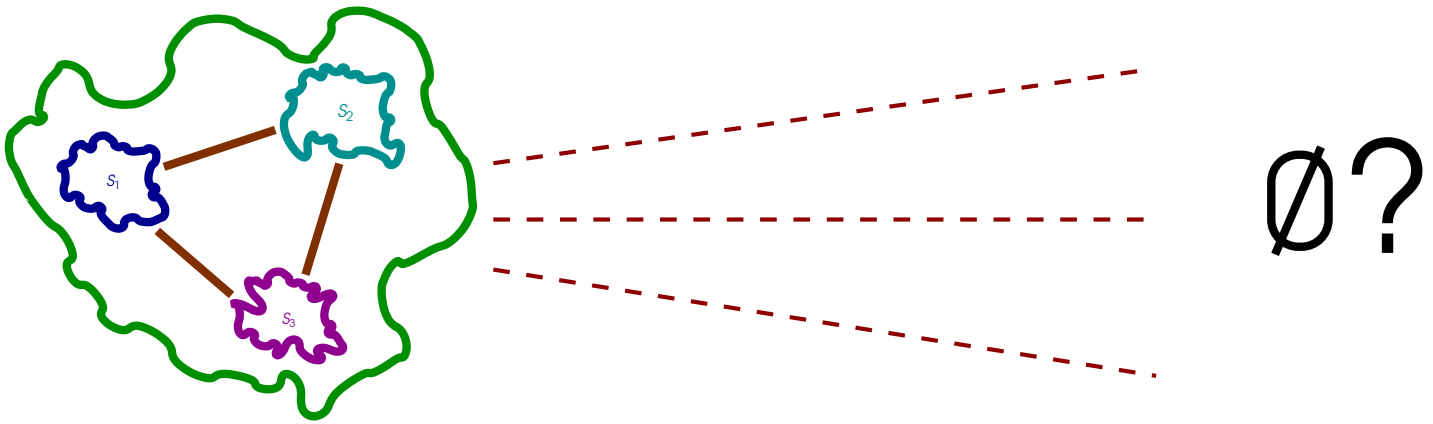
Context (2/2): Consistency



Consistency

Given a specification S, decide whether it admits at least one implementation.

Context (2/2): Consistency

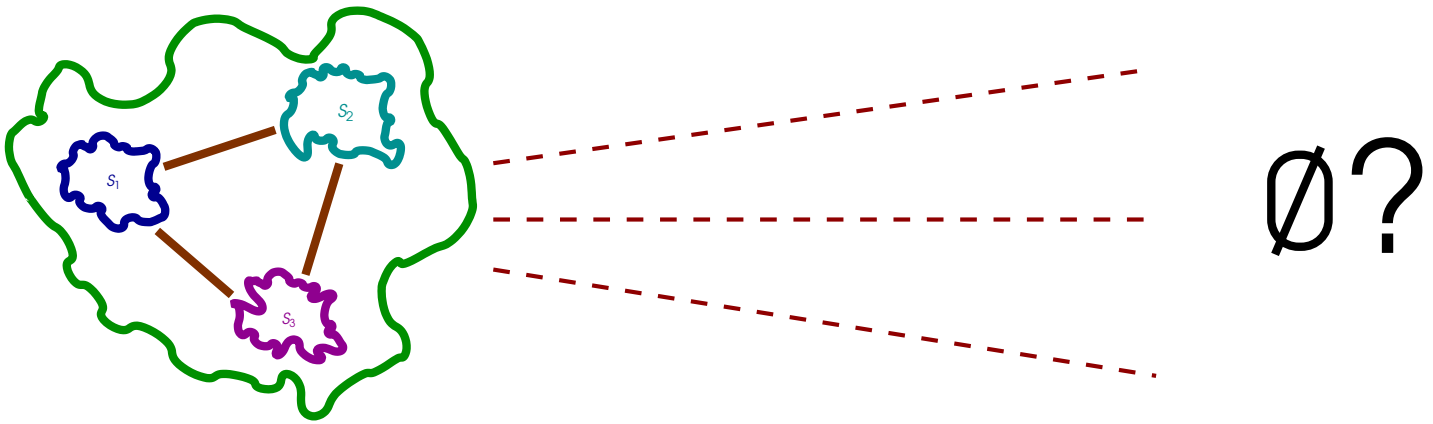


Consistency

Given a specification S, decide whether it admits at least one implementation.

Other questions of interest

Context (2/2): Consistency



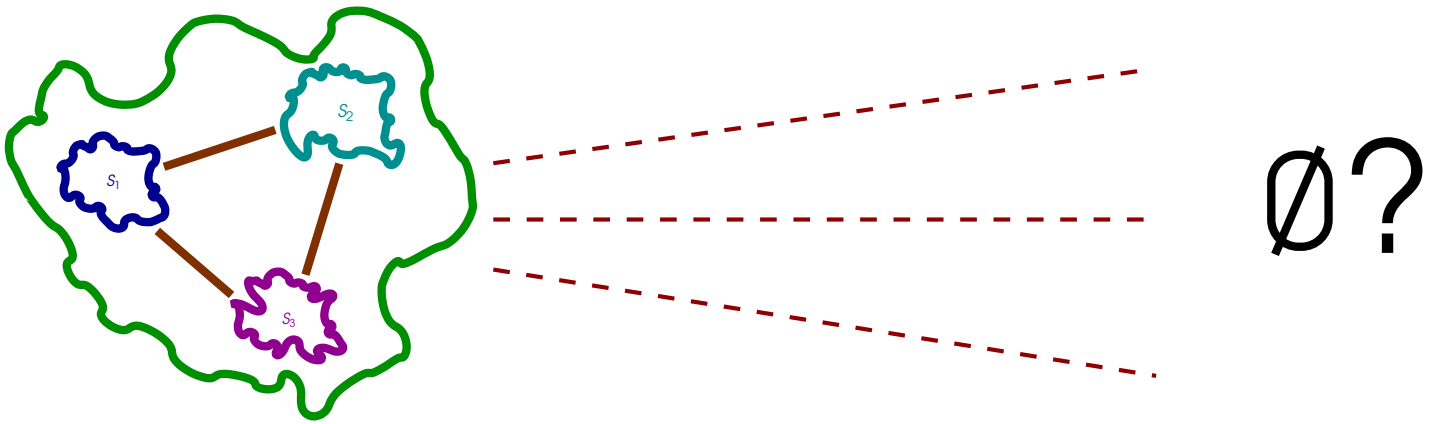
Consistency

Given a specification S, decide whether it admits at least one implementation.

Other questions of interest

- Common implementation

Context (2/2): Consistency



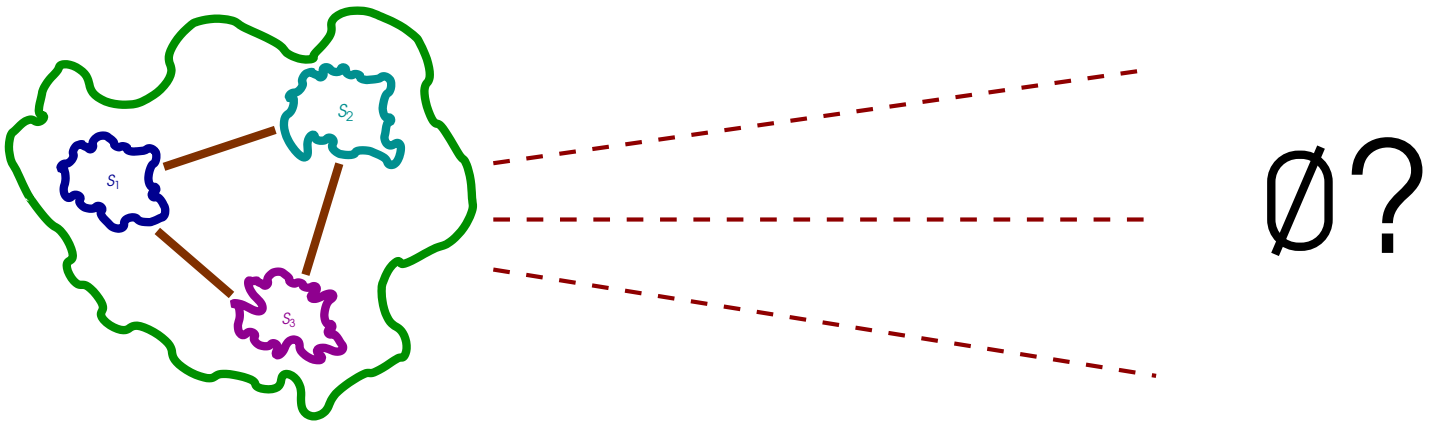
Consistency

Given a specification S, decide whether it admits at least one implementation.

Other questions of interest

- Common implementation
- Refinement

Context (2/2): Consistency



Consistency

Given a specification S, decide whether it admits at least one implementation.

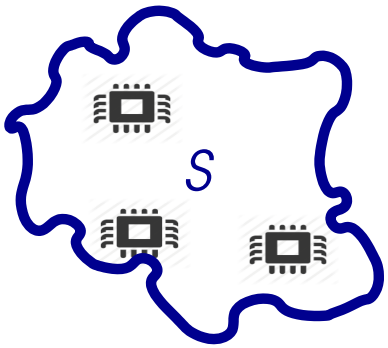
Other questions of interest

- Common implementation
- Refinement
- Composition
  Logical / Structural

Motivation: Why put parameters on probabilities?

Component failure rate

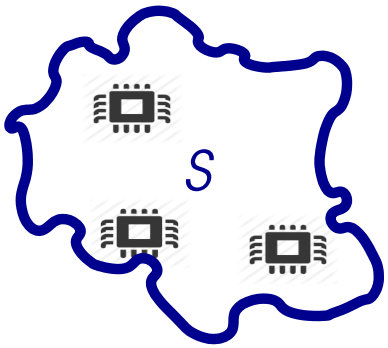
Can be chosen, with adequate cost



Motivation: Why put parameters on probabilities?

Component failure rate

Can be chosen, with adequate cost

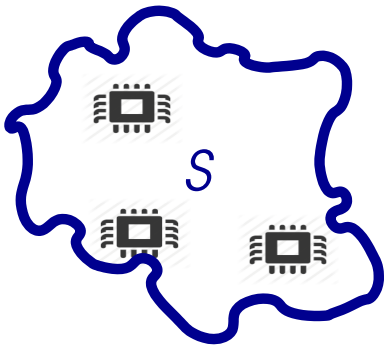


- Decide whether there exists a value for the failure rate for which the system is feasible (i.e. **consistent**)

Motivation: Why put parameters on probabilities?

Component failure rate

Can be chosen, with adequate cost

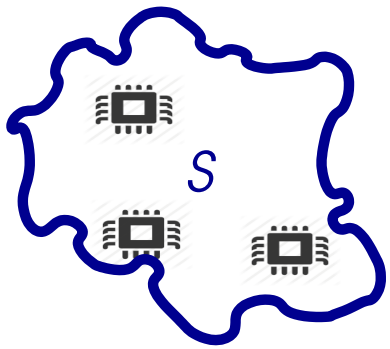


- Decide whether there exists a value for the failure rate for which the system is feasible (i.e. **consistent**)
- Compute the best compromise between failure rate and cost

Motivation: Why put parameters on probabilities?

