Game-based Abstraction and Controller Synthesis for Probabilistic Hybrid Systems

Ernst Moritz Hahn, Saarland University

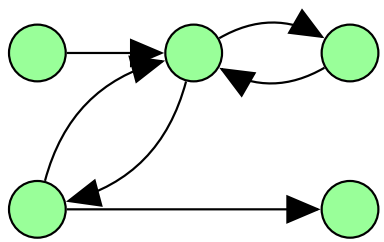
joint work with

Gethin Norman, University of Glasgow, UK
David Parker, University of Oxford, UK
Björn Wachter, University of Oxford, UK
Lijun Zhang, Technical University of Denmark

ROCKS, October 5th, 2011

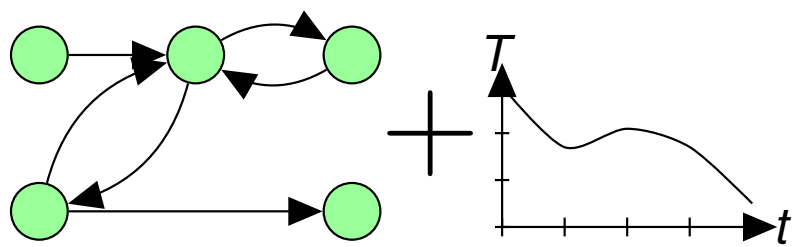
Motivation (1/2)

- many systems have discrete



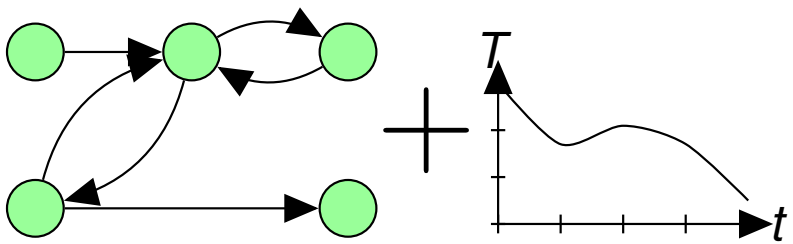
Motivation (1/2)

- many systems have discrete
- and continuous behaviour



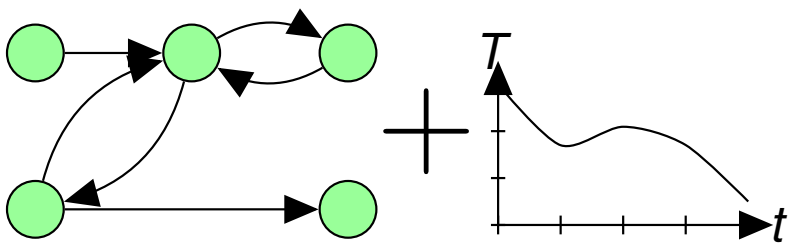
Motivation (1/2)

- many systems have discrete
- and continuous behaviour



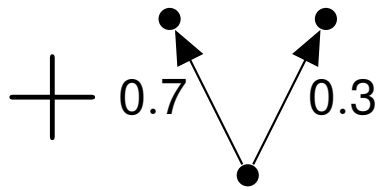
Motivation (1/2)

- many systems have discrete
- and continuous behaviour
- $\Rightarrow$ hybrid automata (handled by PHAVer, etc.)



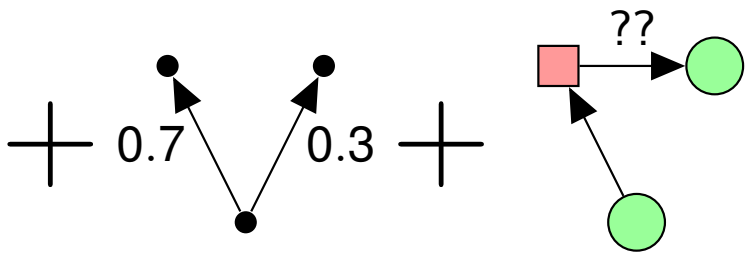
Motivation (2/2)

- often also probabilistic aspects (failures, etc.)



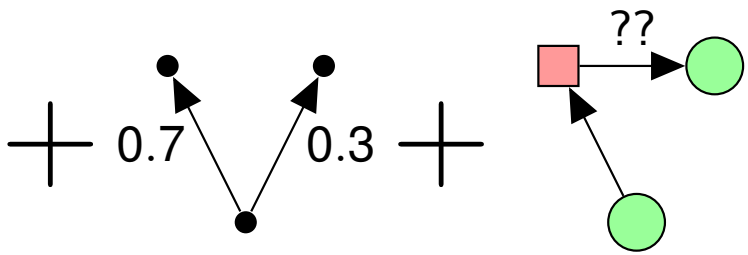
Motivation (2/2)

- often also probabilistic aspects (failures, etc.)
- only partially controllable

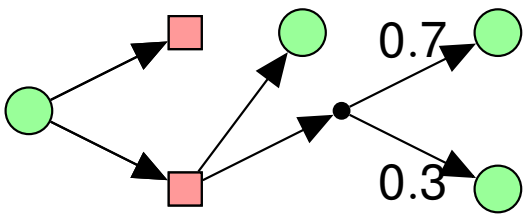


Motivation (2/2)

- often also probabilistic aspects (failures, etc.)
- only partially controllable

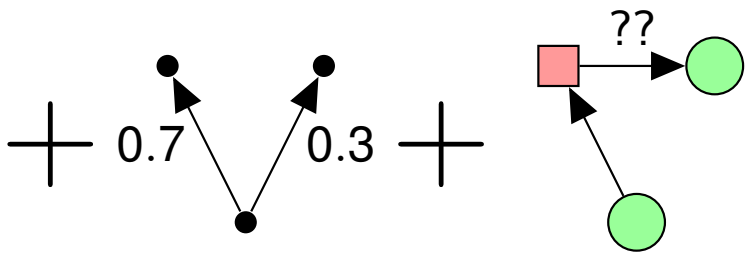


- $\Rightarrow$ game-based semantics for probabilistic hybrid systems

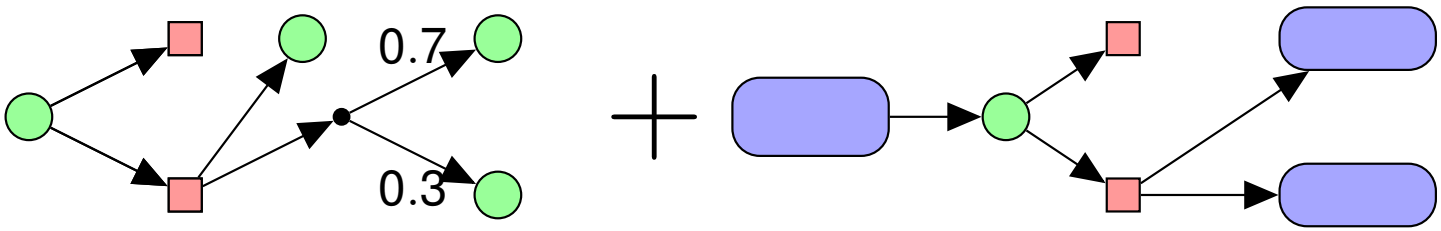


Motivation (2/2)

- often also probabilistic aspects (failures, etc.)
- only partially controllable

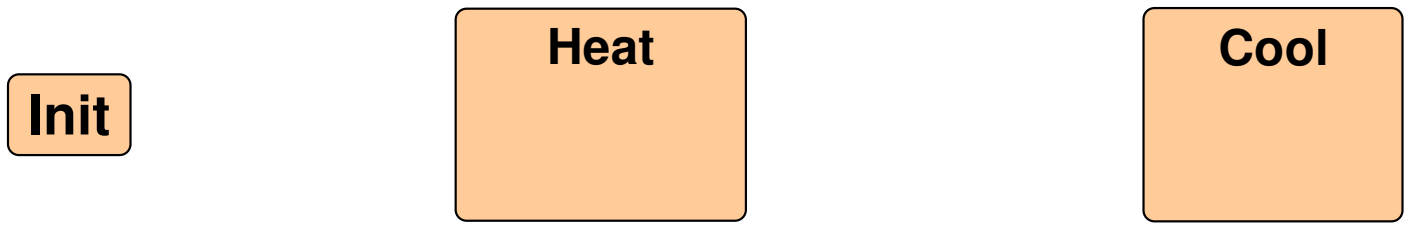


- $\Rightarrow$ game-based semantics for probabilistic hybrid systems
- $\Rightarrow$ finite abstractions to compute results



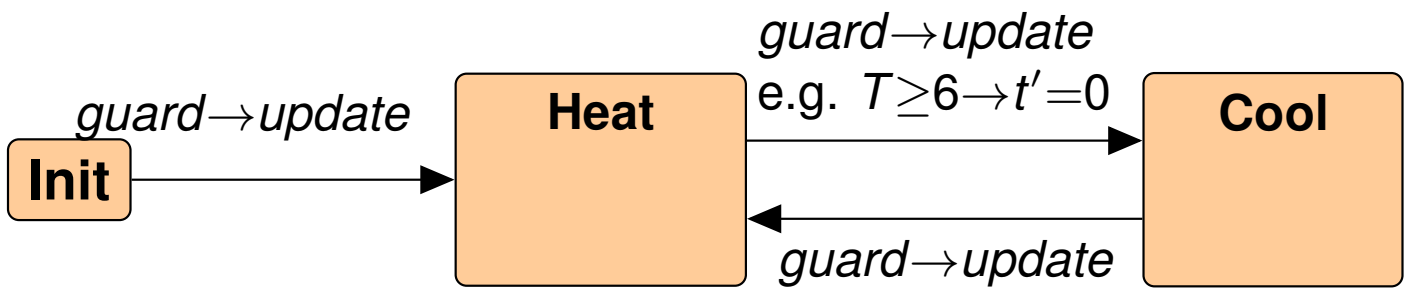
Hybrid Automata

- finite set of modes



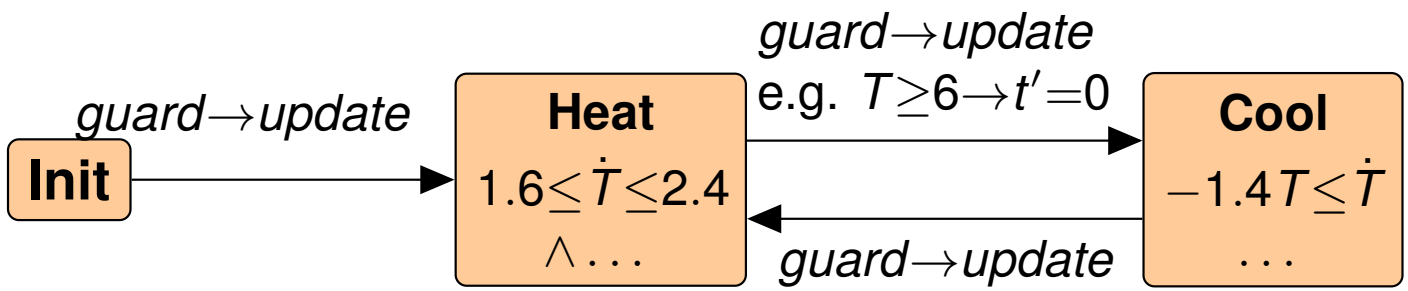
Hybrid Automata

- finite set of modes
- discrete behaviour, set of guarded commands



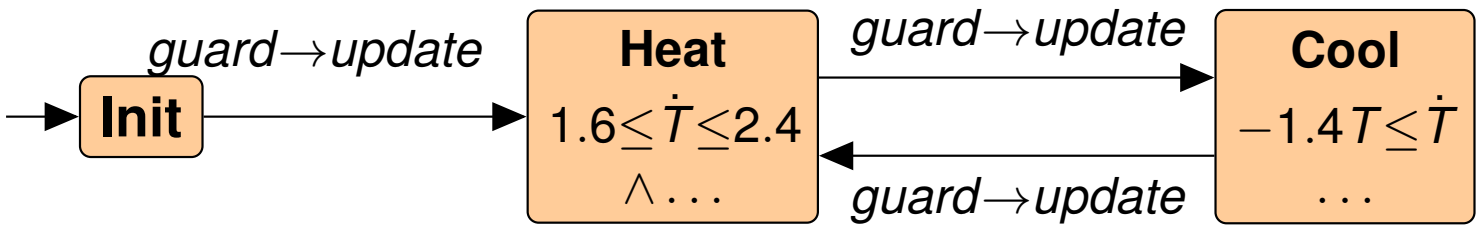
Hybrid Automata

- finite set of modes
- discrete behaviour, set of guarded commands
- continuous behaviour in modes



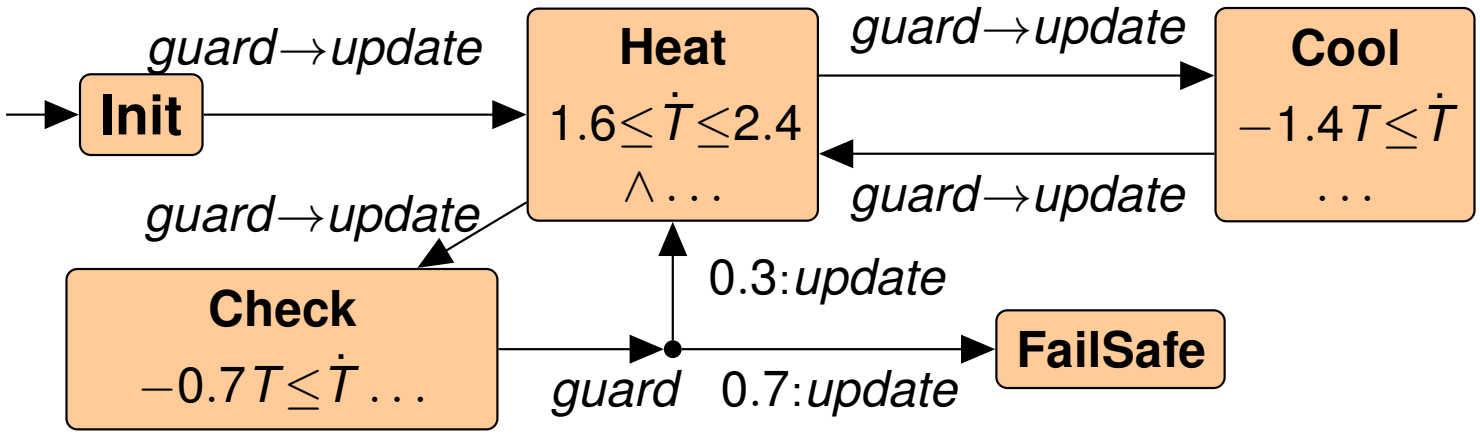
Probabilistic Hybrid Automata

- finite set of modes
- discrete behaviour, set of guarded commands
- continuous variables and behaviour in modes



