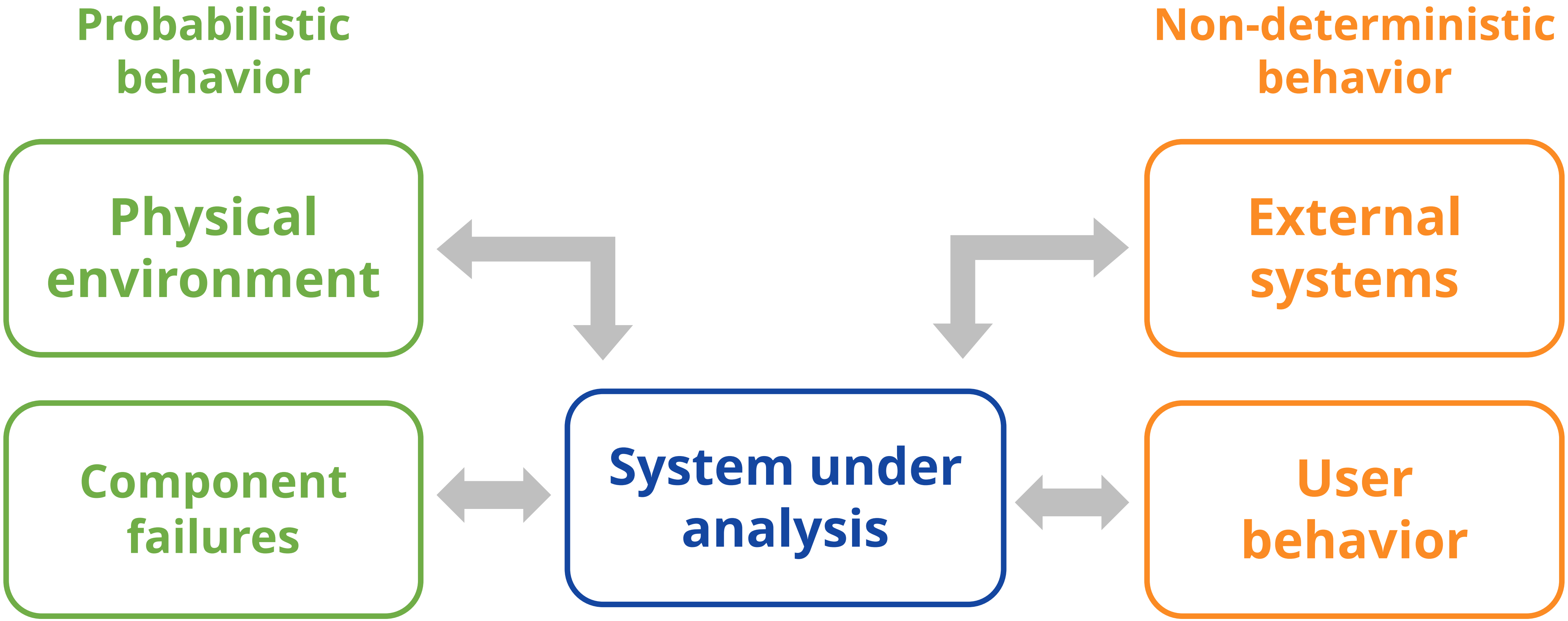
Lazy Abstraction for Markov Decision Processes

Dániel Szekeres

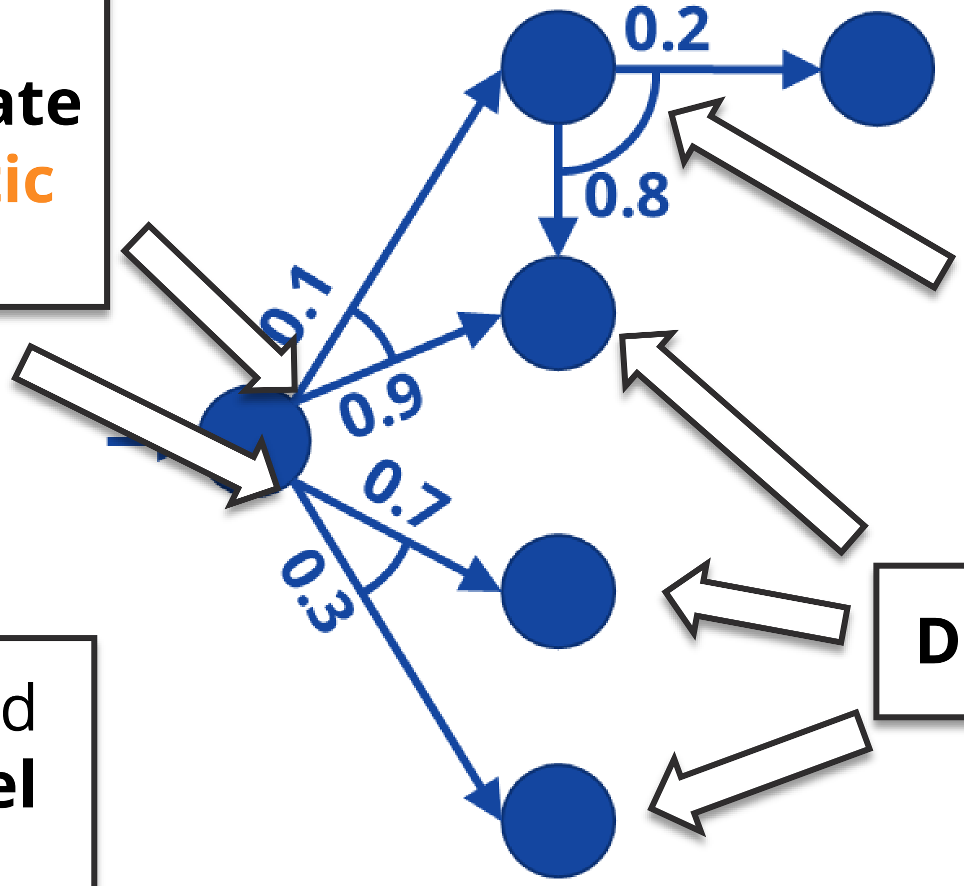
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Critical Systems Research Group

Context: Reliability analysis



Markov Decision Processes (MDP)

Multiple actions available in each state
→ Non-deterministic behavior



Resulting state sampled from a distribution
→ Probabilistic behavior

Discrete set of states

Commonly described through **higher-level formalisms**

Probabilistic Guarded Commands

- A set of **state variables**
- A set of **commands**, each having:
  - A Boolean **guard** expression over the state variables
  - A **probability distribution over effects** changing the variables

$$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$$

$$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \land y' := y), 0.2 : (x' := x \land y' := y)$$

$$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \land x' := 1)$$

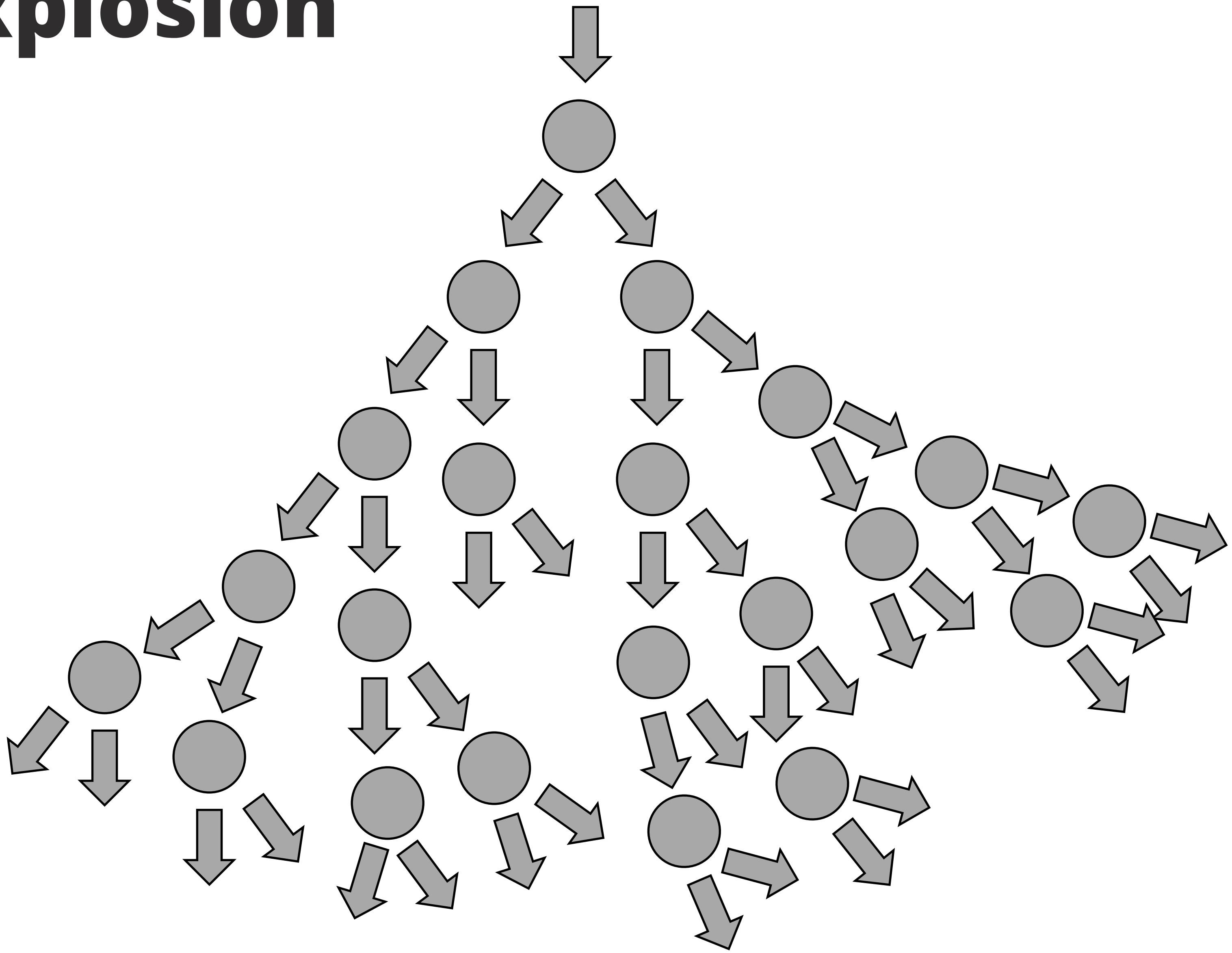
$$\mathbf{c_3} : [x == 2 \land y == 2]\ 1.0 : (y' := 3 \land x' := x)$$

State-space explosion

Exponentially large state space in the description size

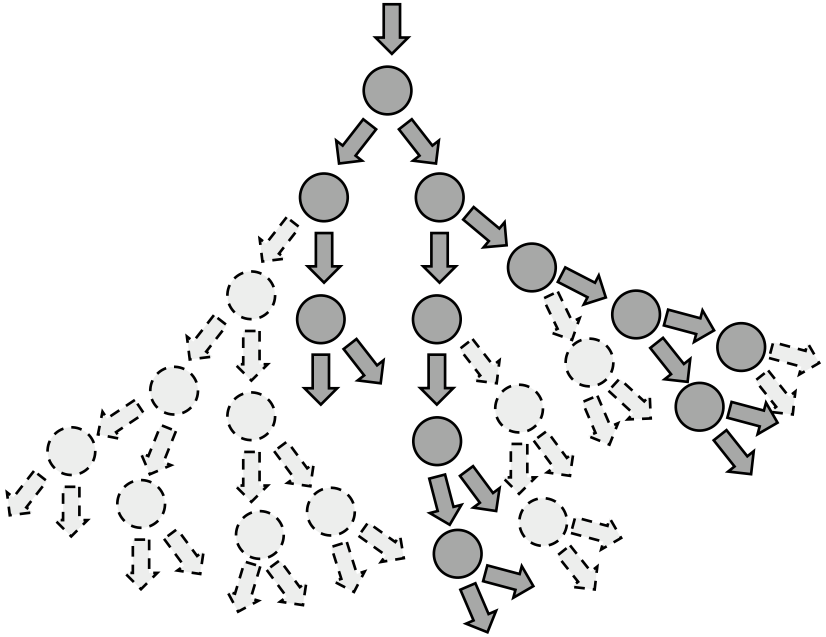
Hinders verifying complex systems in practice

