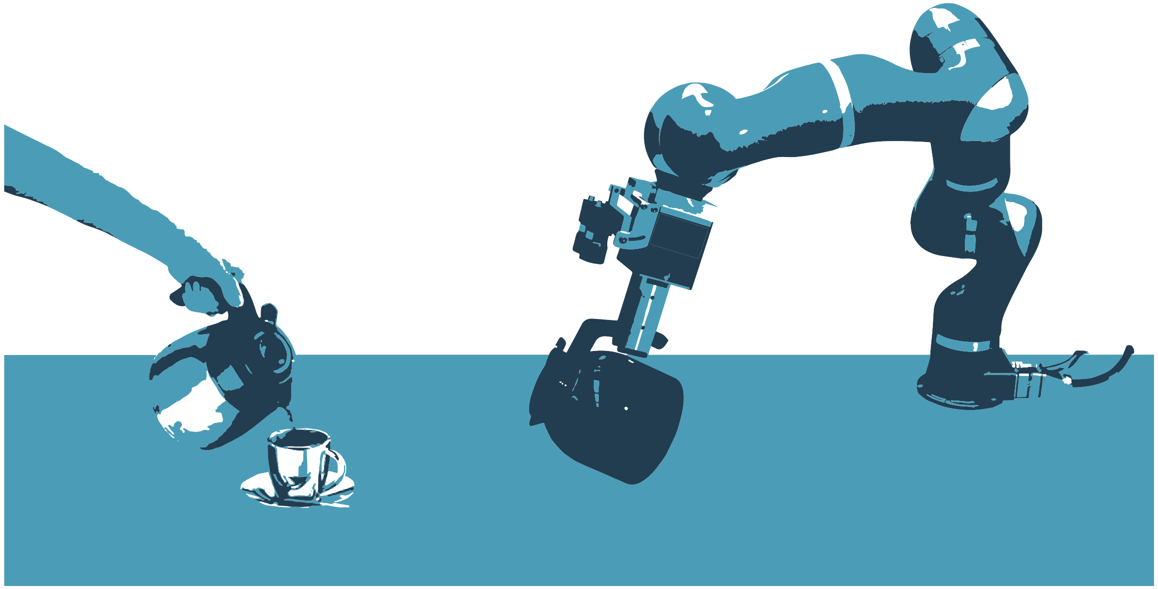
Safety-constrained Reinforcement Learning for MDPs