Probabilistic Hybrid Automata

- finite set of modes
- discrete behaviour, set of guarded commands
- continuous variables and behaviour in modes



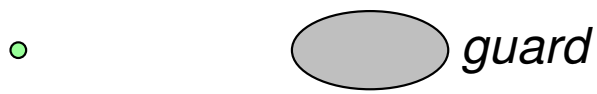
- probabilistic jump targets

Discrete Behaviour

- controller chooses command to execute

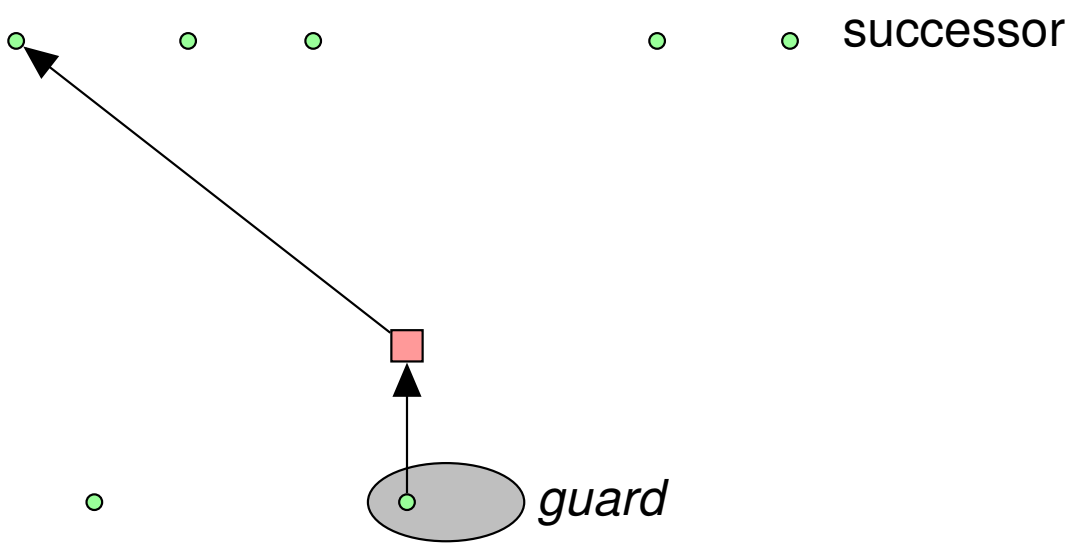
Discrete Behaviour

- controller chooses command to execute
- if state does not fulfill guard cannot execute



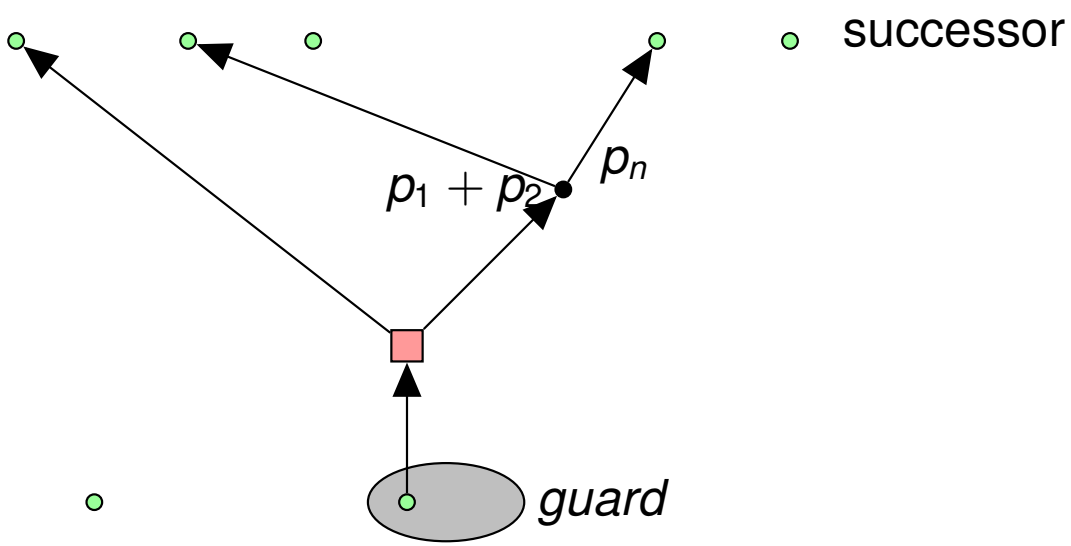
Discrete Behaviour

- controller chooses command to execute
- if state does not fulfill guard cannot execute
- else environment chooses exact effect



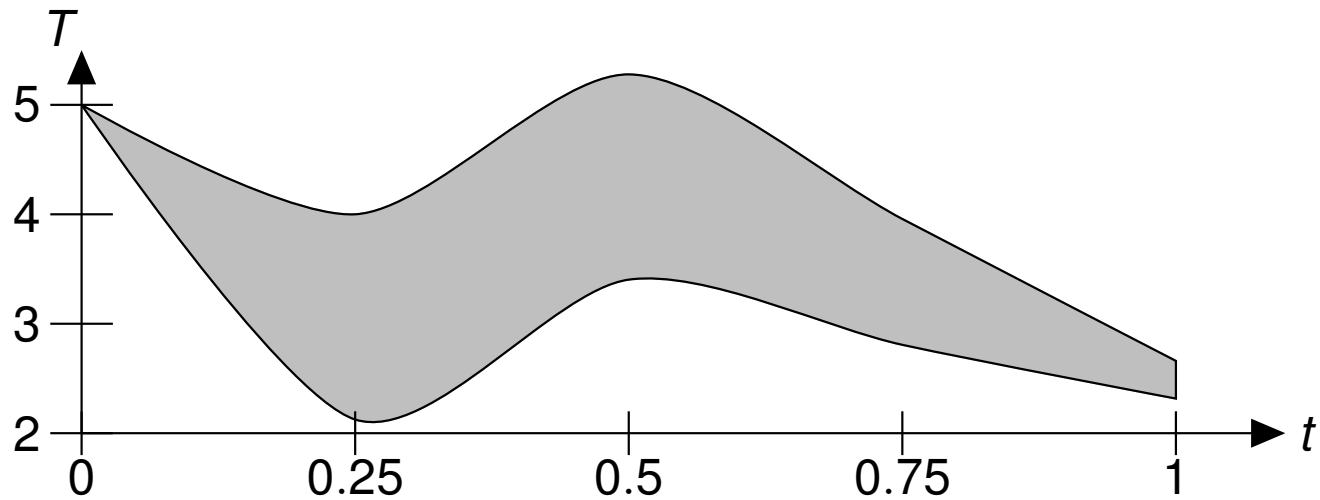
Discrete Behaviour

- controller chooses command to execute
- if state does not fulfill guard cannot execute
- else environment chooses exact effect
- possibly also probabilistic choice



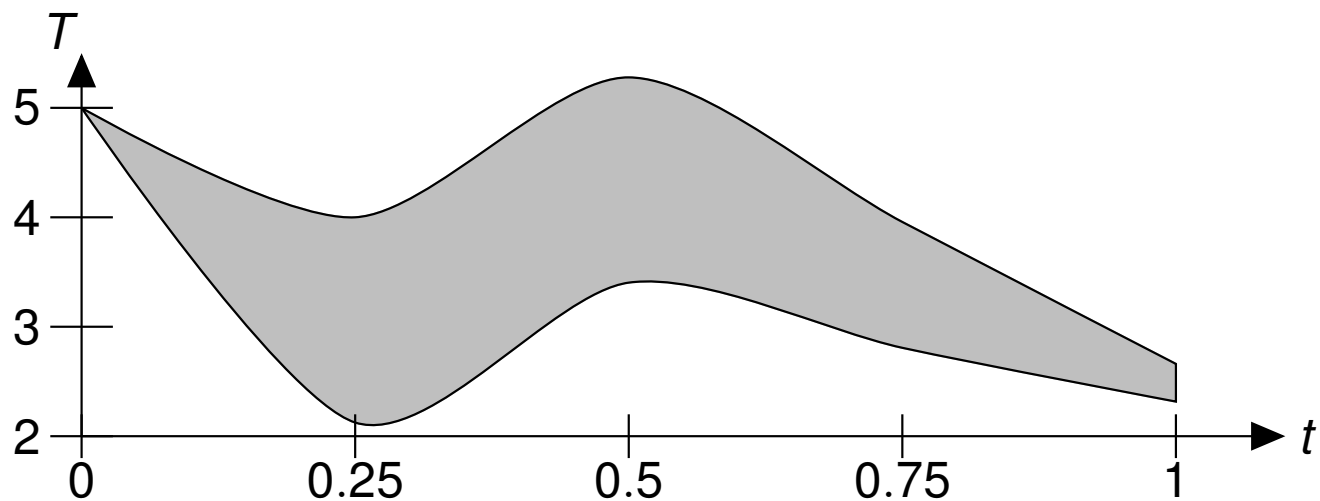
Continuous Behaviour

- described by **differential inequation**



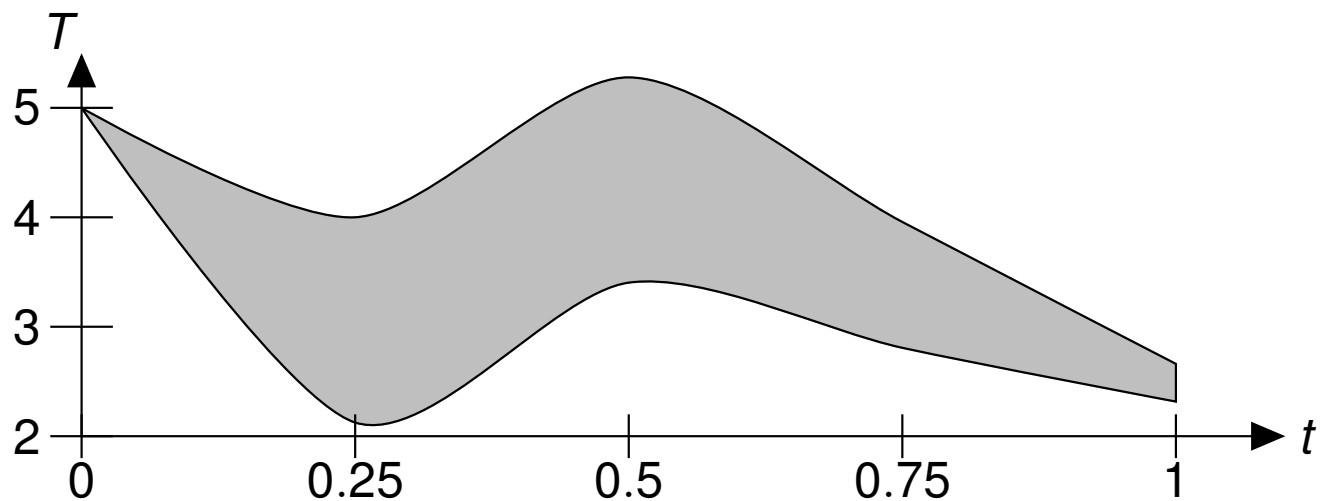
Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$



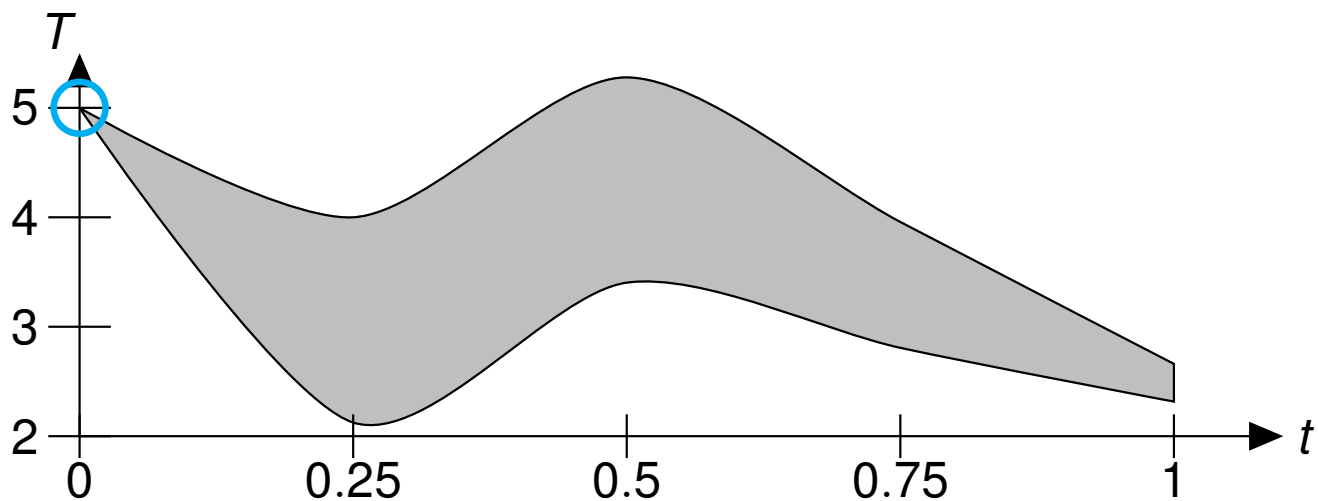
Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$
- function $x(\cdot)$ valid if