Component failure rate

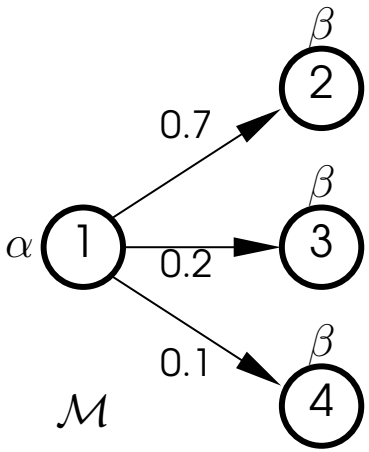
Can be chosen, with adequate cost $\Rightarrow$ **parameter**

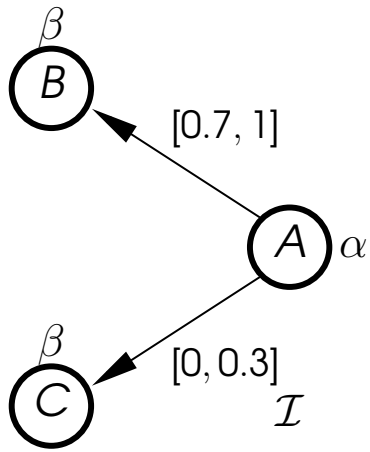


- Decide whether there exists a value for the failure rate for which the system is feasible (i.e. **consistent**)
- Compute the best compromise between failure rate and cost

Outline

(Interval) Markov Chains

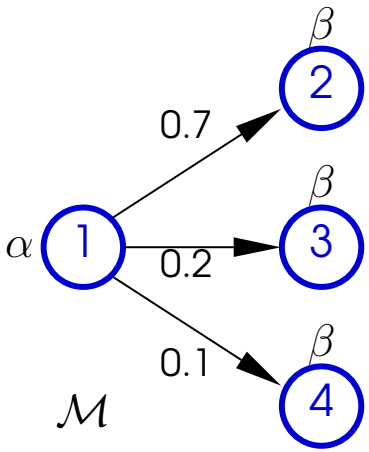


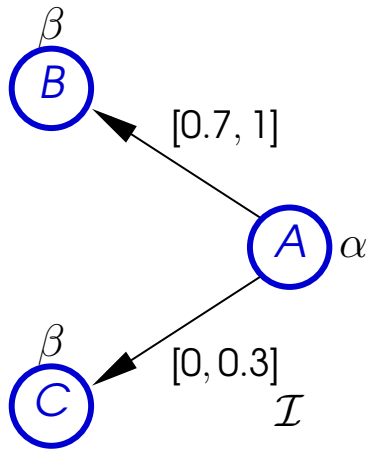


$$\mathcal{M} = (S, s_0, M, A, V)$$

$$\mathcal{I} = (S, s_0, \varphi, A, V)$$

(Interval) Markov Chains

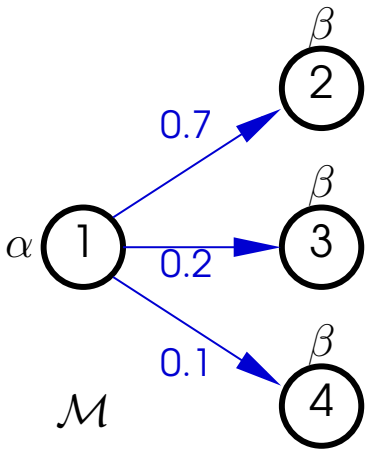


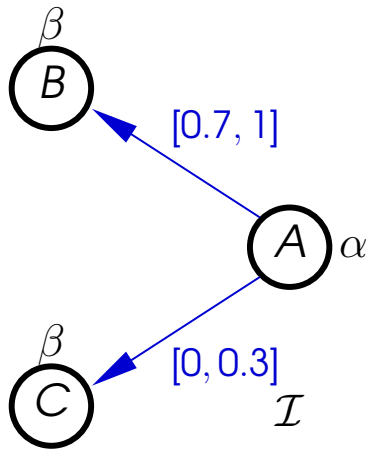


$$\mathcal{M} = (S, s_0, M, A, V)$$

$$\mathcal{I} = (S, s_0, \varphi, A, V)$$

(Interval) Markov Chains

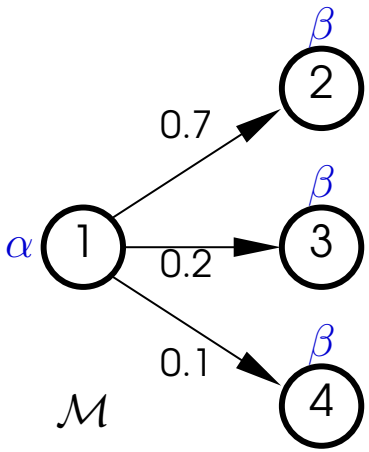


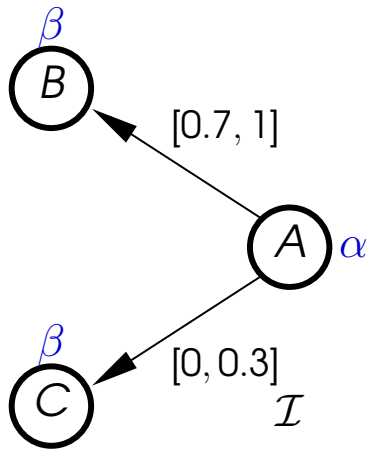


$$\mathcal{M} = (S, s_0, M, A, V)$$

$$\mathcal{I} = (S, s_0, \varphi, A, V)$$

(Interval) Markov Chains

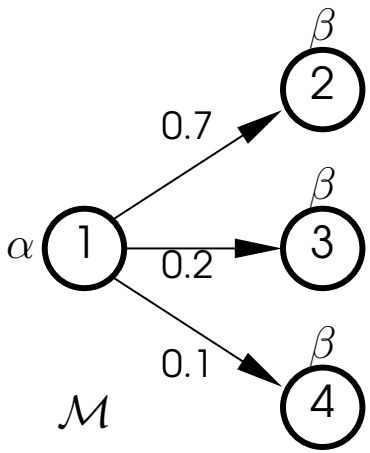


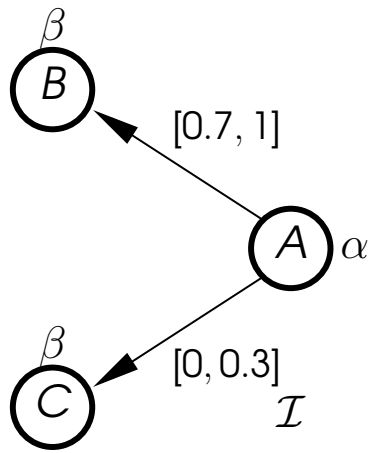


$$\mathcal{M} = (S, s_0, M, A, V)$$

$$\mathcal{I} = (S, s_0, \varphi, A, V)$$

Satisfaction

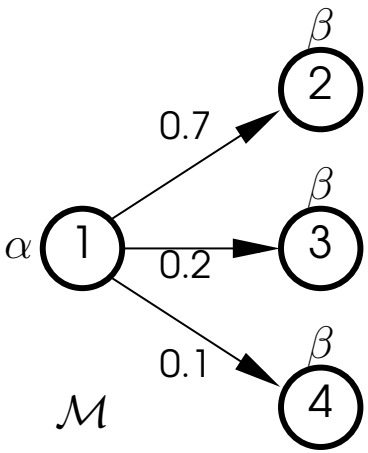


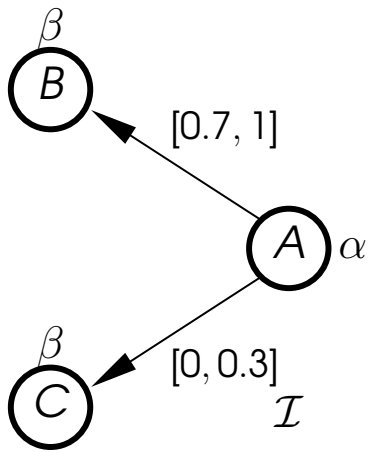


Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

Satisfaction



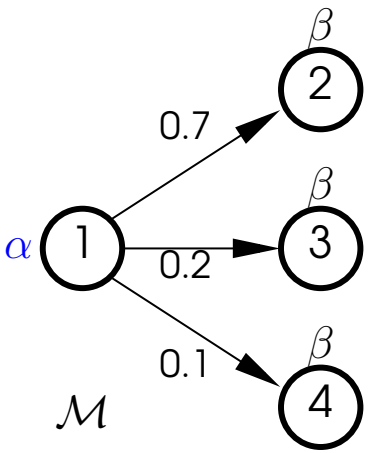


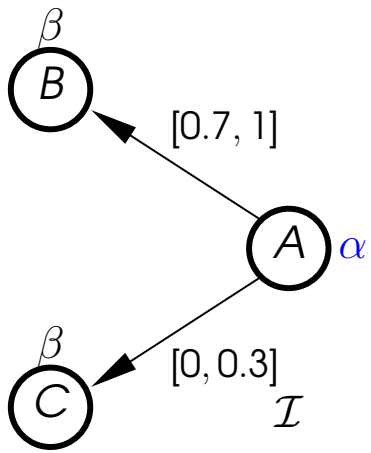
Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

- Valuations of t and s match

Satisfaction



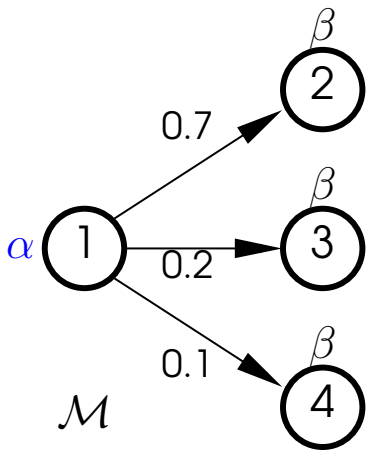


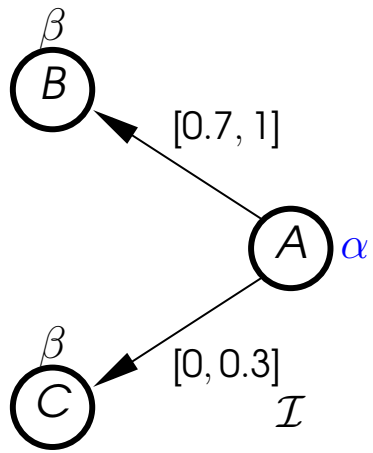
Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

- Valuations of t and s match

Satisfaction



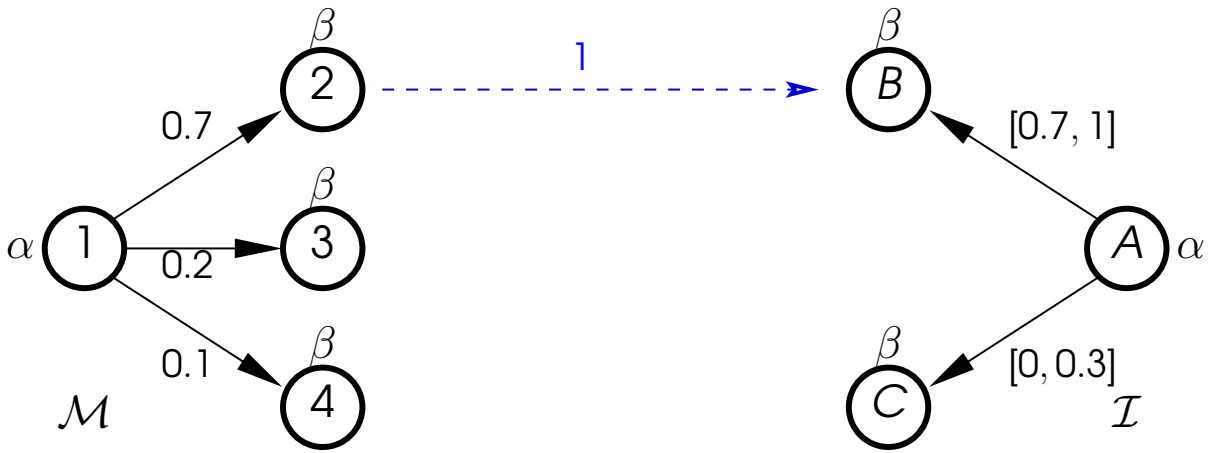


Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

- Valuations of t and s match
- Outgoing transitions of t must be **redistributed** to outgoing transitions of s