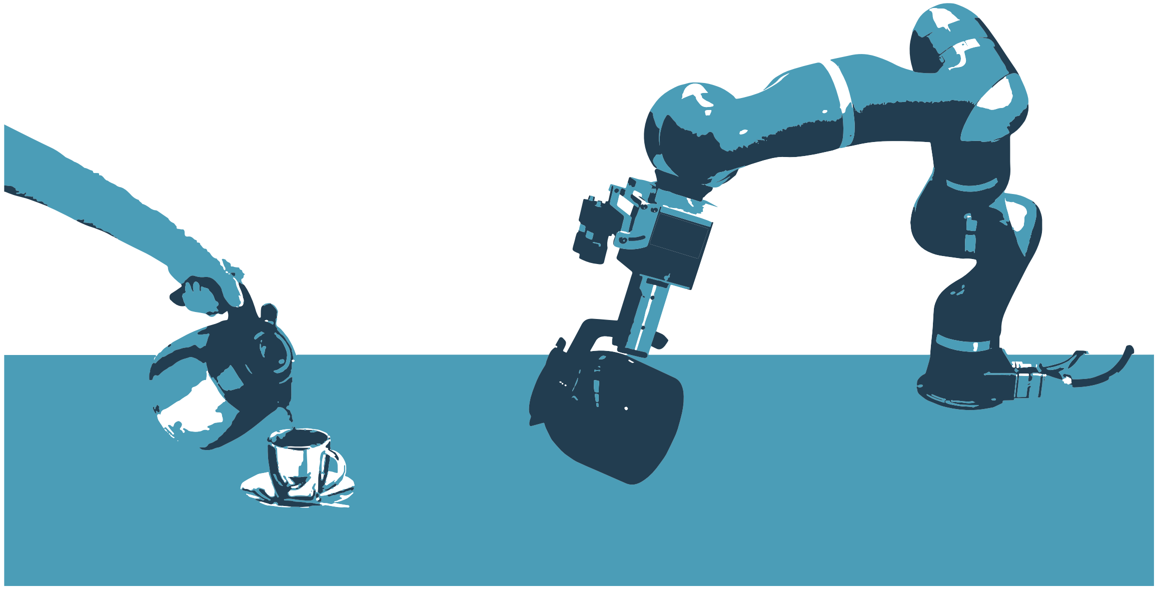
HCSS 2016

Nils Jansen

INSTITUTE FOR COMPUTATIONAL ENGINEERING & SCIENCES

THE UNIVERSITY OF TEXAS AT AUSTIN

Some results on Controller Synthesis for Probabilistic Systems



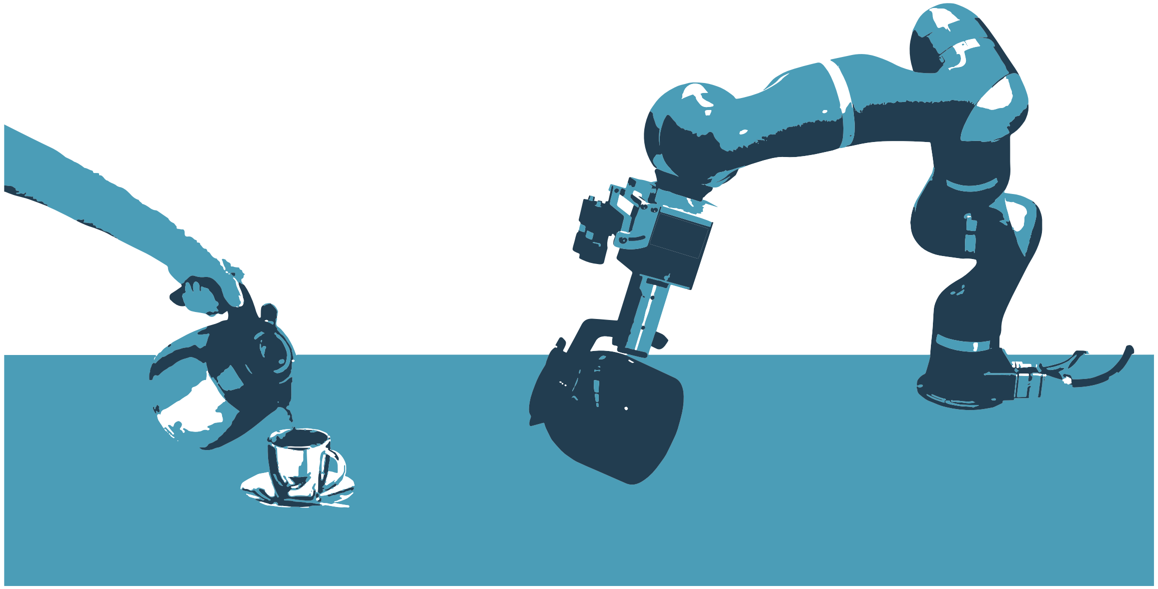
HCSS 2016

Nils Jansen

INSTITUTE FOR COMPUTATIONAL ENGINEERING & SCIENCES

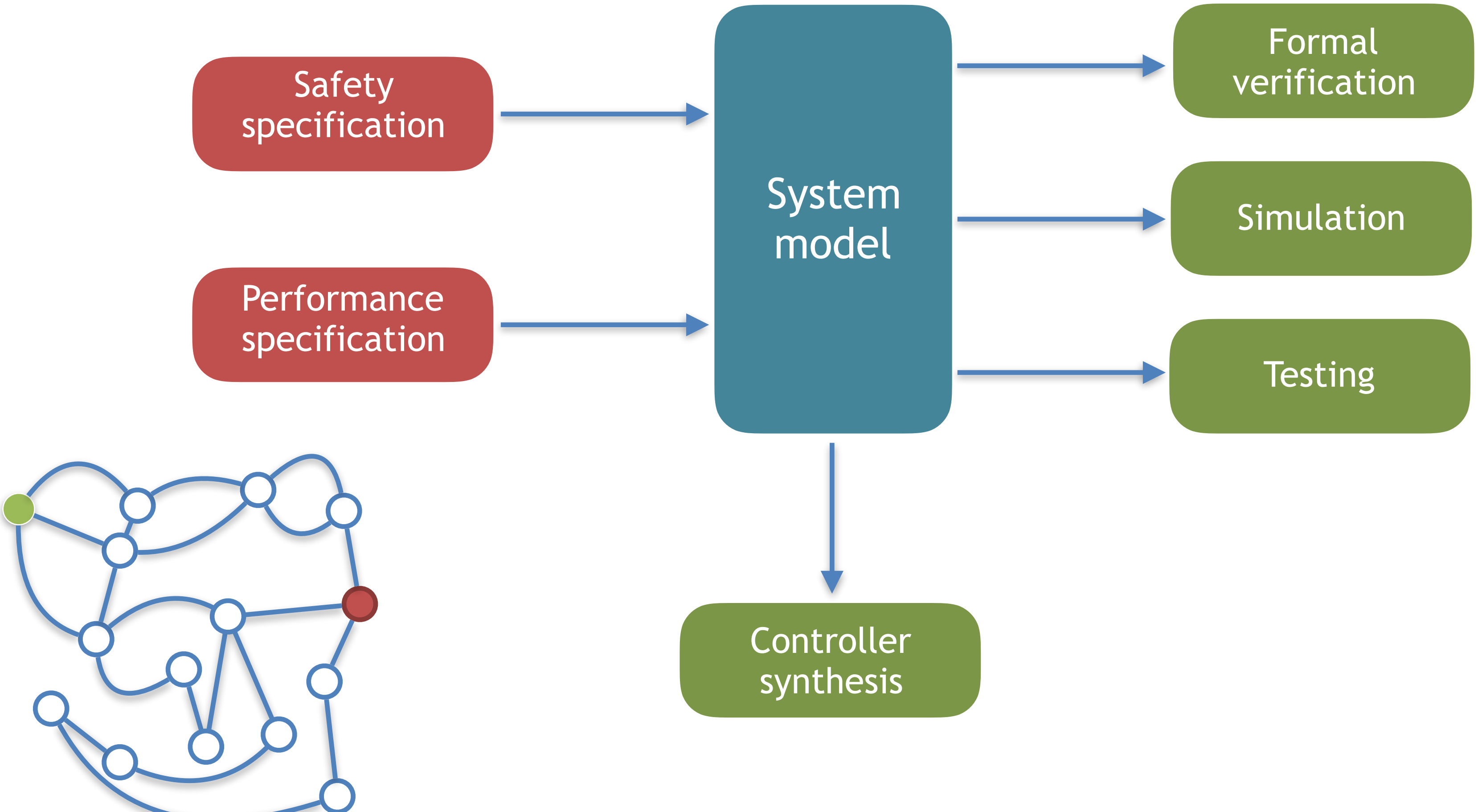
THE UNIVERSITY OF TEXAS AT AUSTIN

Safety-constrained Reinforcement Learning for MDPs

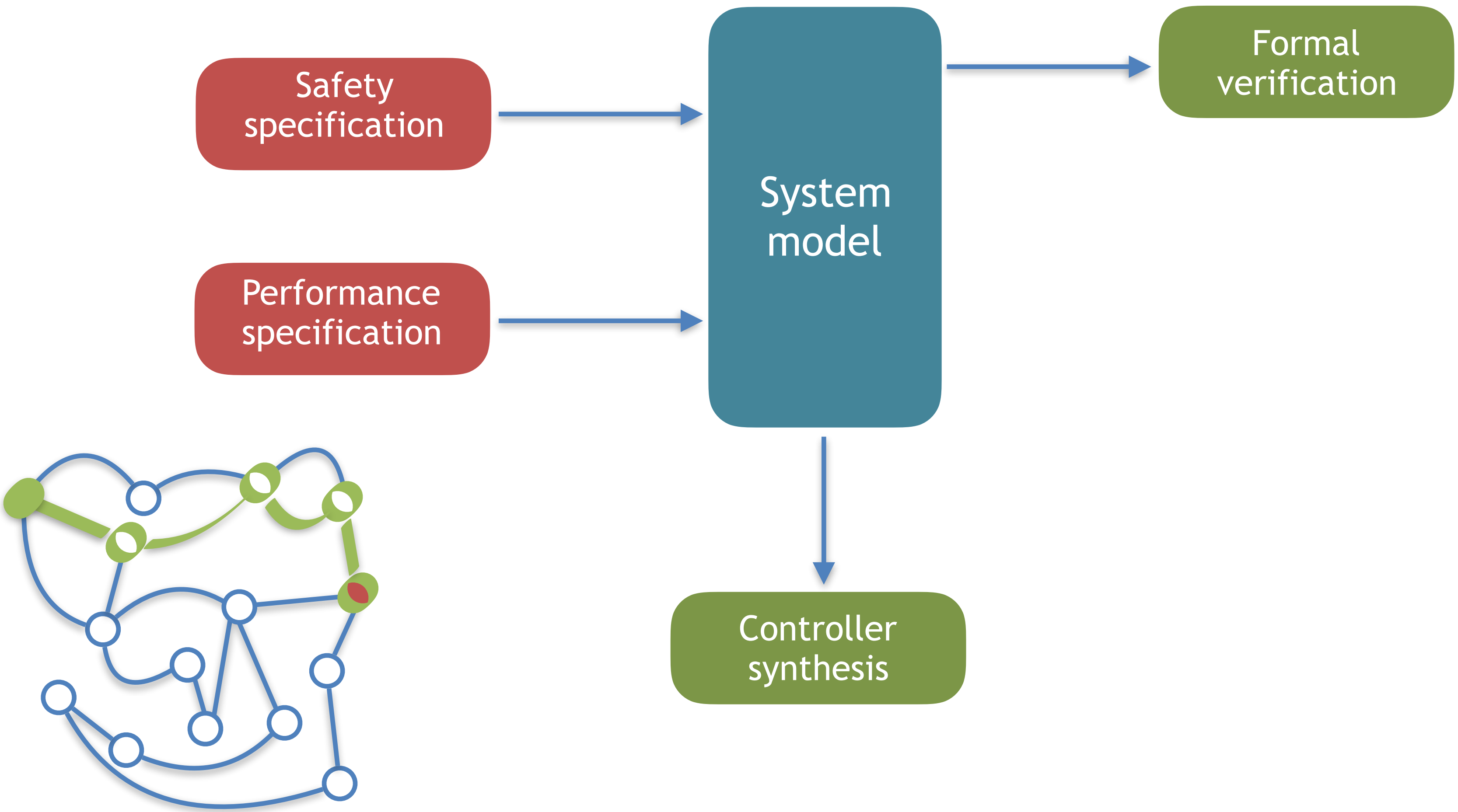


Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen

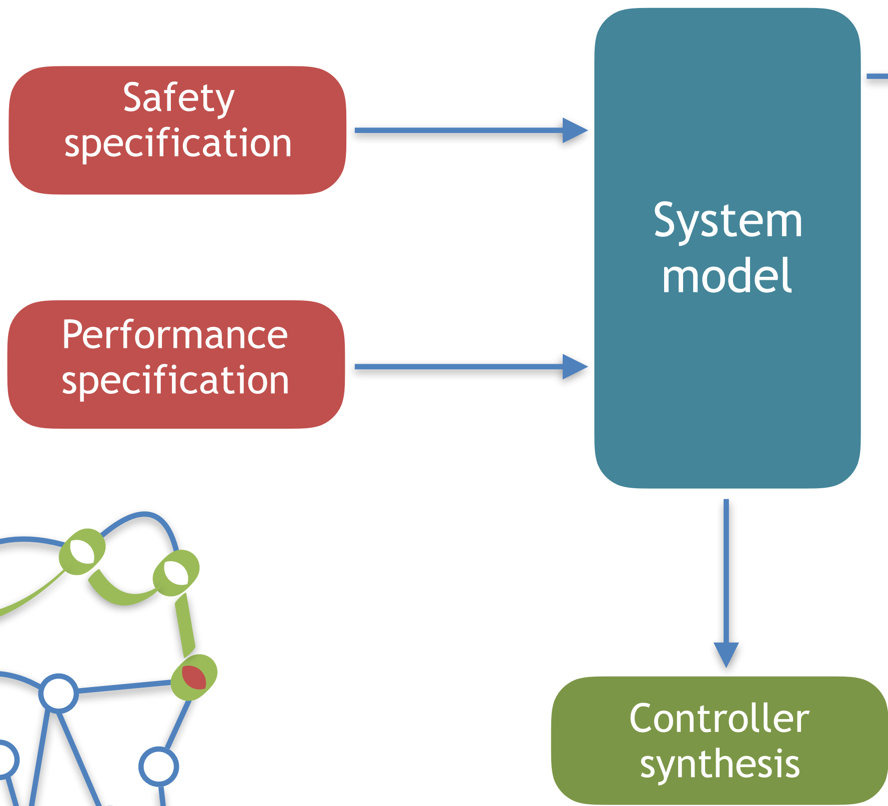
Motivation - Safety-critical Systems

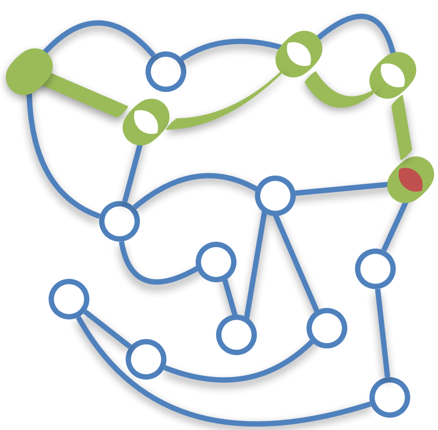


Motivation - Safety-critical Systems

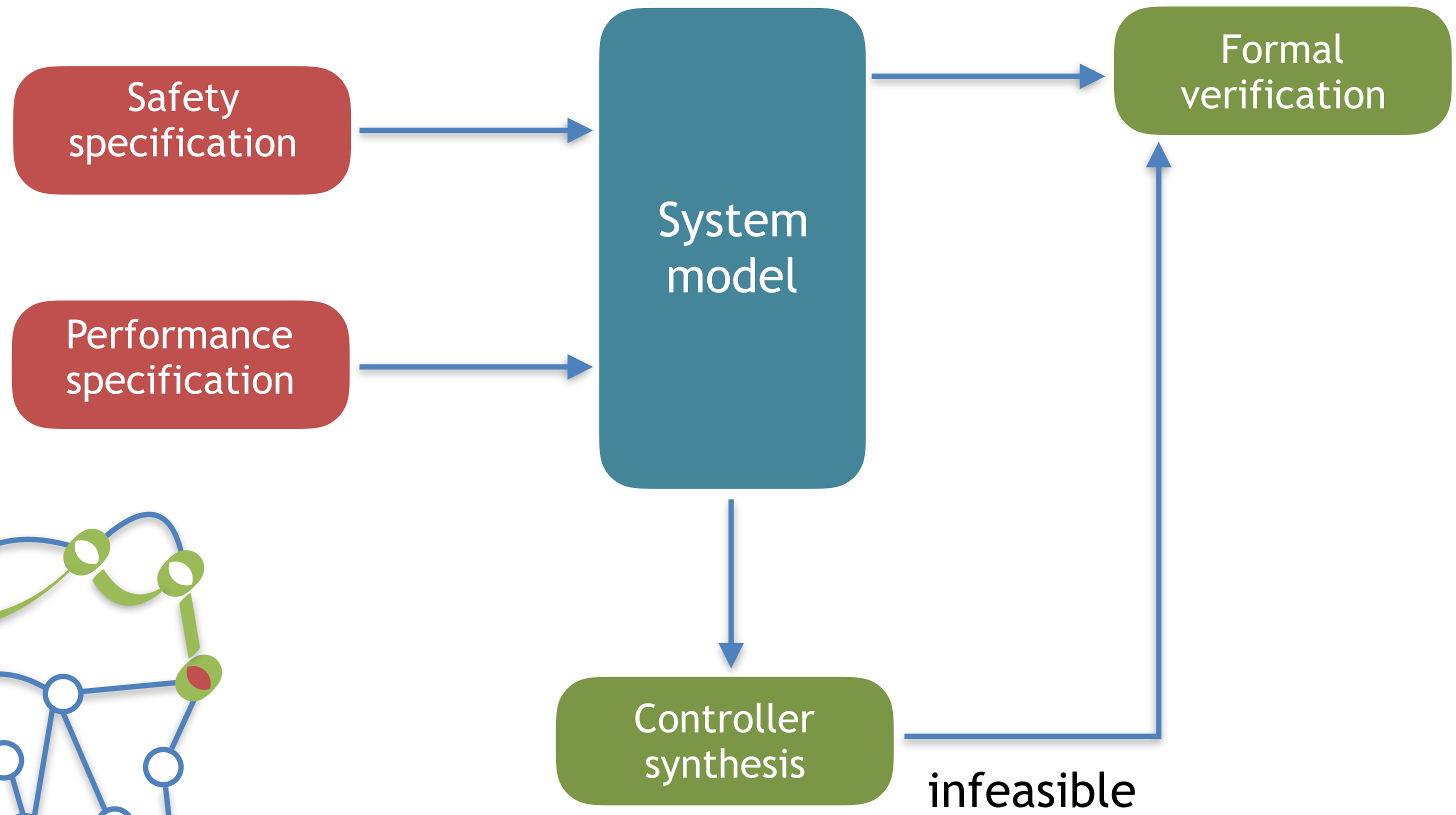


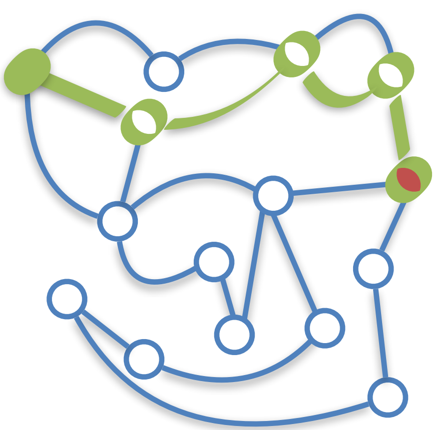
Motivation - Safety-critical Systems



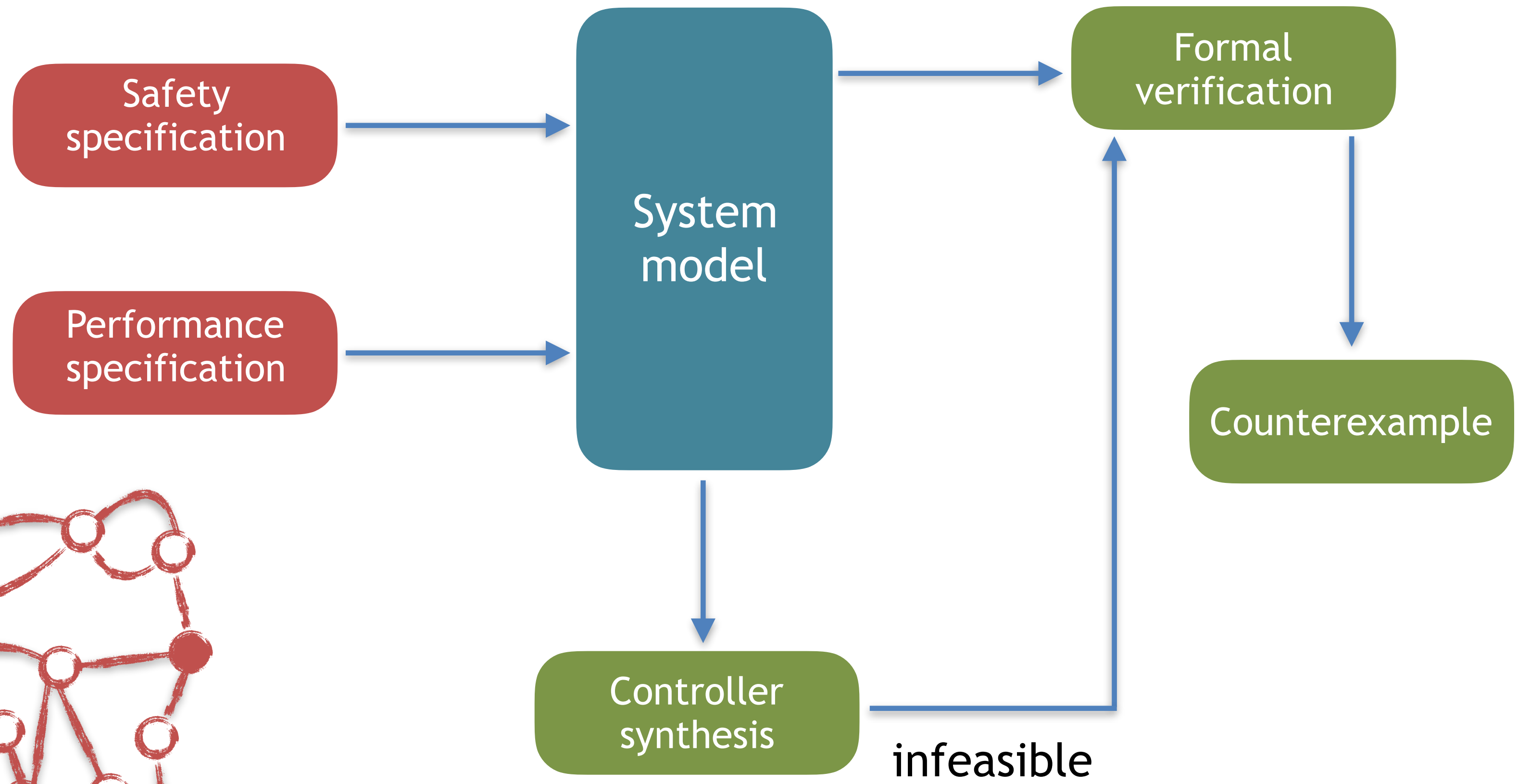


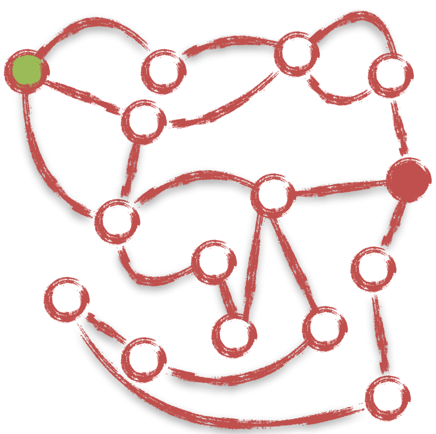
Motivation - Safety-critical Systems



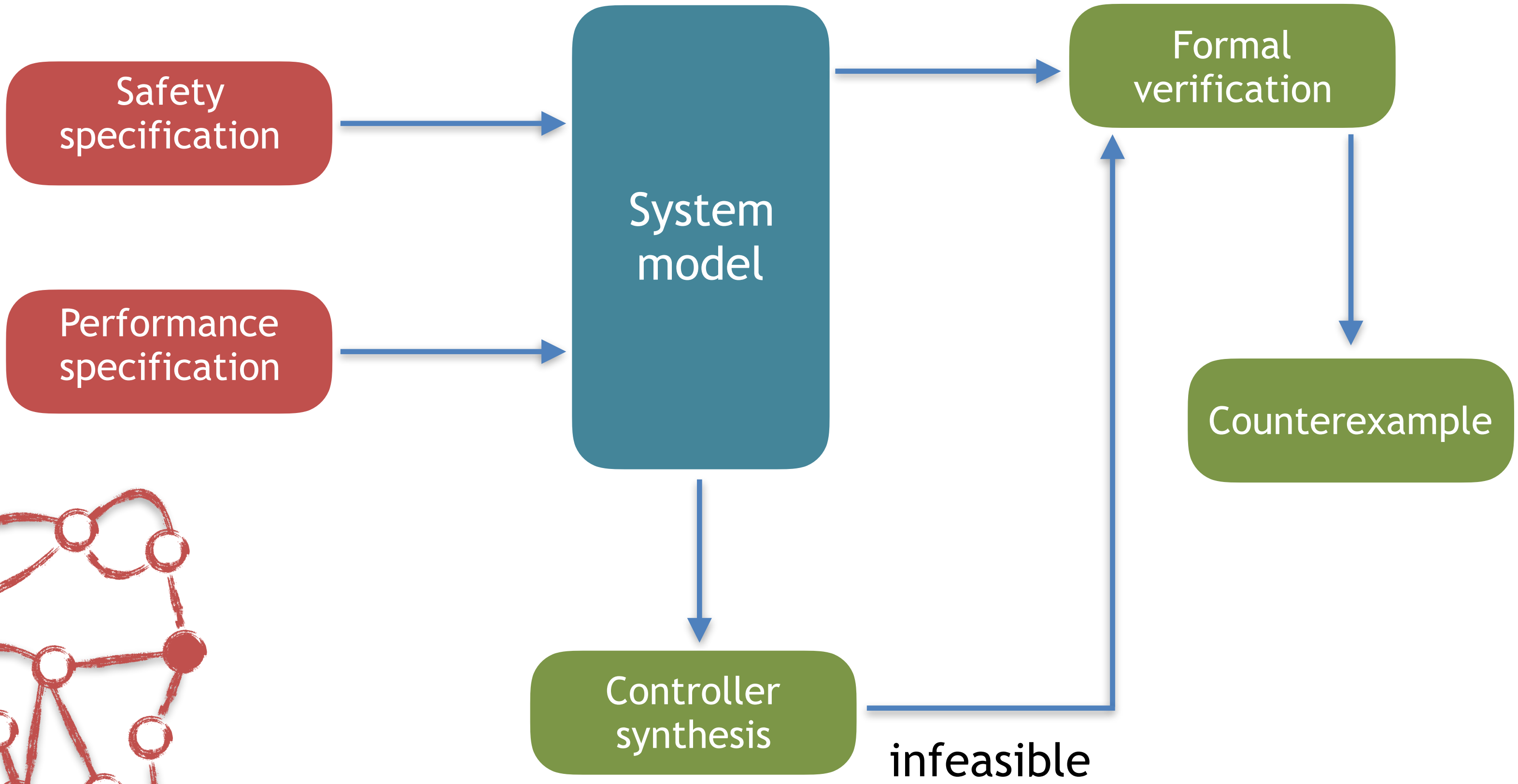


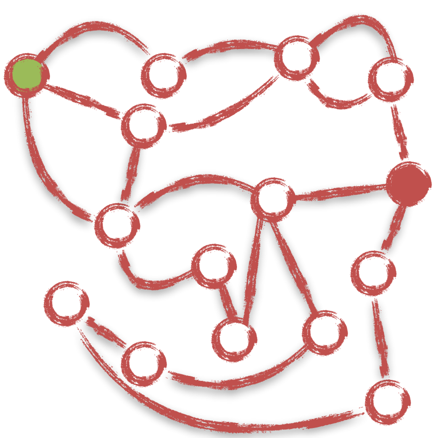
Motivation - Safety-critical Systems



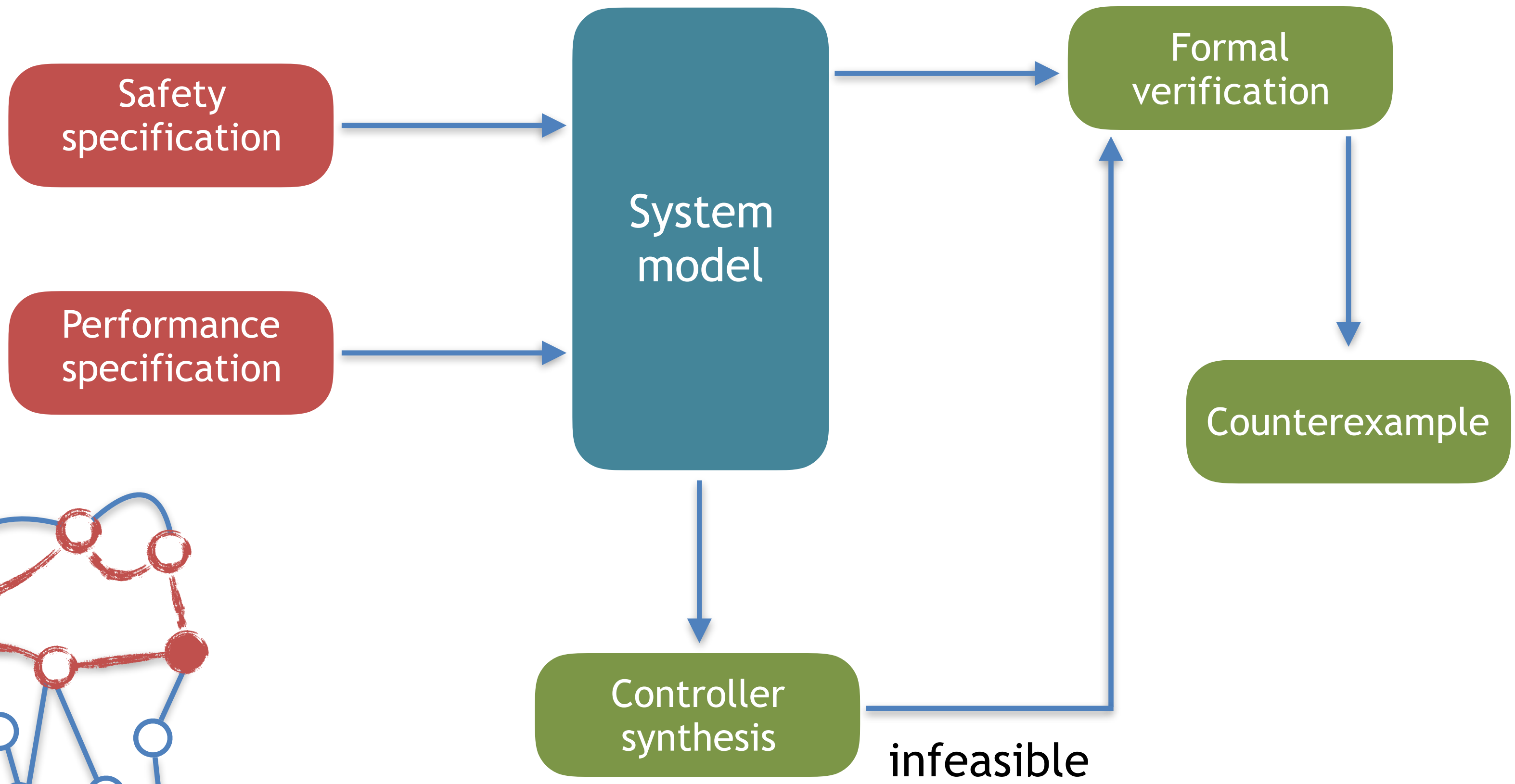


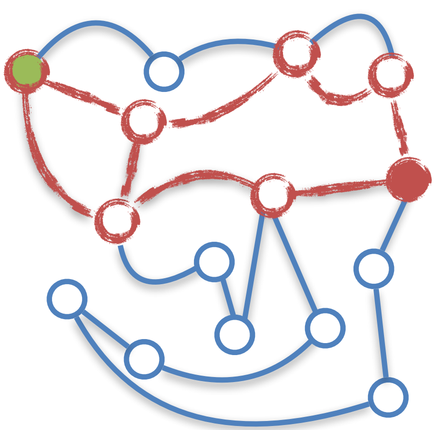
Motivation - Safety-critical Systems



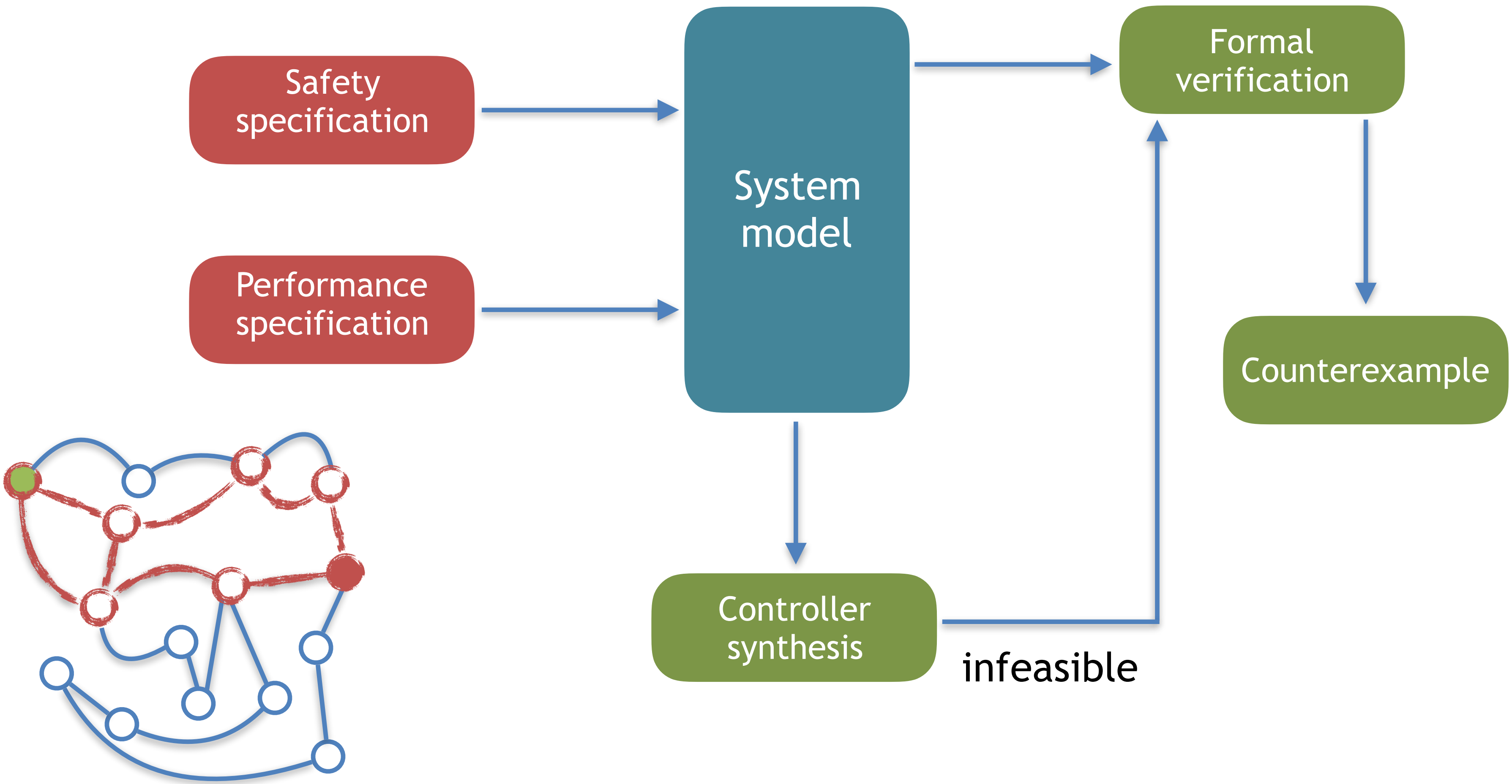


Motivation - Safety-critical Systems

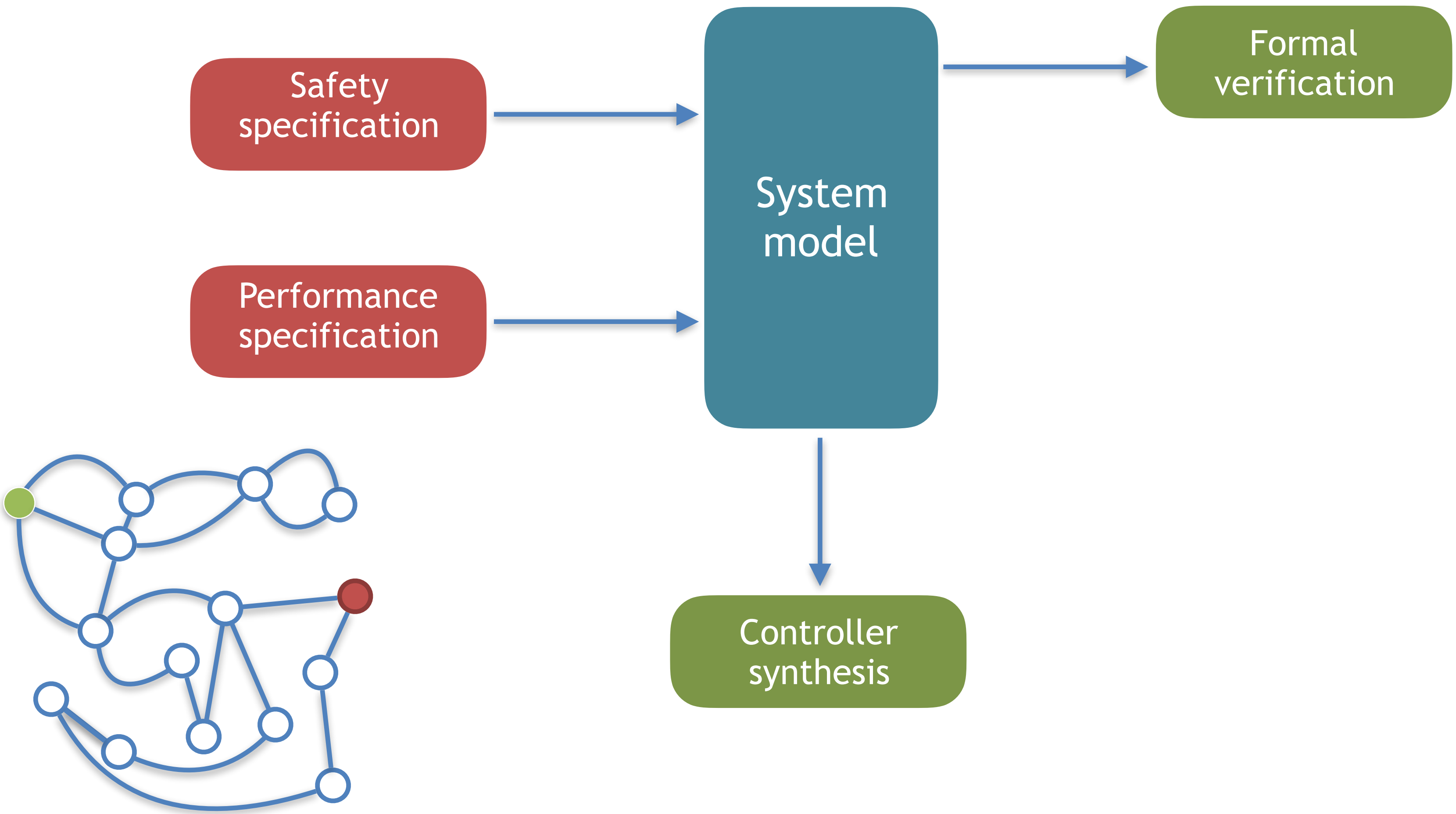




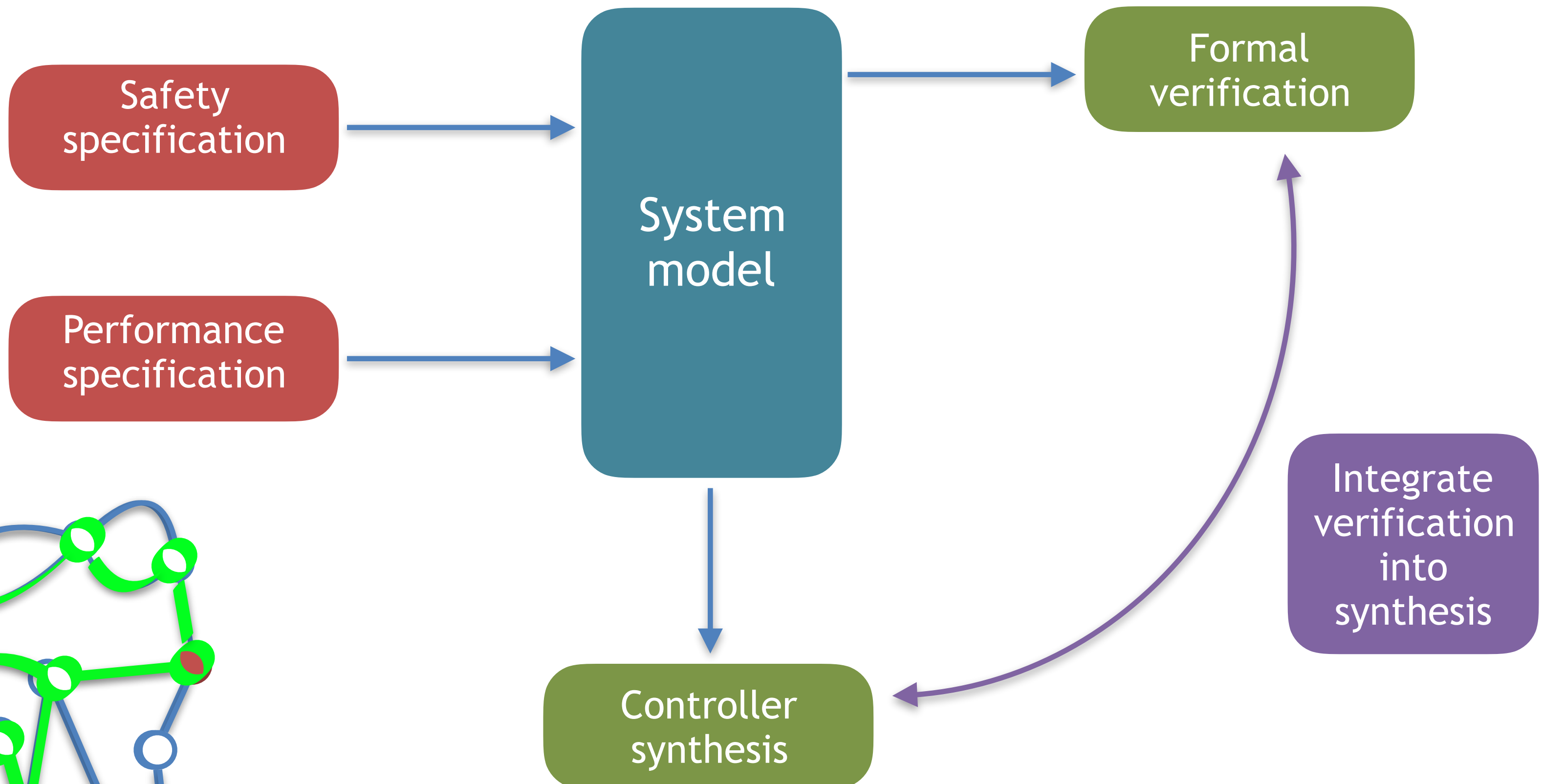
Motivation - Safety-critical Systems

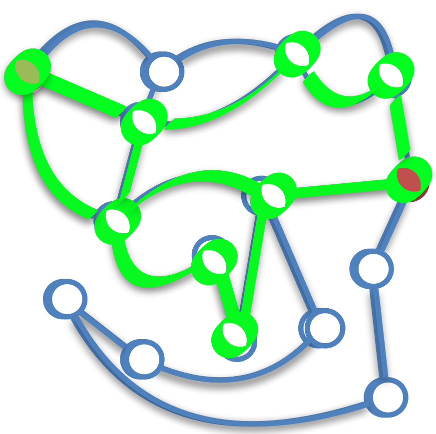


Motivation - Safety-critical Systems



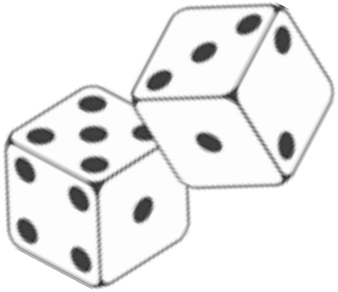
Motivation - Safety-critical Systems