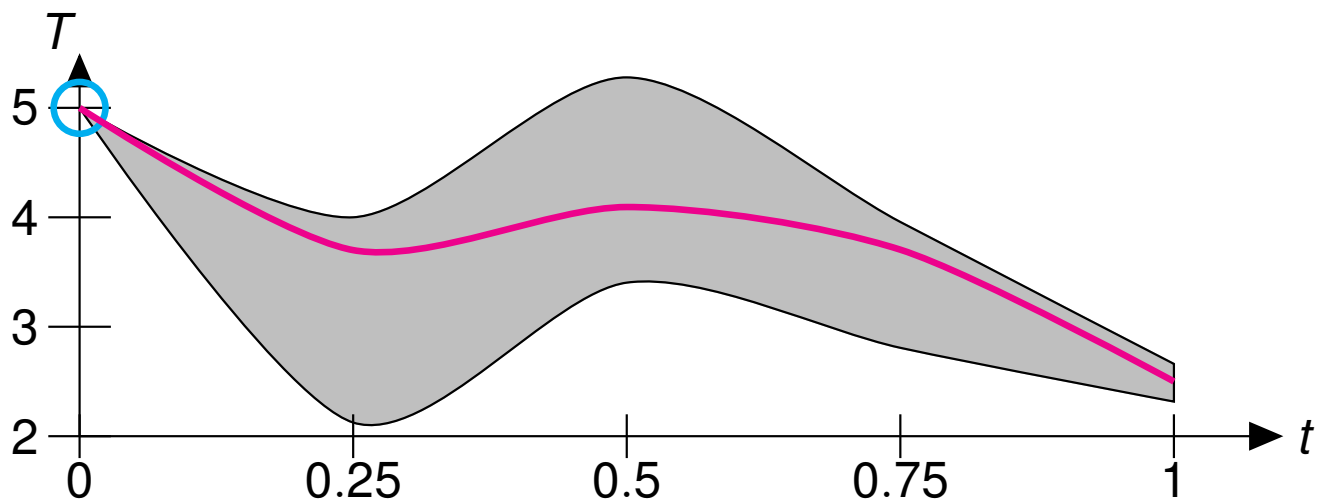
Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$
- function $x(\cdot)$ valid if
  - $x(0)$ describes an initial value



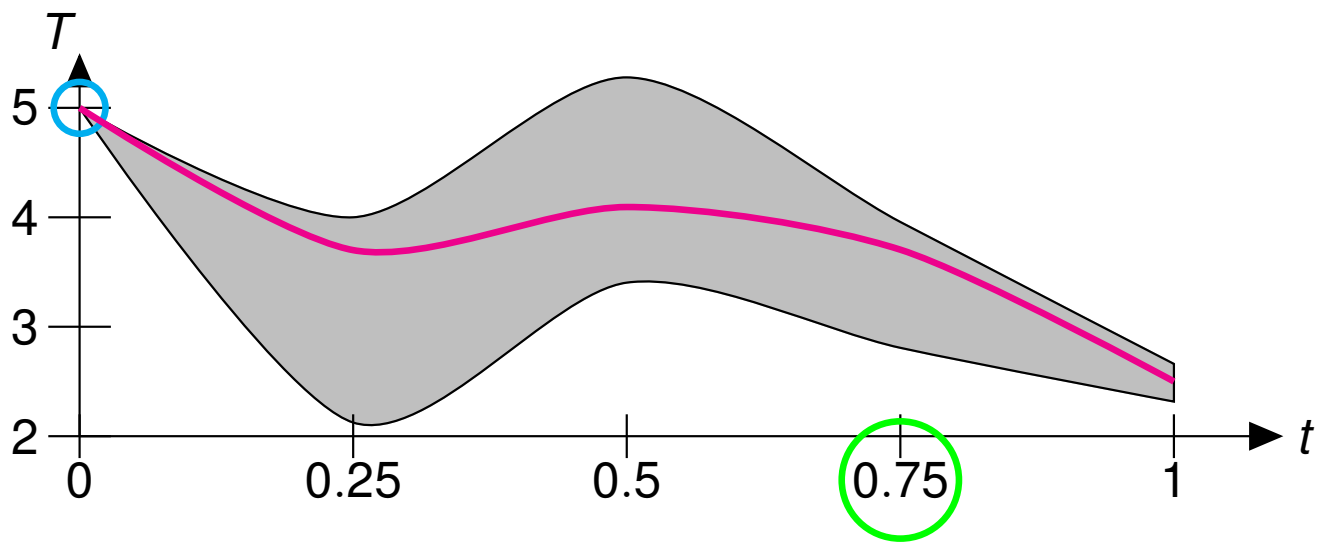
Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$
- function $x(\cdot)$ valid if
  - $x(0)$ describes an initial value
  - $x(\cdot)$ and $\dot{x}(\cdot)$ fulfill inequation



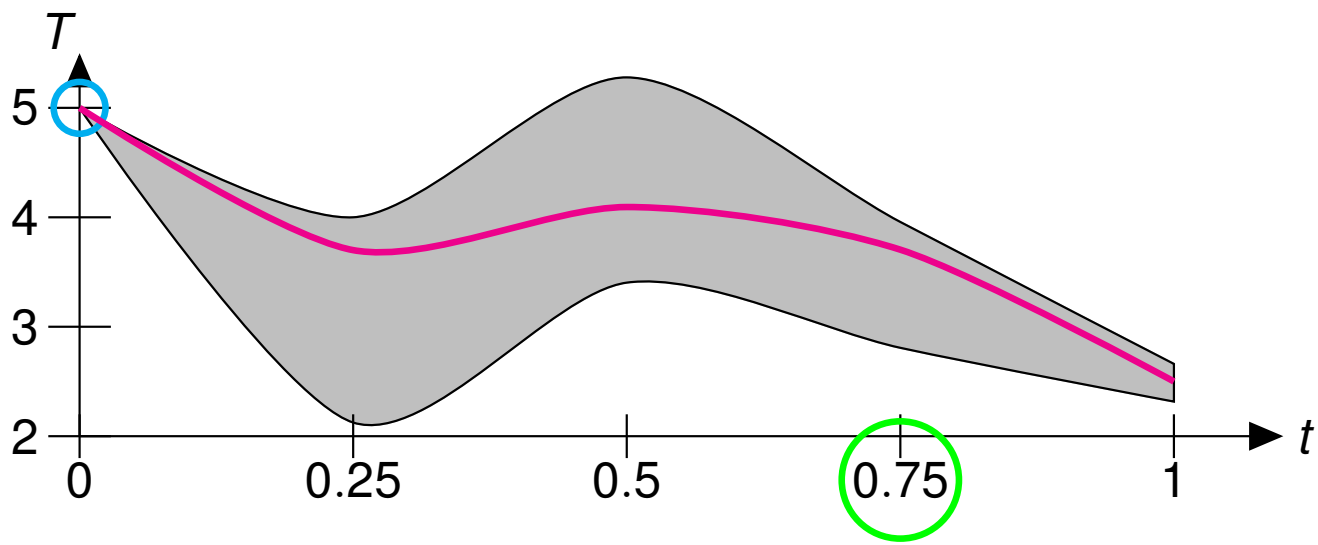
Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$
- function $x(\cdot)$ valid if
  - $x(0)$ describes an initial value
  - $x(\cdot)$ and $\dot{x}(\cdot)$ fulfill inequation
- controller chooses duration t



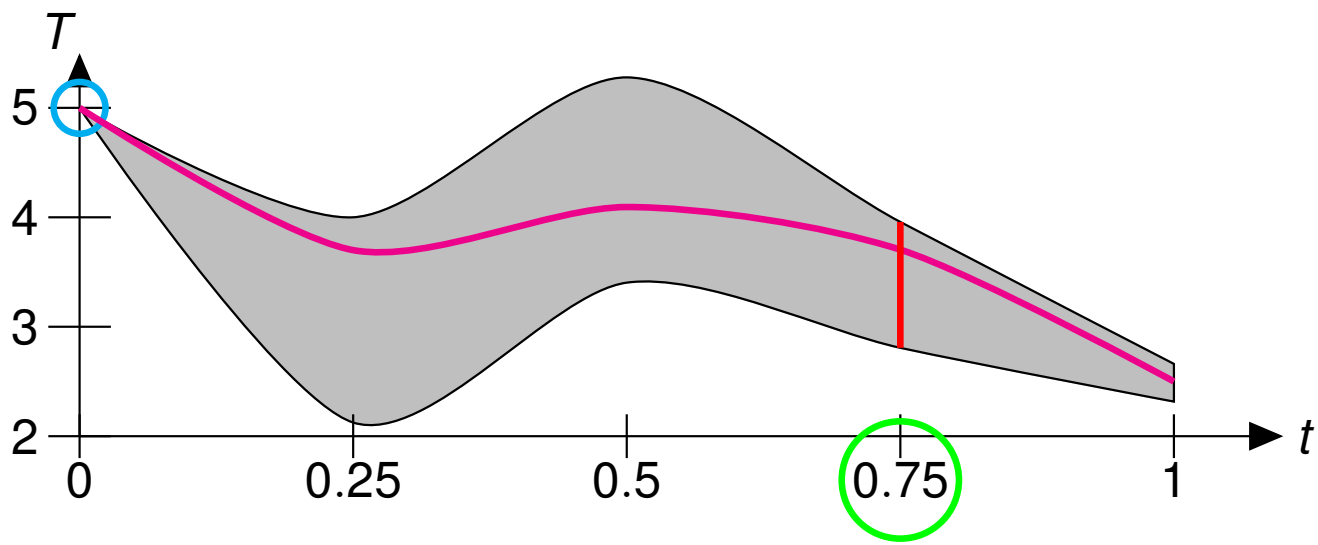
Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$
- function $x(\cdot)$ valid if
  - $x(0)$ describes an initial value
  - $x(\cdot)$ and $\dot{x}(\cdot)$ fulfill inequation
- controller chooses duration t
- environment chooses valid x



Continuous Behaviour

- described by **differential inequation**
- e.g. $1 \leq x \leq 3 \wedge -1 \leq \dot{x} \leq 1$
- function $x(\cdot)$ valid if
  - $x(0)$ describes an initial value
  - $x(\cdot)$ and $\dot{x}(\cdot)$ fulfill inequation
- controller chooses duration t
- environment chooses valid x
- effect given by $x(t)$

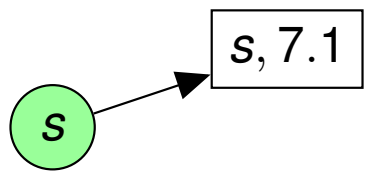


Semantics

- controller

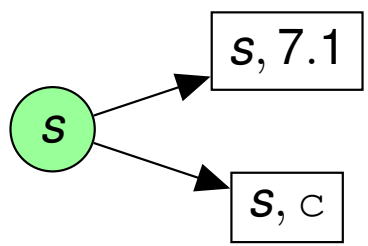
Semantics

- controller
  - chooses whether to wait, and how long



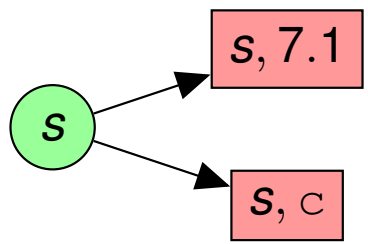
Semantics

- controller
  - chooses whether to wait, and how long
  - or chooses which command to execute



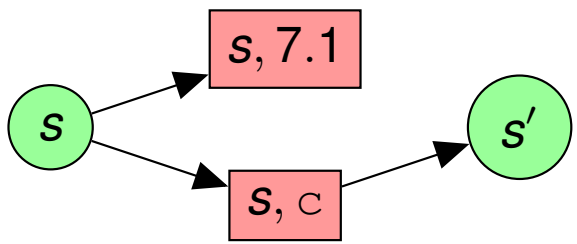
Semantics

- controller
  - chooses whether to wait, and how long
  - or chooses which command to execute
- environment



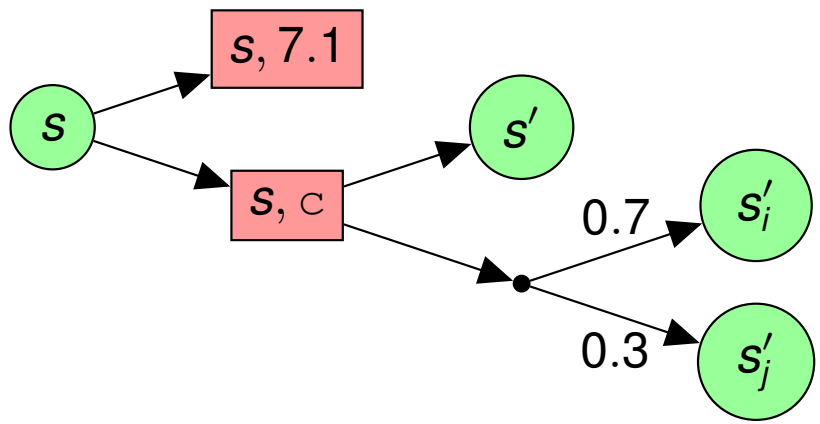
Semantics

- controller
  - chooses whether to wait, and how long
  - or chooses which command to execute
- environment
  - resolves remaining nondeterminism