Satisfaction

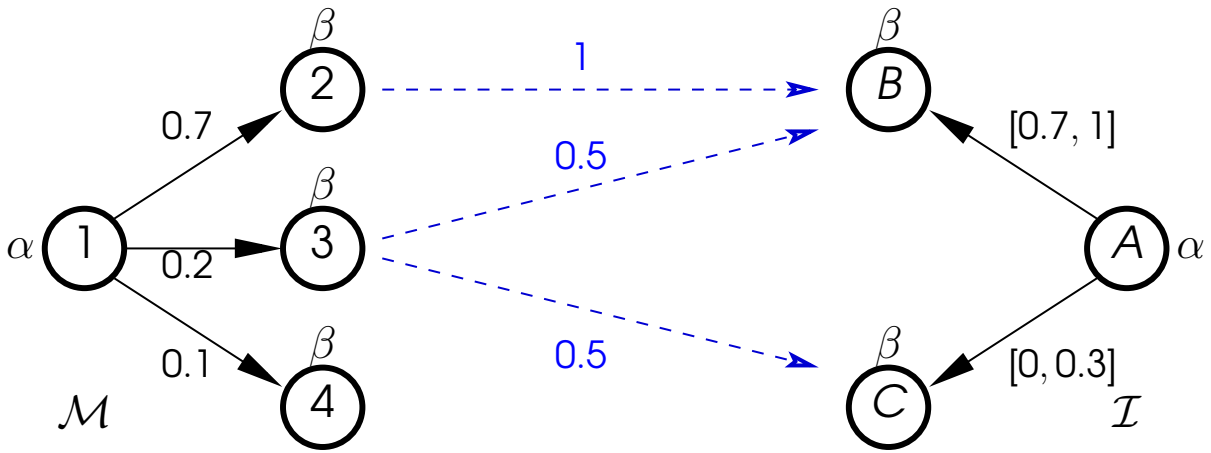


Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

- Valuations of t and s match
- Outgoing transitions of t must be **redistributed** to outgoing transitions of s

Satisfaction

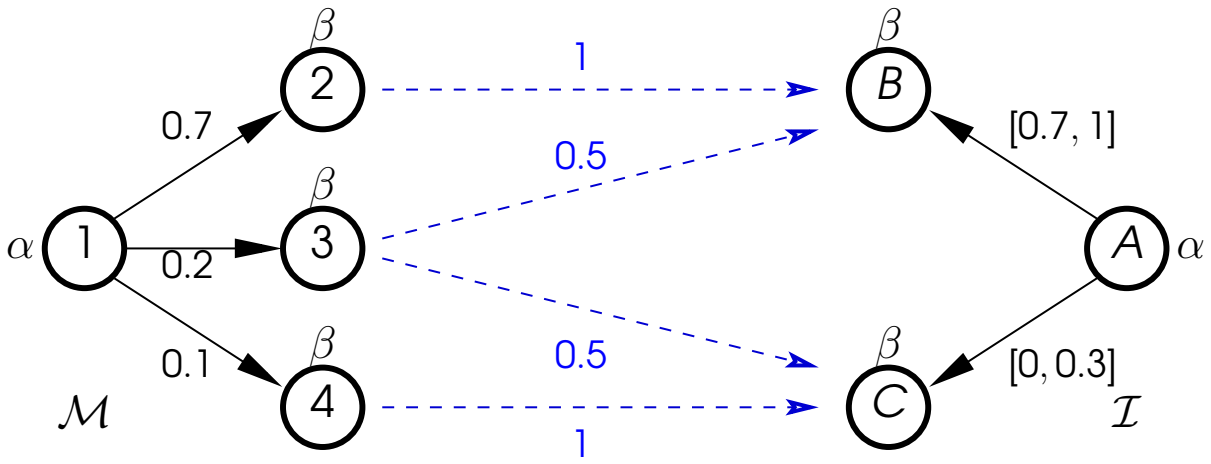


Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

- Valuations of t and s match
- Outgoing transitions of t must be **redistributed** to outgoing transitions of s

Satisfaction

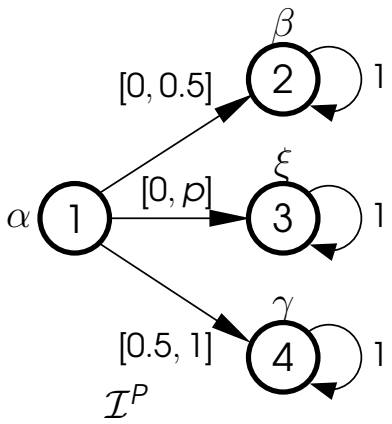


Satisfaction Relation $\mathcal{R}$

Whenever $(t, s) \in \mathcal{R}$, we have

- Valuations of t and s match
- Outgoing transitions of t must be **redistributed** to outgoing transitions of s

Parametric Interval Markov Chains

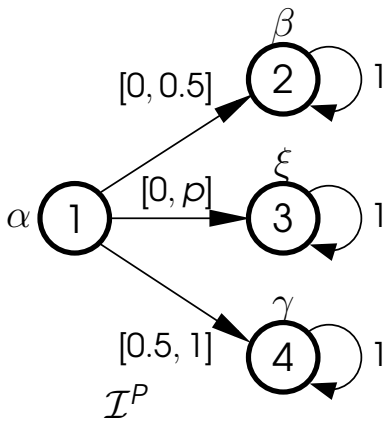


pIMCs

$$\mathcal{I}^P$$

Intervals can contain parameters from set P

Parametric Interval Markov Chains



pIMCs

$$\mathcal{I}^P$$

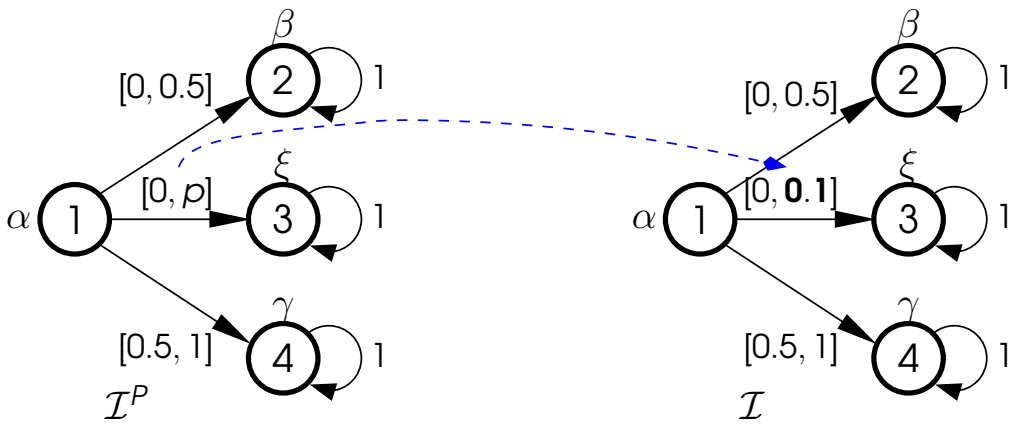
Intervals can contain parameters from set P

Instantiation

$$\mathcal{I} \vdash \mathcal{I}^P$$

Assign values to parameters

Parametric Interval Markov Chains



pIMCs

$$\mathcal{I}^P$$

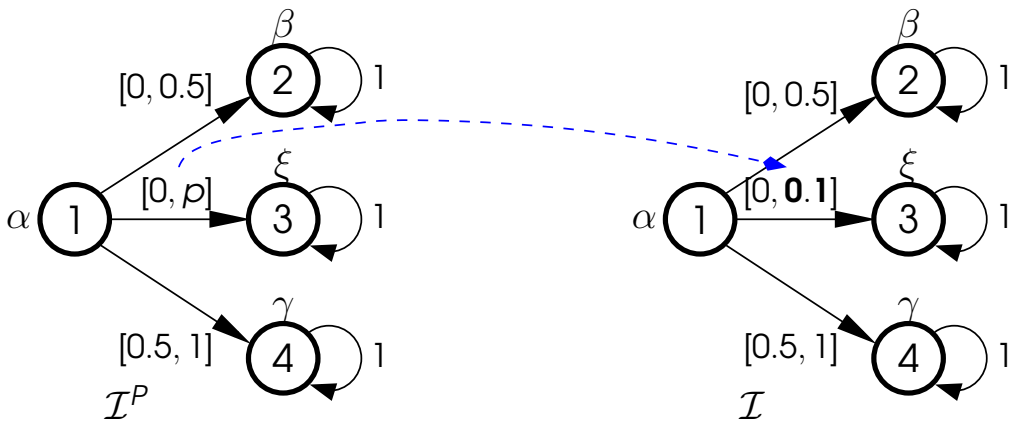
Intervals can contain parameters from set P

Instantiation

$$\mathcal{I} \vdash \mathcal{I}^P$$

Assign values to parameters

Parametric Interval Markov Chains



pIMCs

$\mathcal{I}^P$

Intervals can contain parameters from set P

Instantiation

$\mathcal{I} \vdash \mathcal{I}^P$

Assign values to parameters

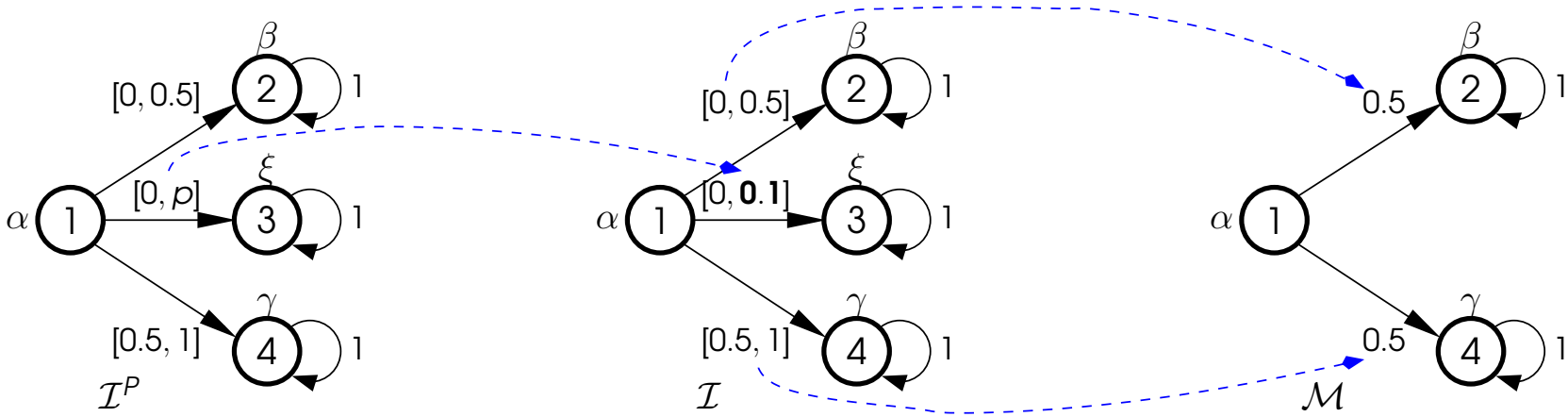
Satisfaction

$\mathcal{M} \models \mathcal{I}^P$

iff $\exists \mathcal{I}$ s.t.

$\mathcal{M} \models \mathcal{I} \vdash \mathcal{I}^P$

Parametric Interval Markov Chains



pIMCs

$\mathcal{I}^P$

Intervals can contain parameters from set P

Instantiation

$\mathcal{I} \vdash \mathcal{I}^P$

Assign values to parameters

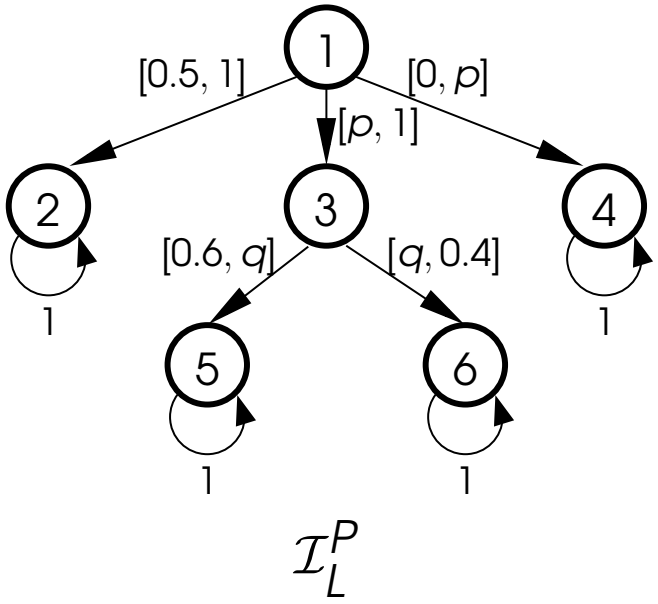
Satisfaction

$\mathcal{M} \models \mathcal{I}^P$

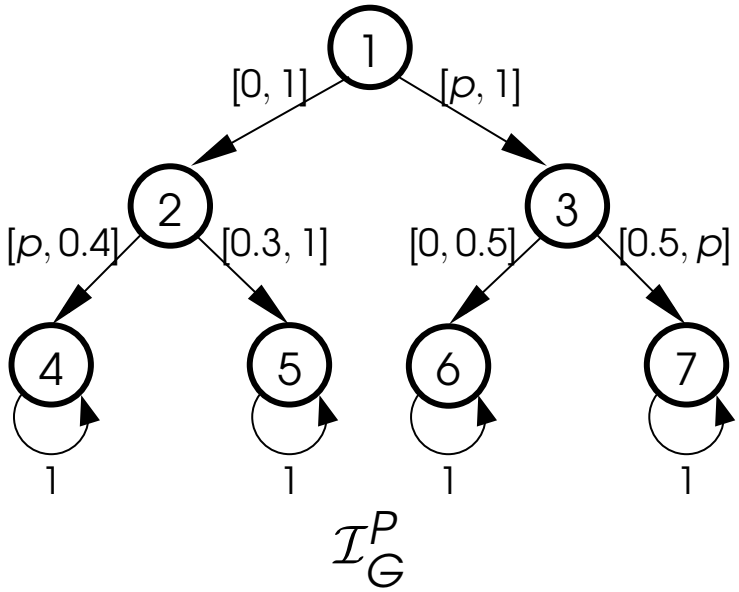
iff $\exists \mathcal{I}$ s.t.