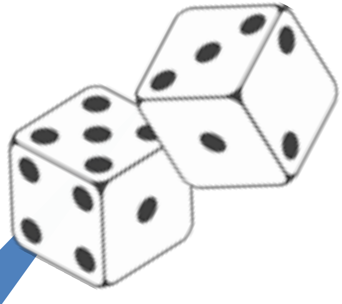


Exclude safety-critical behavior

Randomization



Randomization



Algorithms speed-up

- Probabilistic quicksort
- Rabin-Miller primality test
- Verification of matrix multiplication

Randomization

Deterministic techniques fail - symmetry breaking

- Dinning philosopher problem (Lehmann & Rabin '81)
- Leader election (Angluin '80)
- Ethernet's randomized exponential backoff (IEEE 802.3)

Algorithms speed-up

- Probabilistic quicksort
- Rabin-Miller primality test
- Verification of matrix multiplication

Randomization

Probabilistic programs are simple and intuitive to understand

```
repeat
  c := coin_flip(0.5)
until (c=heads)
```

Deterministic techniques fail - symmetry breaking

- Dinning philosopher problem (Lehmann & Rabin ’81)
- Leader election (Angluin ’80)
- Ethernet’s randomized exponential backoff (IEEE 802.3)

Algorithms speed-up

- Probabilistic quicksort
- Rabin-Miller primality test
- Verification of matrix multiplication

Randomization

Probabilistic programs are simple and intuitive to understand

repeat
$\quad c := \texttt{coin\_flip}(0.5)$
until $(c{=}heads)$

Deterministic techniques fail - symmetry breaking

- Dinning philosopher problem (Lehmann & Rabin ’81)
- Leader election (Angluin ’80)
- Ethernet’s randomized exponential backoff (IEEE 802.3)

Algorithms speed-up

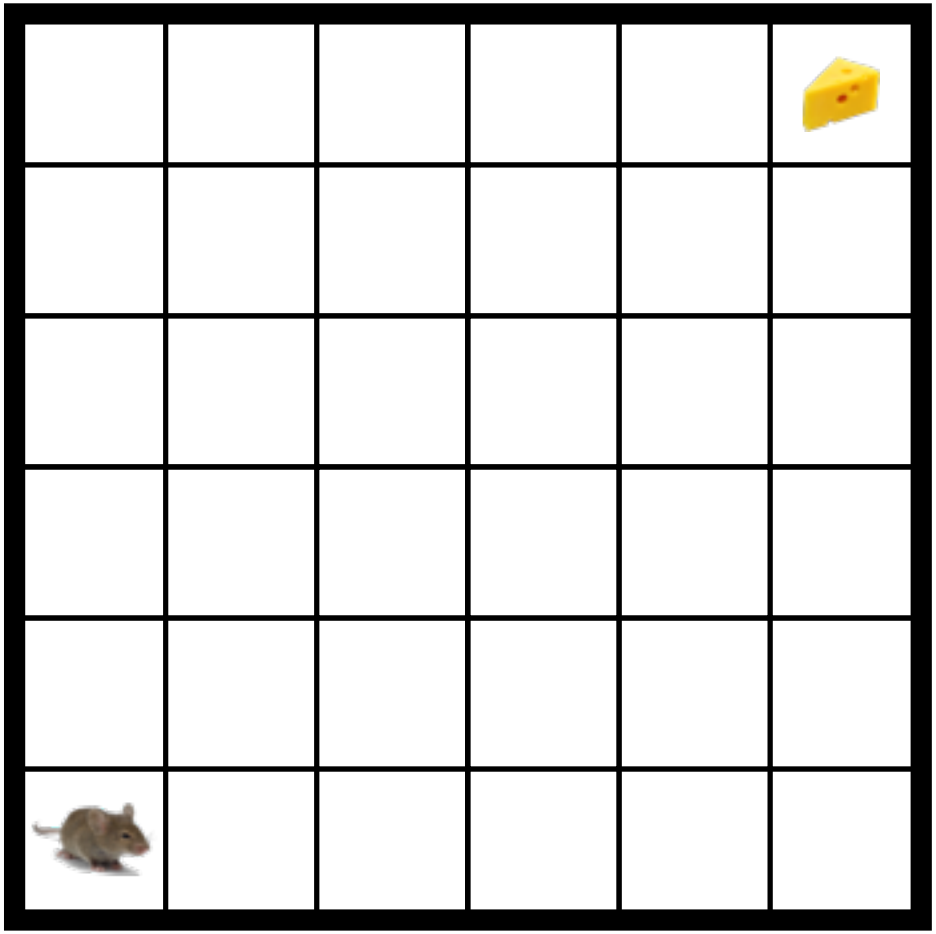
- Probabilistic quicksort
- Rabin-Miller primality test
- Verification of matrix multiplication

Applications in several domains

- cryptography
- biology
- communication
- computer vision
- robotics
- data management
- optimization