Exacerbated by numerical computations in probabilistic model checking



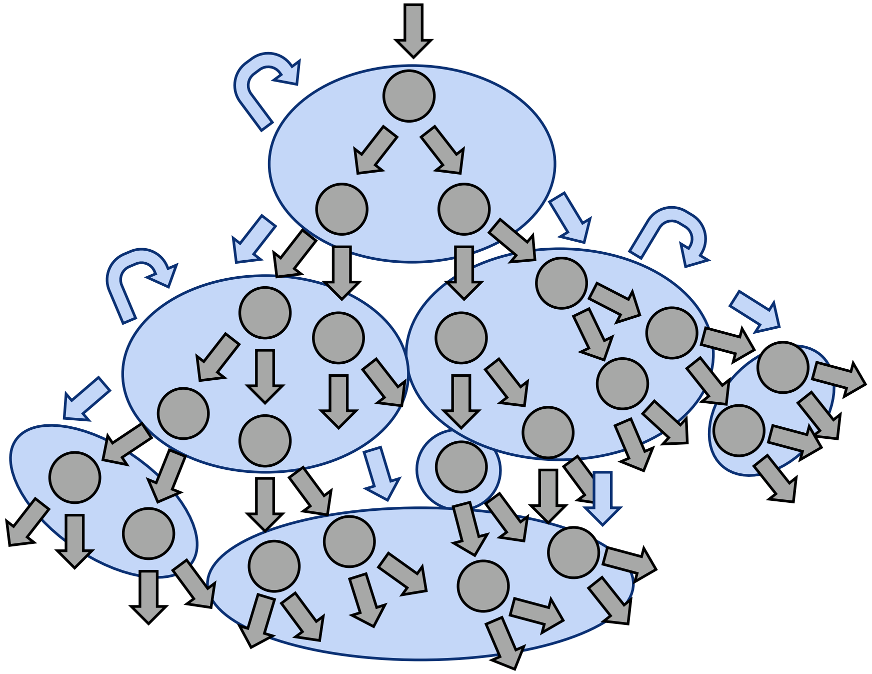
Counteracting state space explosion

Partial state space exploration



- Stop exploring new states when enough information is available

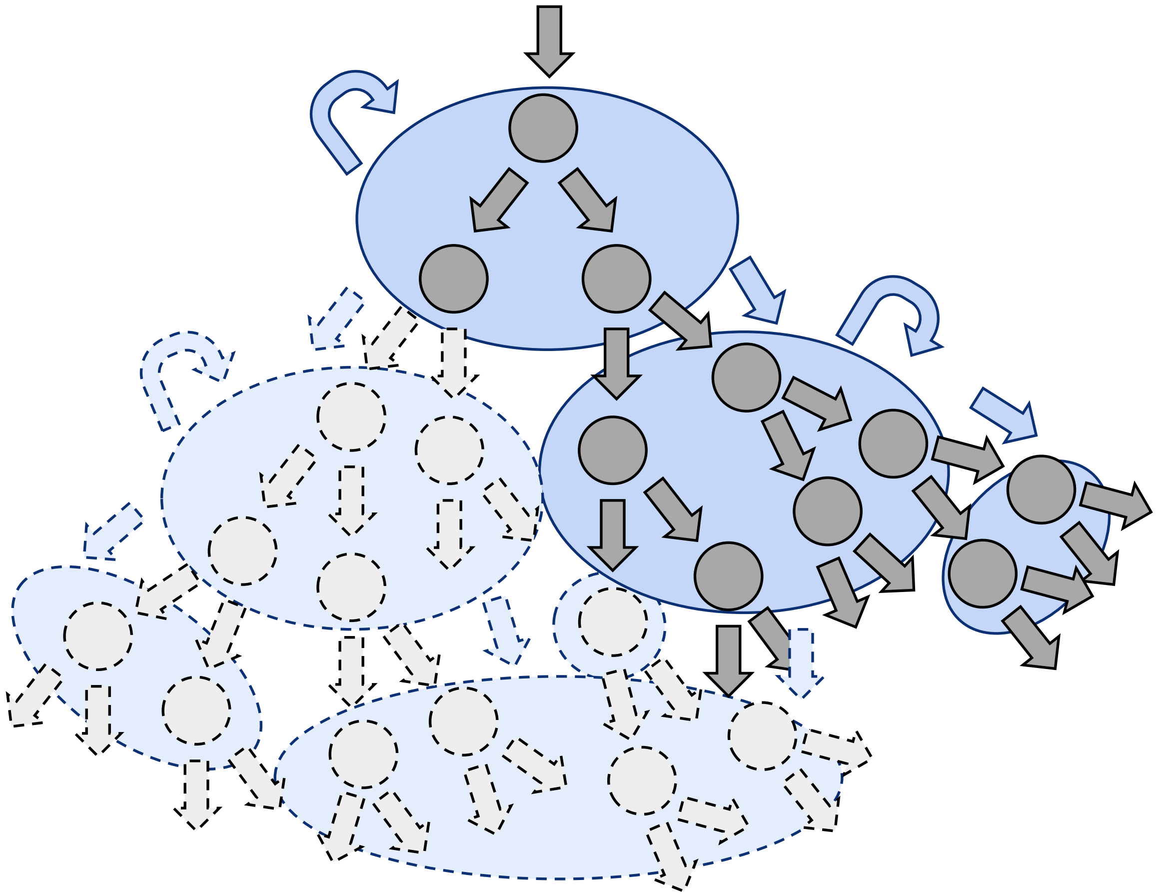
Abstraction



- Merges similar concrete states into abstract states
- Needs to be conservative

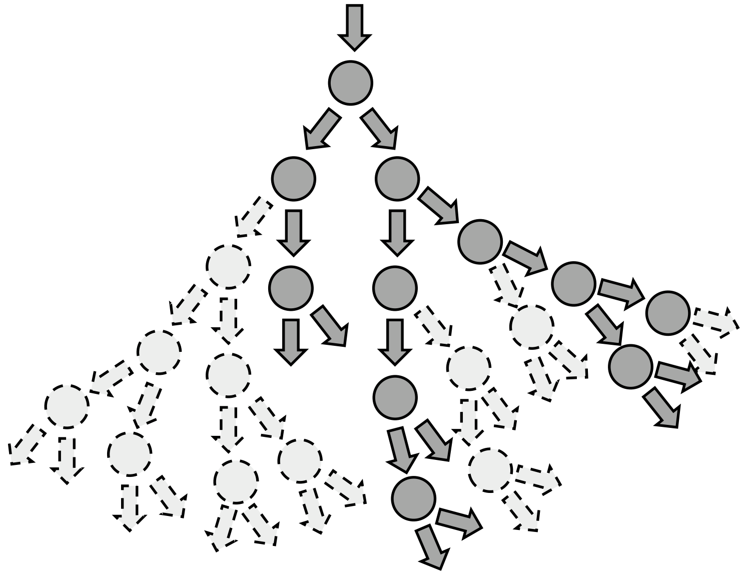
Counteracting state space explosion

Partial state space exploration + Abstraction

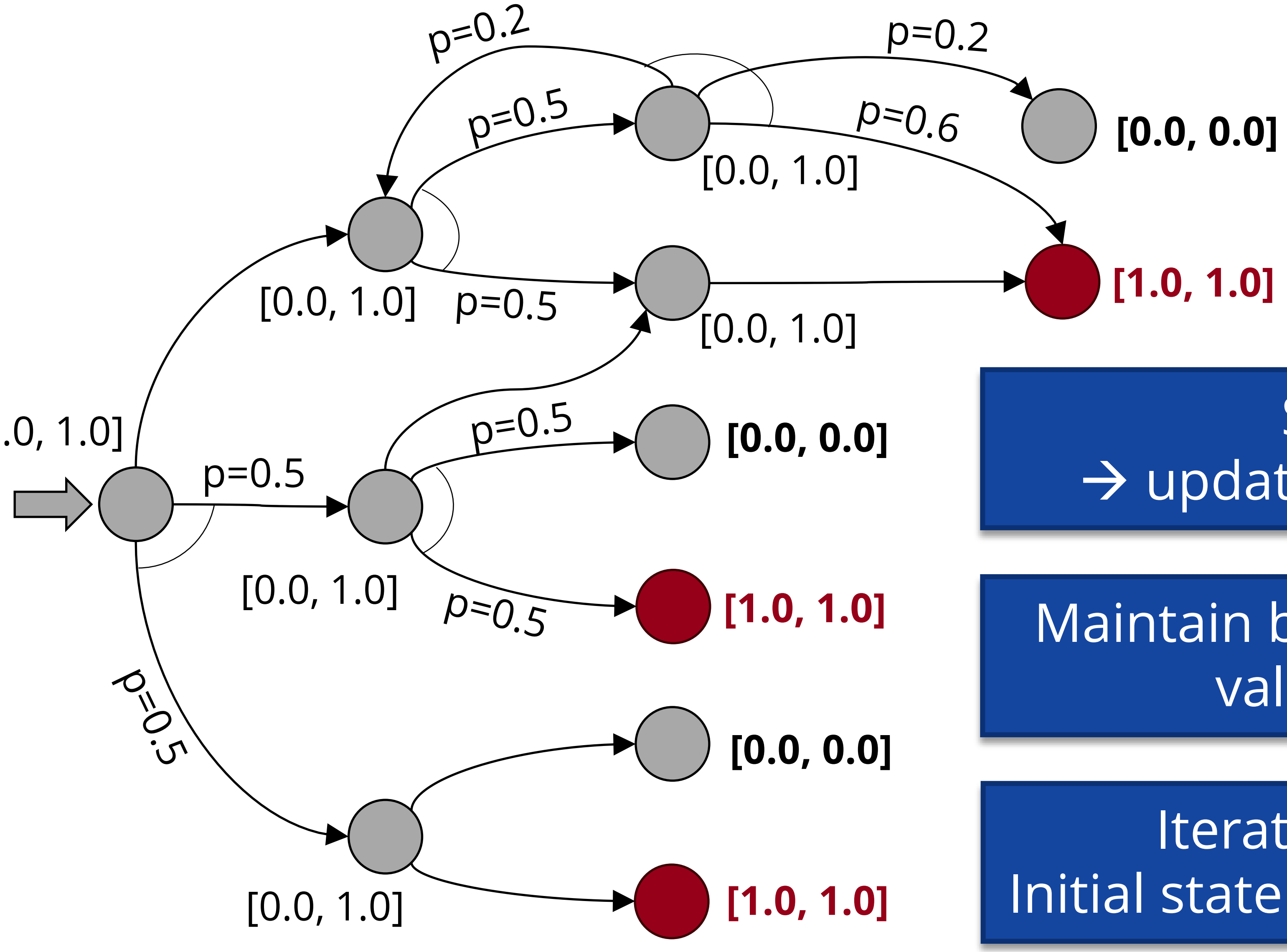


- Explore only a part of the *abstract* state space
- Already used in non-probabilistic abstraction-based model-checking
- Not in probabilistic model-checking
  - Existing MDP abstraction-refinement algorithms rely on the whole abstract state space
  - Lazy abstraction synergizes much better with partial exploration
    → needs to be adapted for MDPs

Partial state-space exploration for MDPs: BRTDP



Bounded Real-Time Dynamic Programming (BRTDP)

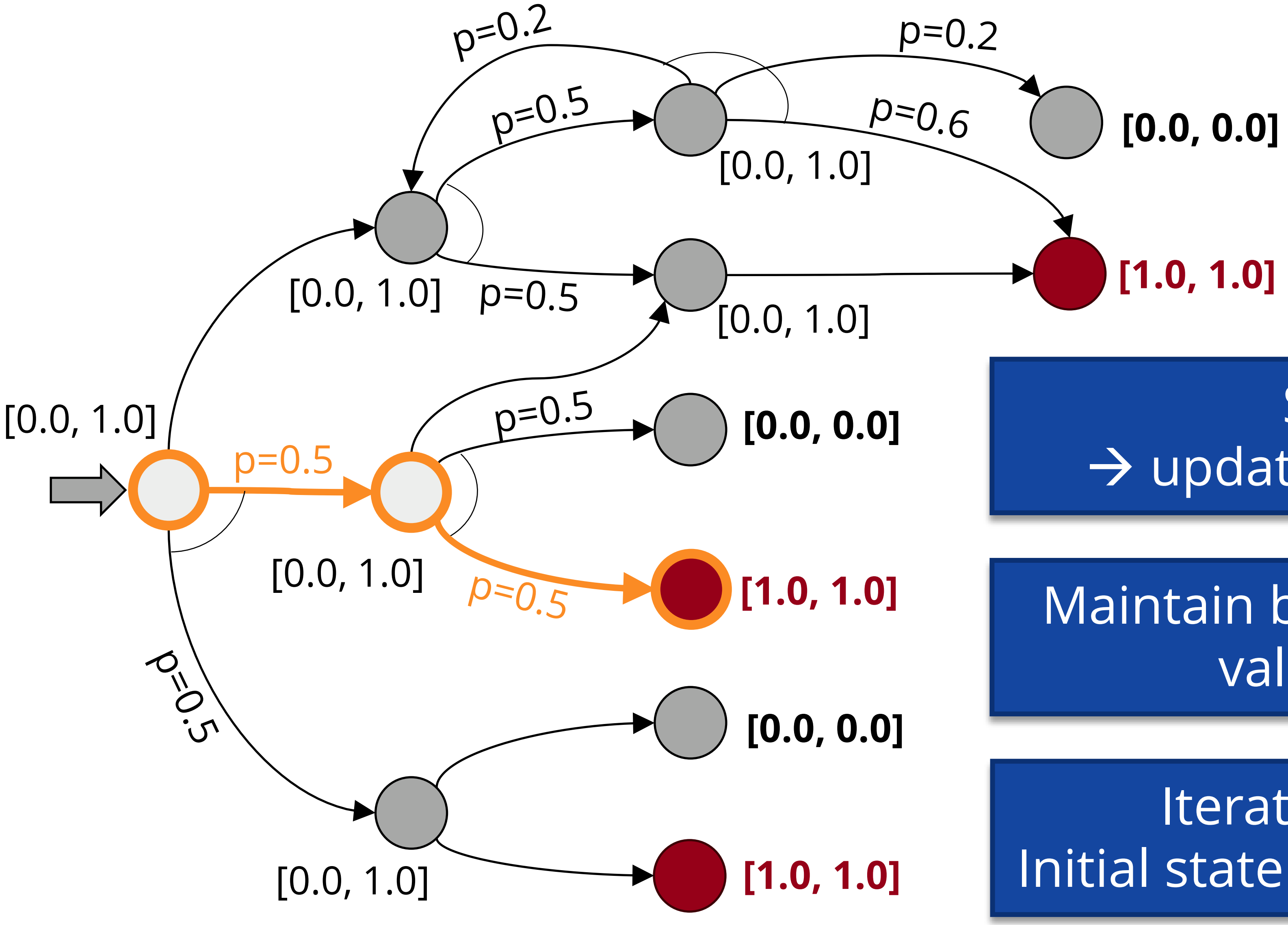


Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

Bounded Real-Time Dynamic Programming (BRTDP)