$\mathcal{M} \models \mathcal{I} \vdash \mathcal{I}^P$

Local vs Global Parameters



$\mathcal{I}_L^P$



$\mathcal{I}_G^P$

Local Parameters

p is **local** if it appears in outgoing transitions of **at most 1 state**

Global Parameters

p is **global** if it appears in outgoing transitions of **at least 2 state**

Outline

Consistency of IMCs

IMC $\mathcal{I}$ is consistent iff there exists at least one MC $\mathcal{M}$ such that

$$\mathcal{M} \models \mathcal{I}$$

Consistency of IMCs

IMC $\mathcal{I}$ is consistent iff there exists at least one MC $\mathcal{M}$ such that

$$\mathcal{M} \models \mathcal{I}$$

- Decidable ([JLAP'12])

Consistency of IMCs

IMC $\mathcal{I}$ is consistent iff there exists at least one MC $\mathcal{M}$ such that

$$\mathcal{M} \models \mathcal{I}$$

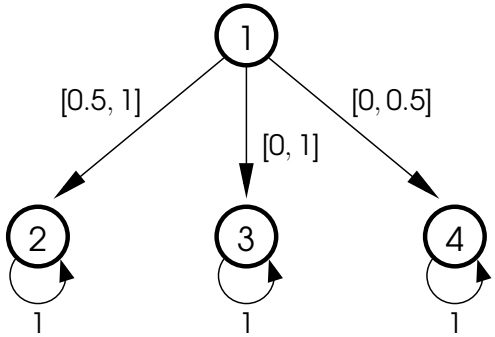
- Decidable ([JLAP'12])
- (Complex) Algorithm based on *Consistency Relations*

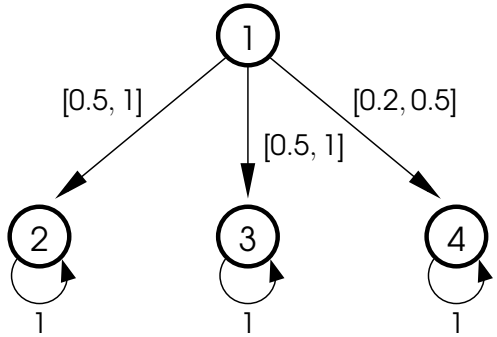
Local state-consistency

A state is **locally consistent** iff there exists a distribution ρ matching its outgoing intervals.

Local state-consistency

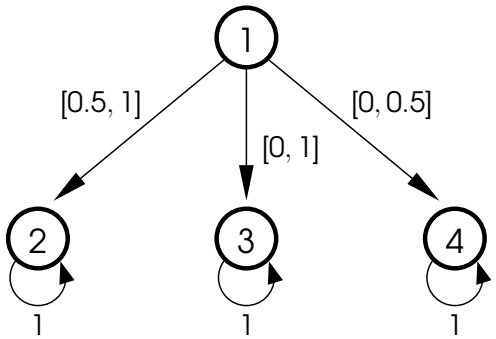
A state is **locally consistent** iff there exists a distribution ρ matching its outgoing intervals.



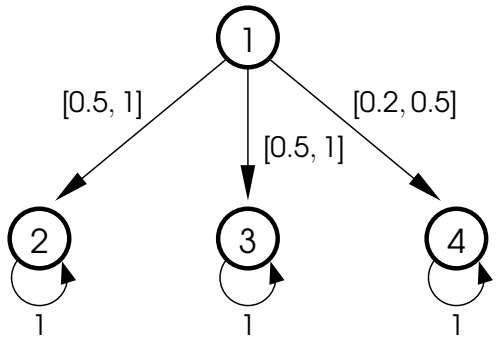


Local state-consistency

A state is **locally consistent** iff there exists a distribution ρ matching its outgoing intervals.

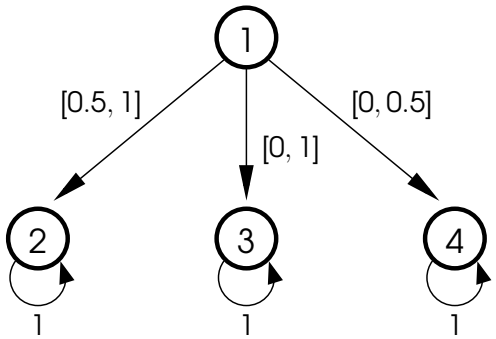


Consistent!

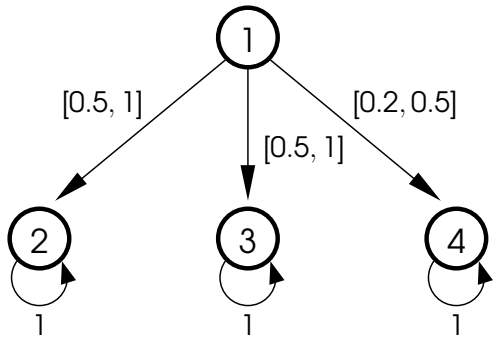


Local state-consistency

A state is **locally consistent** iff there exists a distribution ρ matching its outgoing intervals.



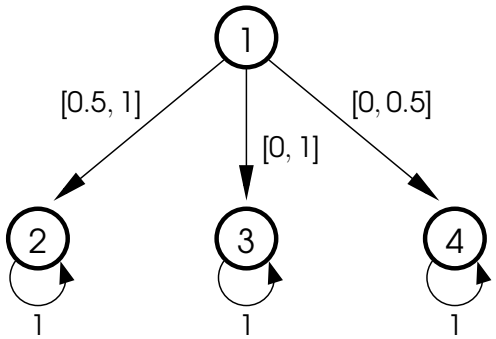
Consistent!



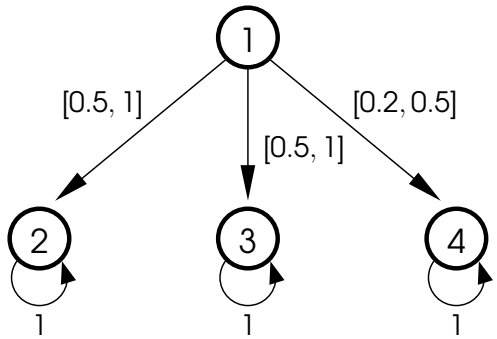
Inconsistent!

Local state-consistency

A state is **locally consistent** iff there exists a distribution ρ matching its outgoing intervals.



Consistent!



Inconsistent!

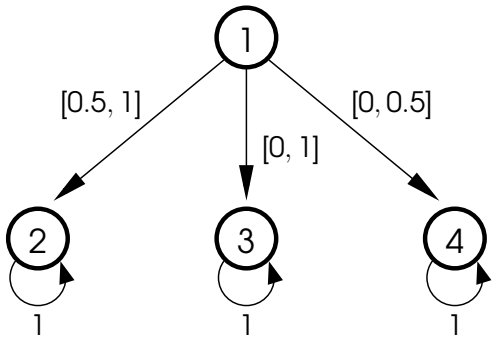
Local state-consistency

s is **locally consistent** iff

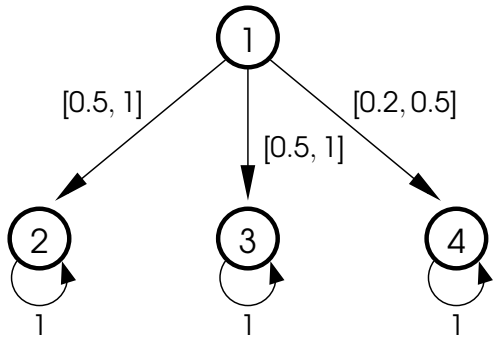
$$\sum_{s' \in S} \mathtt{Low}(\varphi(s, s')) \leq 1 \leq \sum_{s' \in S} \mathtt{Up}(\varphi(s, s'))$$

Local state-consistency

A state is **locally consistent** iff there exists a distribution ρ matching its outgoing intervals.



Consistent!



Inconsistent!

Local state-consistency

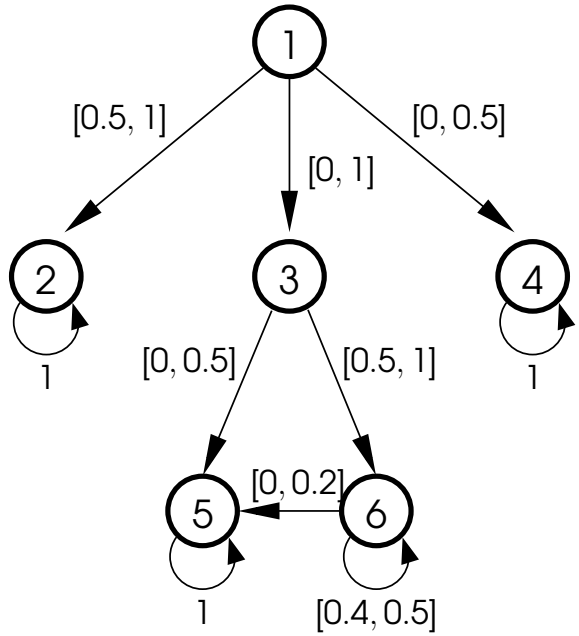
s is **locally consistent** iff

$$\sum_{s' \in S} \mathtt{Low}(\varphi(s, s')) \leq 1 \leq \sum_{s' \in S} \mathtt{Up}(\varphi(s, s'))$$

(if all intervals are **well-formed**)