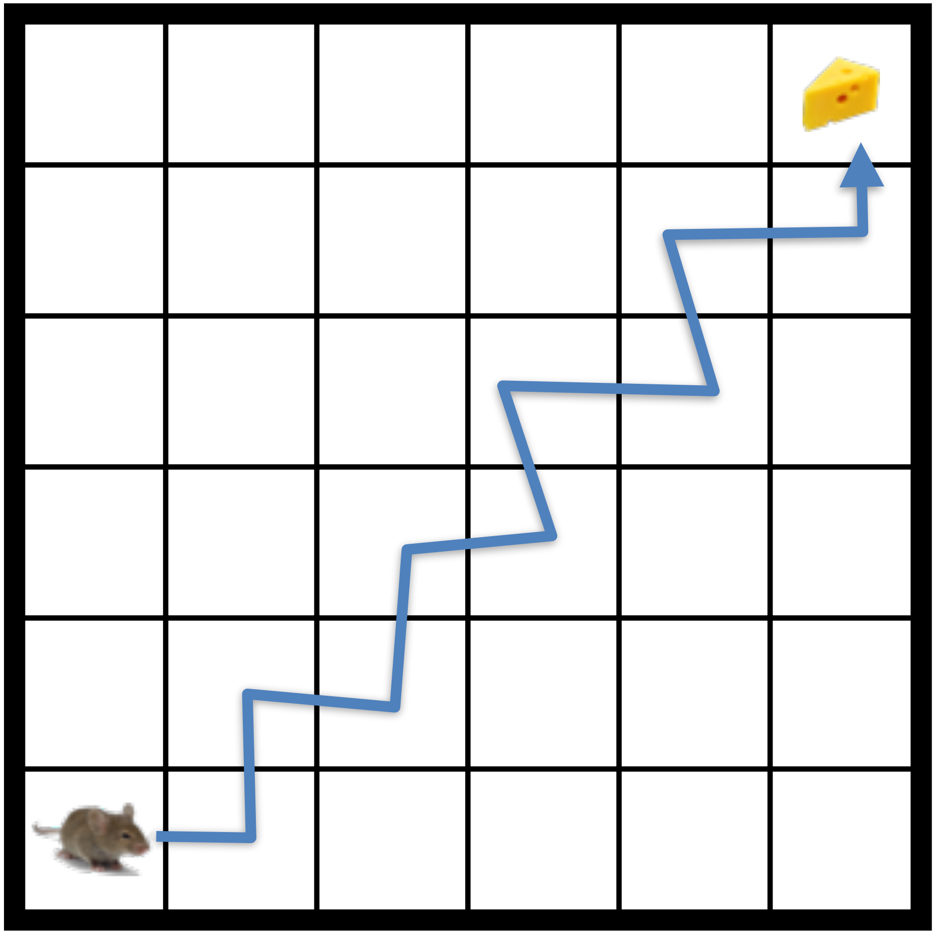
Help the Mouse

Find the best way to the cheese



Help the Mouse

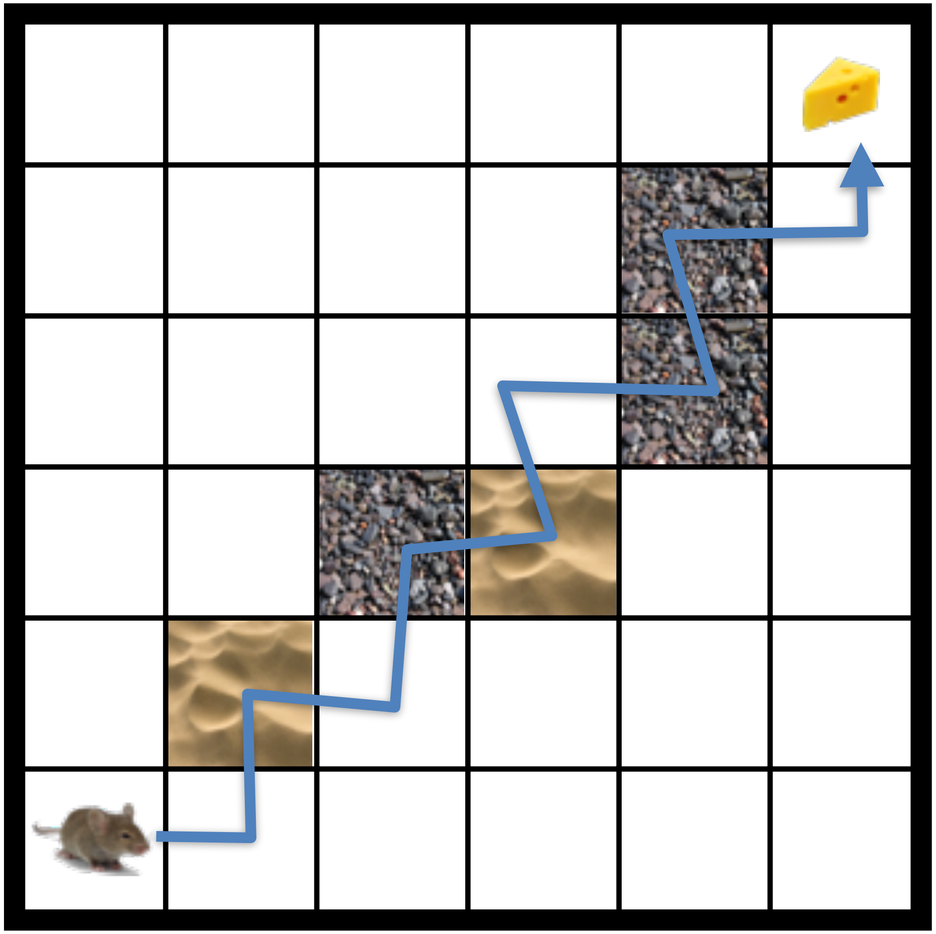
Find the best way to the cheese



Help the Mouse

Find the best way to the cheese

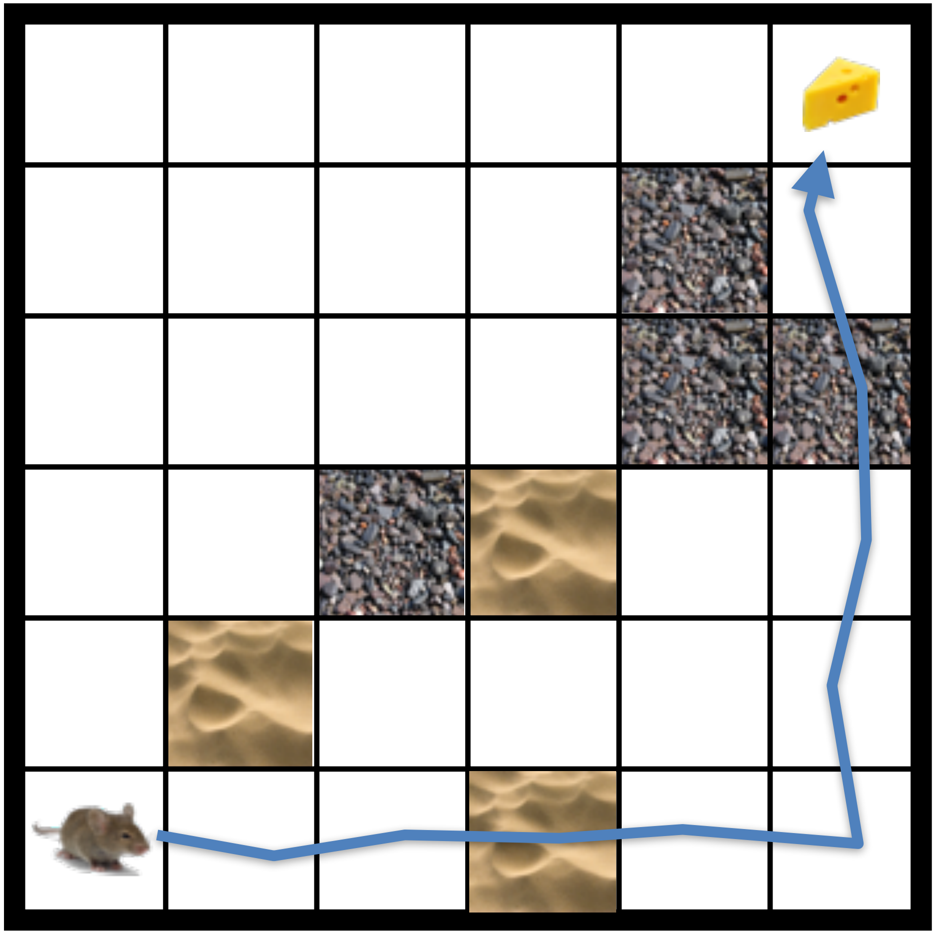
While moving mouse discovers exhausting surfaces



Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

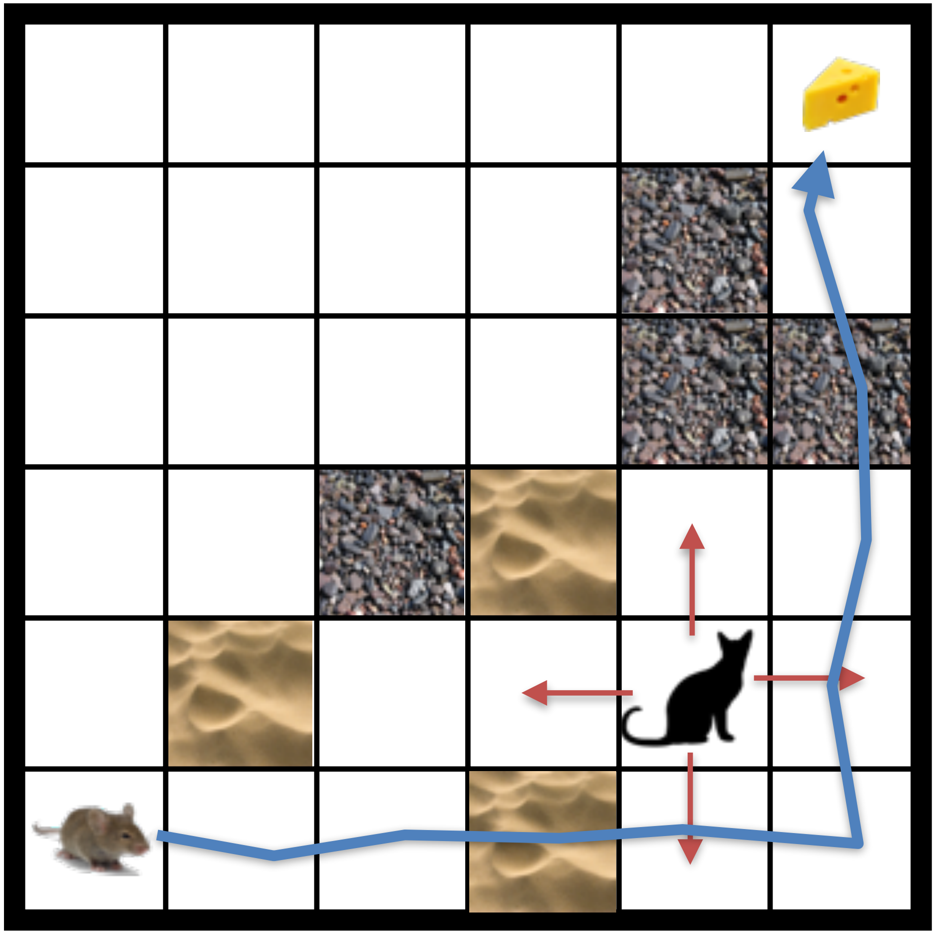


Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

Avoid randomly moving cat



Find safe and cost-optimal strategy to get to the cheese

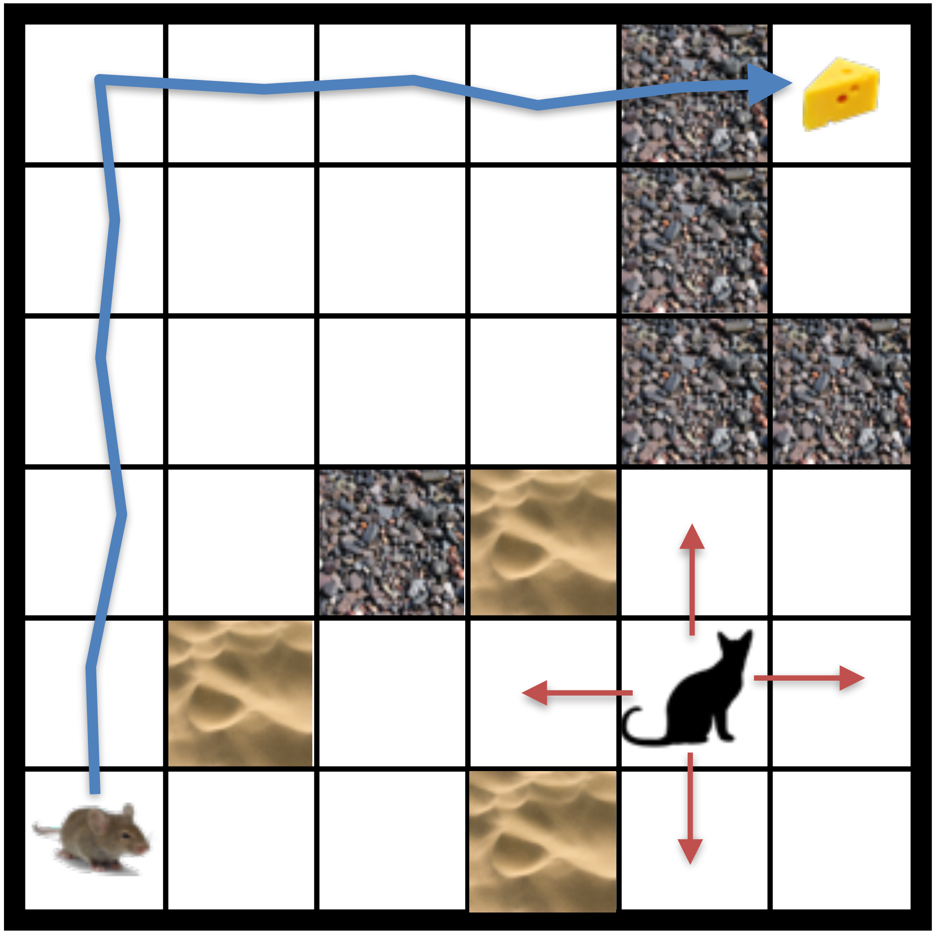
multi-objective model checking

Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

Avoid randomly moving cat



Find safe and cost-optimal strategy to get to the cheese

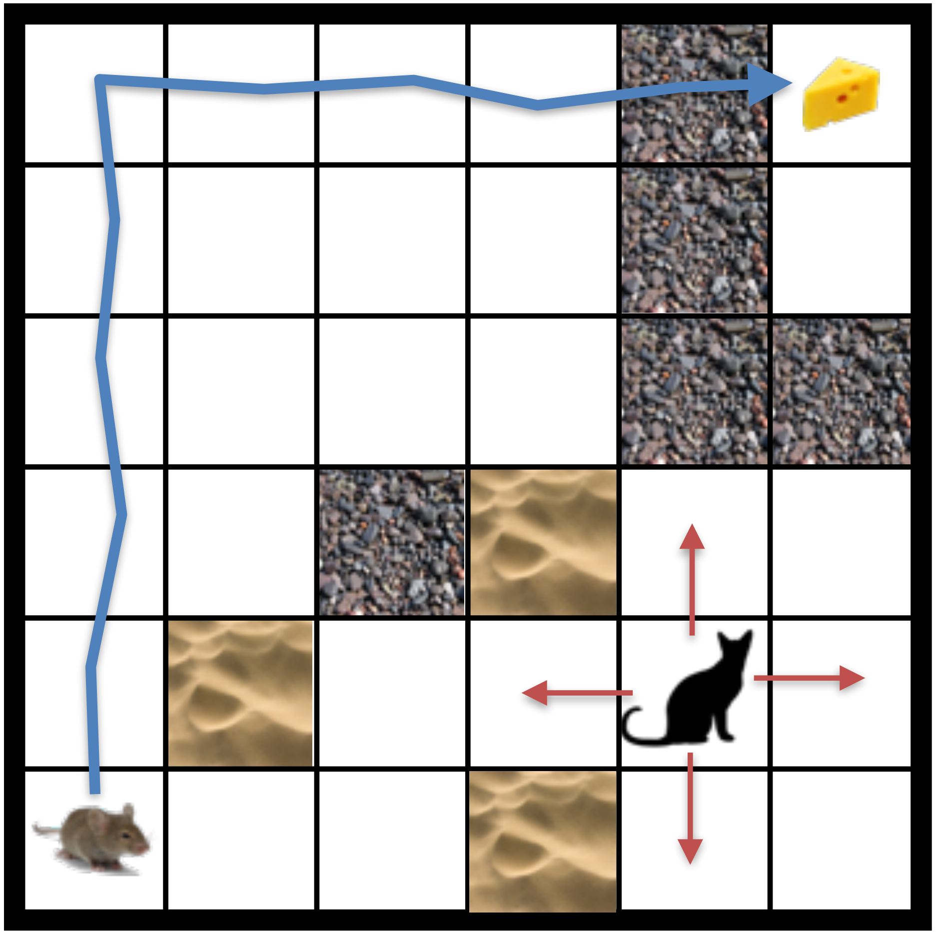
multi-objective model checking

Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

Avoid randomly moving cat



Find safe and cost-optimal strategy to get to the cheese

Cost is not known prior to exploring the grid

multi-objective model checking

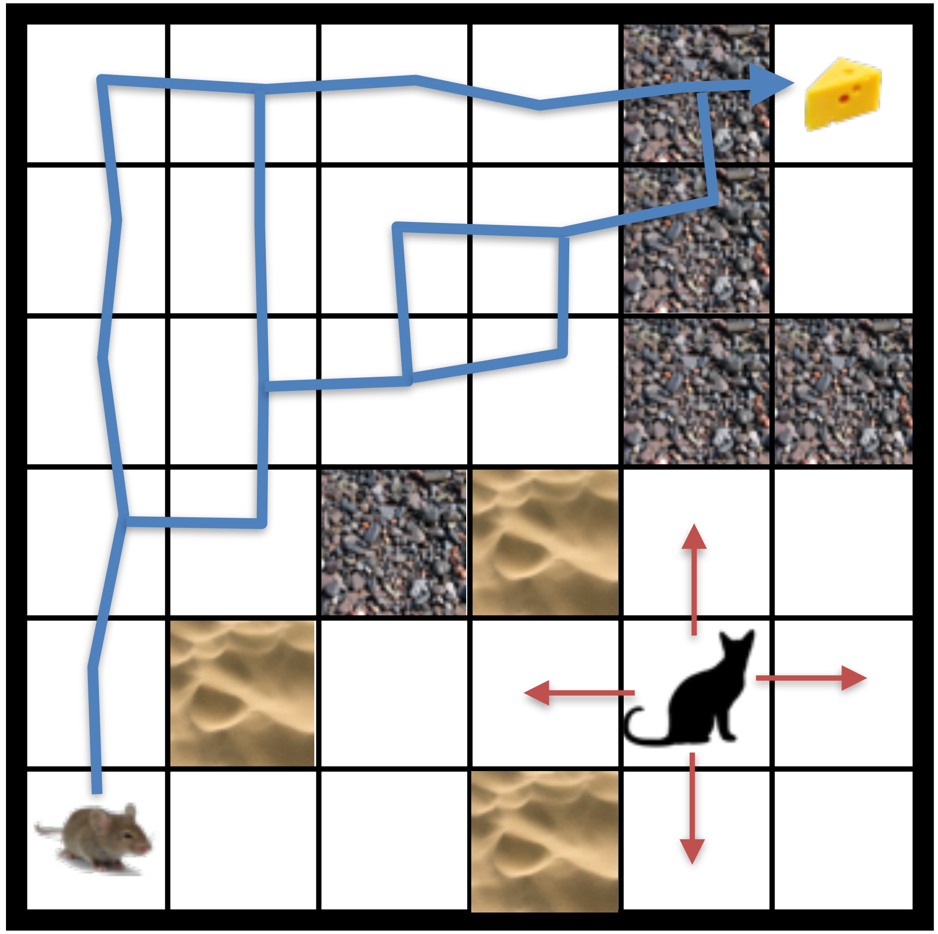
Help the Mouse

Find the best way to the cheese

While moving mouse discovers exhausting surfaces

Avoid randomly moving cat

Try all safe ways to the cheese (for future mice)



Find safe and cost-optimal strategy to get to the cheese

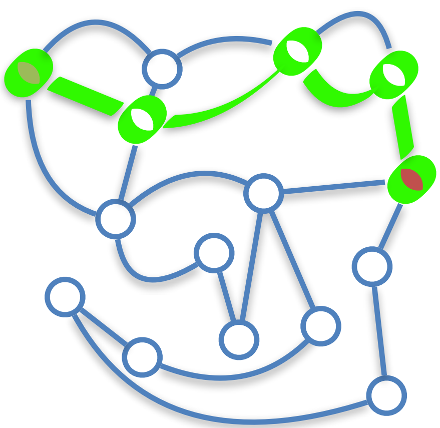
Cost is not known prior to exploring the grid

Deploy multiple strategies for safe exploration (permissive strategy)