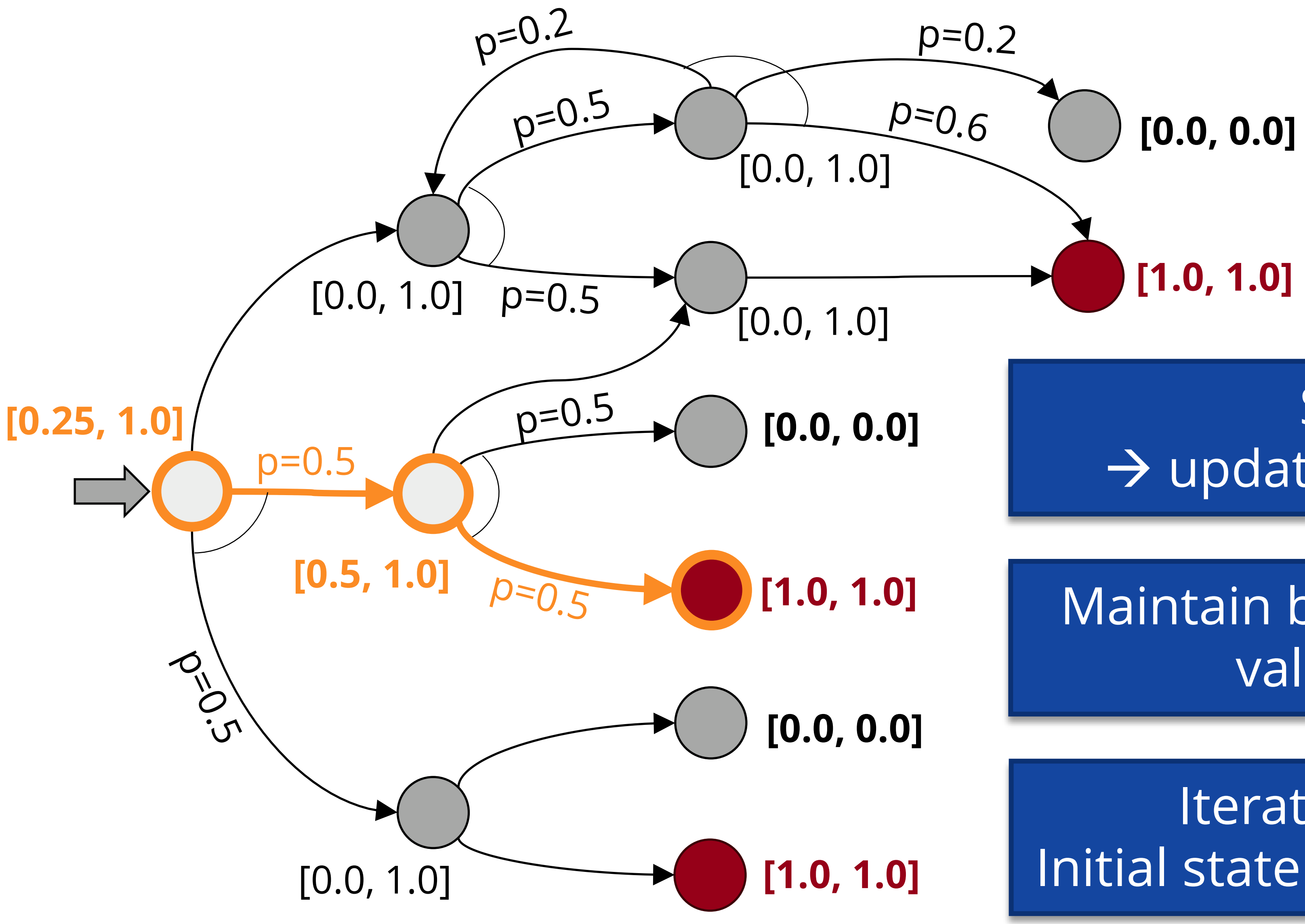


Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

Bounded Real-Time Dynamic Programming (BRTDP)



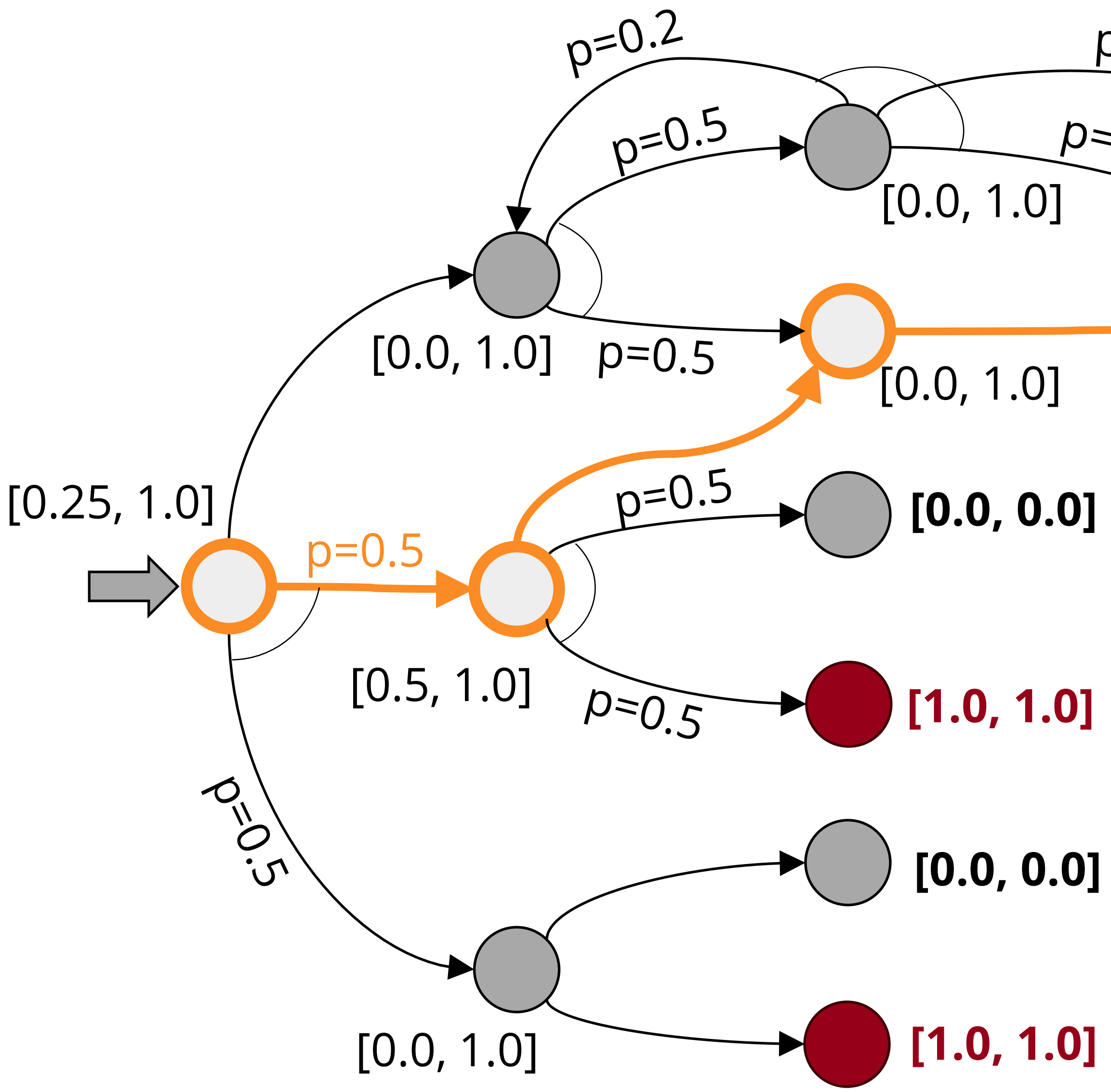
Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

ftsrg

Bounded Real-Time Dynamic Programming (BRTDP)

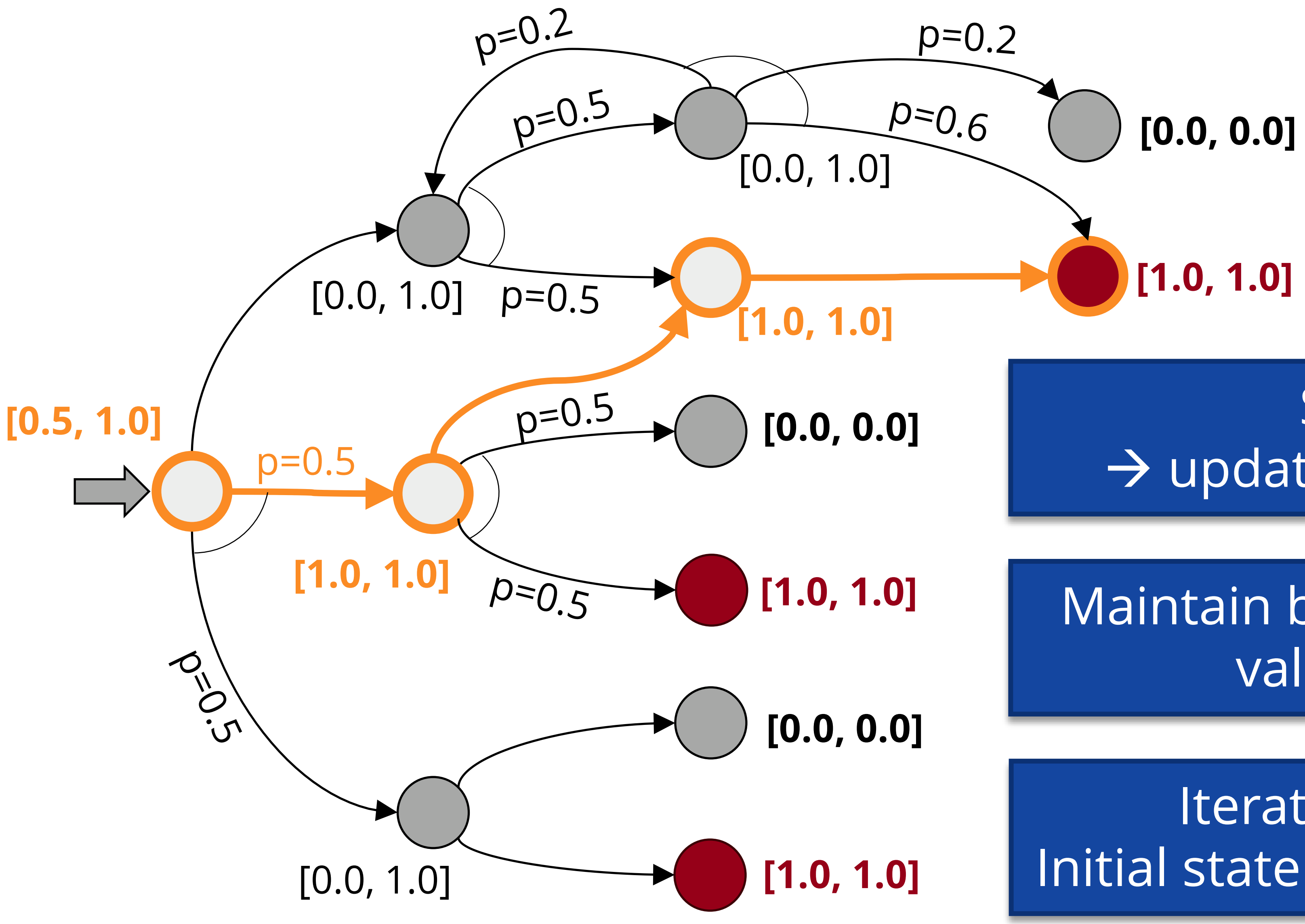


Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

Bounded Real-Time Dynamic Programming (BRTDP)

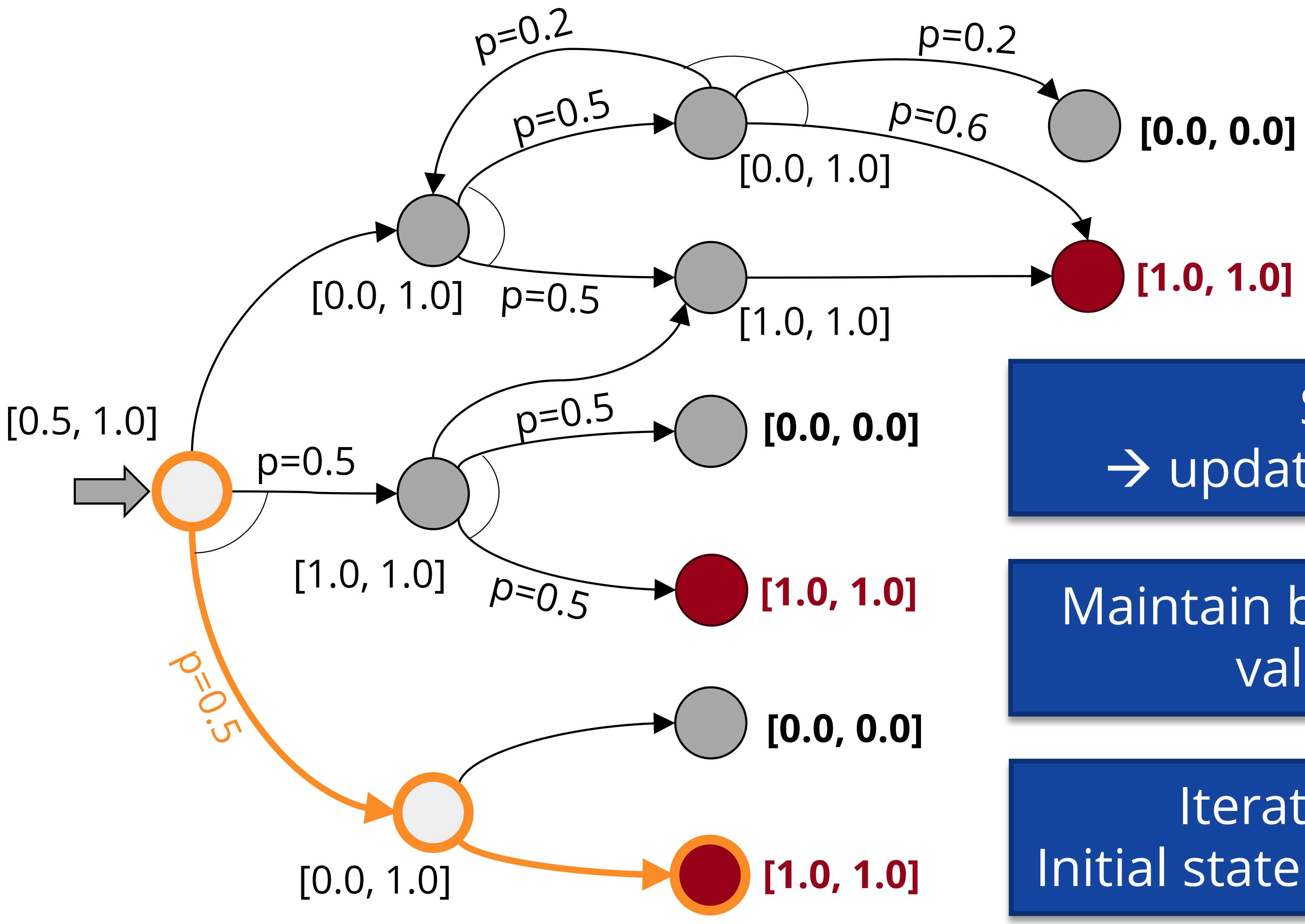


Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

Bounded Real-Time Dynamic Programming (BRTDP)