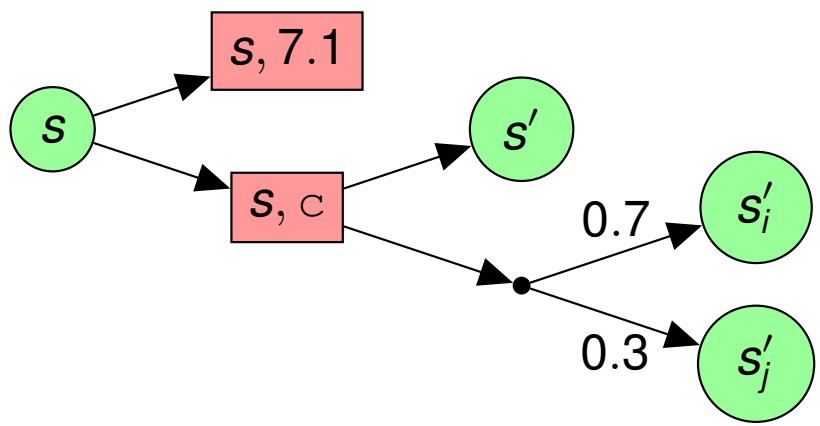
Semantics

- controller
  - chooses whether to wait, and how long
  - or chooses which command to execute
- environment
  - resolves remaining nondeterminism
- probabilistic choice



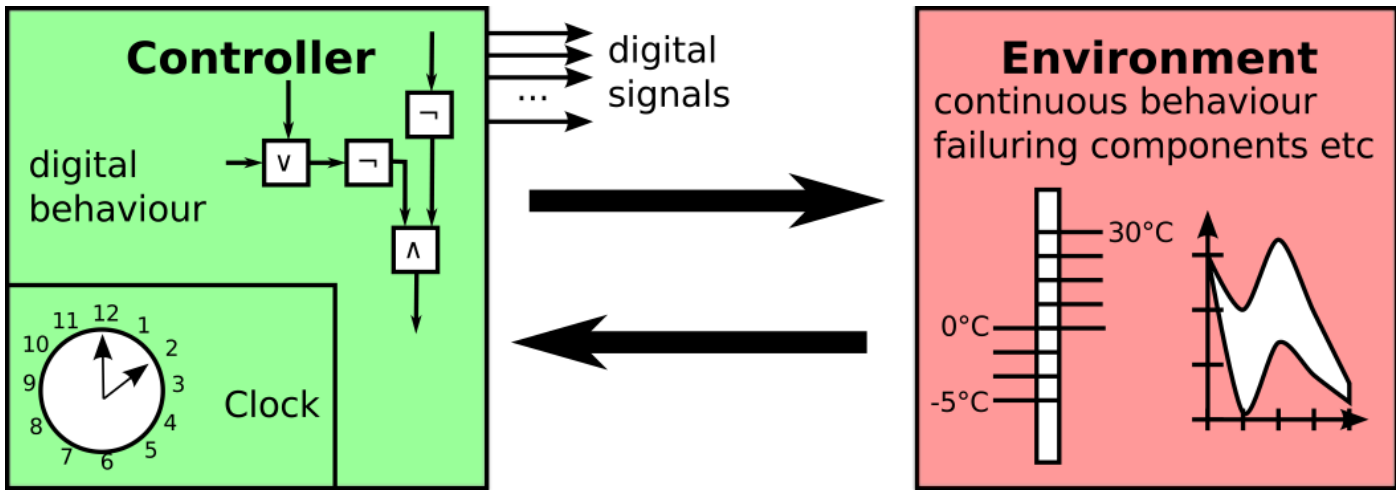
Semantics

- controller
  - chooses whether to wait, and how long
  - or chooses which command to execute
- environment
  - resolves remaining nondeterminism
- probabilistic choice
- $\Rightarrow$ infinite probabilistic two-player game



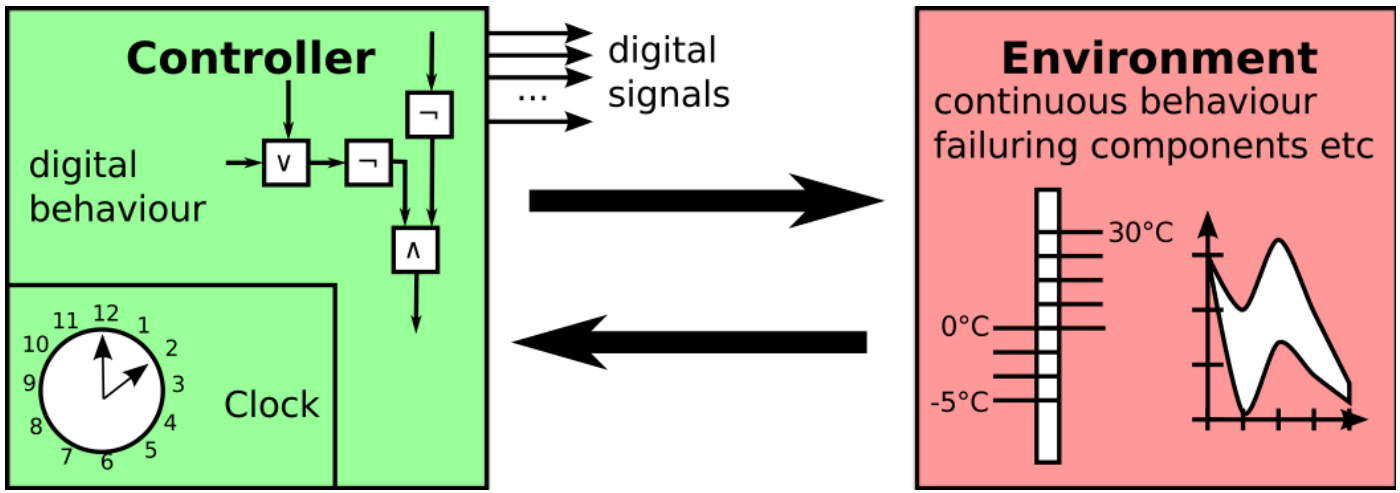
Why Exactly This Semantics?

- natural



Why Exactly This Semantics?

- natural



- fits to existing tools

...

HSolver **PHAVer**

...

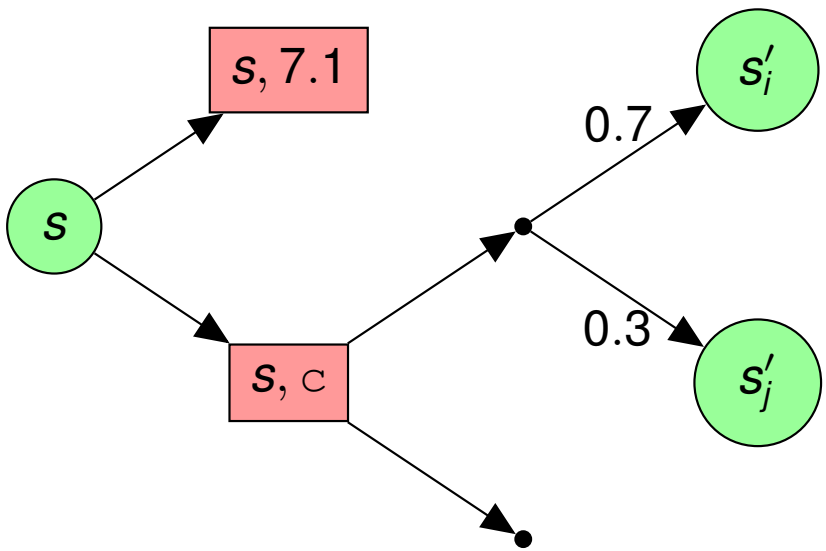
SpaceEx

d/dt

... ...

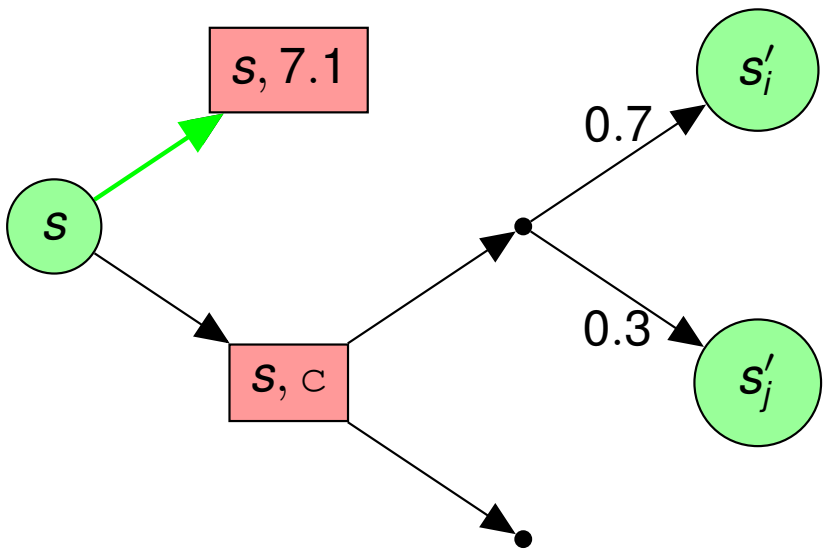
Properties

- **strategy** for given player: chooses successor



Properties

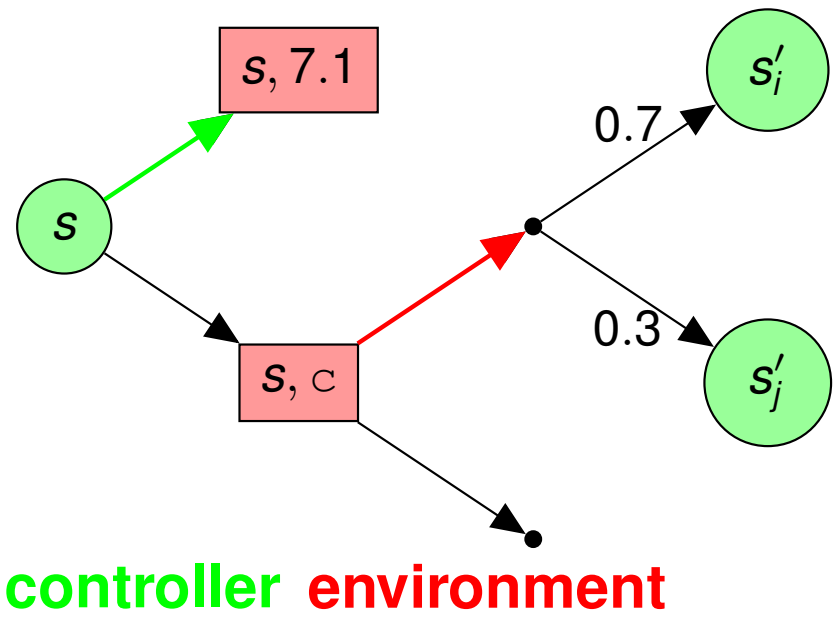
- **strategy** for given player: chooses successor



controller

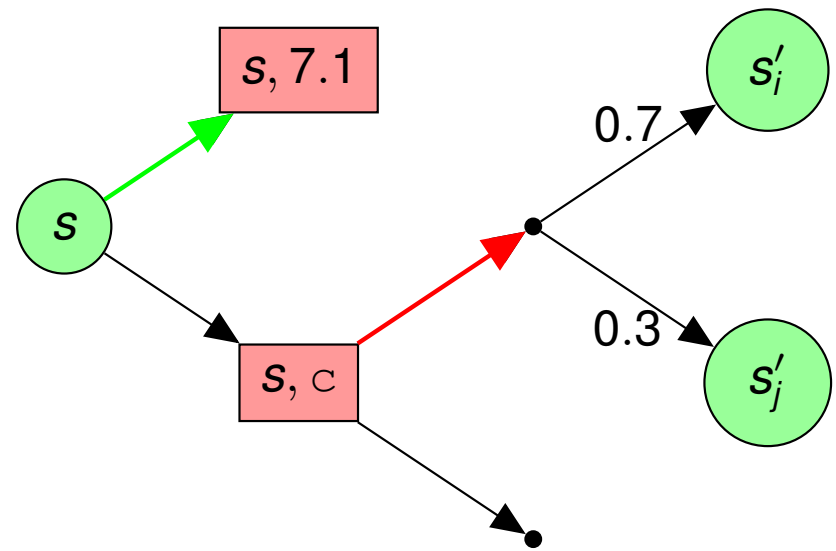
Properties

- **strategy** for given player: chooses successor

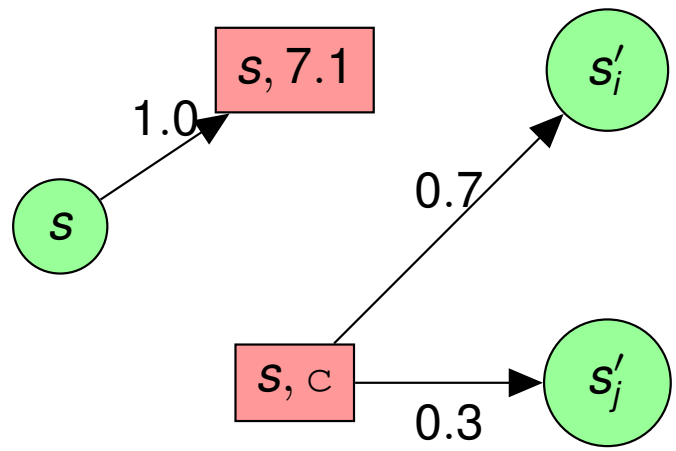


Properties

- **strategy** for given player: chooses successor
- **stochastic process**: obtained by resolving **all** choices