Overview - Naive Approach

Safety Specification → Robot behavior

Performance Specification → MDP



Overview - Naive Approach

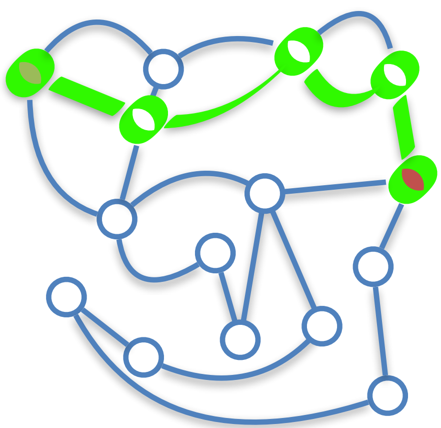
Safety Specification

Performance Specification

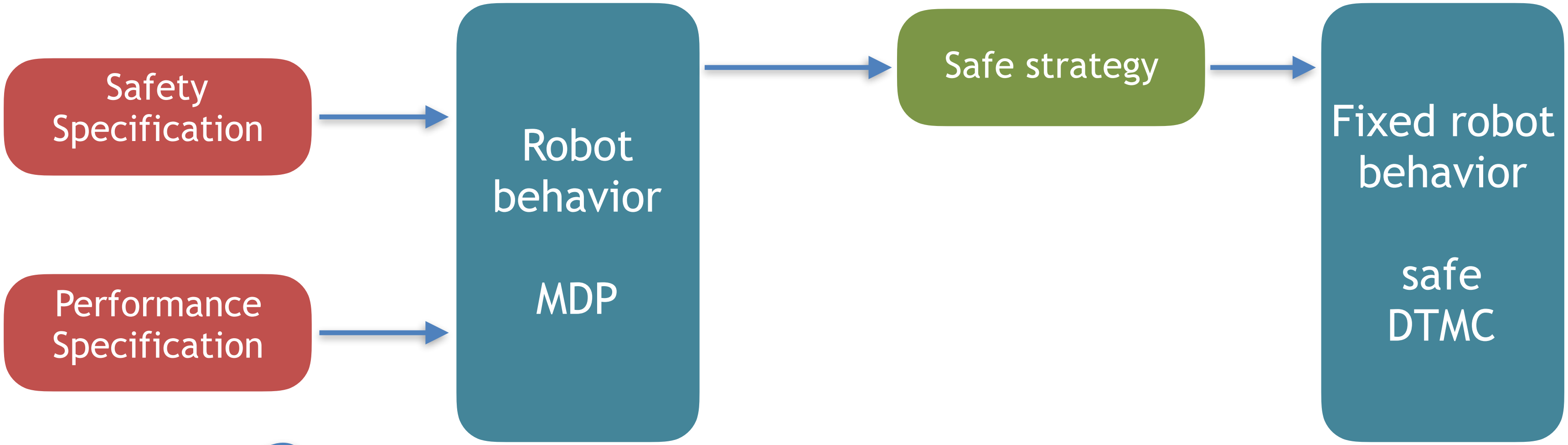
Robot behavior

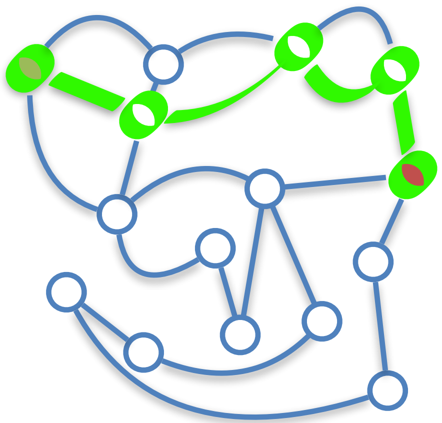
MDP

Safe strategy



Overview - Naive Approach





Overview - Naive Approach

Safety Specification

Performance Specification

Robot behavior

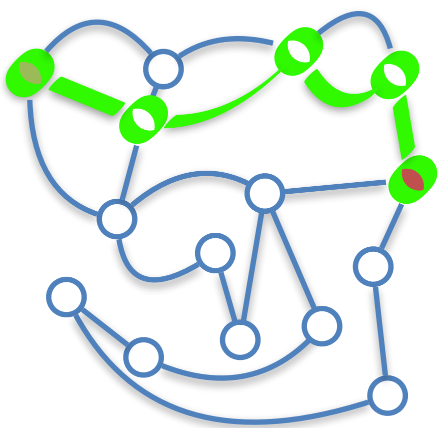
MDP

Safe strategy

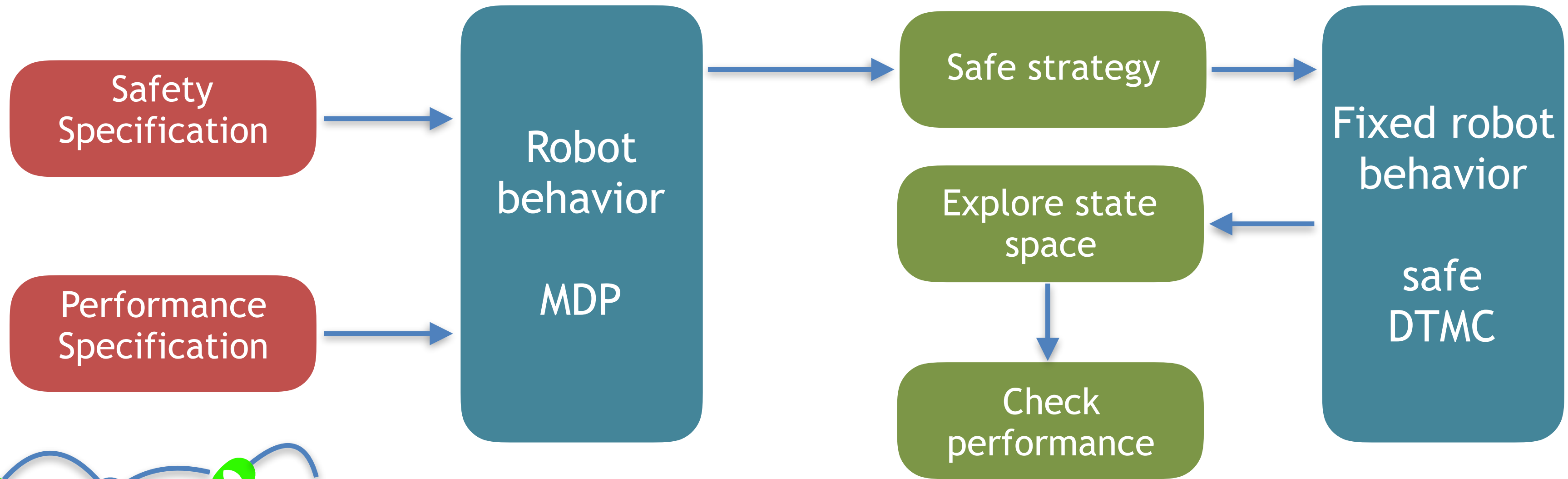
Explore state space

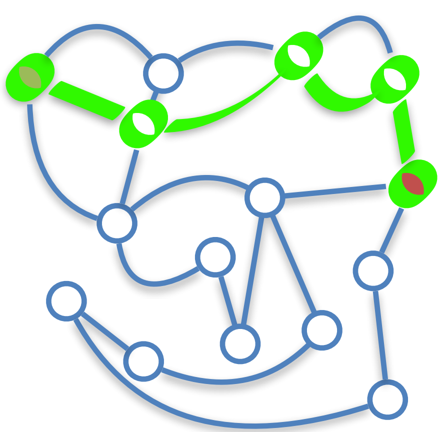
Fixed robot behavior

safe DTMC

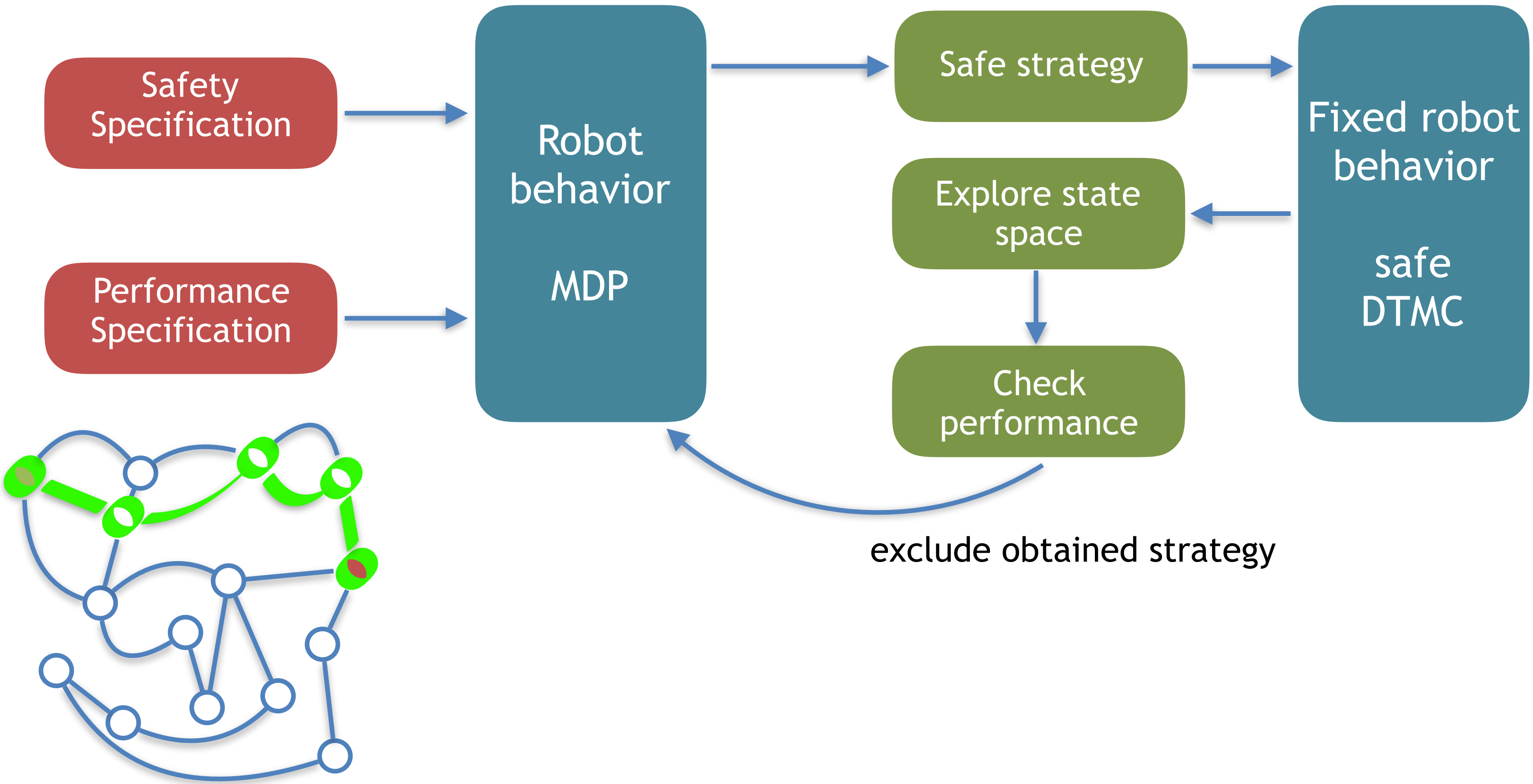


Overview - Naive Approach

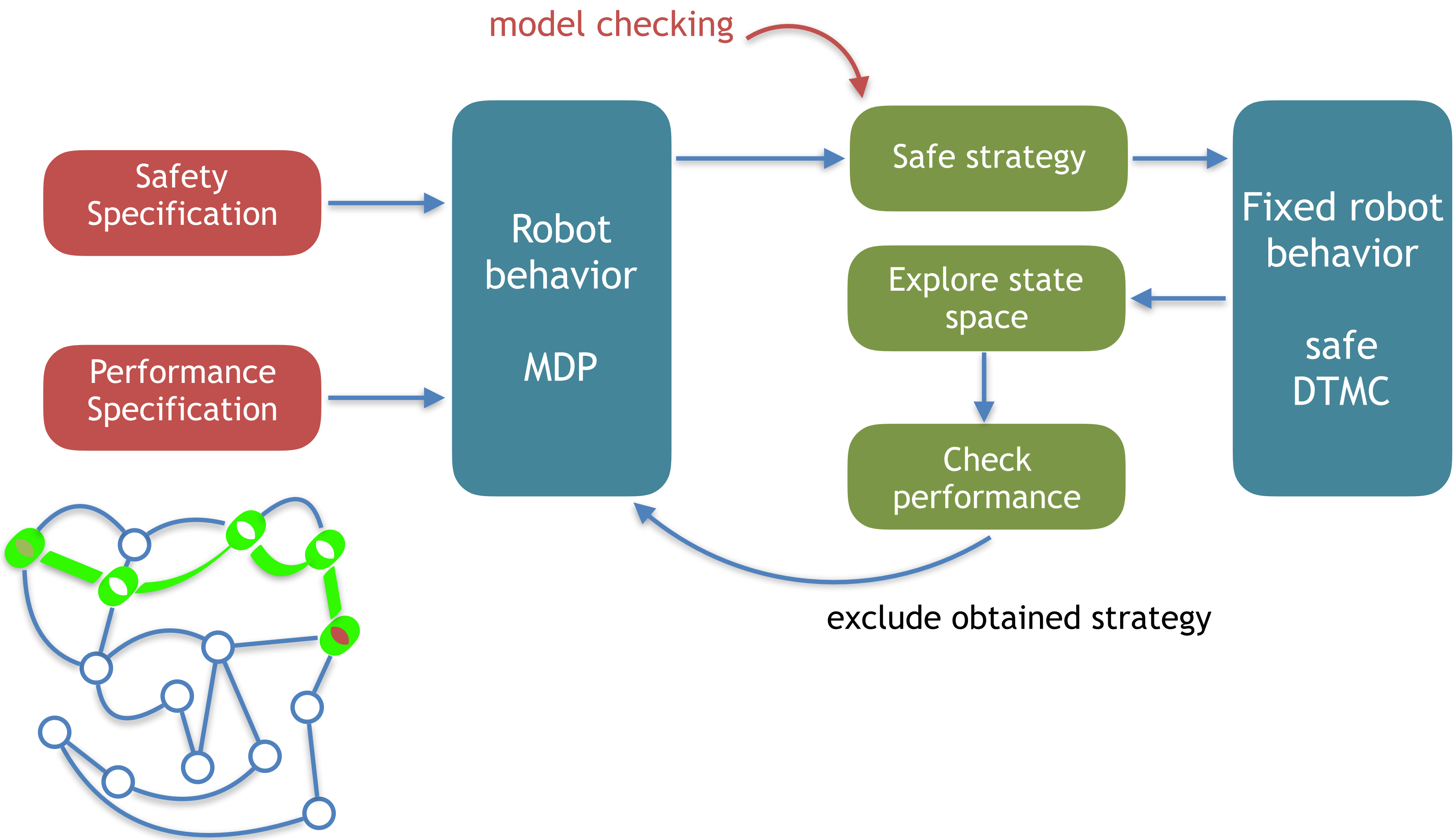




Overview - Naive Approach



Overview - Naive Approach



Overview - Naive Approach

model checking

deployment on robot

Safety Specification

Performance Specification

Robot behavior

MDP

Safe strategy

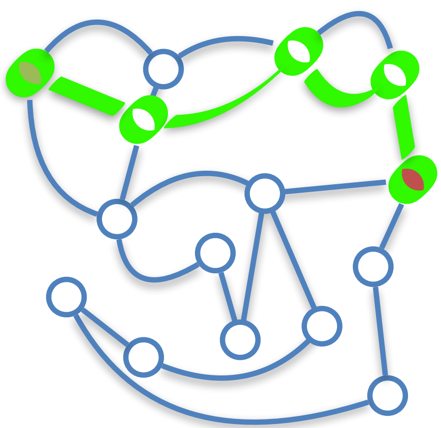
Explore state space

Check performance

Fixed robot behavior

safe DTMC

exclude obtained strategy



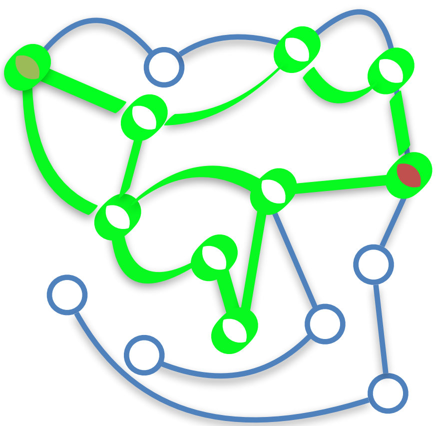
Overview - Permissive Approach

Safety Specification

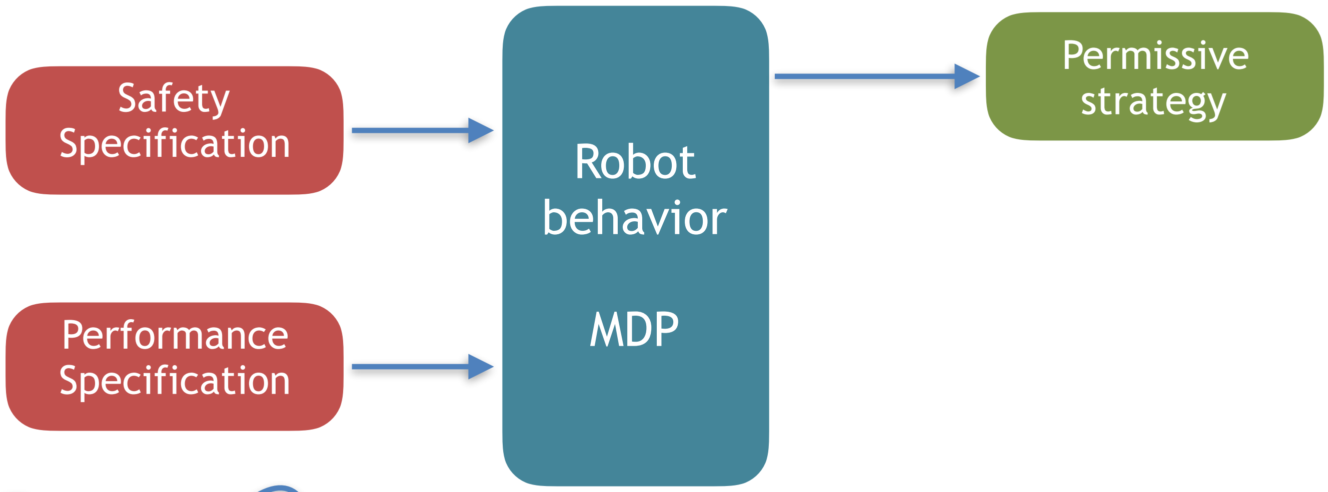
Performance Specification

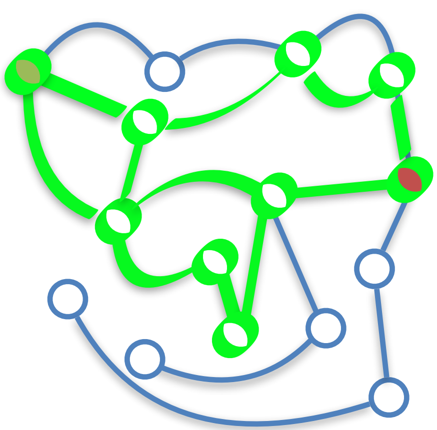
Robot behavior

MDP



Overview - Permissive Approach



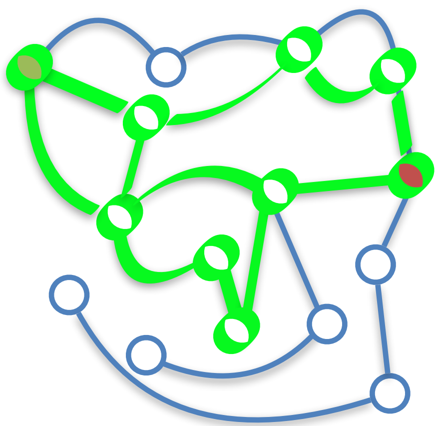


Overview - Permissive Approach

Safety Specification → Robot behavior MDP

Performance Specification → Robot behavior MDP