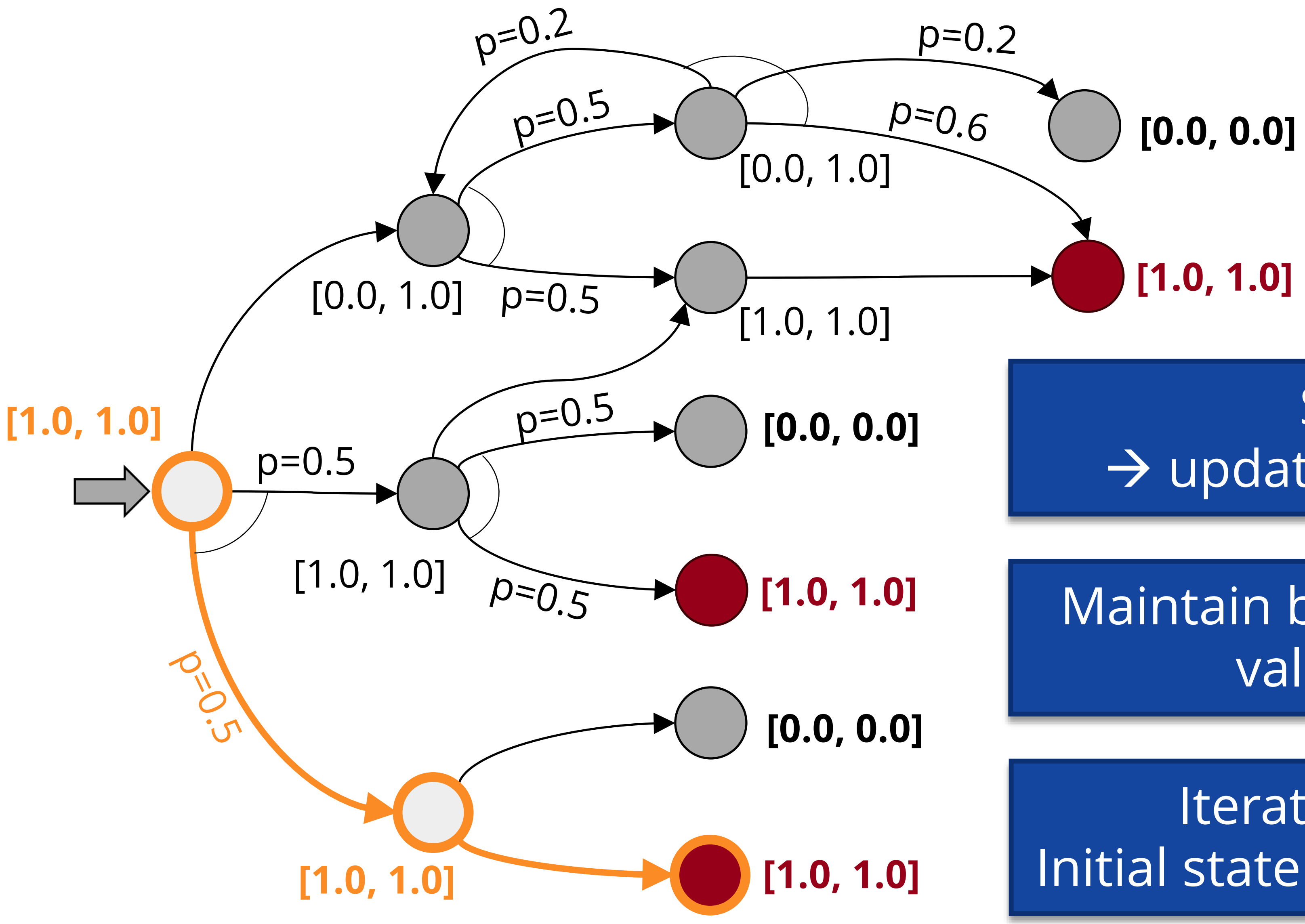


Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

Bounded Real-Time Dynamic Programming (BRTDP)

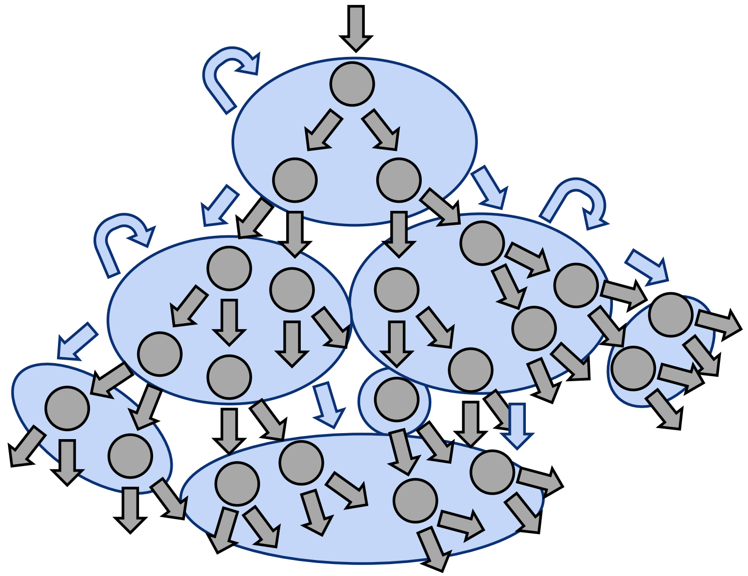


Simulate traces
→ update only simulated states

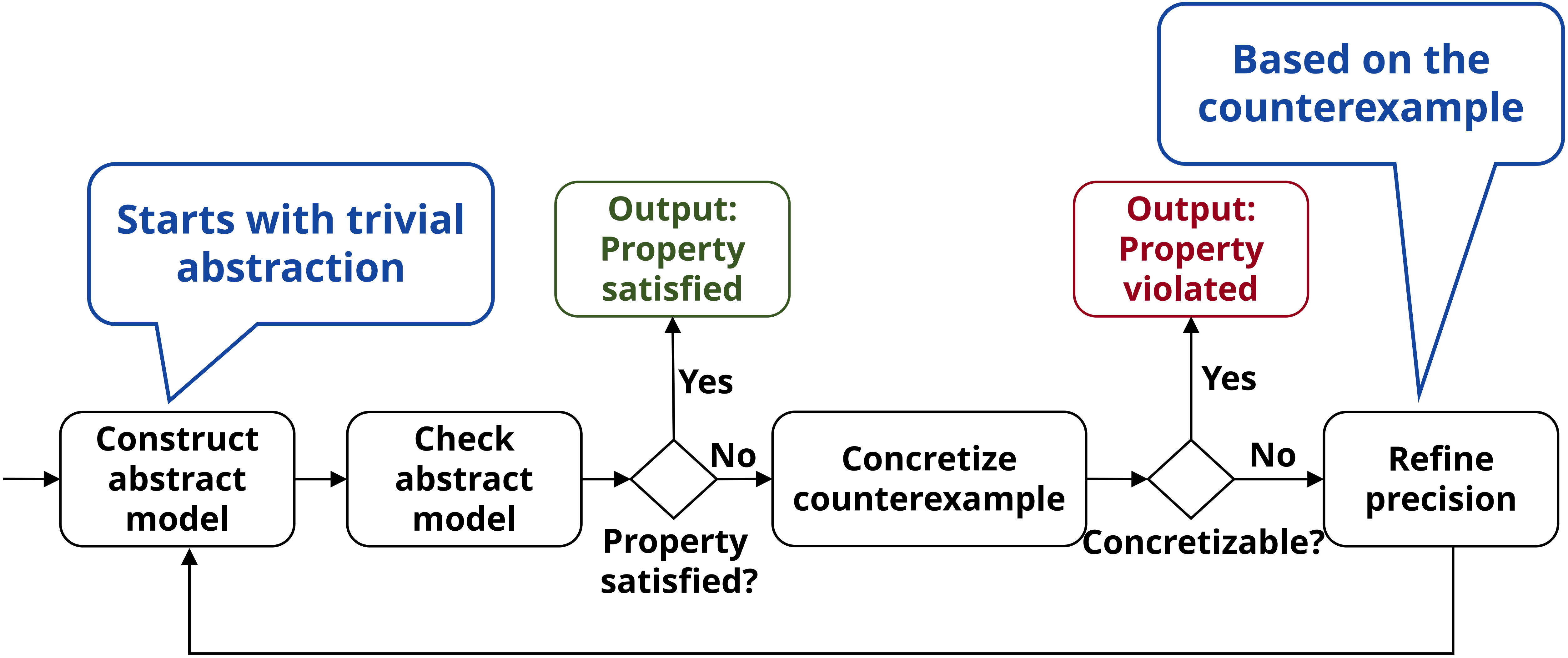
Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

Lazy abstraction for MDPs

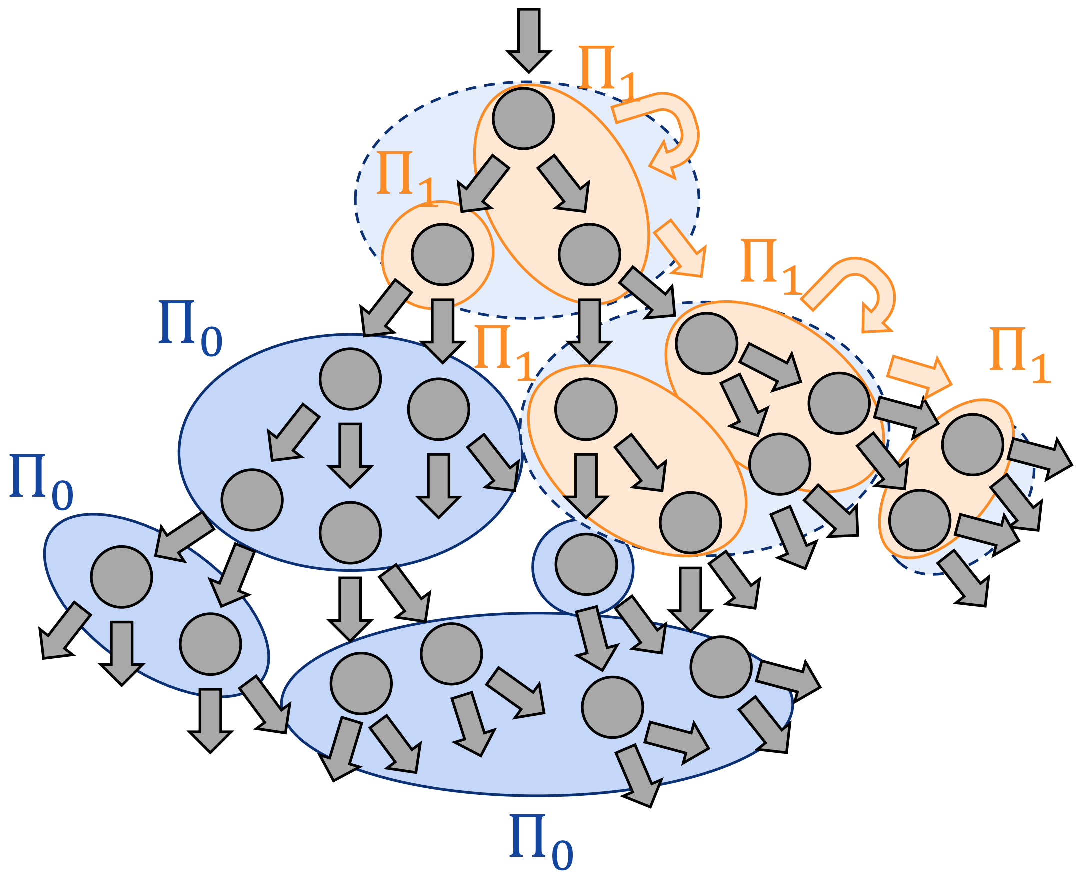


CounterExample-Guided Abstraction Refinement



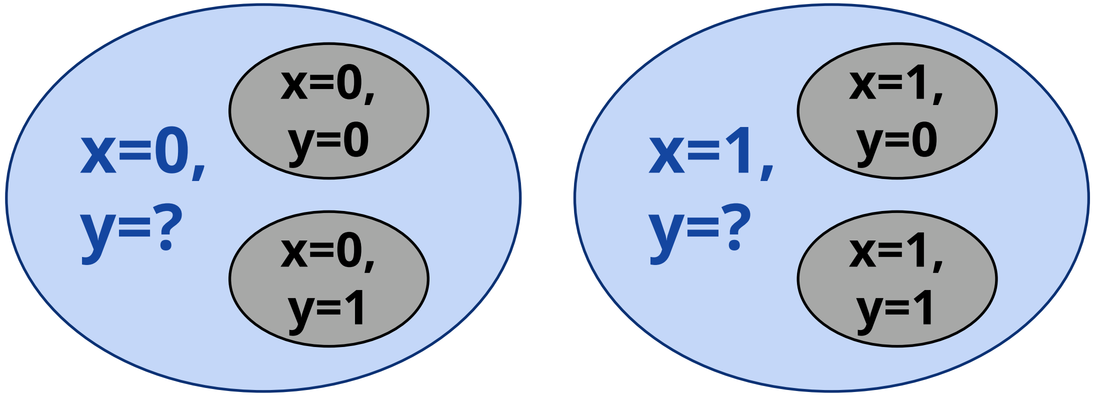
Lazy abstraction

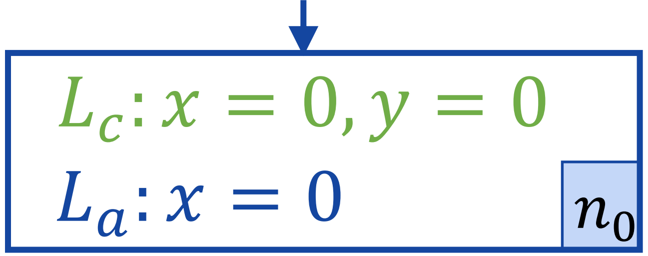
- Builds on the idea of **CEGAR**
- **Merged** abstract exploration and refinement
- **Precision** is **local to each node** in the abstract state graph
- **Refinement** is performed **locally** on the required nodes
- **Better suited for** combination with **BRTDP** than non-lazy probabilistic CEGAR approaches