Pruning operator for IMCs

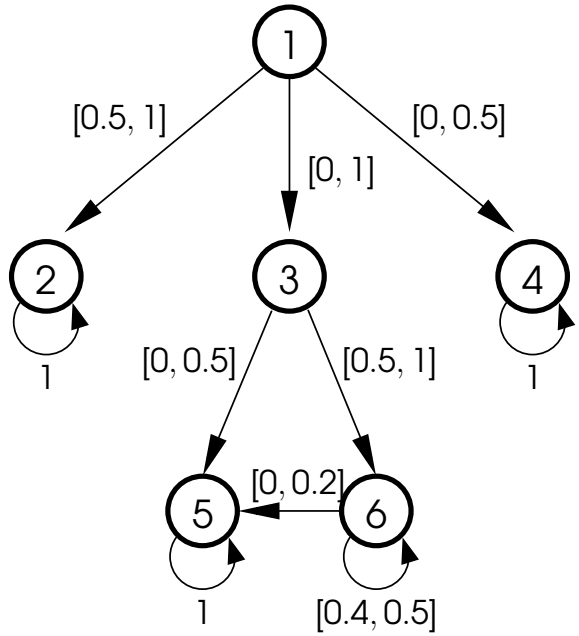
Remove all inconsistent behaviors



Pruning operator for IMCs

Remove all inconsistent behaviors

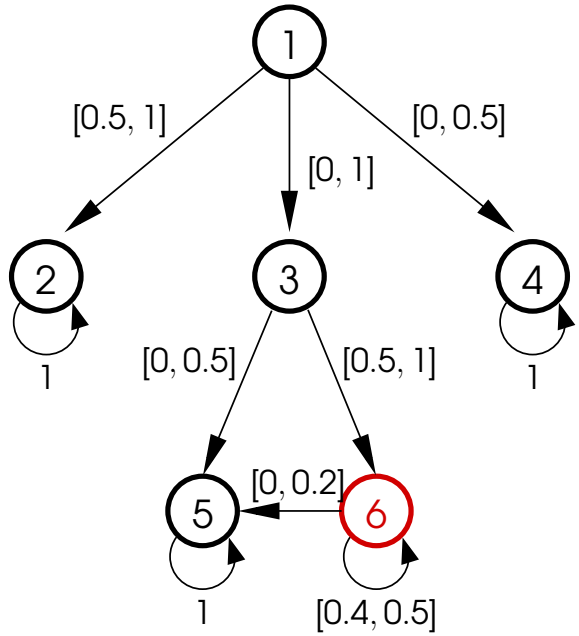
- Identify locally inconsistent states



Pruning operator for IMCs

Remove all inconsistent behaviors

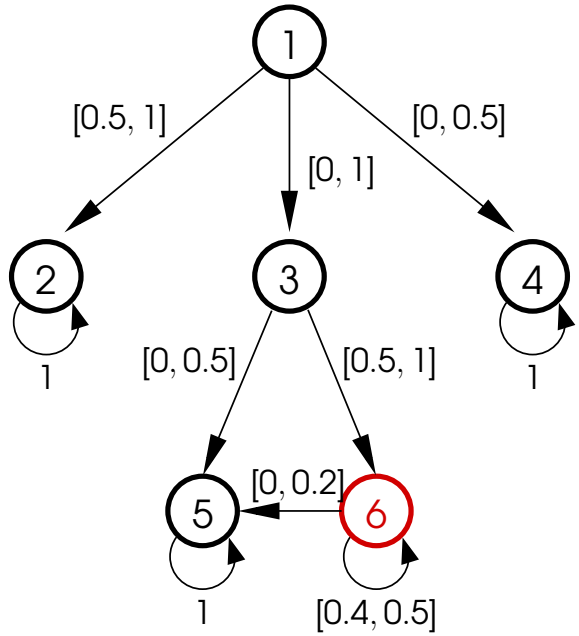
- Identify locally inconsistent states



Pruning operator for IMCs

Remove all inconsistent behaviors

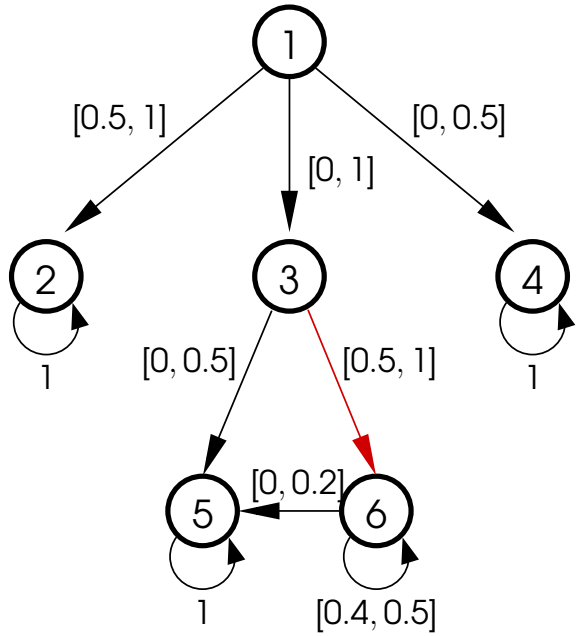
- Identify locally inconsistent states
- Make them unreachable



Pruning operator for IMCs

Remove all inconsistent behaviors

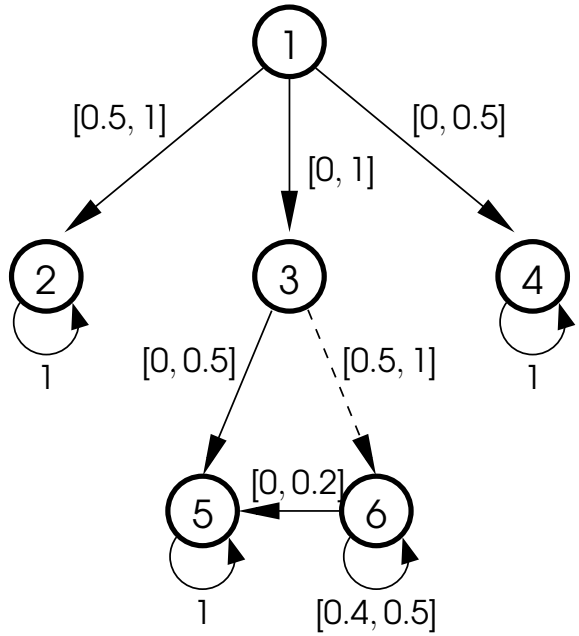
- Identify locally inconsistent states
- Make them unreachable



Pruning operator for IMCs

Remove all inconsistent behaviors

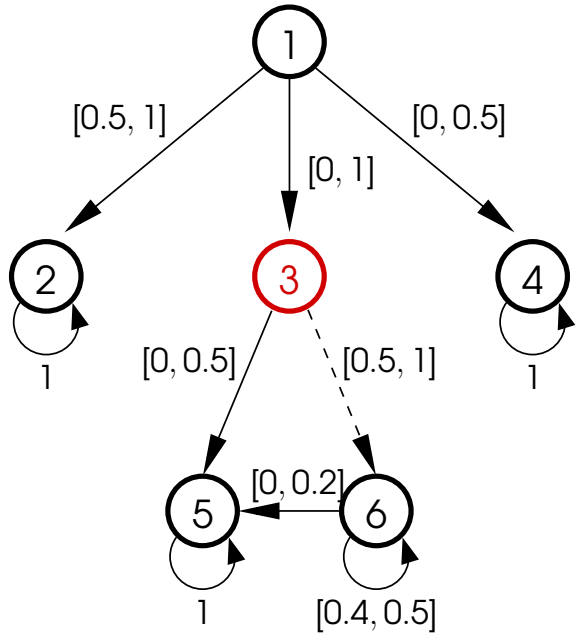
- Identify locally inconsistent states
- Make them unreachable



Pruning operator for IMCs

Remove all inconsistent behaviors

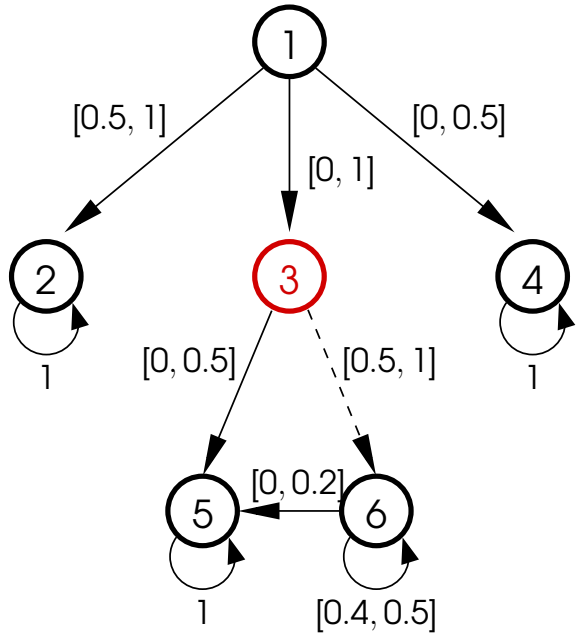
- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear



Pruning operator for IMCs

Remove all inconsistent behaviors

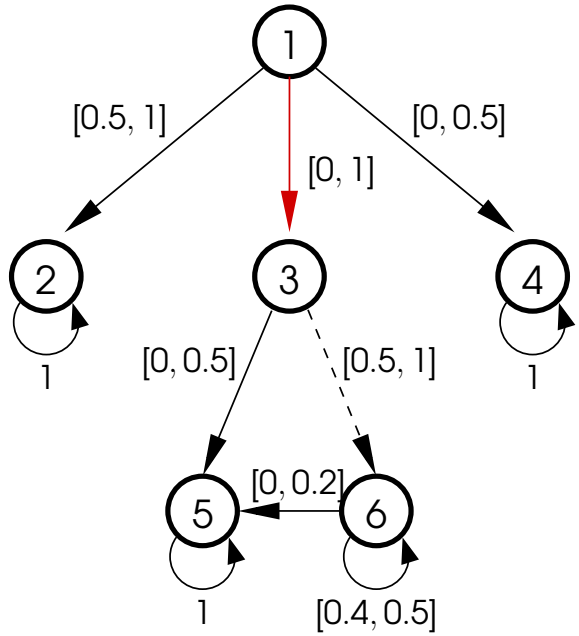
- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



Pruning operator for IMCs

Remove all inconsistent behaviors

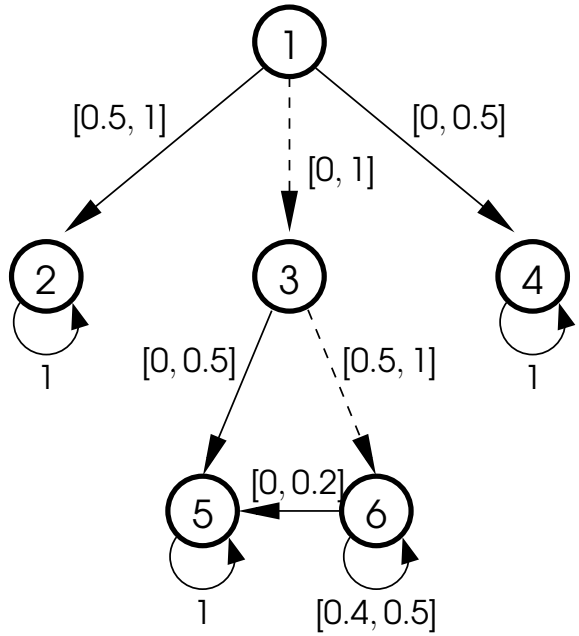
- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



Pruning operator for IMCs

Remove all inconsistent behaviors

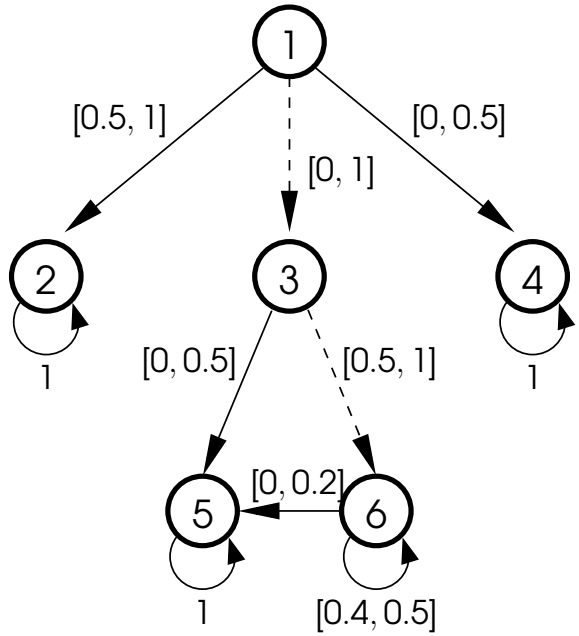
- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



Pruning operator for IMCs

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



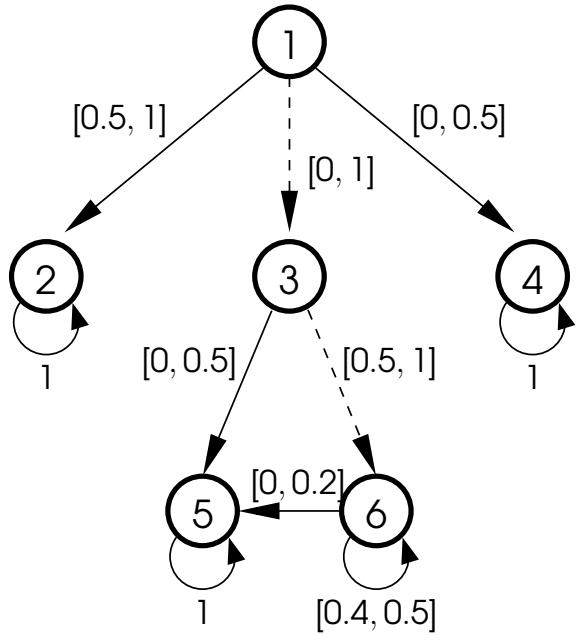
How to make state s_i unreachable?

For all other states $s \neq s_i$, two cases:

Pruning operator for IMCs

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



How to make state s_i unreachable?