Robot behavior MDP → Permissive strategy → Restricted robot behavior safe MDP



Overview - Permissive Approach

Safety Specification

Performance Specification

Robot behavior

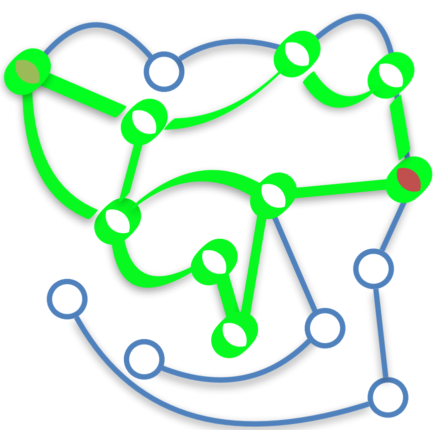
MDP

Permissive strategy

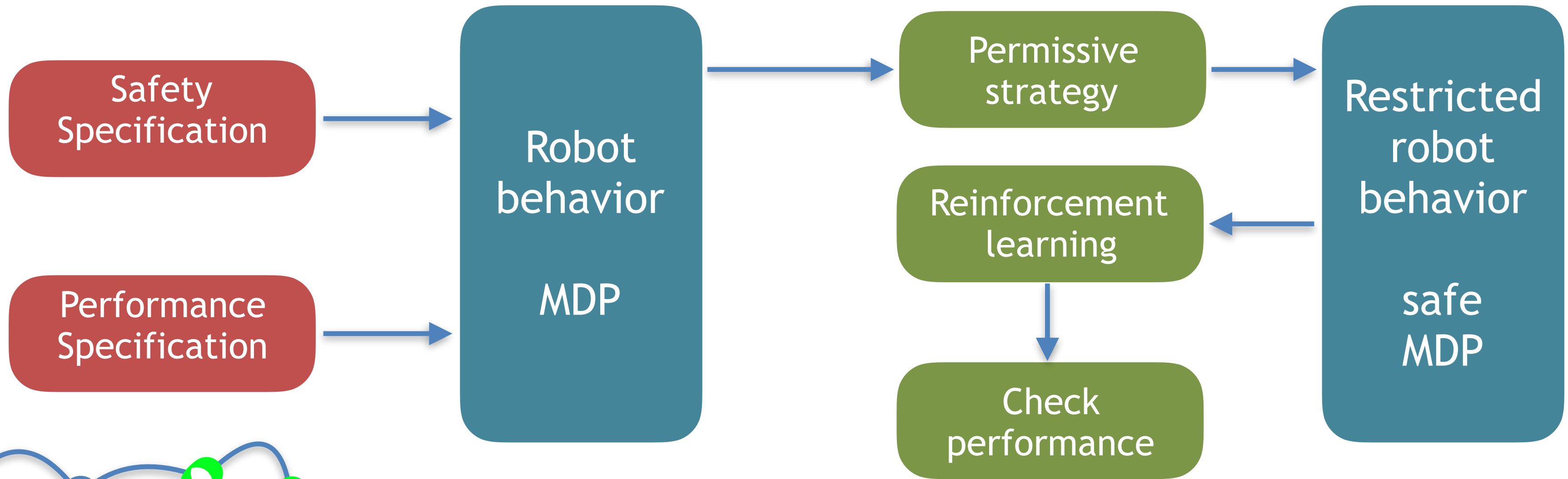
Reinforcement learning

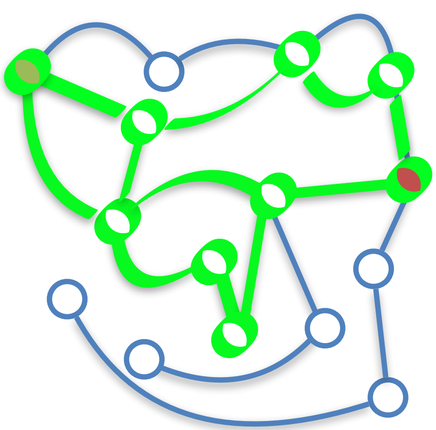
Restricted robot behavior

safe MDP

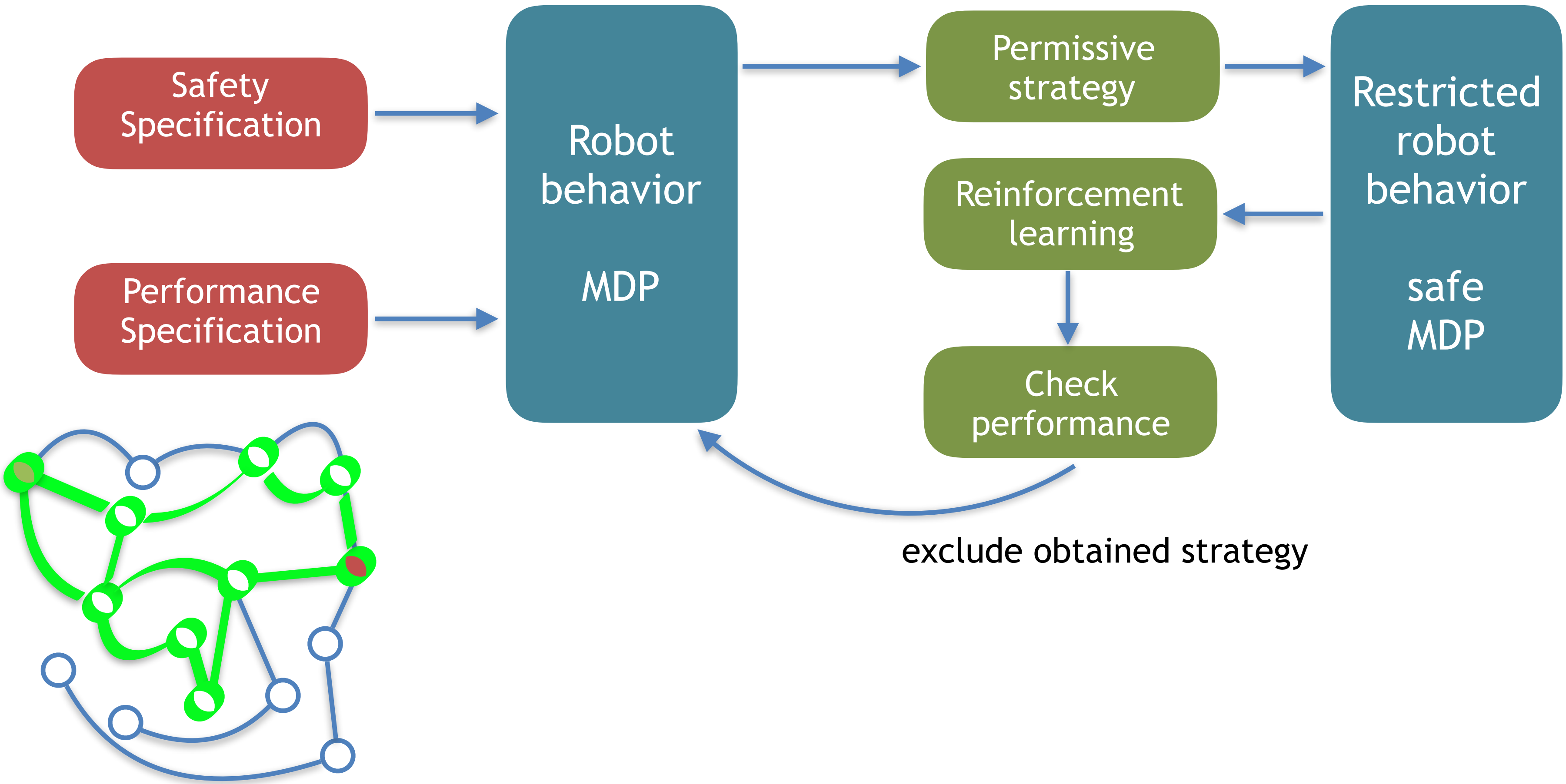


Overview - Permissive Approach

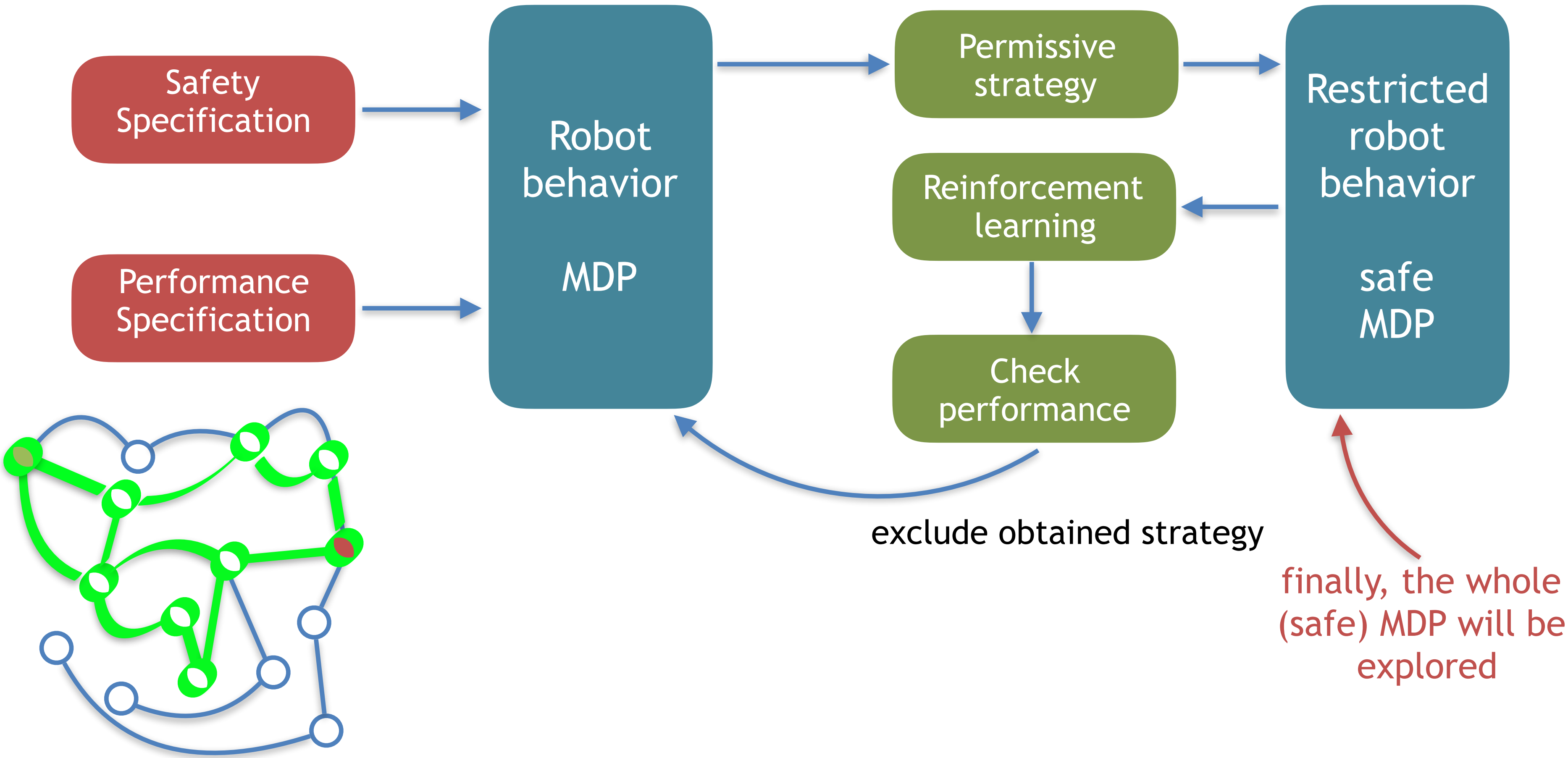




Overview - Permissive Approach



Overview - Permissive Approach



Related Work

Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, Mateusz Ujma:
Permissive Controller Synthesis for Probabilistic Systems. LMCS 2015

Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, Mateusz Ujma:
Permissive Controller Synthesis for Probabilistic Systems. TACAS 2014

Related Work

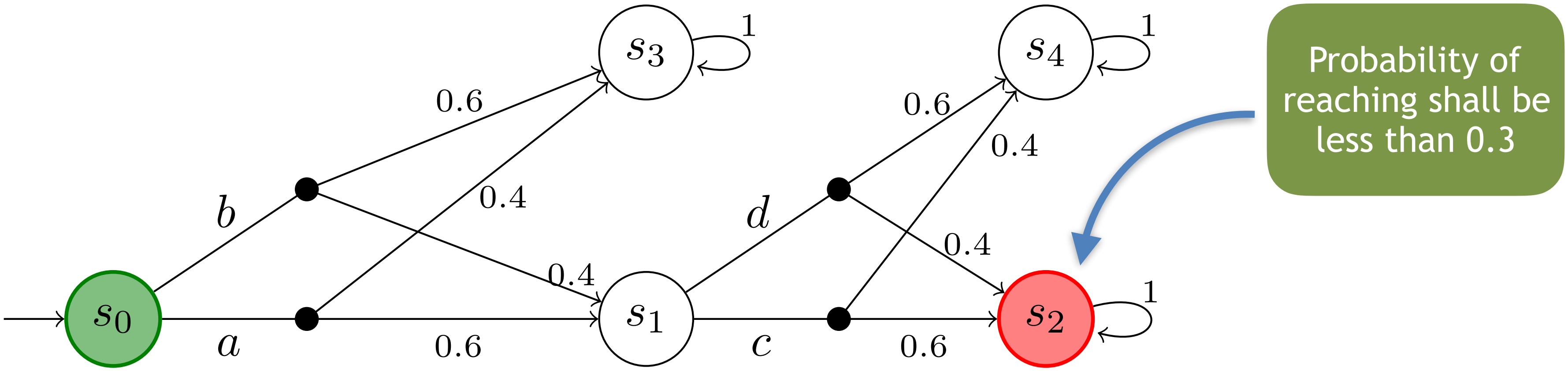
Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, Mateusz Ujma:
Permissive Controller Synthesis for Probabilistic Systems. LMCS 2015

Klaus Dräger, Vojtech Forejt, Marta Z. Kwiatkowska, David Parker, Mateusz Ujma:
Permissive Controller Synthesis for Probabilistic Systems. TACAS 2014

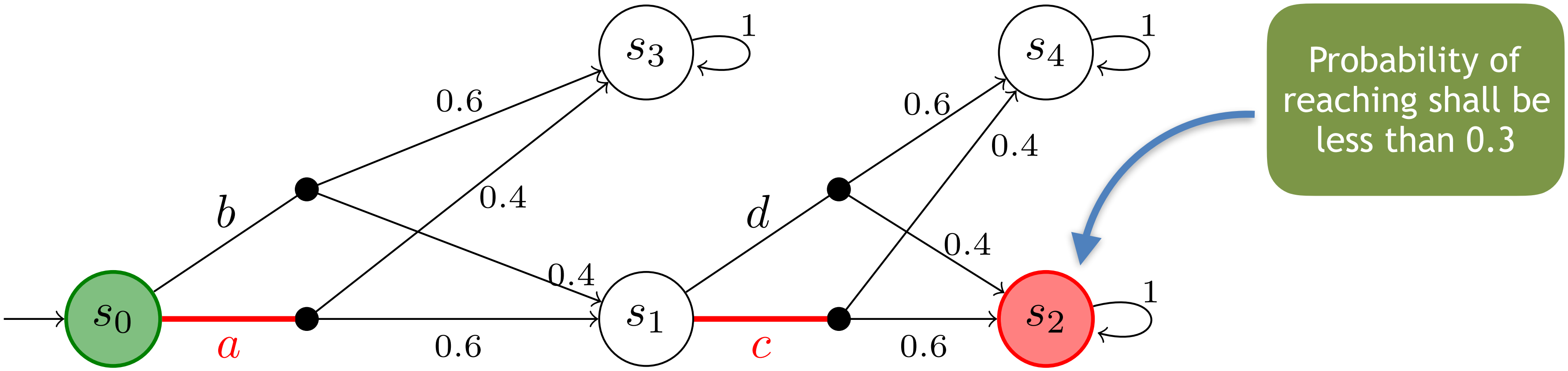
Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikucionis, Jakob Haahr Taankvist:
Uppaal Stratego. TACAS 2015

Kim G. Larsen, Marius Mikucionis, Marco Muniz, Jiri Srba and Jakob Haahr Taankvist:
Online and Compositional Learning of Controllers with Application to Floor Heating. TACAS 2016