Lazy abstraction for MDPs

- Several different lazy abstraction implementations (BLAST, Impact, etc.)
  → We use an **Adaptive Simulation Graph**-based version
- Abstract model: **Probabilistic Adaptive Simulation Graph** (PASG)
- **Domain-agnostic** in general
- Currently implemented with **Explicit Value Abstraction**:
  Some variables are tracked exactly, others are unknown





$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$

$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \wedge y' := y), 0.2 : (x' := x \wedge y' := y)$

$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \wedge x' := 1)$

$\mathbf{c_3} : [x == 2 \wedge y == 2]\ 1.0 : (y' := 3 \wedge x' := x)$

Probabilistic Adaptive Simulation Graph (ASG):

- Nodes are labeled by a **concrete state**
- and an **abstract state (describing a set of concrete states)** that contains it
- The **concrete state** represents all states in the **abstract state** w.r.t. available “behaviors” (action sequences)

Initial node:

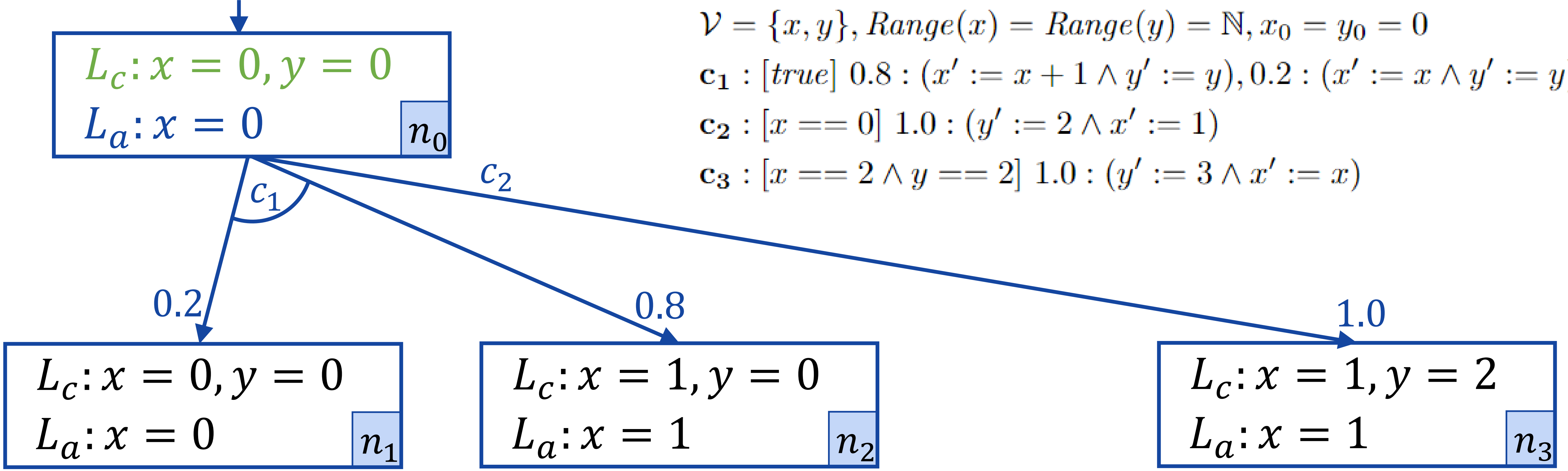
- **concrete label** is the **concrete initial state**
- **abstract label** is **as coarse as possible**

$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$

$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \wedge y' := y), 0.2 : (x' := x \wedge y' := y)$

$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \wedge x' := 1)$

$\mathbf{c_3} : [x == 2 \wedge y == 2]\ 1.0 : (y' := 3 \wedge x' := x)$

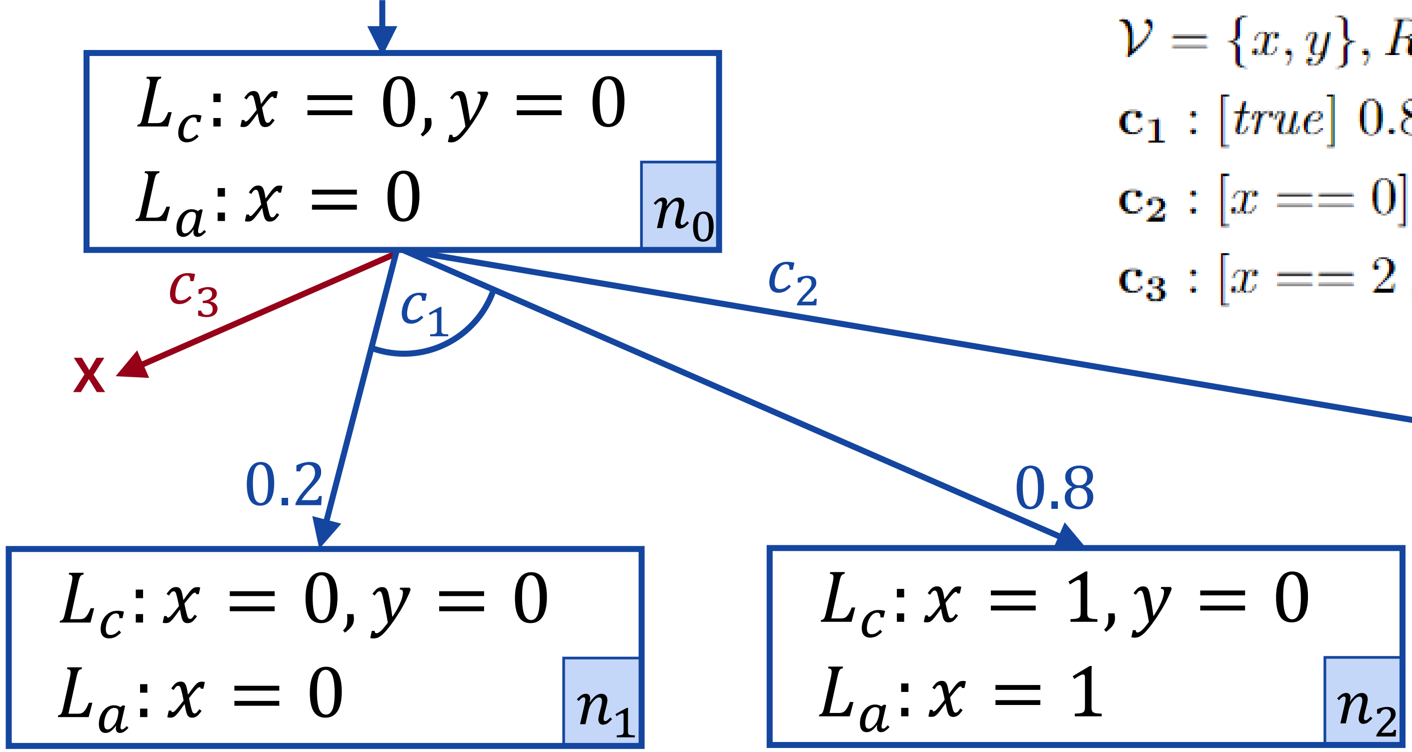


Probabilistic Adaptive Simulation Graph (ASG):

- Nodes are labeled by a **concrete state**
- and an **abstract state (describing a set of concrete states)** that contains it
- The **concrete state** represents all states in the **abstract state** w.r.t. available “behaviors” (action sequences)

Expansion:

- Select an action enabled in the **concrete state**
- Compute the image of the **concrete state**
- Overapproximate the **image** of the **abstract state**



$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$

$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \wedge y' := y), 0.2 : (x' := x \wedge y' := y)$

$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \wedge x' := 1)$

$\mathbf{c_3} : [x == 2 \wedge y == 2]\ 1.0 : (y' := 3 \wedge x' := x)$

If an action is **not enabled** in any part of the **abstract state**, it is ignored

Expansion:

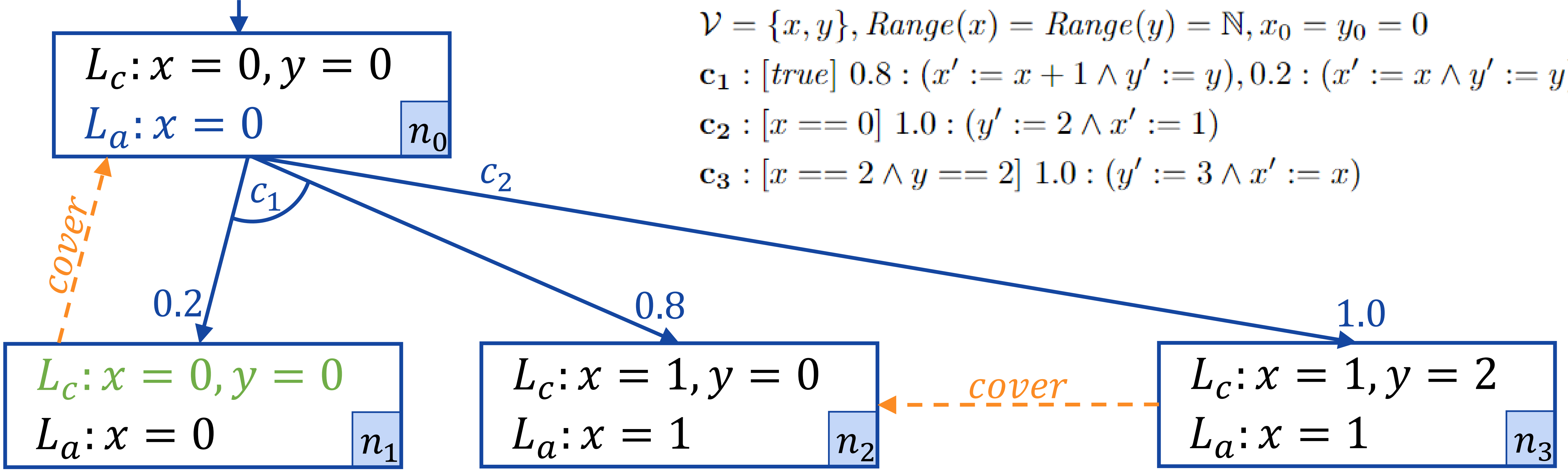
- Select an action enabled in the **concrete state**
- Compute the image of the **concrete state**
- Overapproximate the **image** of the **abstract state**

$$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$$

$$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \wedge y' := y), 0.2 : (x' := x \wedge y' := y)$$

$$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \wedge x' := 1)$$

$$\mathbf{c_3} : [x == 2 \wedge y == 2]\ 1.0 : (y' := 3 \wedge x' := x)$$



Covering:

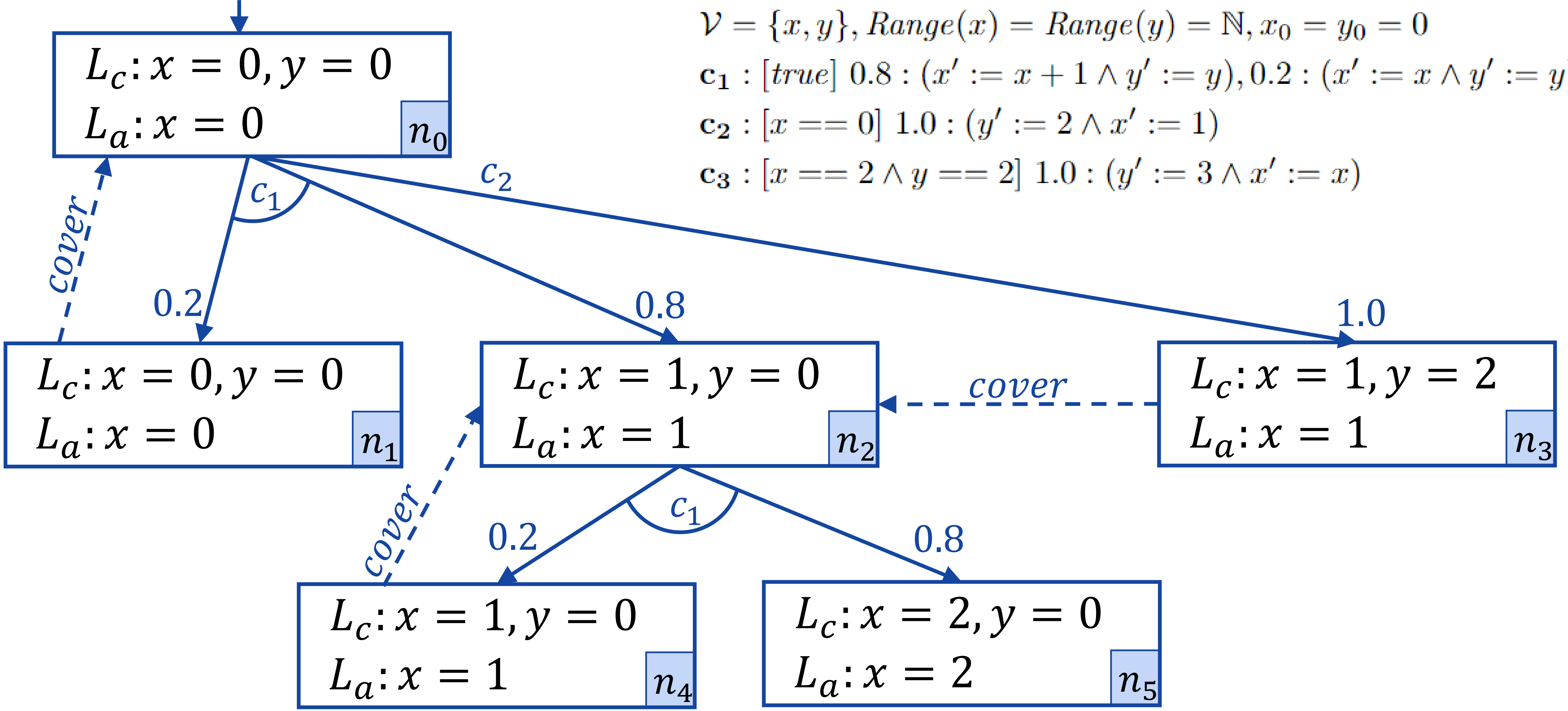
- If the new **concrete state** after expansion is already contained in another **abstract state**
- A **cover edge** is created
- Expansion of the covered node can be **skipped**

$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$

$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \wedge y' := y), 0.2 : (x' := x \wedge y' := y)$

$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \wedge x' := 1)$

$\mathbf{c_3} : [x == 2 \wedge y == 2]\ 1.0 : (y' := 3 \wedge x' := x)$

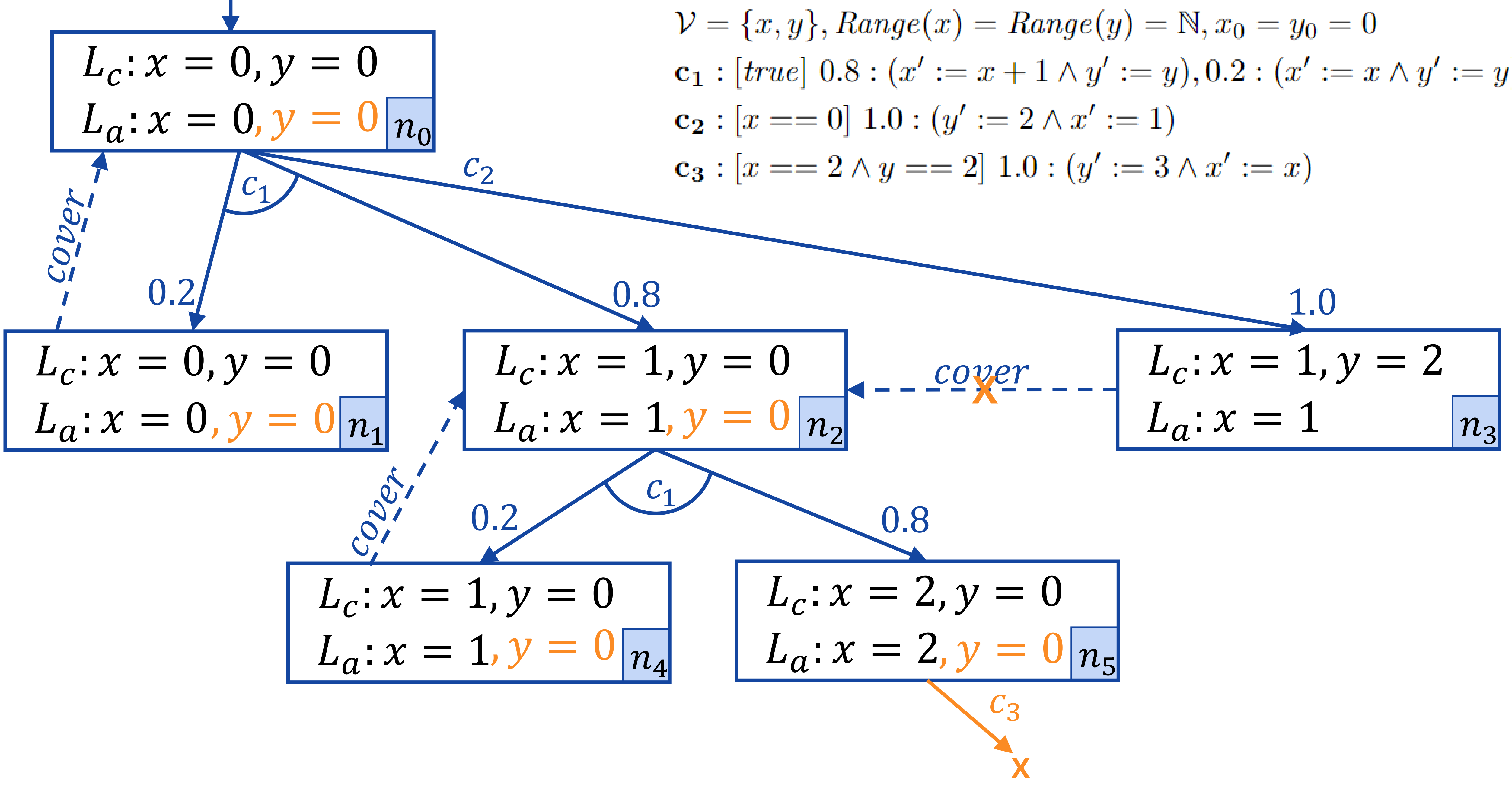


$\mathcal{V} = \{x, y\}, Range(x) = Range(y) = \mathbb{N}, x_0 = y_0 = 0$

$\mathbf{c_1} : [true]\ 0.8 : (x' := x + 1 \wedge y' := y), 0.2 : (x' := x \wedge y' := y)$

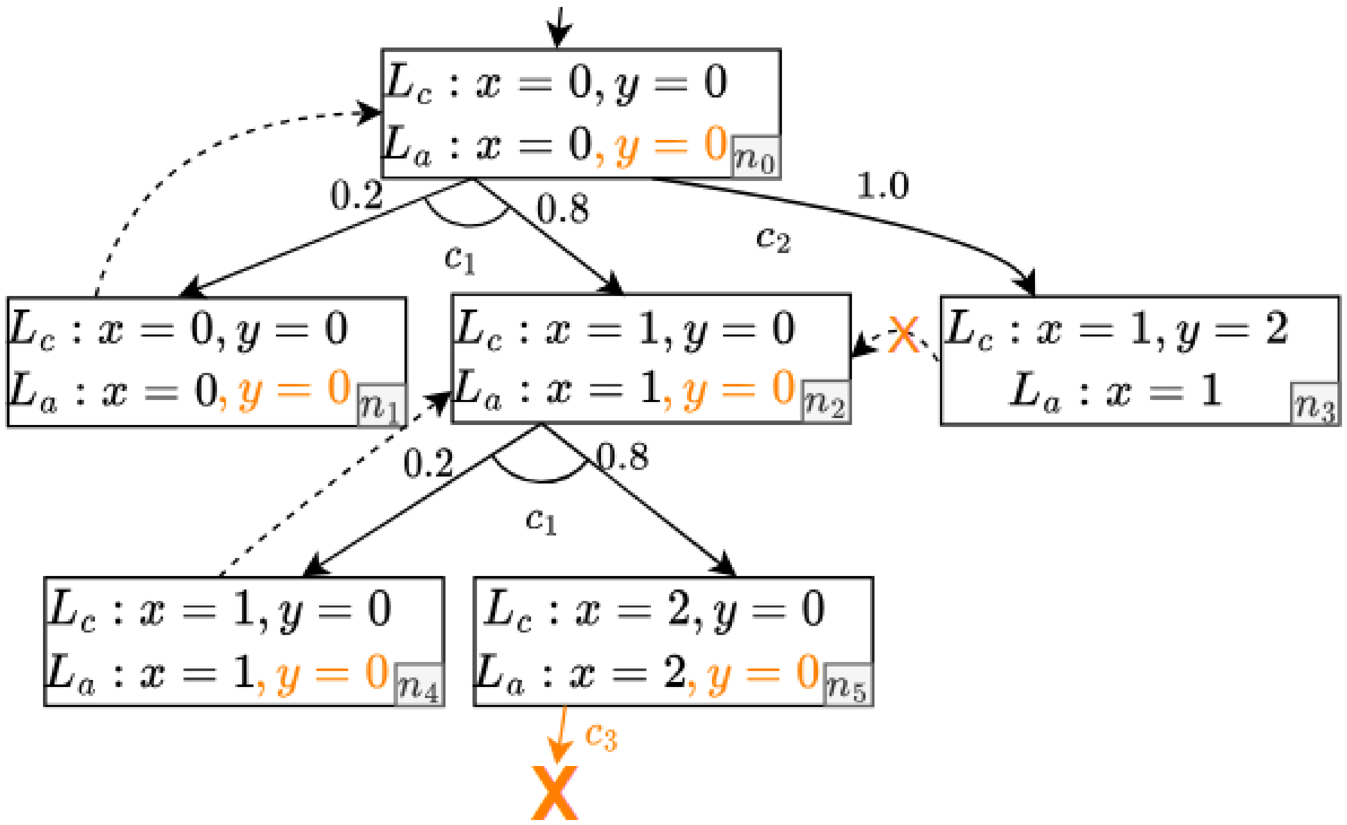
$\mathbf{c_2} : [x == 0]\ 1.0 : (y' := 2 \wedge x' := 1)$

$\mathbf{c_3} : [x == 2 \wedge y == 2]\ 1.0 : (y' := 3 \wedge x' := x)$



PASG versions

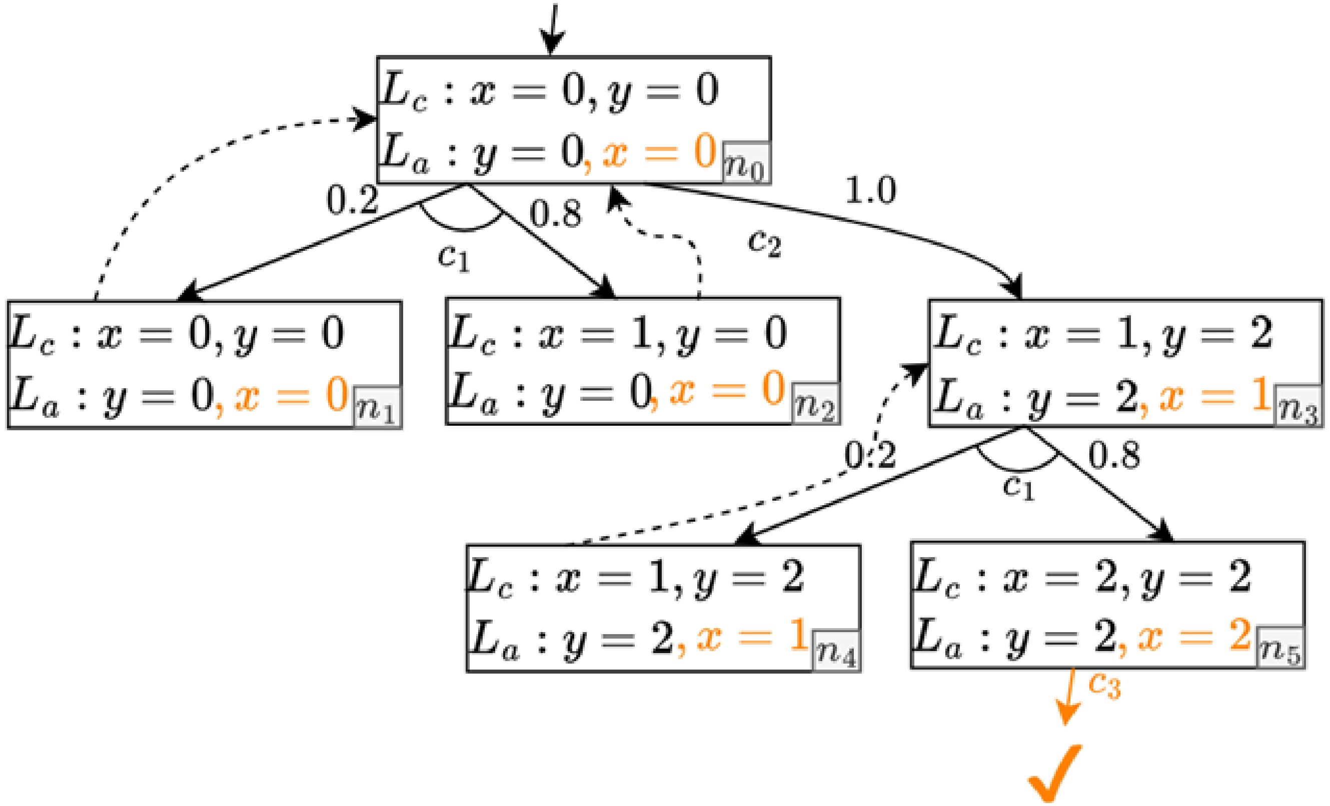
Upper-cover:
- Direct adaptation of the original ASG for MDPs
- Action that might be **enabled** somewhere in the abstract label must be enabled in the concrete
- Upper approximation