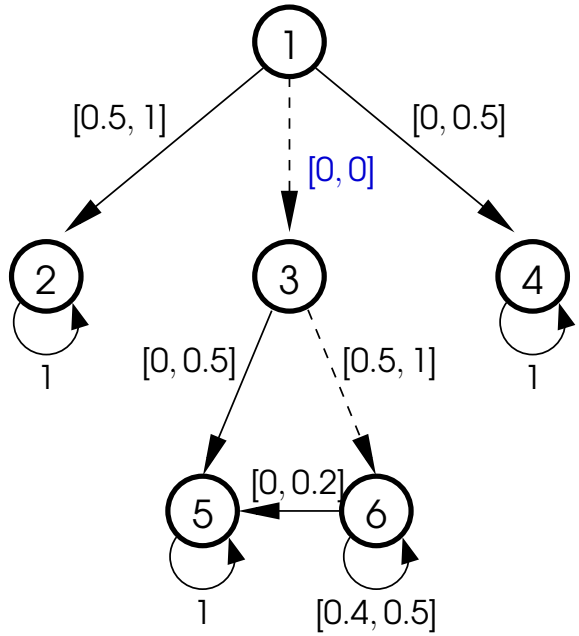
For all other states $s \neq s_i$, two cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$

Pruning operator for IMCs

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



How to make state s_i unreachable?

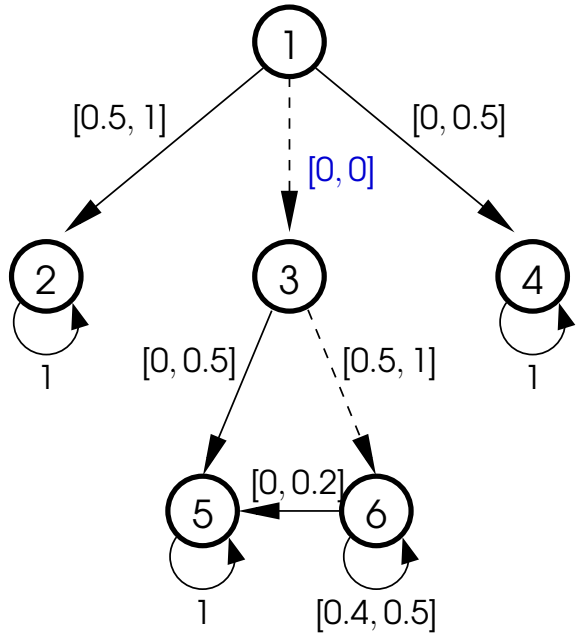
For all other states $s \neq s_i$, two cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$

Pruning operator for IMCs

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



How to make state s_i unreachable?

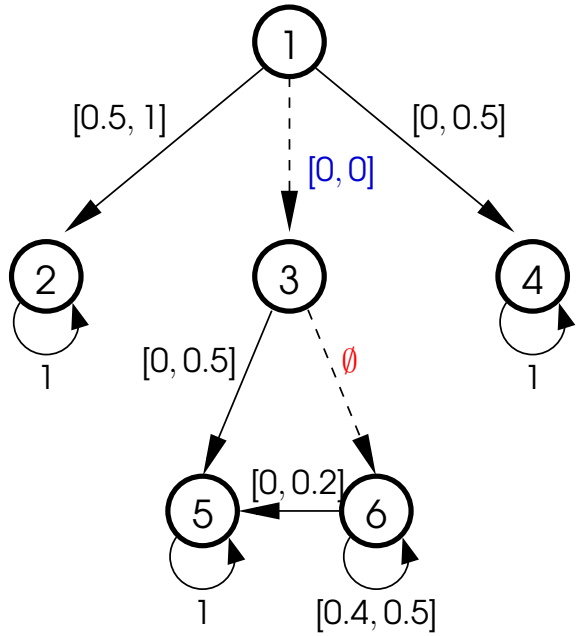
For all other states $s \neq s_i$, two cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$
- If $0 \notin \varphi(s, s_i)$, then $\varphi'(s, s_i) = \emptyset$

Pruning operator for IMCs

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable



How to make state s_i unreachable?

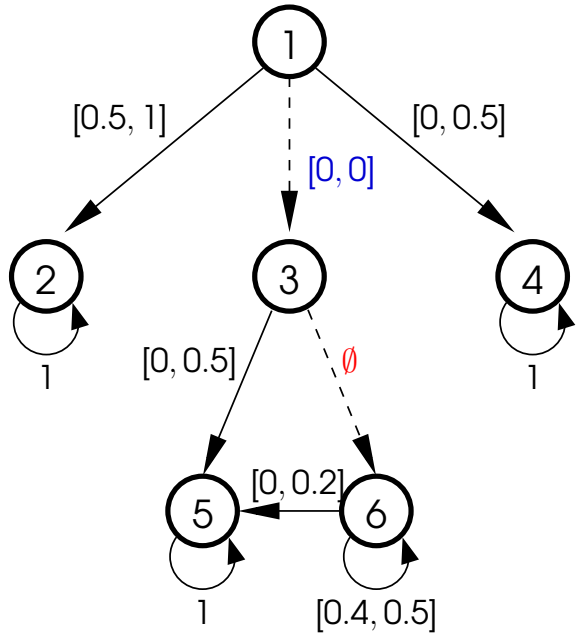
For all other states $s \neq s_i$, two cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$
- If $0 \notin \varphi(s, s_i)$, then $\varphi'(s, s_i) = \emptyset$

Pruning operator for IMCs

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- New inconsistent states may appear
- Repeat until no more inconsistent states are reachable

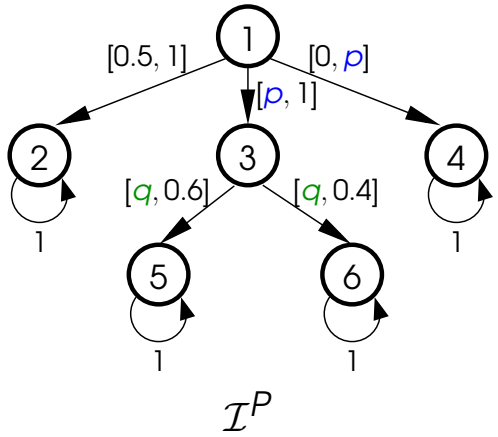


How to make state s_i unreachable?

For all other states $s \neq s_i$, two cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$
- If $0 \notin \varphi(s, s_i)$, then $\varphi'(s, s_i) = \emptyset \Rightarrow$ Makes s inconsistent

Consistency of pIMCs

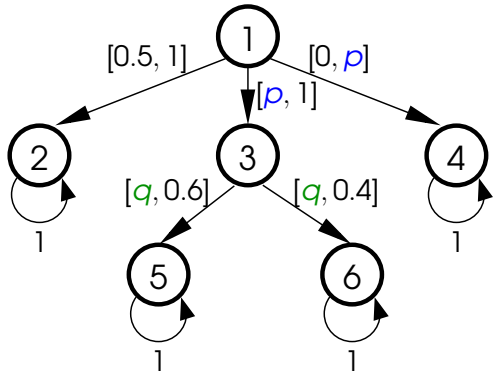


$\mathcal{I}^P$

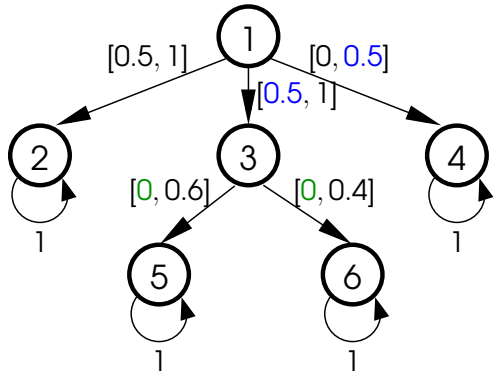
pIMC $\mathcal{I}^P$ is consistent iff there exists at least one IMC $\mathcal{I}$ and one MC $\mathcal{M}$ such that

$$\mathcal{M} \models \mathcal{I} \vdash \mathcal{I}^P$$

Consistency of pIMCs



$\mathcal{I}^P$

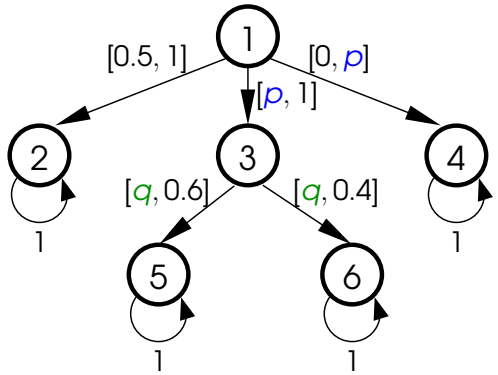


$\mathcal{I}$

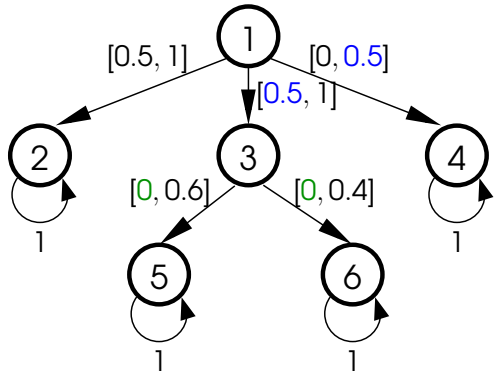
pIMC $\mathcal{I}^P$ is consistent iff there exists at least one IMC $\mathcal{I}$ and one MC $\mathcal{M}$ such that

$$\mathcal{M} \models \mathcal{I} \vdash \mathcal{I}^P$$

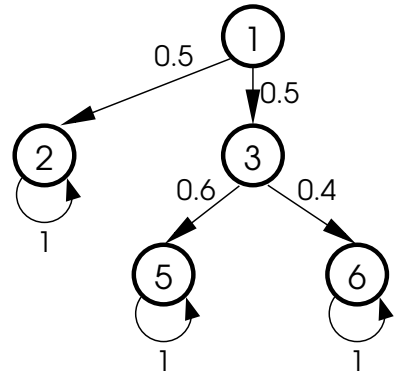
Consistency of pIMCs



$\mathcal{I}^P$



$\mathcal{I}$



$\mathcal{M}$

pIMC $\mathcal{I}^P$ is consistent iff there exists at least one IMC $\mathcal{I}$ and one MC $\mathcal{M}$ such that

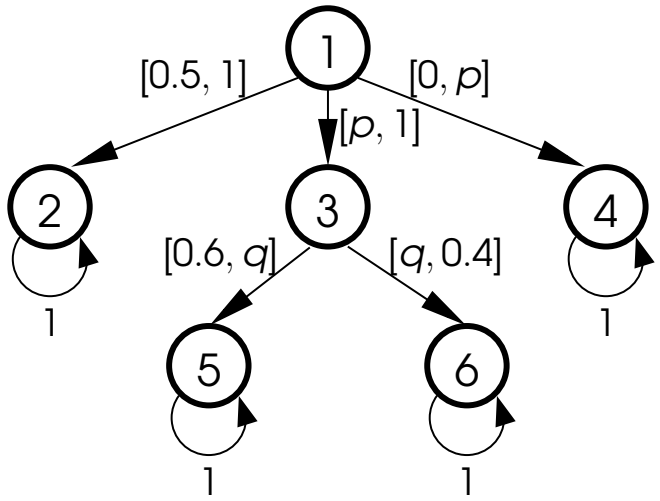
$$\mathcal{M} \models \mathcal{I} \vdash \mathcal{I}^P$$

Local state-consistency (1/2)

A state is **locally consistent** iff there exists a valuation for the parameters and a distribution ρ matching the obtained outgoing intervals.

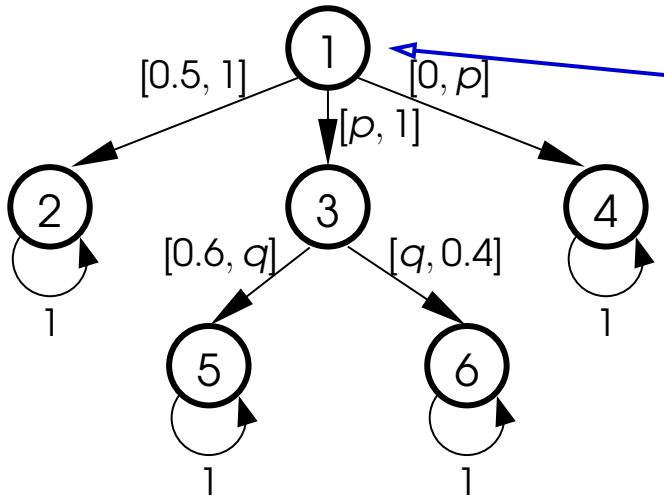
Local state-consistency (1/2)

A state is **locally consistent** iff there exists a valuation for the parameters and a distribution ρ matching the obtained outgoing intervals.