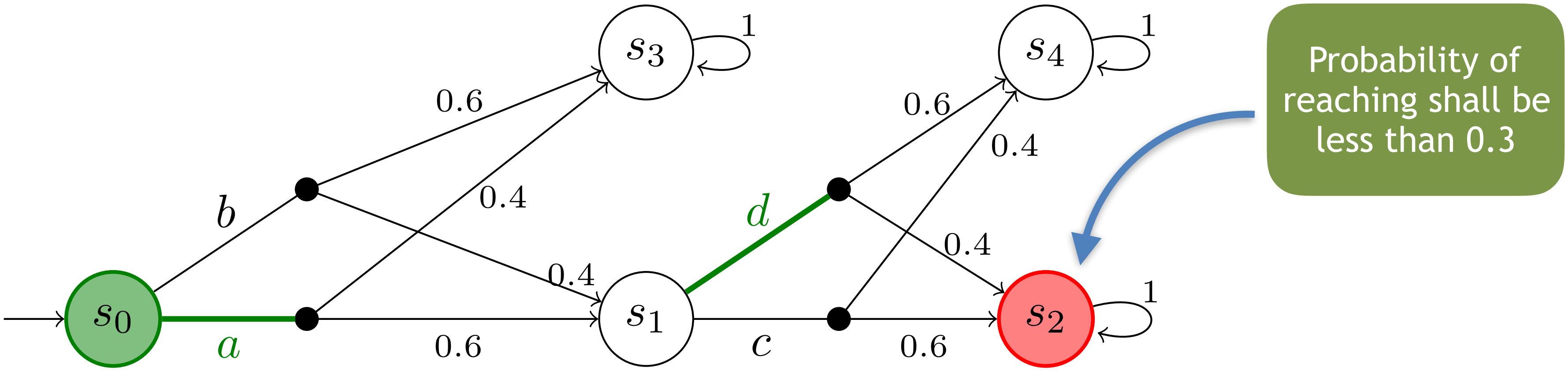
Permissive Strategies



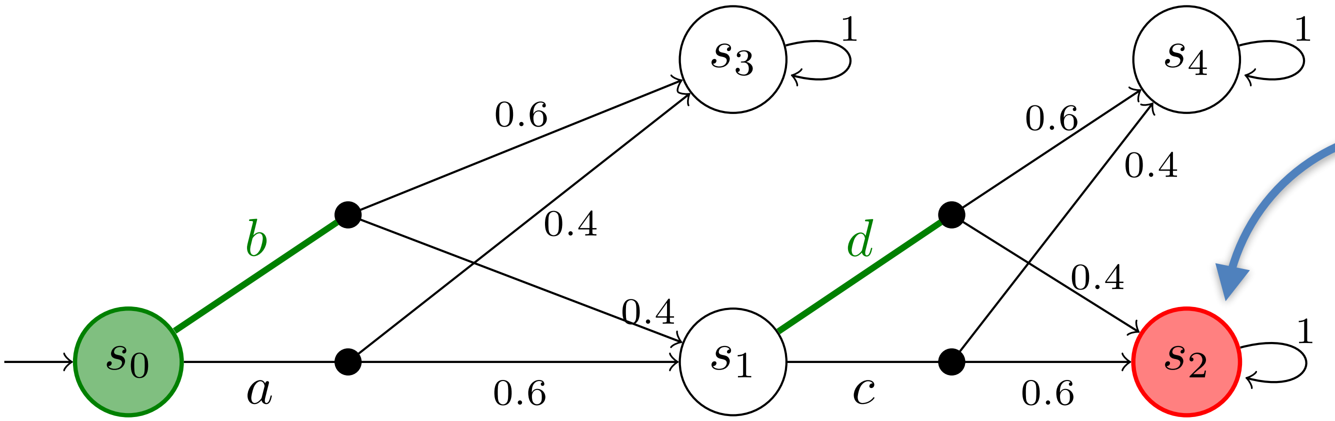
Permissive Strategies



Permissive Strategies

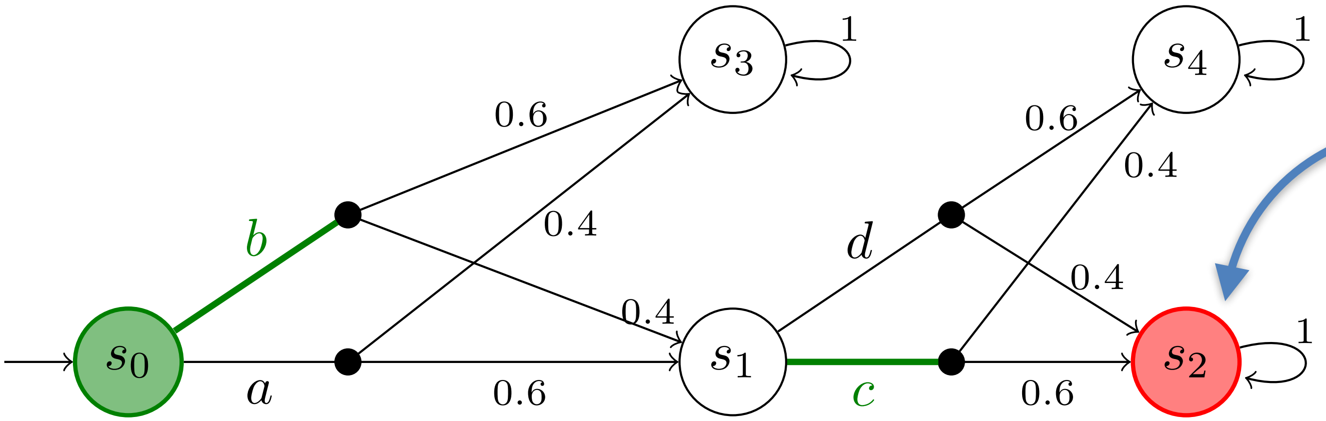


Permissive Strategies



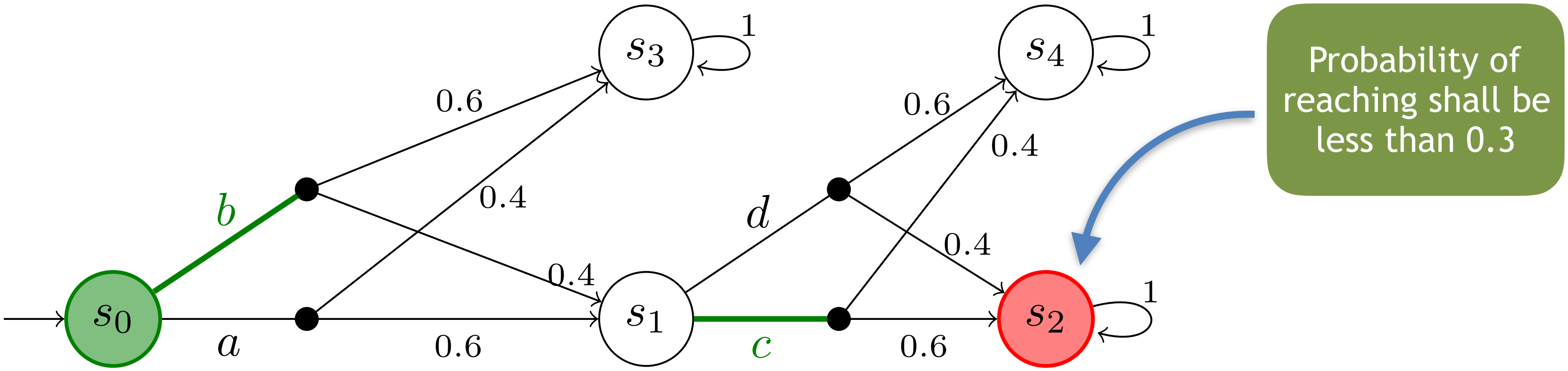
Probability of reaching shall be less than 0.3

Permissive Strategies



Probability of reaching shall be less than 0.3

Permissive Strategies



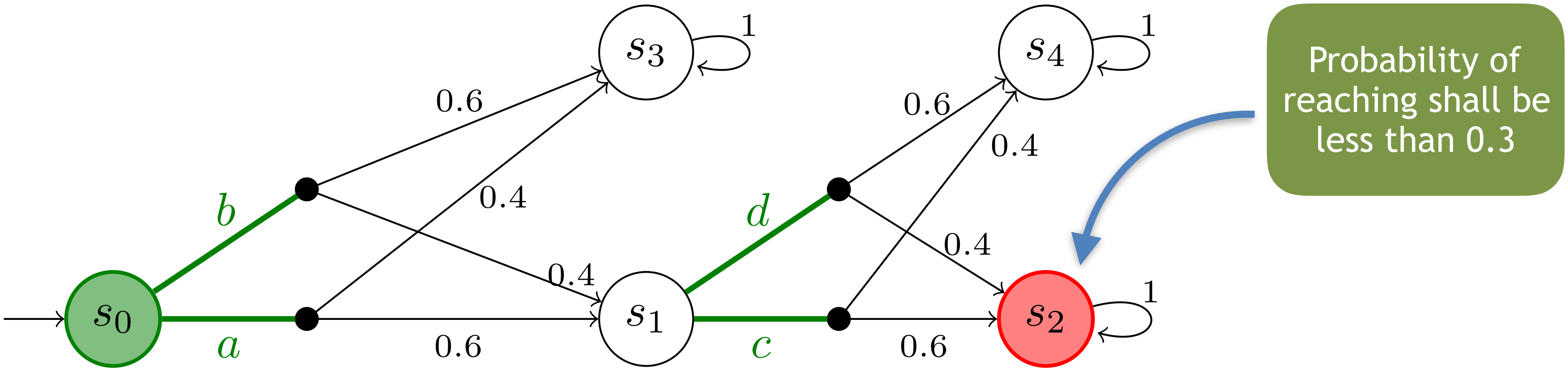
$s_0 \mapsto a,\ s_1 \mapsto d$

$s_0 \mapsto b,\ s_1 \mapsto c$ safe

$s_0 \mapsto b,\ s_1 \mapsto d$

$s_0 \mapsto a,\ s_1 \mapsto c$ unsafe

Permissive Strategies



$s_0 \mapsto a,\ s_1 \mapsto d$

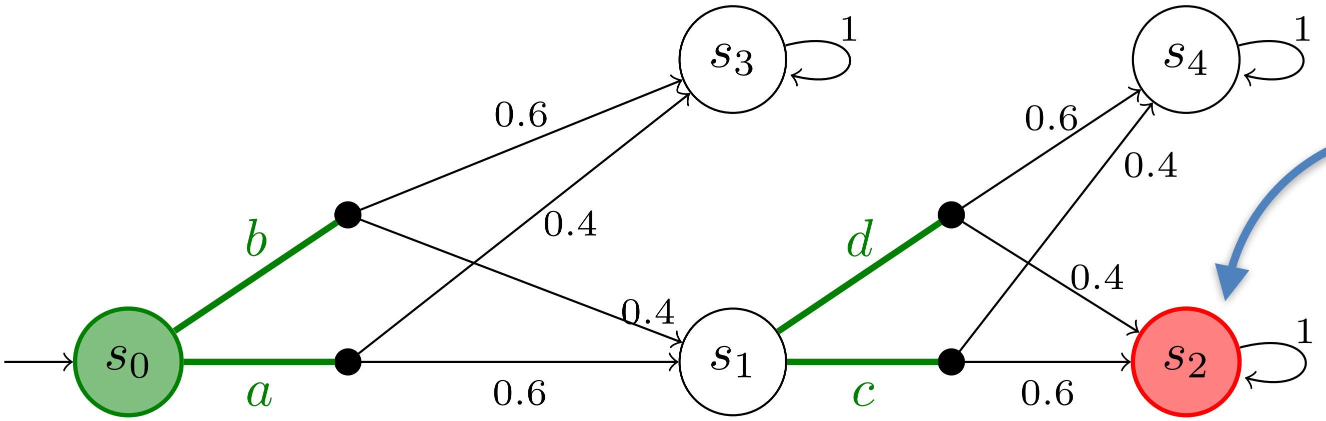
$s_0 \mapsto b,\ s_1 \mapsto c$ safe

$s_0 \mapsto b,\ s_1 \mapsto d$

Find maximal set of safe strategies

$s_0 \mapsto a,\ s_1 \mapsto c$ unsafe

Permissive Strategies



Probability of reaching shall be less than 0.3

$s_0 \mapsto a,\ s_1 \mapsto d$

$s_0 \mapsto b,\ s_1 \mapsto c$ safe

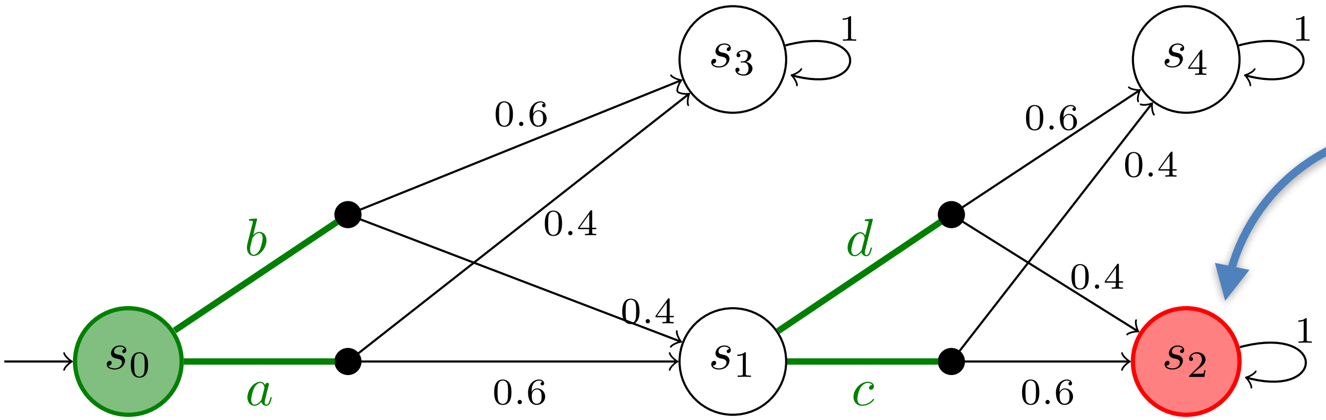
$s_0 \mapsto b,\ s_1 \mapsto d$

Find maximal set of safe strategies

$s_0 \mapsto a,\ s_1 \mapsto c$ unsafe

Exclude conflicting choices

Permissive Strategies



Probability of reaching shall be less than 0.3

$s_0 \mapsto a,\ s_1 \mapsto d$

$s_0 \mapsto b,\ s_1 \mapsto c$ safe

$s_0 \mapsto b,\ s_1 \mapsto d$

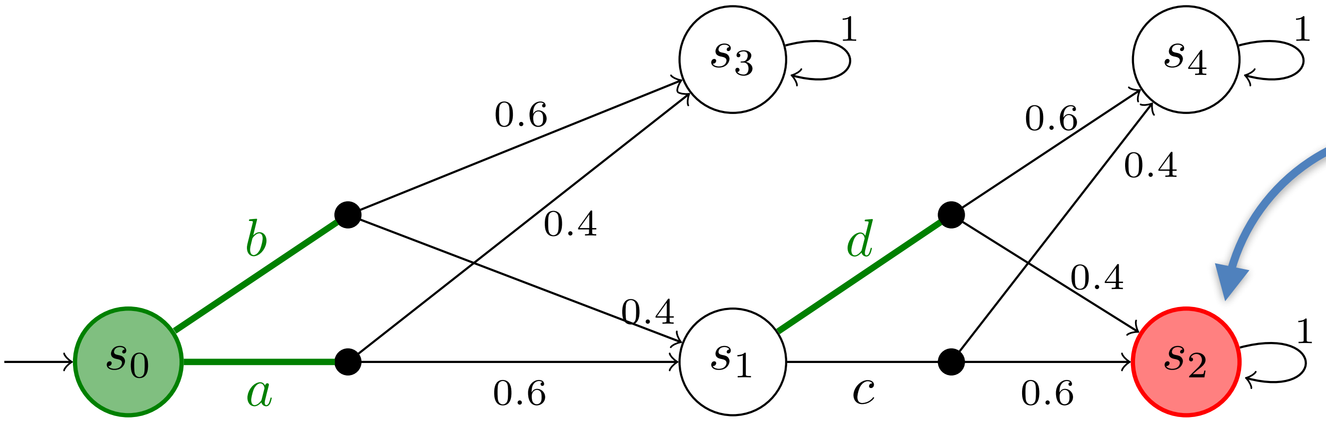
Find maximal set of safe strategies

$s_0 \mapsto a,\ s_1 \mapsto c$ unsafe

Exclude conflicting choices

exponentially many

Permissive Strategies



Probability of reaching shall be less than 0.3

$s_0 \mapsto a,\ s_1 \mapsto d$

$s_0 \mapsto b,\ s_1 \mapsto c$ safe

$s_0 \mapsto b,\ s_1 \mapsto d$

Find maximal set of safe strategies

Find set of safe strategies

$s_0 \mapsto a,\ s_1 \mapsto c$ unsafe

Exclude conflicting choices

exponentially many

SMT for permissive strategies

$$p_{s_I} \leq \lambda$$

probability smaller than threshold

$$\forall s \in S. \quad \bigvee_{a \in Act(s)} y_{s,a}$$

at least one action per state

$$\forall s \in T. \quad p_s = 1$$

probability of target states is one

$$\forall s \in S. \forall a \in Act(s). \quad y_{s,a} \rightarrow p_s \geq \sum_{s' \in S} \mathcal{P}(s,a,s') \cdot p_{s'}$$

probability of each state is assigned the maximum under the permissive strategy

Variables

$y_{s,a}$ action is chosen at state

p_s probability of state

Solution induces safe permissive strategy

SMT for permissive strategies

$$p_{s_I} \leq \lambda$$

probability smaller than threshold

$$\forall s \in S. \quad \bigvee_{a \in Act(s)} y_{s,a}$$

at least one action per state

$$\forall s \in T. \quad p_s = 1$$

probability of target states is one

$$\forall s \in S. \forall a \in Act(s). \quad y_{s,a} \rightarrow p_s \geq \sum_{s' \in S} \mathcal{P}(s,a,s') \cdot p_{s'}$$

probability of each state is assigned the maximum under the permissive strategy

Variables

$y_{s,a}$ action is chosen at state

p_s probability of state

Solution induces safe permissive strategy

MILP
quantify permissiveness

Bounds on the Safe Optimum

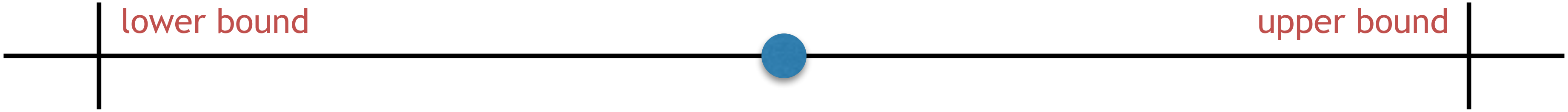
Reinforcement learning explores the state space