PASG versions

Lower-cover:

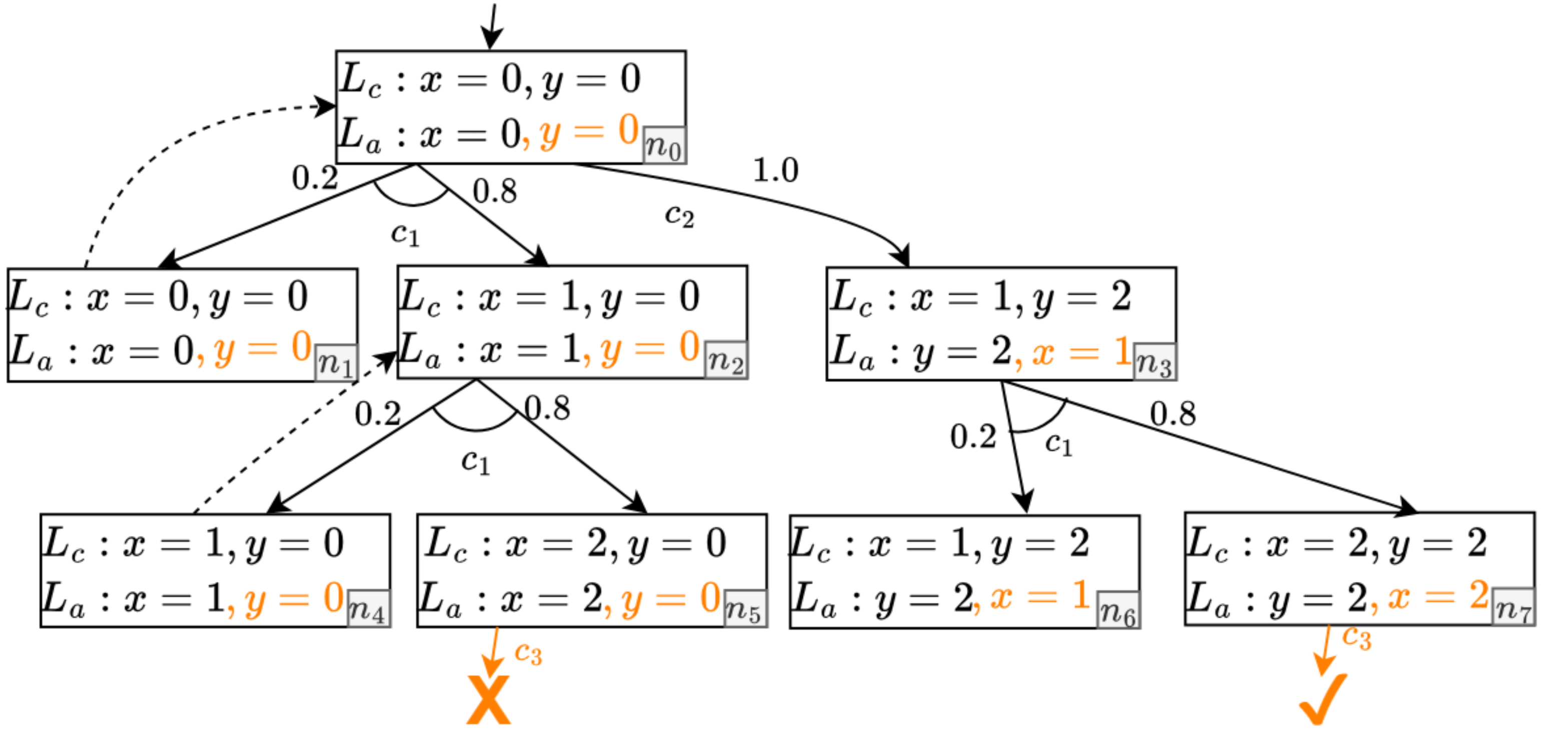
- Inverted representativity requirement
- Action **disabled** somewhere in the abstract label must be disabled in the concrete
- Lower approximation



PASG versions

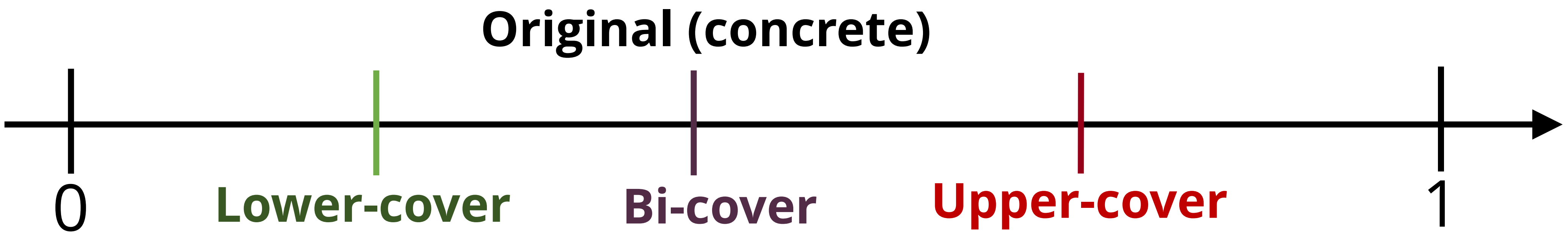
Bi-cover:

- Combines the upper- and lower-cover constraints
- Provides exact numerical results
- Resulting *value* is independent of the order of exploration

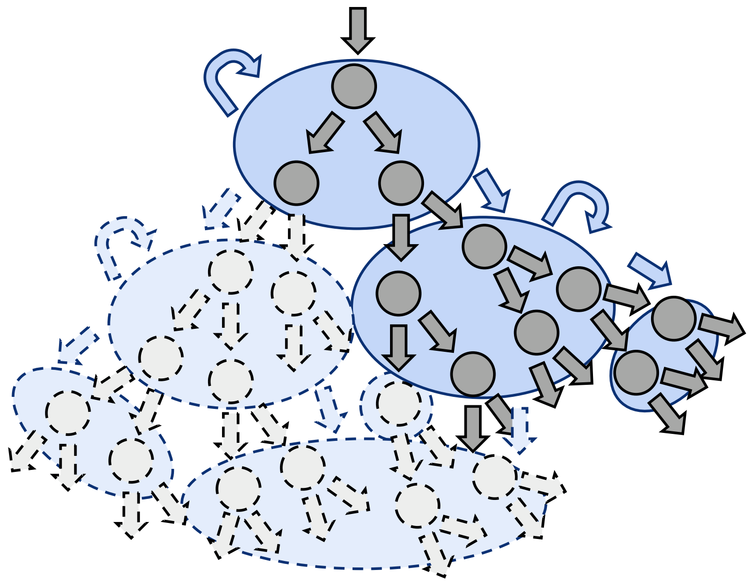


Quantitative Analysis – Full Exploration

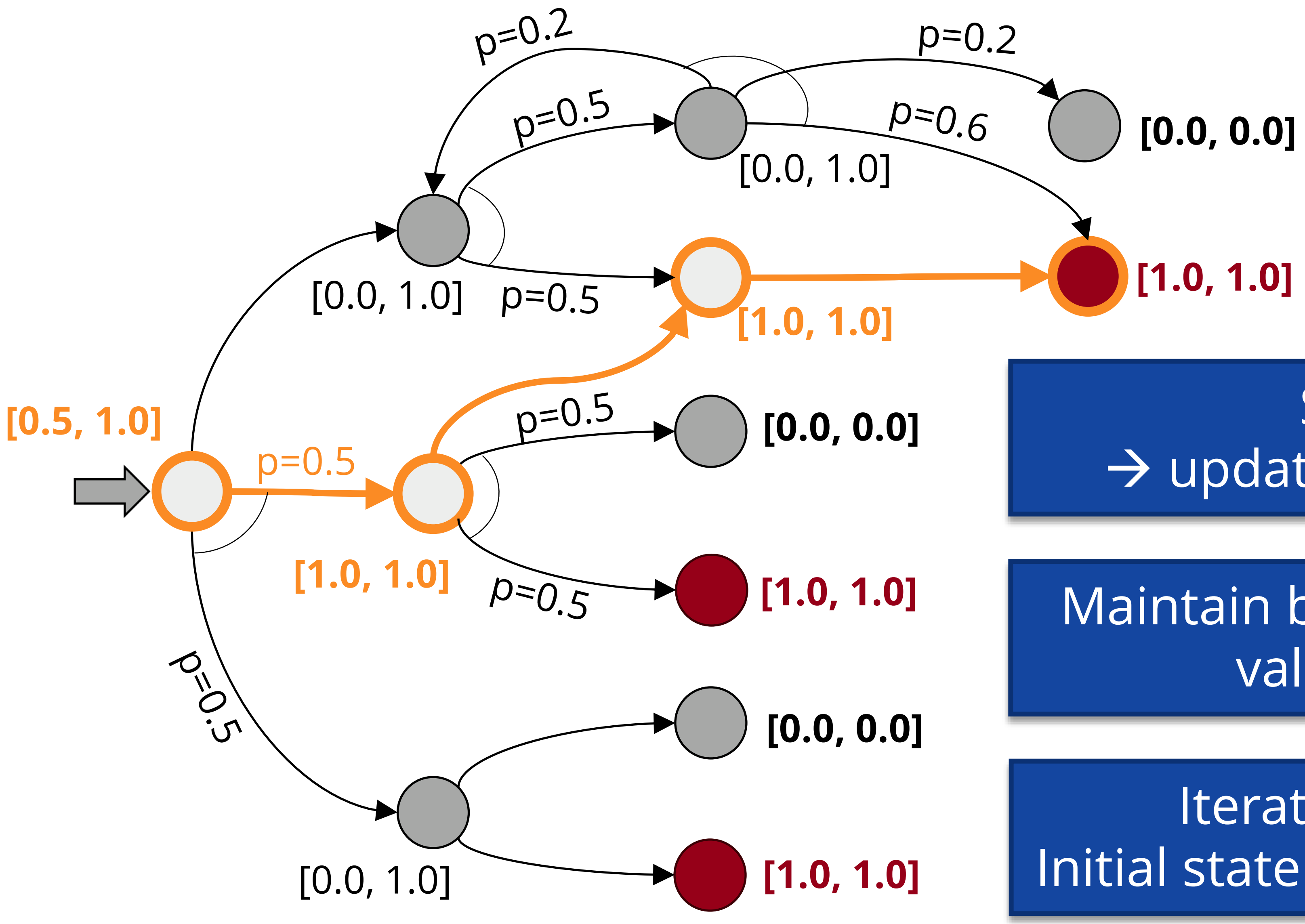
- Construct full PASG → Analyze it as an MDP
- Cover edges are deterministic actions
- Any MDP analysis algorithm can be applied (value iteration variants, policy iteration, linear programming, …)
- Provable guarantees for the target probability:



Lazy abstraction + BRTDP



BRTDP reminder



Simulate traces
→ update only simulated states

Maintain both a *lower* and an *upper* value approximation

Iterate until convergence:
Initial state has small enough interval

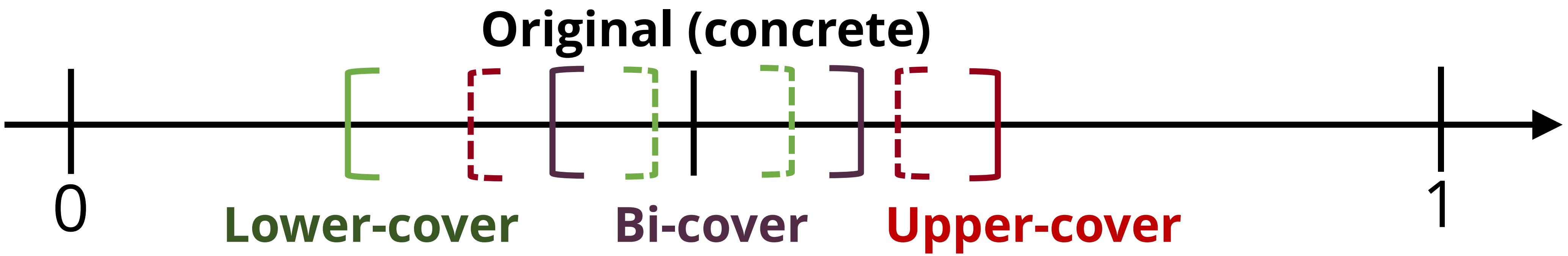
Quantitative Analysis – On-the-fly

- Uses **BRTDP** for analysis
- **Merges** PASG construction and numeric computations
- PASG nodes are constructed during trace simulation