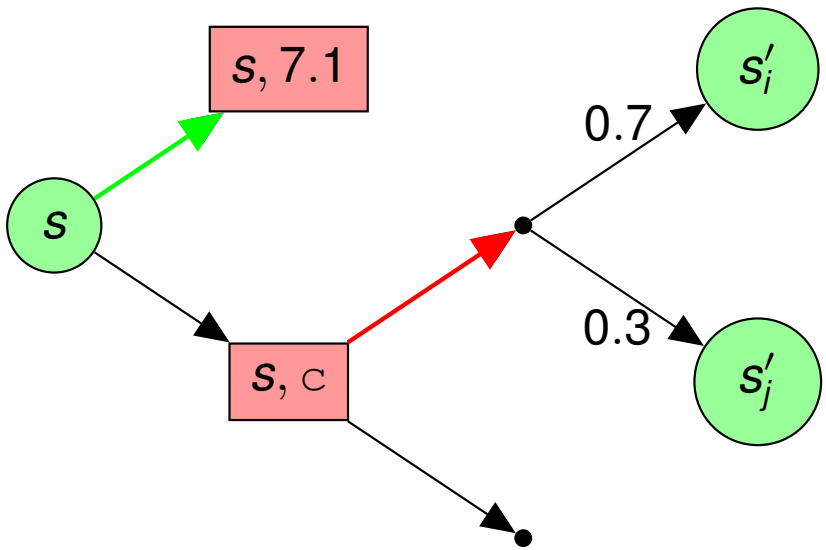
controller **environment**



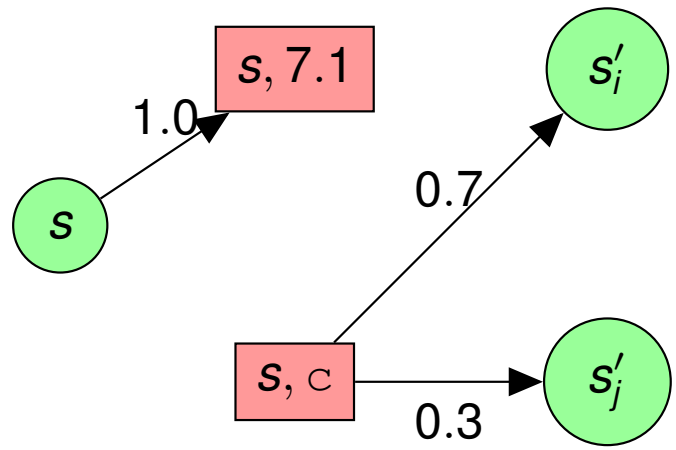
stochastic process

Properties

- **strategy** for given player: chooses successor
- **stochastic process**: obtained by resolving **all** choices
- allows to define **value functions**



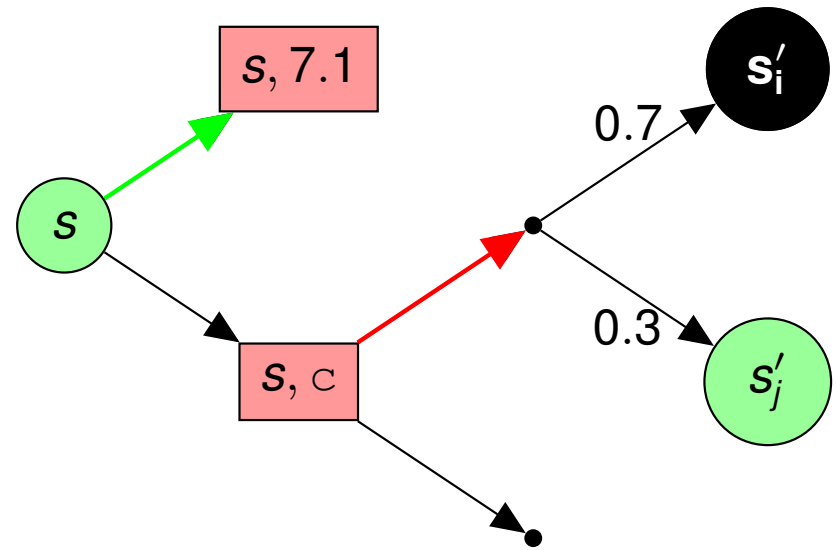
controller **environment**

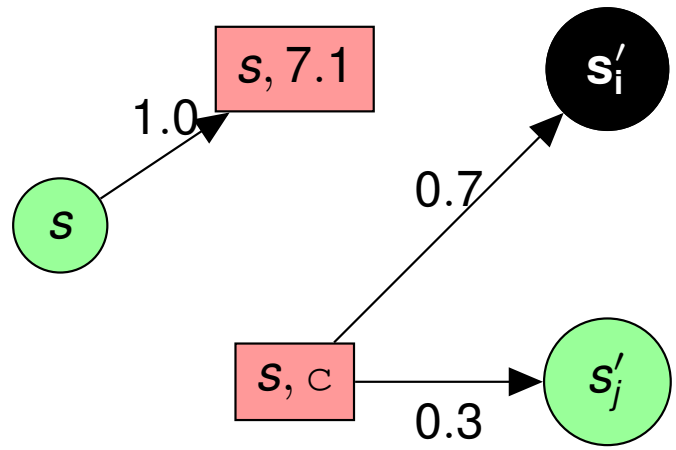


stochastic process

Properties

- **strategy** for given player: chooses successor
- **stochastic process**: obtained by resolving **all** choices
- allows to define **value functions**
- e.g. **probabilistic reachability**: Probability to reach **certain set of states**

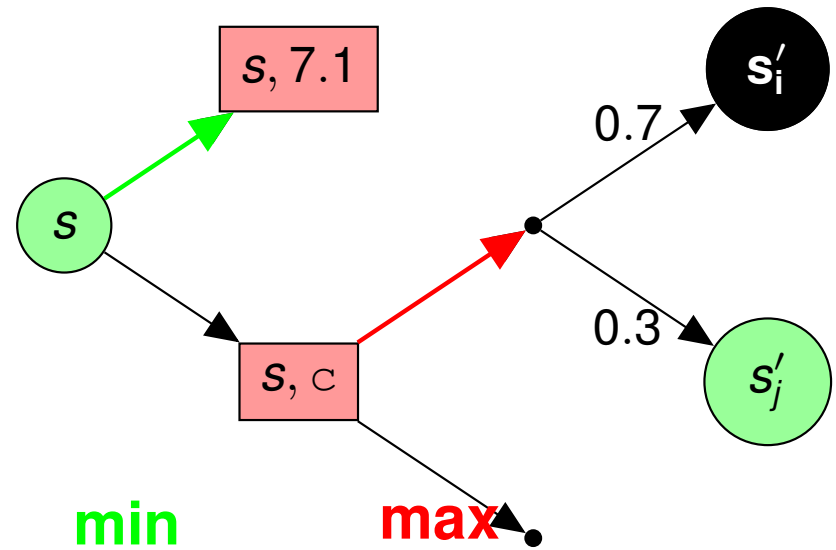


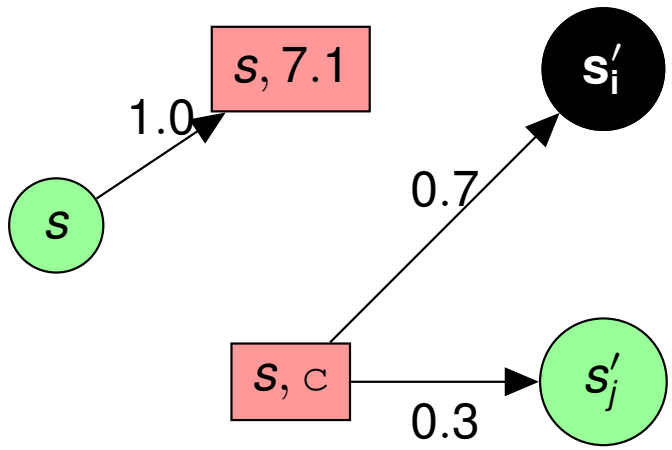


stochastic process

Properties

- **strategy** for given player: chooses successor
- **stochastic process**: obtained by resolving **all** choices
- allows to define **value functions**
- e.g. **probabilistic reachability**: Probability to reach **certain set of states**
- **objective** of player: minimize or maximise value function

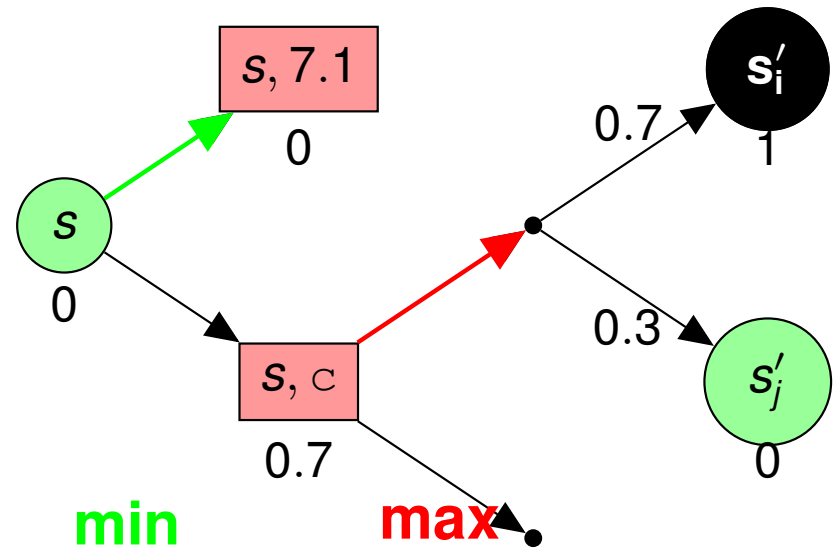


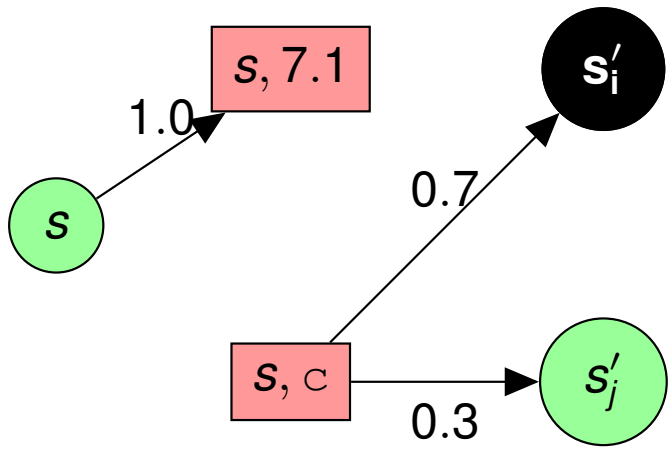


stochastic process

Properties

- **strategy** for given player: chooses successor
- **stochastic process**: obtained by resolving **all** choices
- allows to define **value functions**
- e.g. **probabilistic reachability**: Probability to reach **certain set of states**
- **objective** of player: minimize or maximise value function
- **optimal value**: from optimal strategies objectives

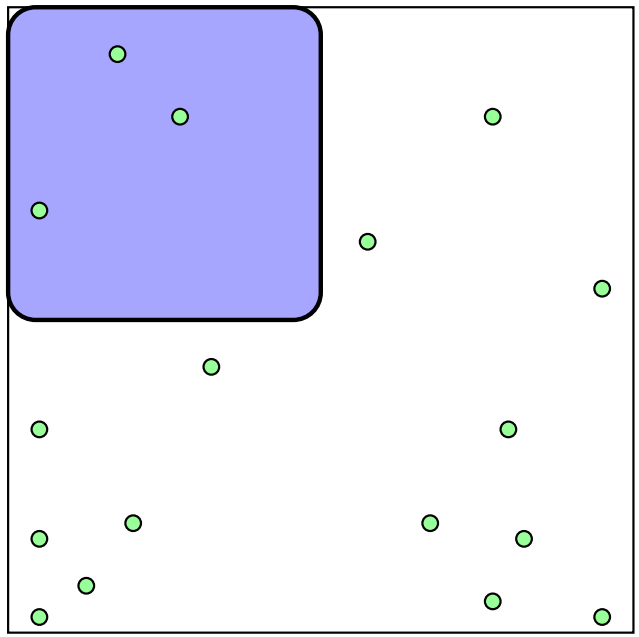




stochastic process

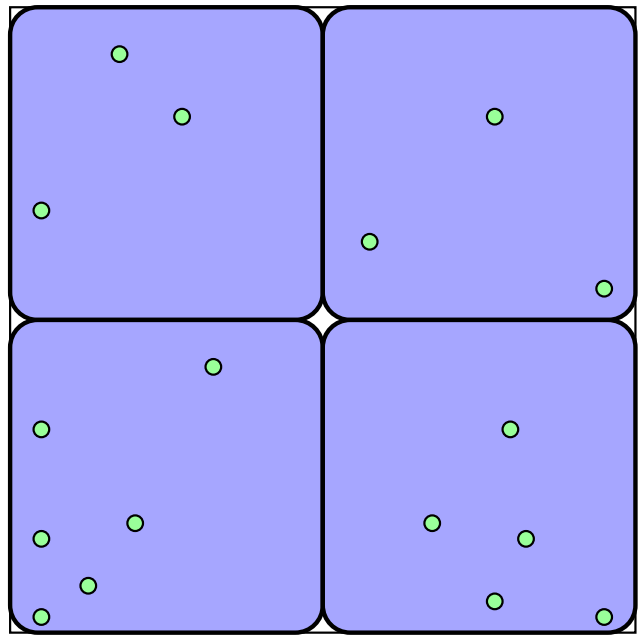
Abstract State Space

- **abstract state:** set of concrete states



Abstract State Space

- **abstract state:** set of concrete states
- **abstract state space:** set of abstract states which contains complete state space



Game-based Abstraction (GBA)

- finite three-player game on abstract state space $\mathcal{A}$

Game-based Abstraction (GBA)