Local state-consistency (1/2)

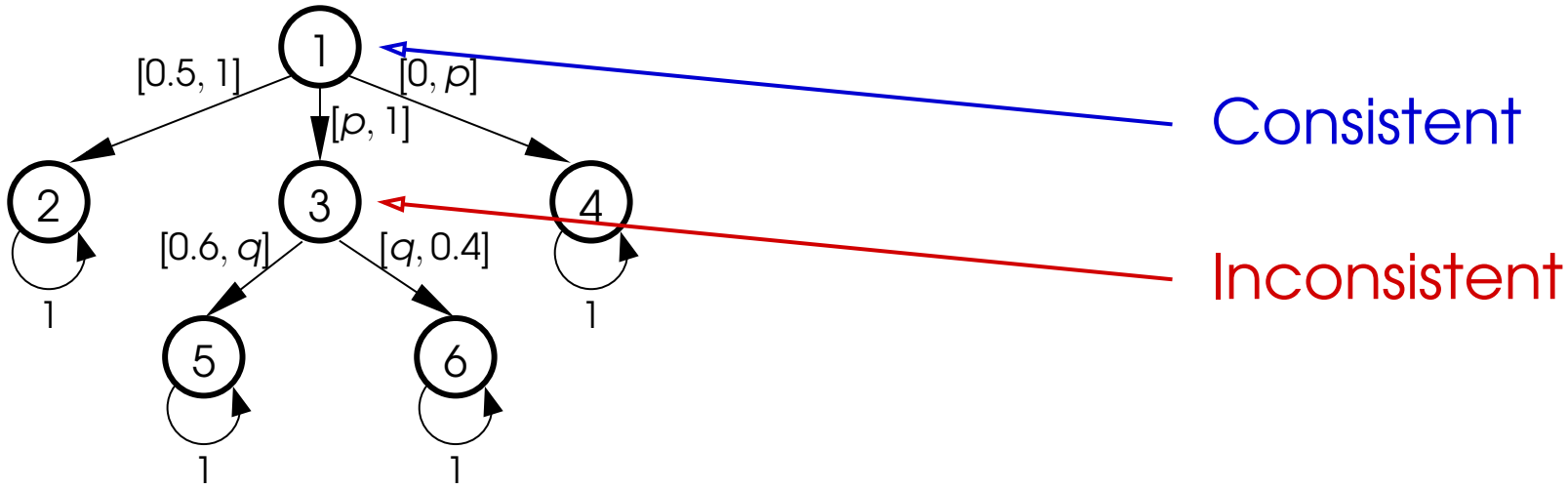
A state is **locally consistent** iff there exists a valuation for the parameters and a distribution ρ matching the obtained outgoing intervals.



Consistent

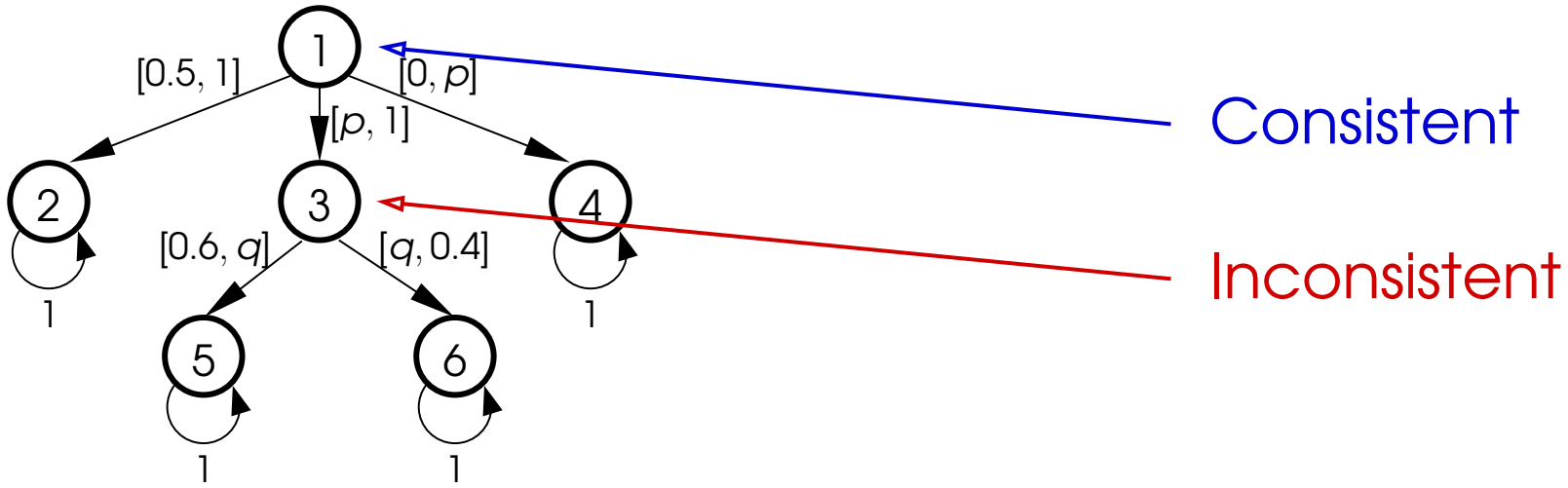
Local state-consistency (1/2)

A state is **locally consistent** iff there exists a valuation for the parameters and a distribution ρ matching the obtained outgoing intervals.



Local state-consistency (1/2)

A state is **locally consistent** iff there exists a valuation for the parameters and a distribution ρ matching the obtained outgoing intervals.



Local state-consistency

s is **locally consistent** iff the following system admits a solution:

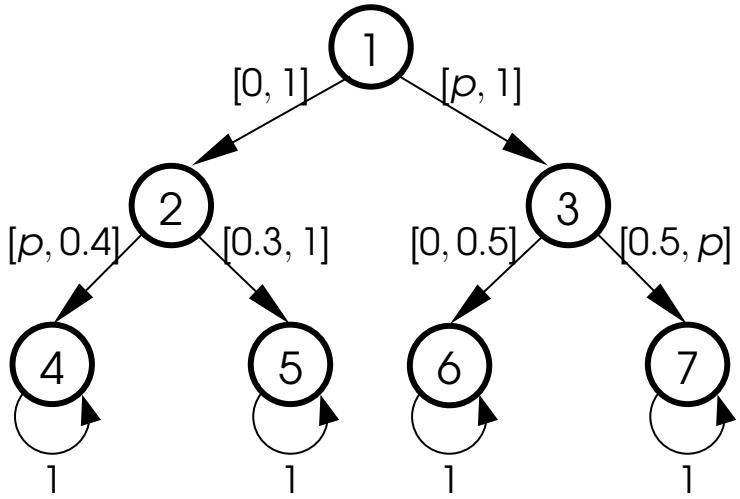
$$\sum_{j=1}^{n} \texttt{Low}(\varphi_P(s, s_j)) \leq 1 \wedge \sum_{j=1}^{n} \texttt{Up}(\varphi_P(s, s_j)) \geq 1 \wedge \texttt{Low}(\varphi_P(s, s_1)) \leq \texttt{Up}(\varphi_P(s, s_2)) \\ \wedge \ldots \wedge \texttt{Low}(\varphi_P(s, s_n)) \leq \texttt{Up}(\varphi_P(s, s_n))$$

Local state-consistency (2/2)

Only for **local** plMCs

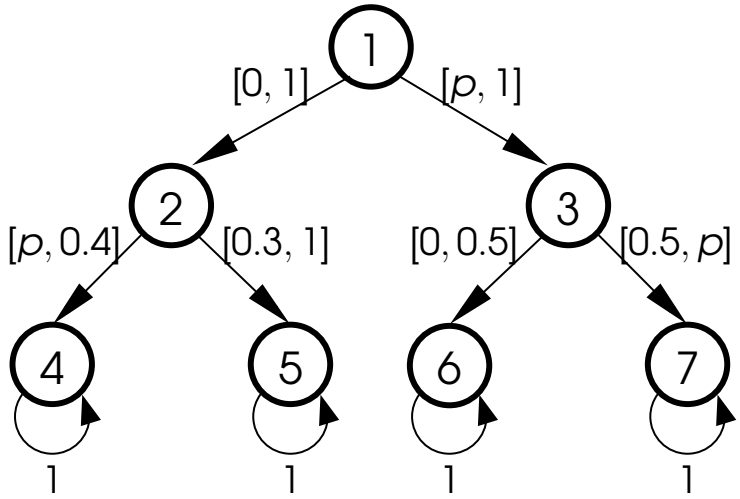
Local state-consistency (2/2)

Only for **local** pIMCs



Local state-consistency (2/2)

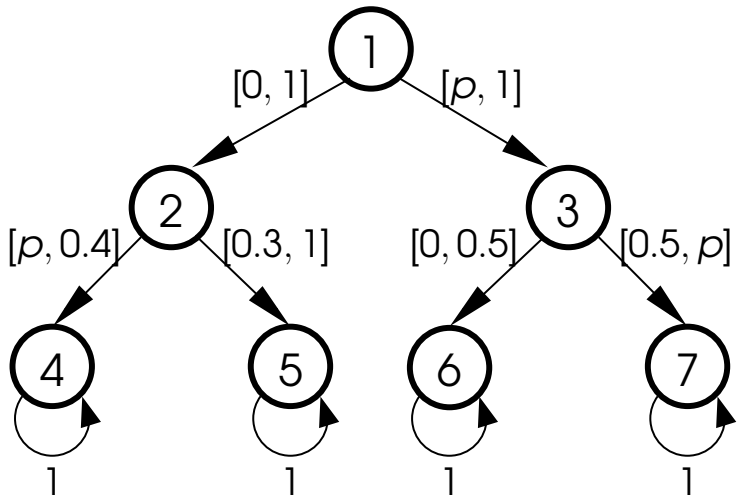
Only for **local** pIMCs



- All states are *locally* consistent

Local state-consistency (2/2)

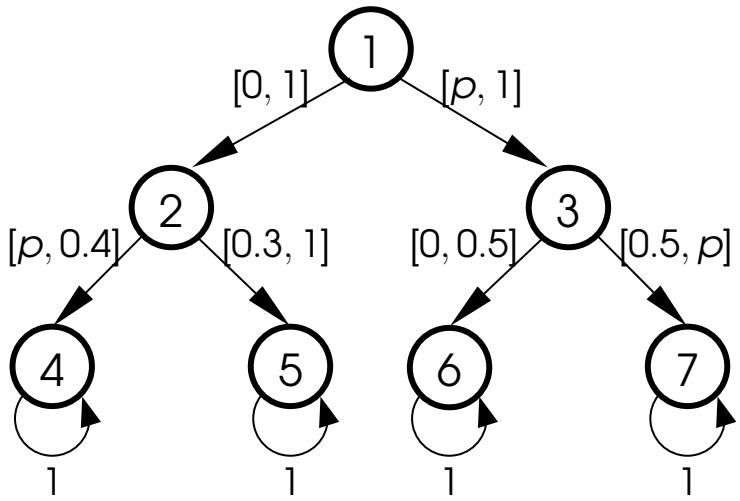
Only for **local** pIMCs



- All states are *locally* consistent
- There is no *global* value of p such that all intervals are well-formed

Local state-consistency (2/2)

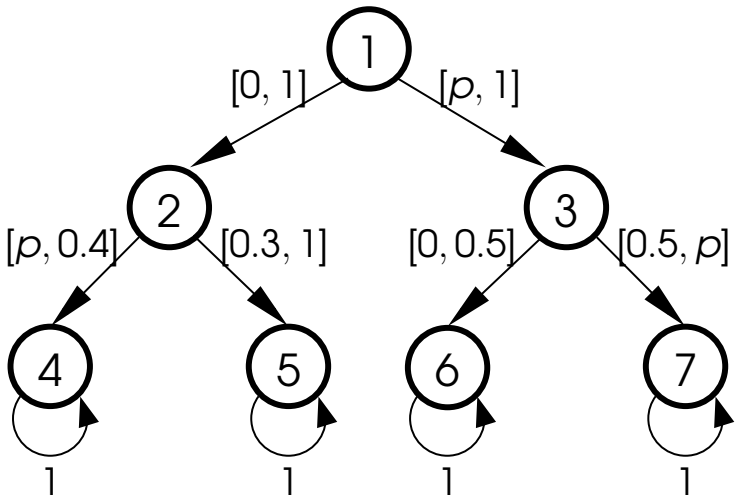
Only for **local** pIMCs

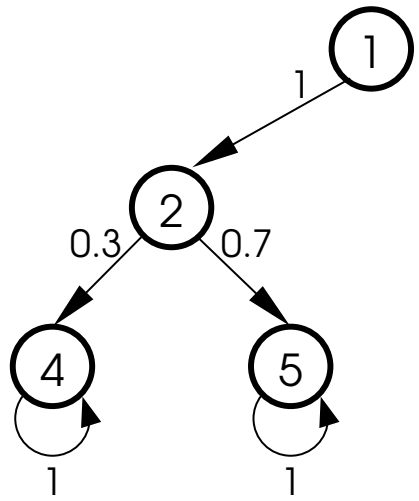


- All states are *locally* consistent
- There is no *global* value of p such that all intervals are well-formed
- The pIMC **is** consistent

Local state-consistency (2/2)

Only for **local** pIMCs





- All states are *locally* consistent
- There is no *global* value of p such that all intervals are well-formed
- The pIMC **is** consistent

Pruning of local pIMCs

Same principle...