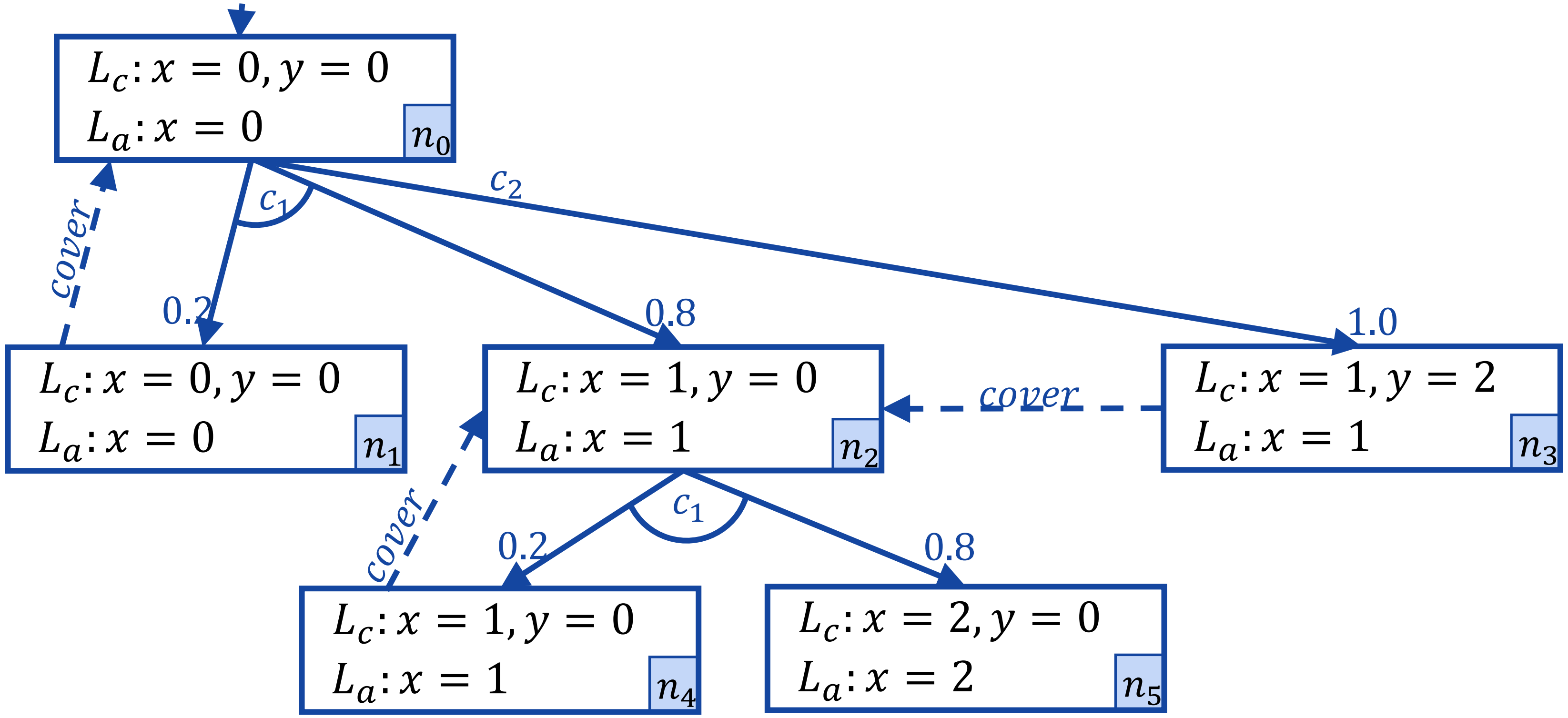
Less states explored → **Less inconsistencies** → **Coarser abstract labels** → **Smaller abstract state space**

Quantitative Analysis – On-the-fly

- Provable guarantees:
- Convergence for finite state spaces: PASG is finished after a finite number of traces + BRTDP convergence results applied to the finished PASG
- Guarantees for the target probability:



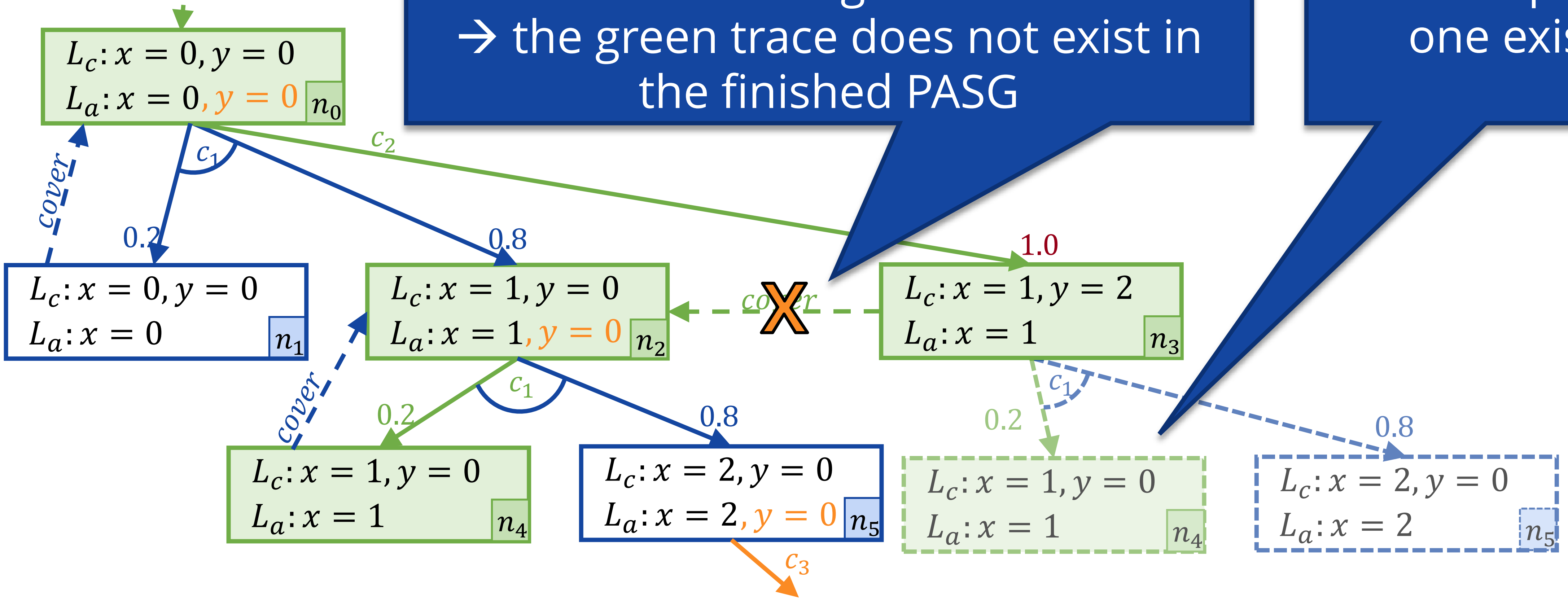
Correctness of the on-the-fly analysis



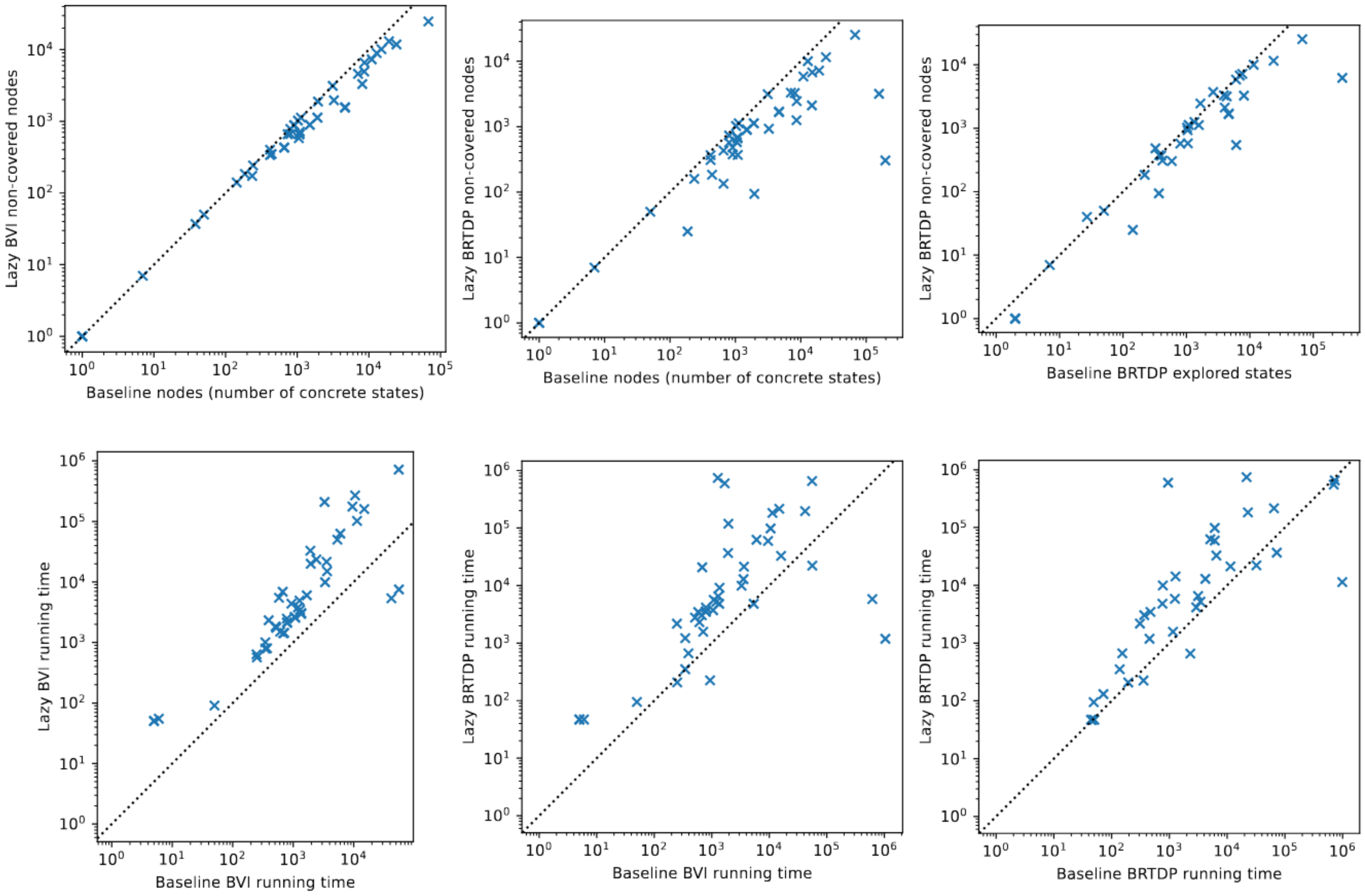
Correctness of the on-the-fly analysis

Refinement when n_5 is expanded
→ the cover edge is removed
→ the green trace does not exist in the finished PASG

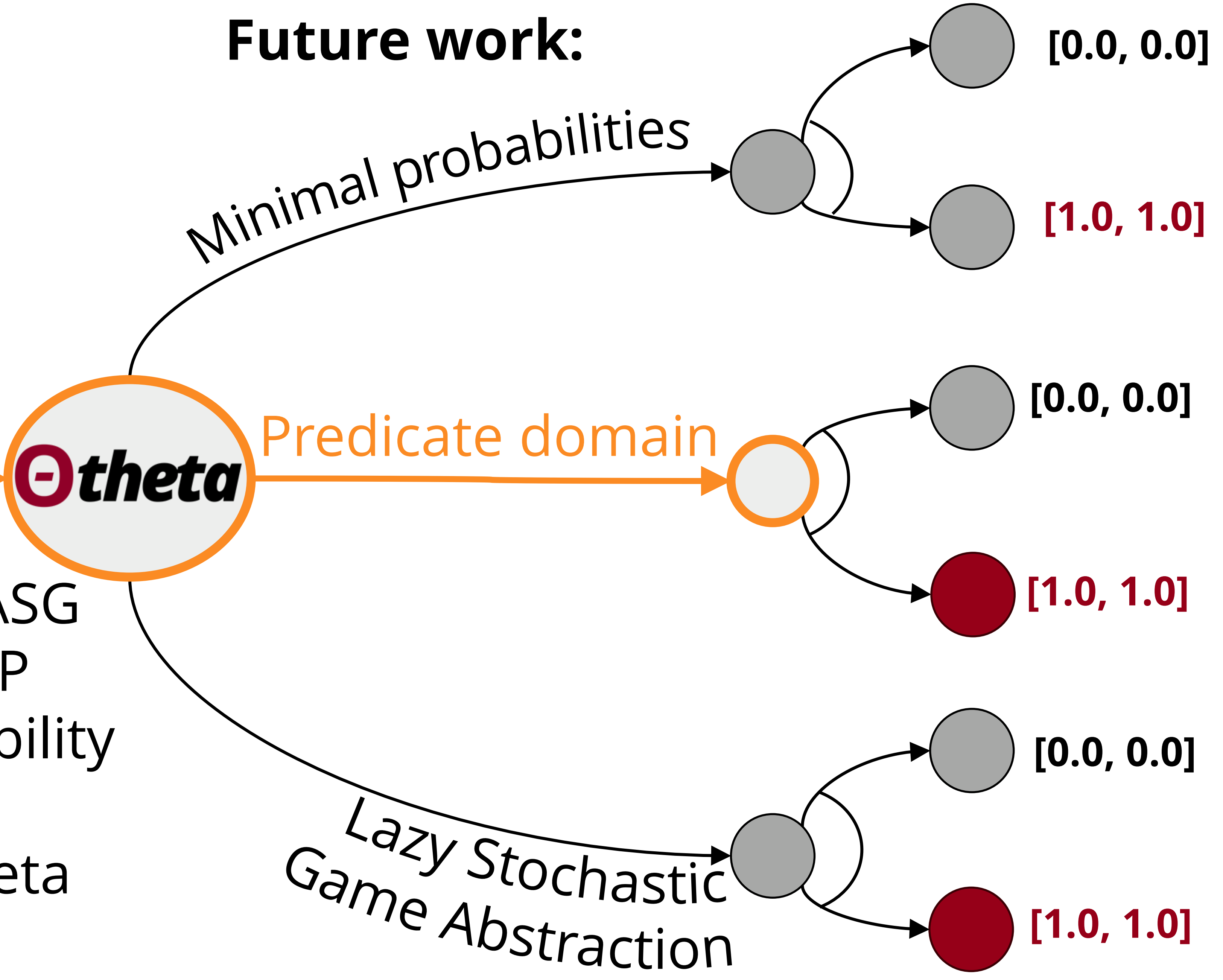
But an equivalent one exists!



(Preliminary) Measurements on the QComp benchmarks



Future work:



Current state:

- Upper/lower/bi-cover PASG
- Full construction / BRTDP
- Only for maximal probability
- Explicit Value Domain

→ Implemented in the Theta model checker

Thank you for your attention