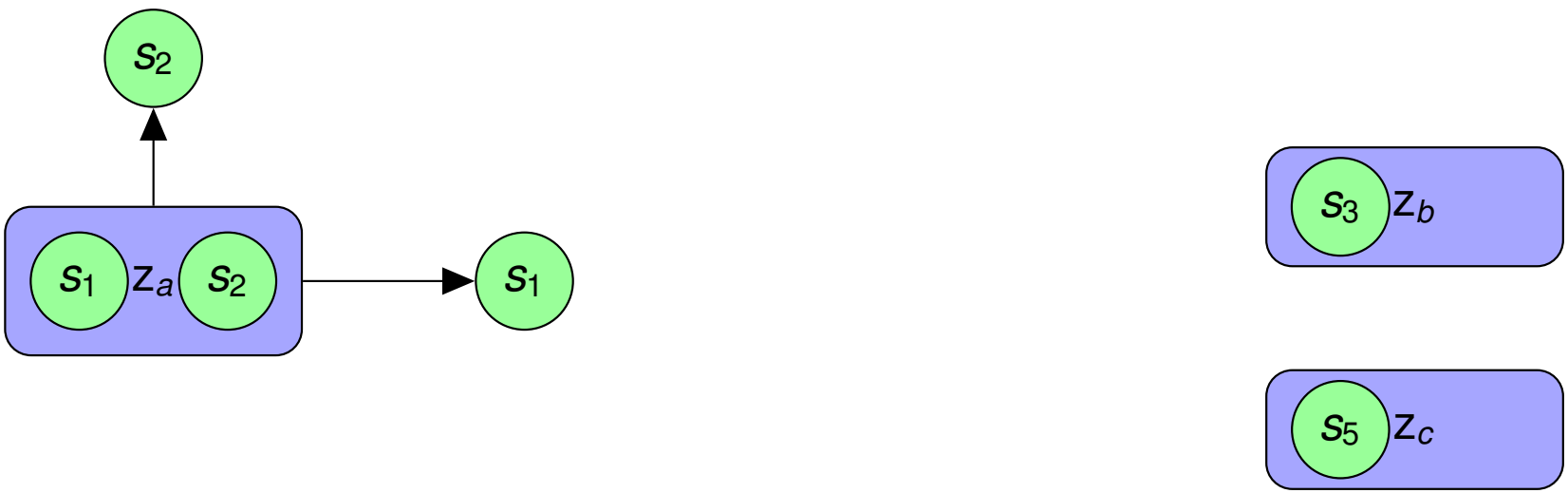
- finite three-player game on abstract state space $\mathcal{A}$
- S_{abs}: abstraction player

Game-based Abstraction (GBA)

- finite three-player game on abstract state space $\mathcal{A}$
- S_{abs}: abstraction player

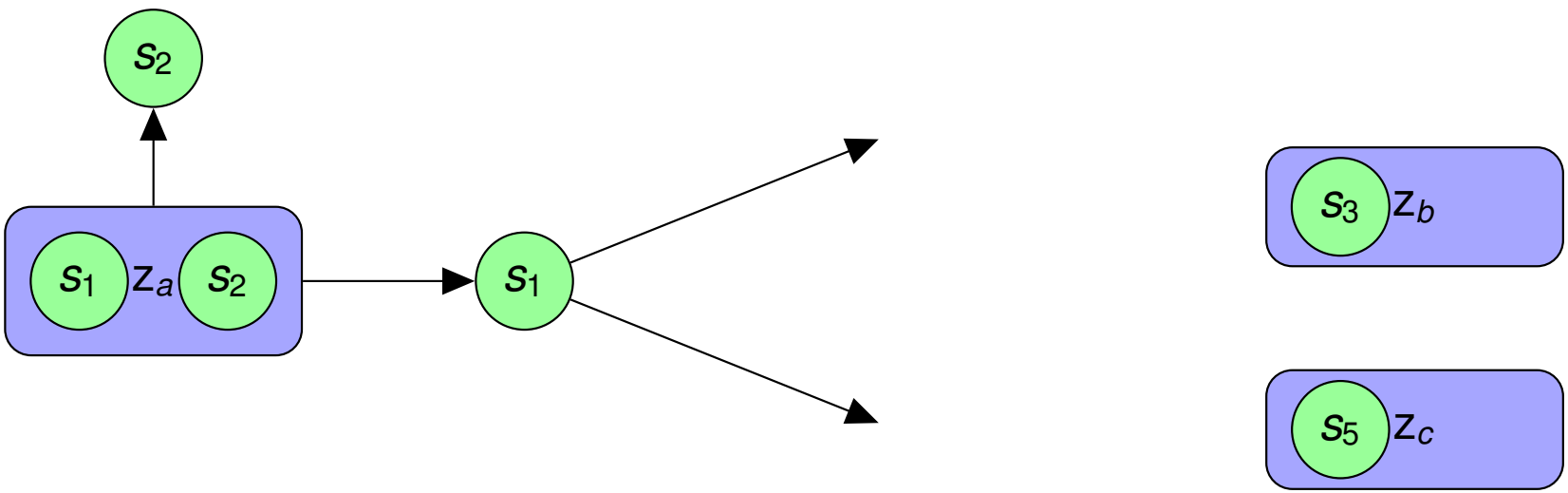
Game-based Abstraction (GBA)

- finite three-player game on abstract state space $\mathcal{A}$
- S_{abs}: abstraction player
- S_{con}: controller player



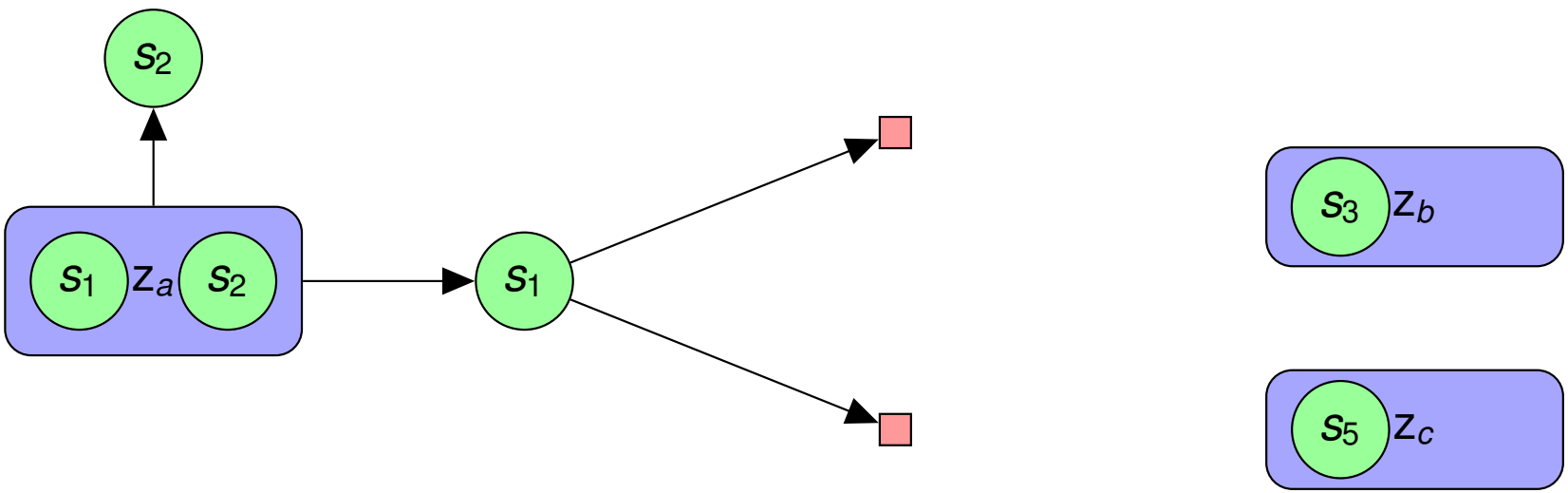
Game-based Abstraction (GBA)

- finite three-player game on abstract state space $\mathcal{A}$
- S_{abs}: abstraction player
- S_{con}: controller player



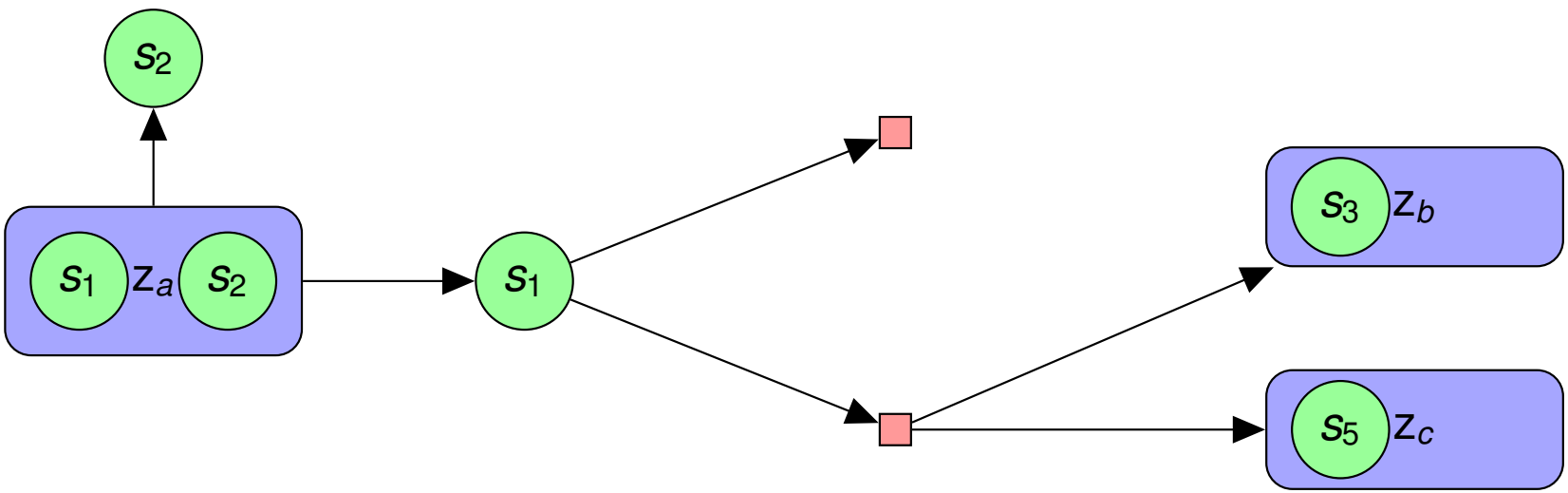
Game-based Abstraction (GBA)

- finite three-player game on abstract state space $\mathcal{A}$
- S_{abs}: abstraction player
- S_{con}: controller player
- S_{env}: environment player



Game-based Abstraction (GBA)

- finite three-player game on abstract state space $\mathcal{A}$
- S_{abs}: abstraction player
- S_{con}: controller player
- S_{env}: environment player



GBA: Analysis

- what to do with third player?

GBA: Analysis

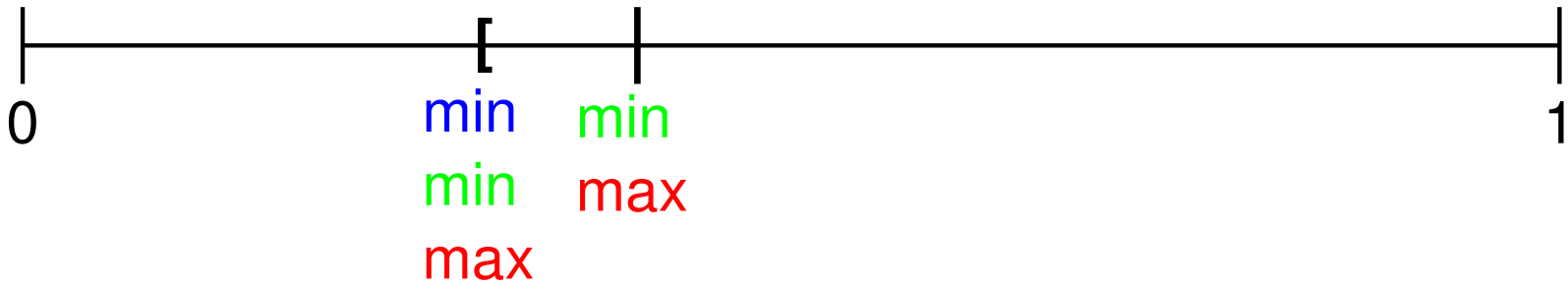
- what to do with third player?
- combine abstraction player with controller or environment

GBA: Analysis

- what to do with third player?
- combine abstraction player with controller or environment

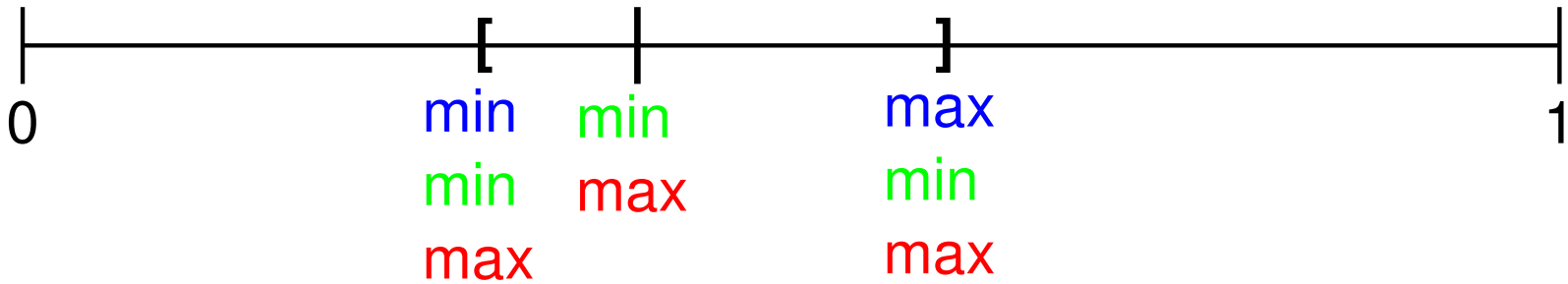
GBA: Analysis

- what to do with third player?
- combine abstraction player with controller or environment