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- Repeat until no more inconsistent states are reachable

Pruning of local pIMCs

Same principle...

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- Repeat until no more inconsistent states are reachable

How to make state s_i unreachable?

For all other states $s \neq s_i$, three cases:

Pruning of local pIMCs

Same principle...

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- Repeat until no more inconsistent states are reachable

How to make state s_i unreachable?

For all other states $s \neq s_i$, three cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$

Pruning of local pIMCs

Same principle...

Remove all inconsistent behaviors

- Identify locally inconsistent states
- Make them unreachable
- Repeat until no more inconsistent states are reachable

How to make state s_i unreachable?

For all other states $s \neq s_i$, three cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$
- If $\varphi(s, s_i) = [p, .]$, then $\varphi'(s, s_i) = [0, 0]$ **and $\mathbf{p} = \mathbf{0}$ in all other intervals**

Pruning of local pIMCs

Same principle...

Remove all inconsistent behaviors

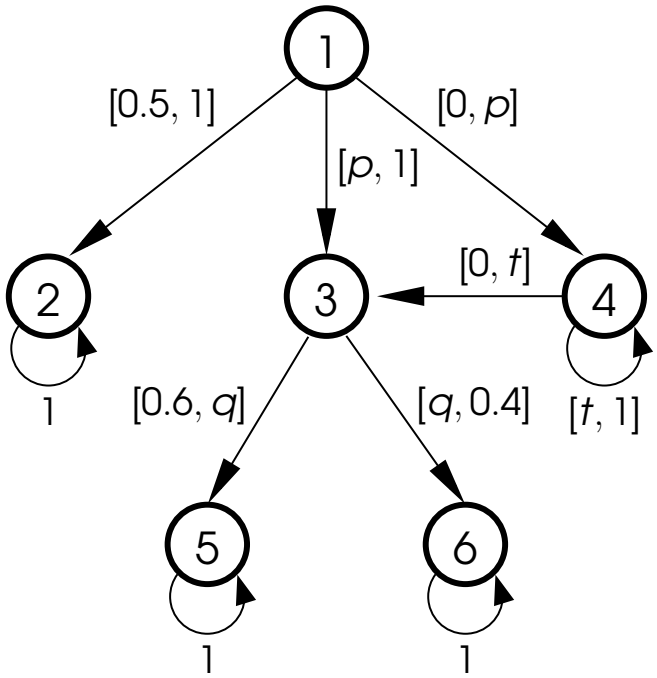
- Identify locally inconsistent states
- Make them unreachable
- Repeat until no more inconsistent states are reachable

How to make state s_i unreachable?

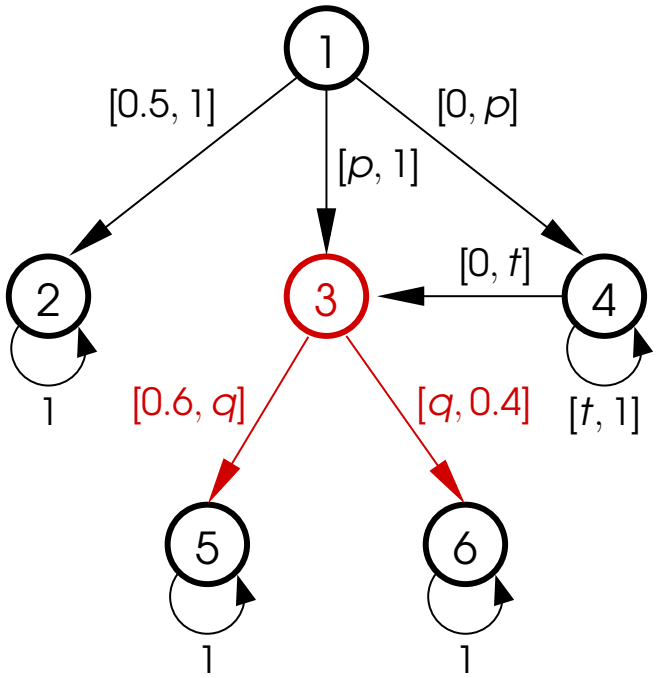
For all other states $s \neq s_i$, three cases:

- If $\varphi(s, s_i) = [0, .]$, then $\varphi'(s, s_i) = [0, 0]$
- If $\varphi(s, s_i) = [p, .]$, then $\varphi'(s, s_i) = [0, 0]$ **and $p = 0$ in all other intervals**
- If $\varphi(s, s_i) = [x, 0]$ with $x > 0$, then $\varphi'(s, s_i) = \emptyset$

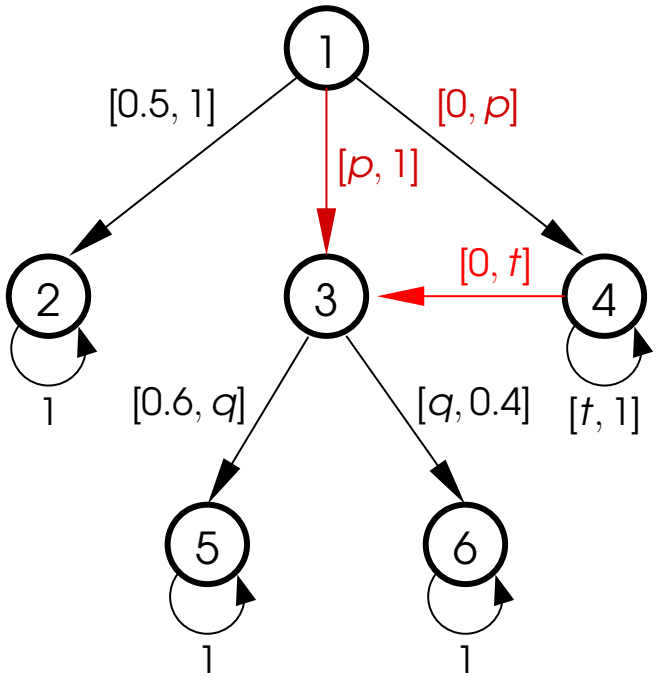
Example



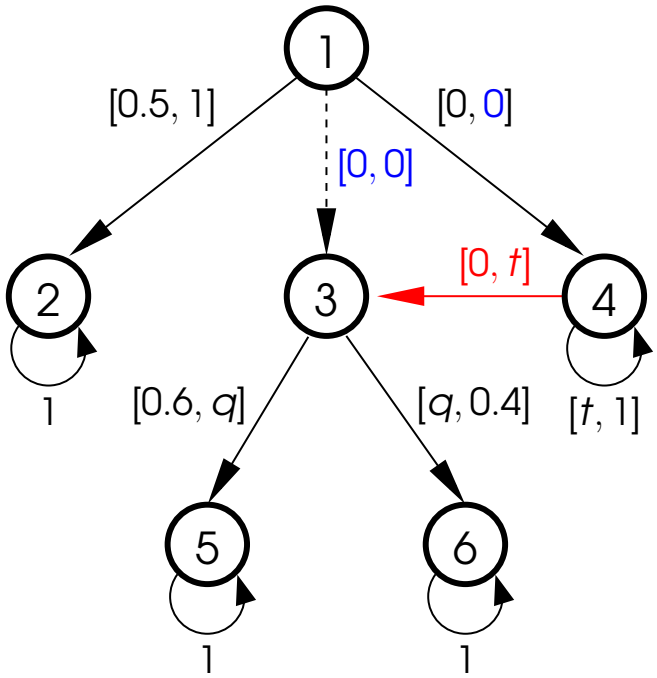
Example



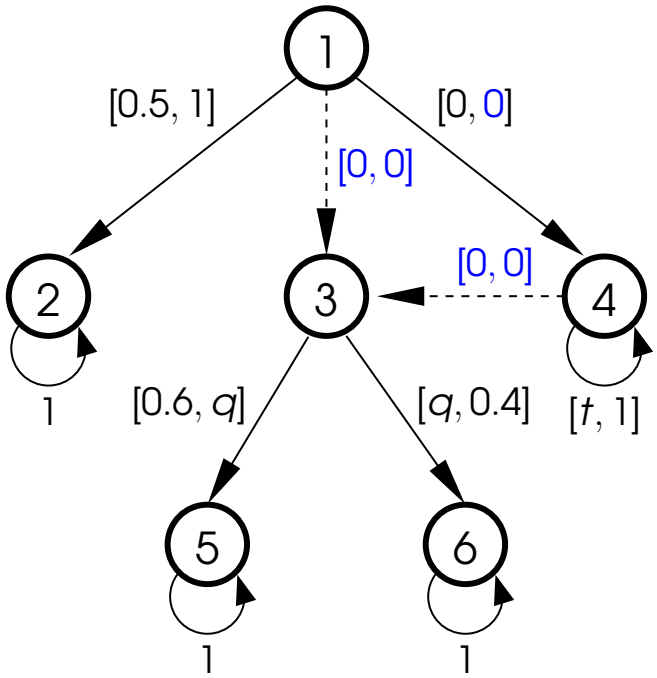
Example



Example



Example



Consistency of global pIMCs - Hints

Brute force algorithm

Inter-consistency: Parameters are local to *groups of states* $\Rightarrow$ combine their systems of inequations.

Consistency of global plMCs - Hints

Brute force algorithm

Inter-consistency: Parameters are local to *groups of states* $\Rightarrow$ combine their systems of inequations.

- Identify inconsistent *groups of states*

Consistency of global pIMCs - Hints

Brute force algorithm

Inter-consistency: Parameters are local to *groups of states* $\Rightarrow$ combine their systems of inequations.

- Identify inconsistent *groups of states*
- Make one state of the group unreachable and start over
  - If consistency is proven $\Rightarrow$ ok
  - If not, backtrack and try another state

Consistency of global pIMCs - Hints

Brute force algorithm

Inter-consistency: Parameters are local to *groups of states* $\Rightarrow$ combine their systems of inequations.

- Identify inconsistent *groups of states*
- Make one state of the group unreachable and start over
  - If consistency is proven $\Rightarrow$ ok
  - If not, backtrack and try another state
- If all possible combinations have failed, the pIMC is inconsistent

Other problems of interest

Other problems of interest

- Global Parameters

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property
  - Reachability

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property
  - Reachability
  - Avoidability

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property
  - Reachability
  - Avoidability
  - ...

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property
  - Reachability
  - Avoidability
  - ...
- Parameters in Constraint Markov Chains/APA

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property
  - Reachability
  - Avoidability
  - ...
- Parameters in Constraint Markov Chains/APA
- Specification theory using pIMCs/pCMCs/pAPAs

Other problems of interest

- Global Parameters
- Synthesize *all* parameter values ensuring consistency
- Synthesize parameter values optimizing some property
  - Reachability
  - Avoidability
  - ...
- Parameters in Constraint Markov Chains/APA
- Specification theory using pIMCs/pCMCs/pAPAs
- ...

The end...

Thank you for your attention!

Questions?