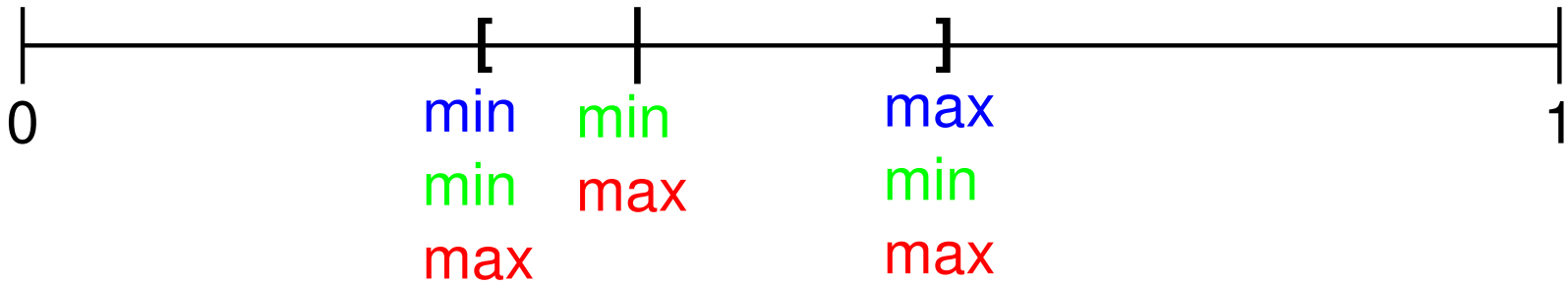
GBA: Analysis

- what to do with third player?
- combine abstraction player with controller or environment



GBA: Analysis

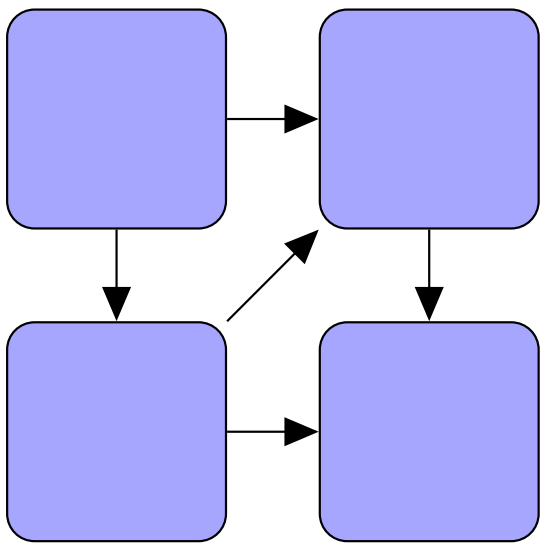
- what to do with third player?
- combine abstraction player with controller or environment



- $\Rightarrow$ finite probabilistic two-player games
- efficient solution techniques exist
  (value iteration or policy iteration)

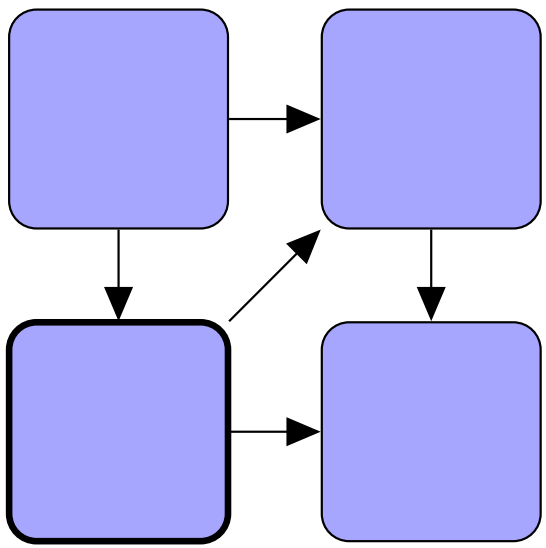
GBA: Refinement

- bounds may be too coarse



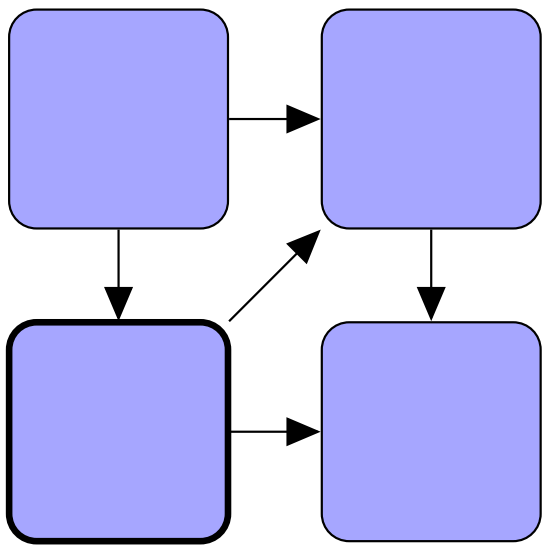
GBA: Refinement

- bounds may be too coarse
- choose abstract state to refine



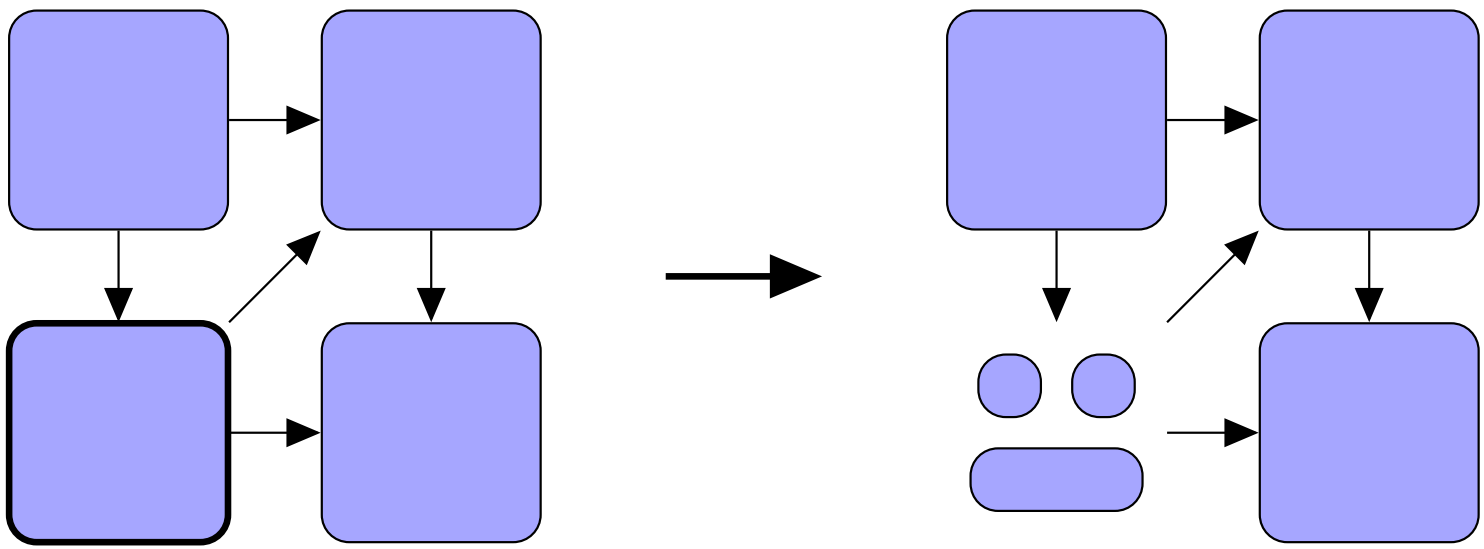
GBA: Refinement

- bounds may be too coarse
- choose abstract state to refine
- choice depends on probabilities and strategies



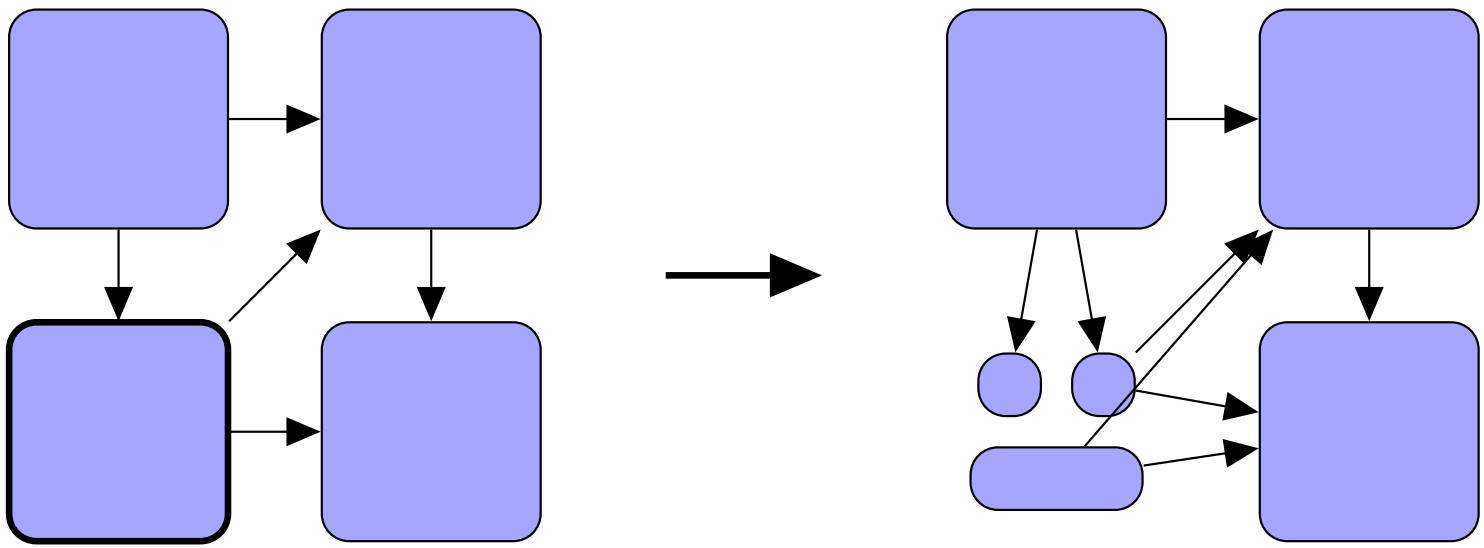
GBA: Refinement

- bounds may be too coarse
- choose abstract state to refine
- choice depends on probabilities and strategies
- split into new states



GBA: Refinement

- bounds may be too coarse
- choose abstract state to refine
- choice depends on probabilities and strategies
- split into new states
- afterwards adapt transitions



Environment Abstraction

- GBA yields safe bounds and allows refinement

Environment Abstraction

- GBA yields safe bounds and allows refinement
- however, underapproximation needed, not easy

Environment Abstraction

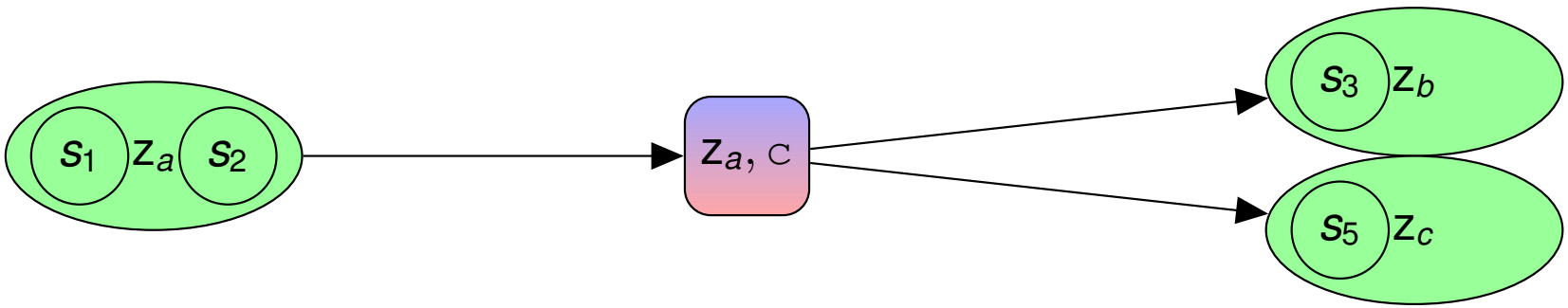
- GBA yields safe bounds and allows refinement
- however, underapproximation needed, not easy
- $\Rightarrow$ **environment abstraction**

Environment Abstraction

- GBA yields safe bounds and allows refinement
- however, underapproximation needed, not easy
- $\Rightarrow$ **environment abstraction**
- can be implemented using existing solvers (e.g. PHAVer)

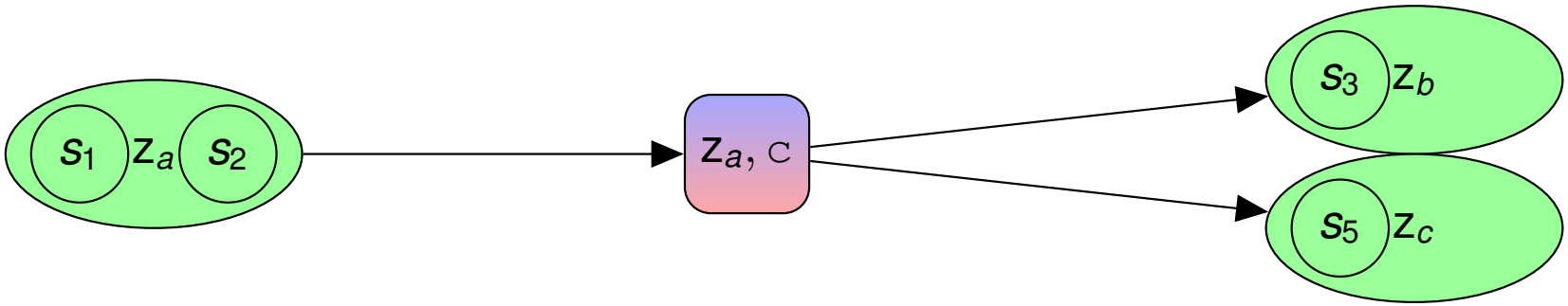
Environment Abstraction

- GBA yields safe bounds and allows refinement
- however, underapproximation needed, not easy
- $\Rightarrow$ **environment abstraction**
- can be implemented using existing solvers (e.g. PHAVer)
- subsume abstraction and environment by construction



Environment Abstraction

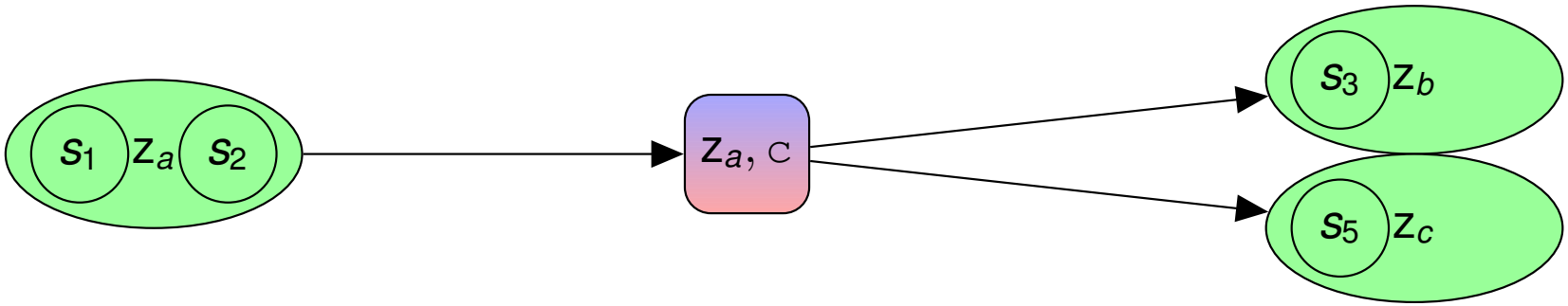
- GBA yields safe bounds and allows refinement
- however, underapproximation needed, not easy
- $\Rightarrow$ **environment abstraction**
- can be implemented using existing solvers (e.g. PHAVer)
- subsume abstraction and environment by construction



- bounds less tight than in GBA

Environment Abstraction

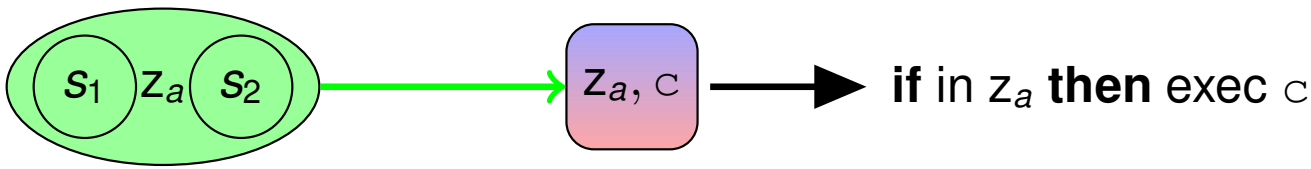
- GBA yields safe bounds and allows refinement
- however, underapproximation needed, not easy
- $\Rightarrow$ **environment abstraction**
- can be implemented using existing solvers (e.g. PHAVer)
- subsume abstraction and environment by construction



- bounds less tight than in GBA

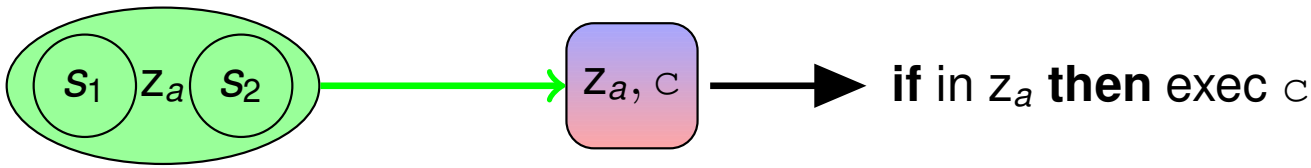
Extensions

- controller synthesis

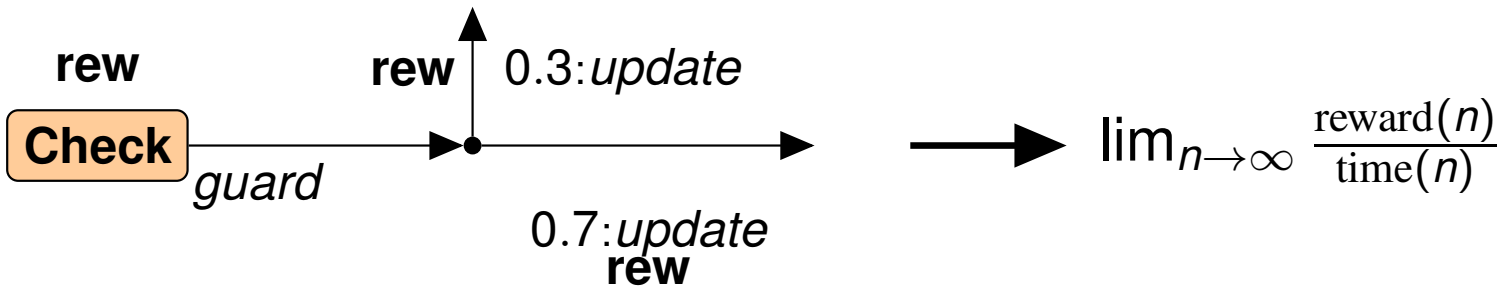


Extensions

- controller synthesis



- long-run average rewards



Experiments

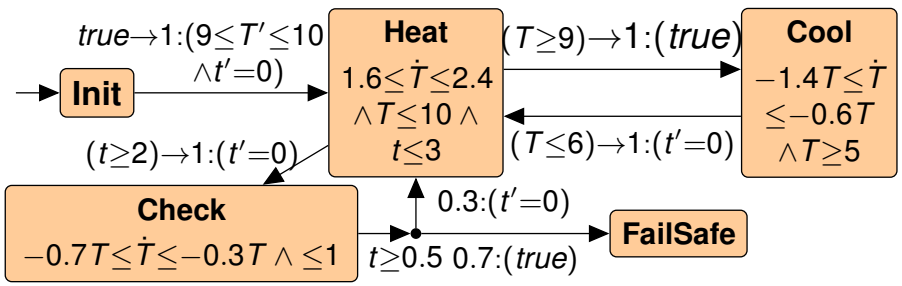
- preliminary implementation of environment abstraction

Experiments

- preliminary implementation of environment abstraction
- build on top of PHAVer

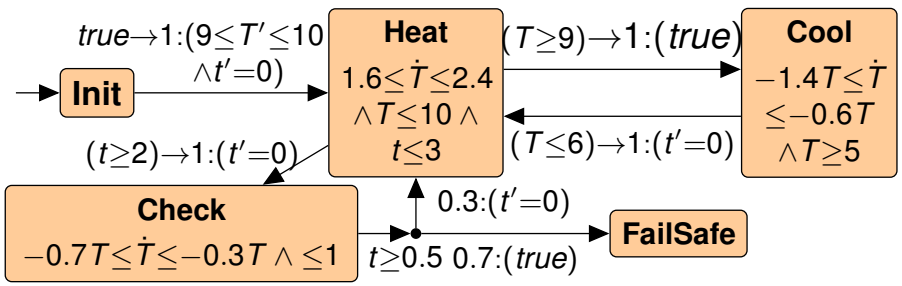
Experiments

- preliminary implementation of environment abstraction
- build on top of PHAVer
- applied on thermostat example



Experiments

- preliminary implementation of environment abstraction
- build on top of PHAVer
- applied on thermostat example
- maximal probability reach safe state within given time B



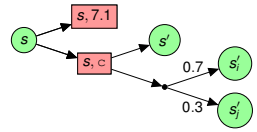
B	coarseness 0.5 prob.	build(s)	#states
5	[0.000, 0.910]	14	1051
10	[0.910, 0.992]	81	4330
15	[0.973, 0.999]	50	3216

B	coarseness 0.2 prob.	build(s)	#states
5	[0.700, 0.910]	54	4066
10	[0.910, 0.992]	413	16773
15	[0.992, 0.999]	2578	53289

coarseness of abstraction: PHAVer-specific

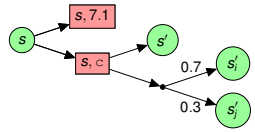
Summary

- game semantics for probabilistic hybrid systems

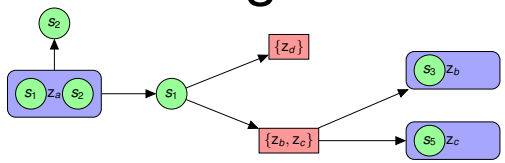


Summary

- game semantics for probabilistic hybrid systems

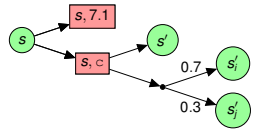


- game-based abstraction: tight bounds, refinement

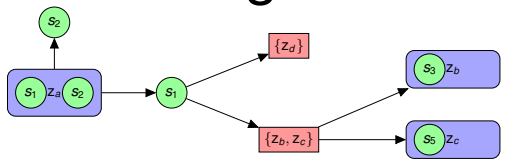


Summary

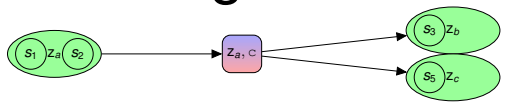
- game semantics for probabilistic hybrid systems



- game-based abstraction: tight bounds, refinement

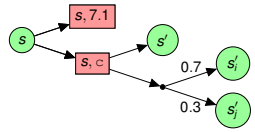


- environment abstraction: general

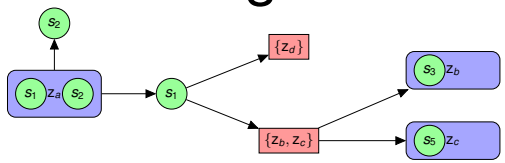


Summary

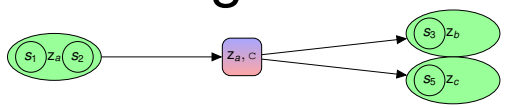
- game semantics for probabilistic hybrid systems



- game-based abstraction: tight bounds, refinement



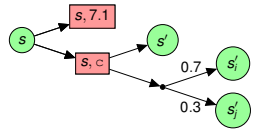
- environment abstraction: general



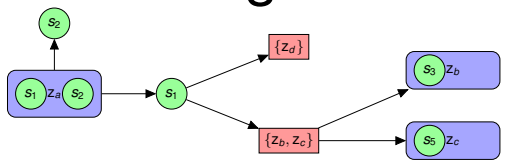
- controller synthesis

Summary

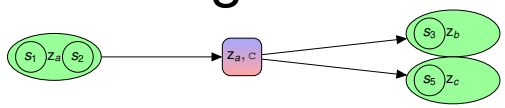
- game semantics for probabilistic hybrid systems



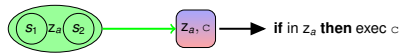
- game-based abstraction: tight bounds, refinement



- environment abstraction: general



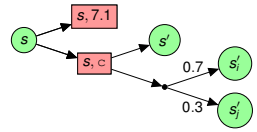
- controller synthesis



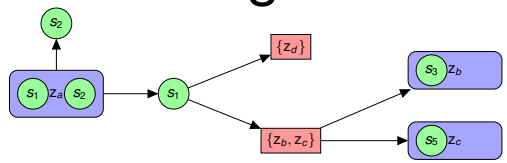
- long-run average values

Summary

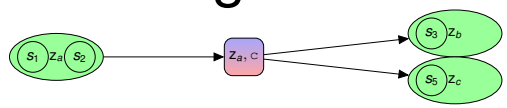
- game semantics for probabilistic hybrid systems



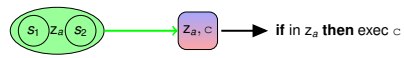
- game-based abstraction: tight bounds, refinement



- environment abstraction: general



- controller synthesis



- long-run average values

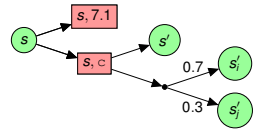
- preliminary experimental results

B	coarseness 0.5 prob.	build(s)	#states
5	[0.000, 0.910]	14	1051
10	[0.910, 0.992]	81	4330
15	[0.973, 0.999]	50	3216

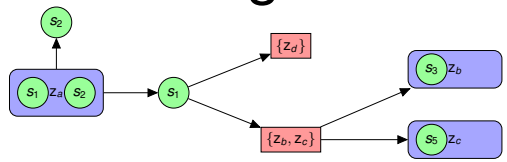
B	coarseness 0.2 prob.	build(s)	#states
5	[0.700, 0.910]	54	4066
10	[0.910, 0.992]	413	16773
15	[0.992, 0.999]	2578	53289

Summary

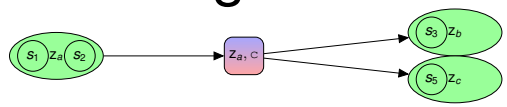
- game semantics for probabilistic hybrid systems



- game-based abstraction: tight bounds, refinement

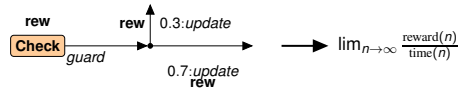


- environment abstraction: general



- controller synthesis

- long-run average values



- preliminary experimental results

B	coarseness 0.5 prob.	build(s)	#states
5	[0.000, 0.910]	14	1051
10	[0.910, 0.992]	81	4330
15	[0.973, 0.999]	50	3216

B	coarseness 0.2 prob.	build(s)	#states
5	[0.700, 0.910]	54	4066
10	[0.910, 0.992]	413	16773
15	[0.992, 0.999]	2578	53289

- future: combine with continuous jump targets (HSCC'11)