- (unknown) cost function $\rho\colon S \to \mathbb{R}$ is refined and
- unknown cost values are instantiated by given lower bounds yielding $\rho_l\colon S \to \mathbb{R}$

Bounds on the Safe Optimum

Reinforcement learning explores the state space

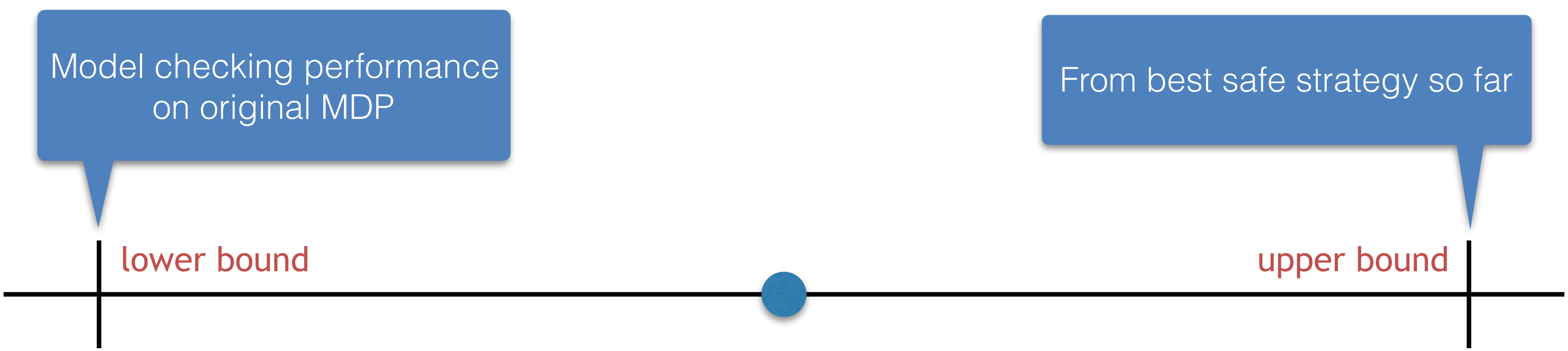
- (unknown) cost function $\rho\colon S \to \mathbb{R}$ is refined and
- unknown cost values are instantiated by given lower bounds yielding $\rho_l\colon S \to \mathbb{R}$



Bounds on the Safe Optimum

Reinforcement learning explores the state space

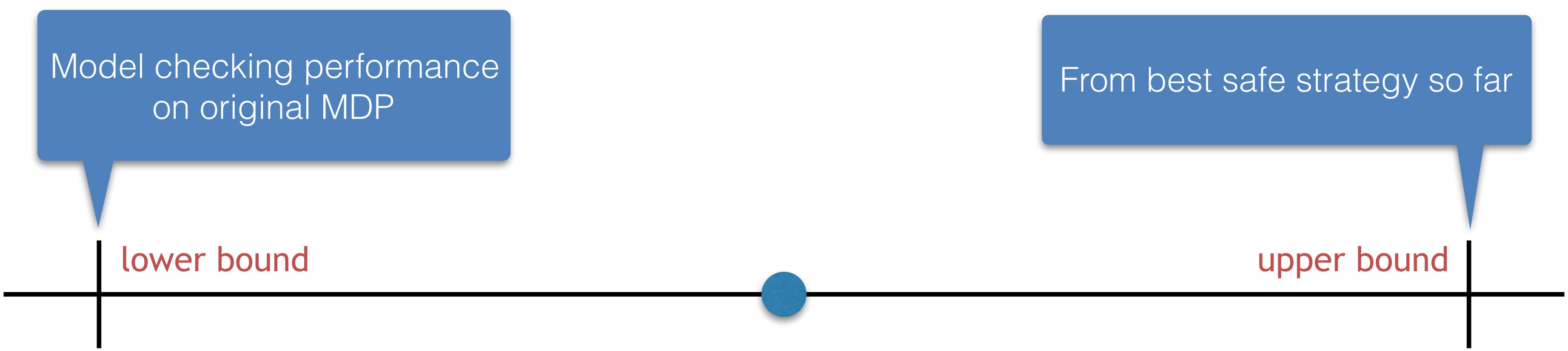
- (unknown) cost function $\rho\colon S \to \mathbb{R}$ is refined and
- unknown cost values are instantiated by given lower bounds yielding $\rho_l\colon S \to \mathbb{R}$



Bounds on the Safe Optimum

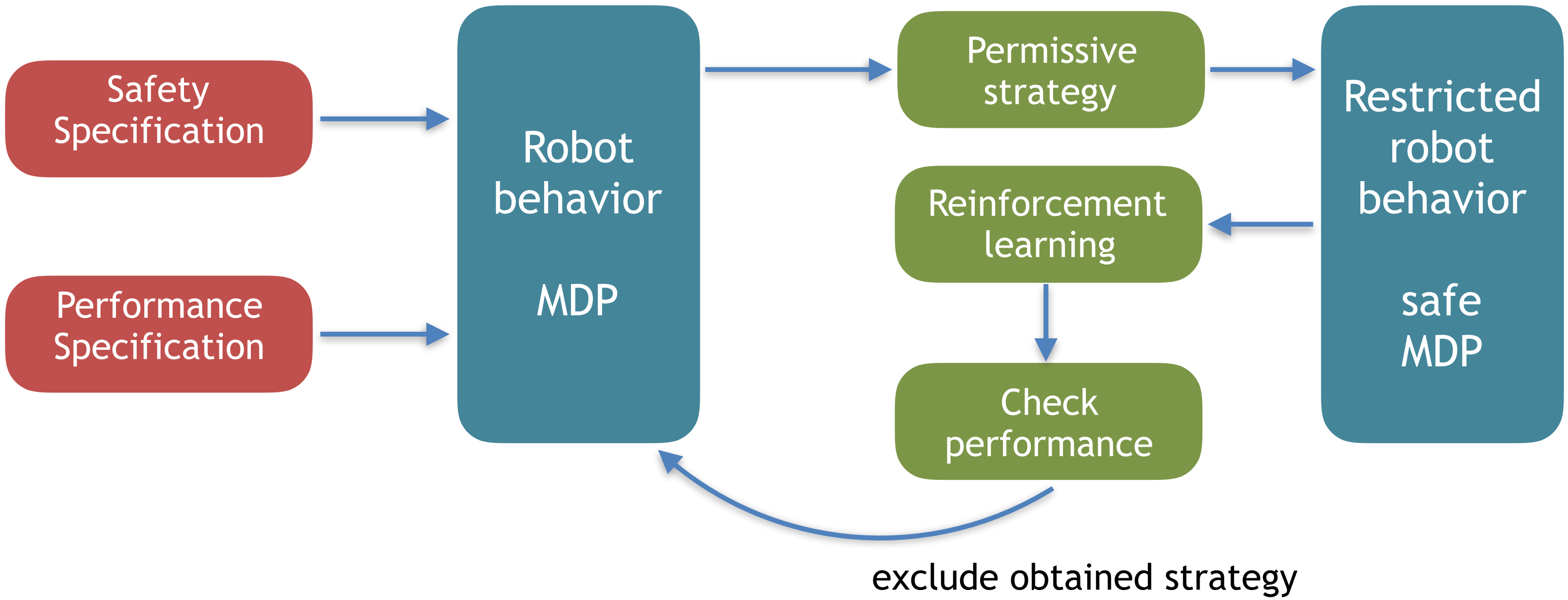
Reinforcement learning explores the state space

- (unknown) cost function $\rho\colon S \to \mathbb{R}$ is refined and
- unknown cost values are instantiated by given lower bounds yielding $\rho_l\colon S \to \mathbb{R}$

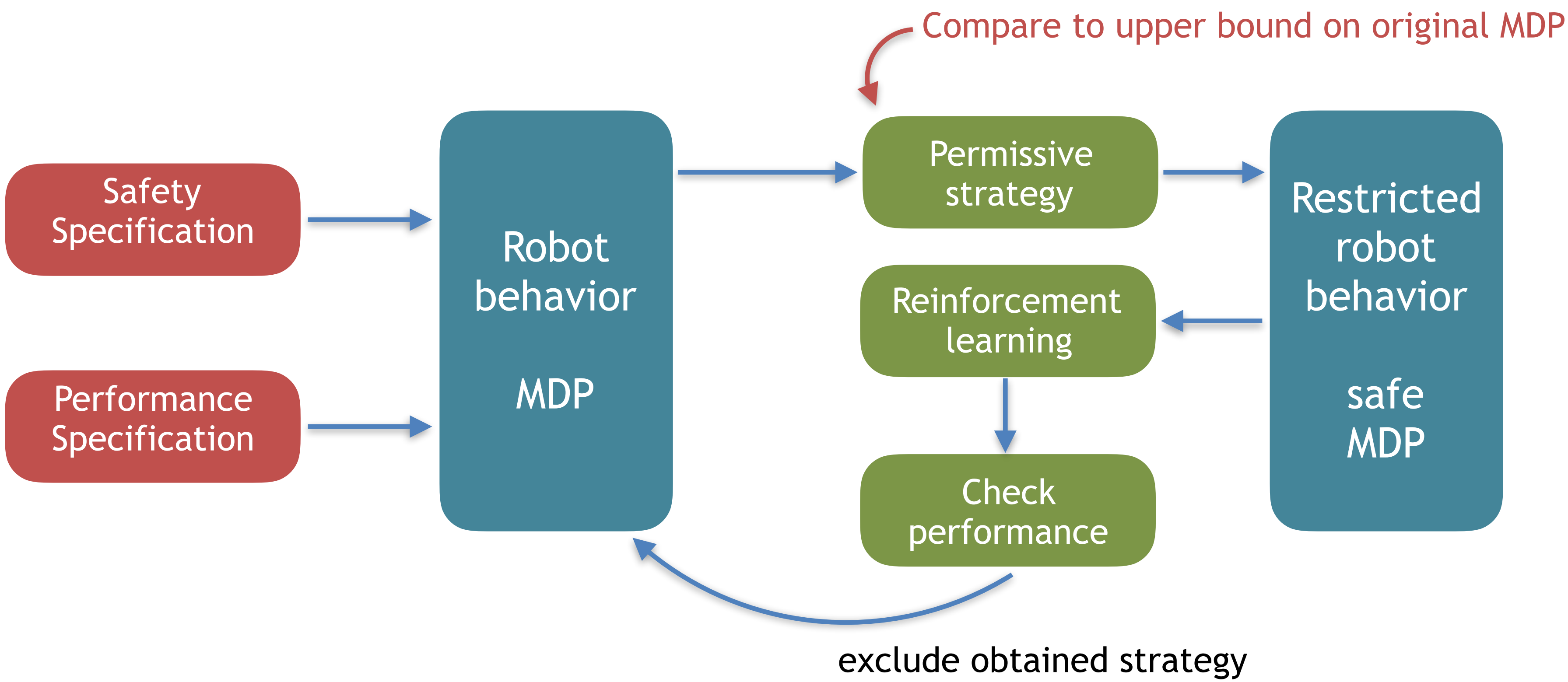


Multi-objective model checking yields even tighter bounds.

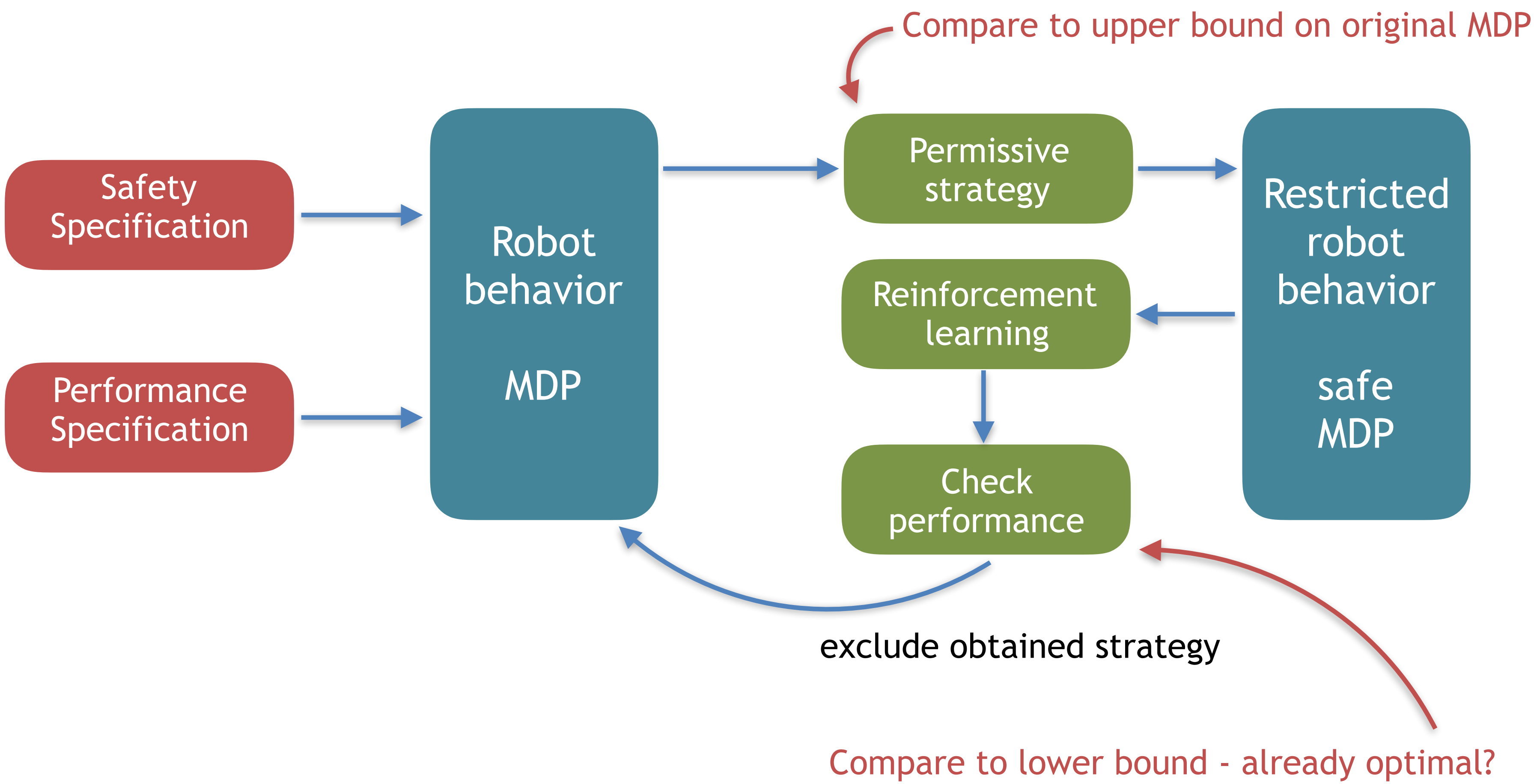
Incorporating the Bounds



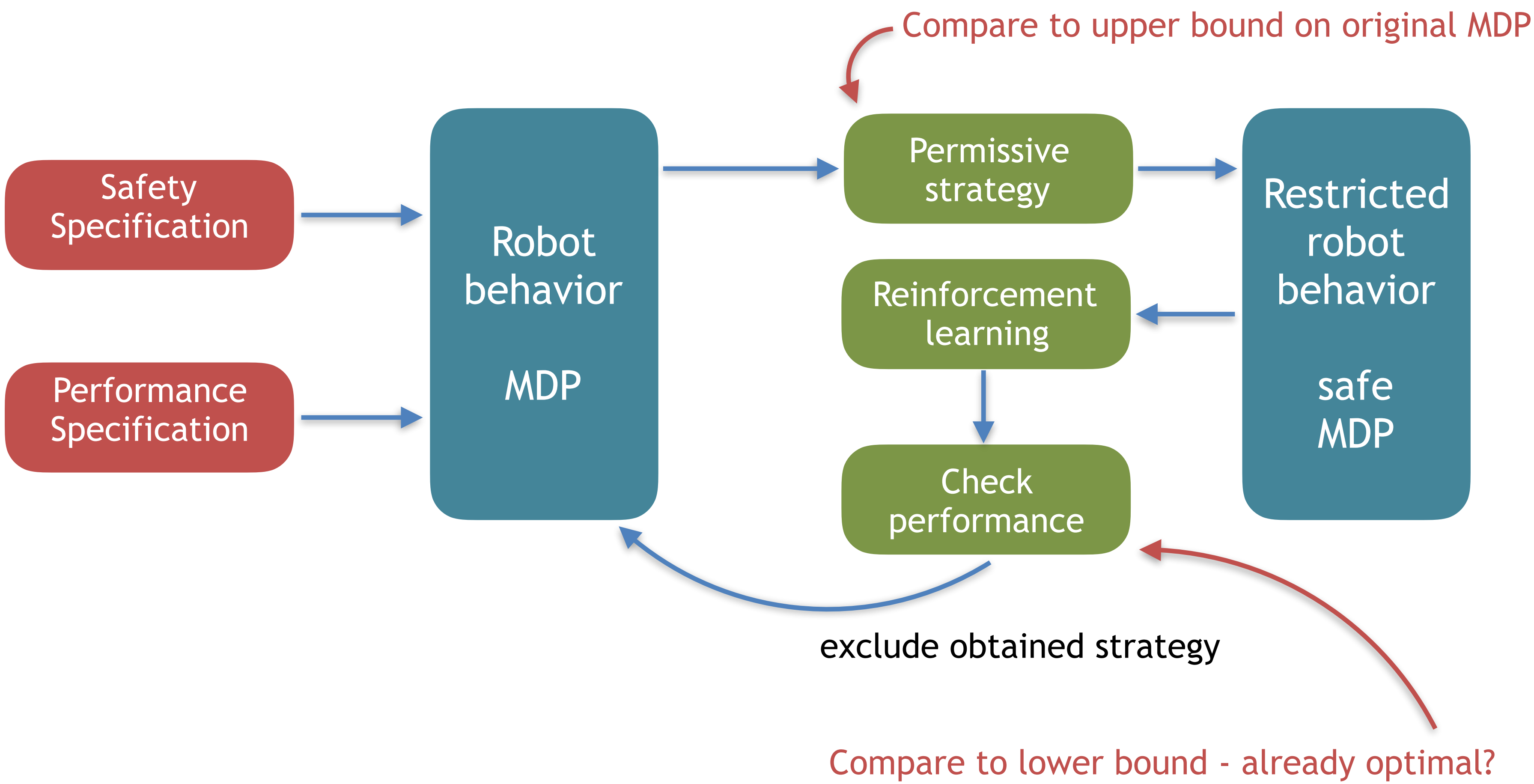
Incorporating the Bounds



Incorporating the Bounds

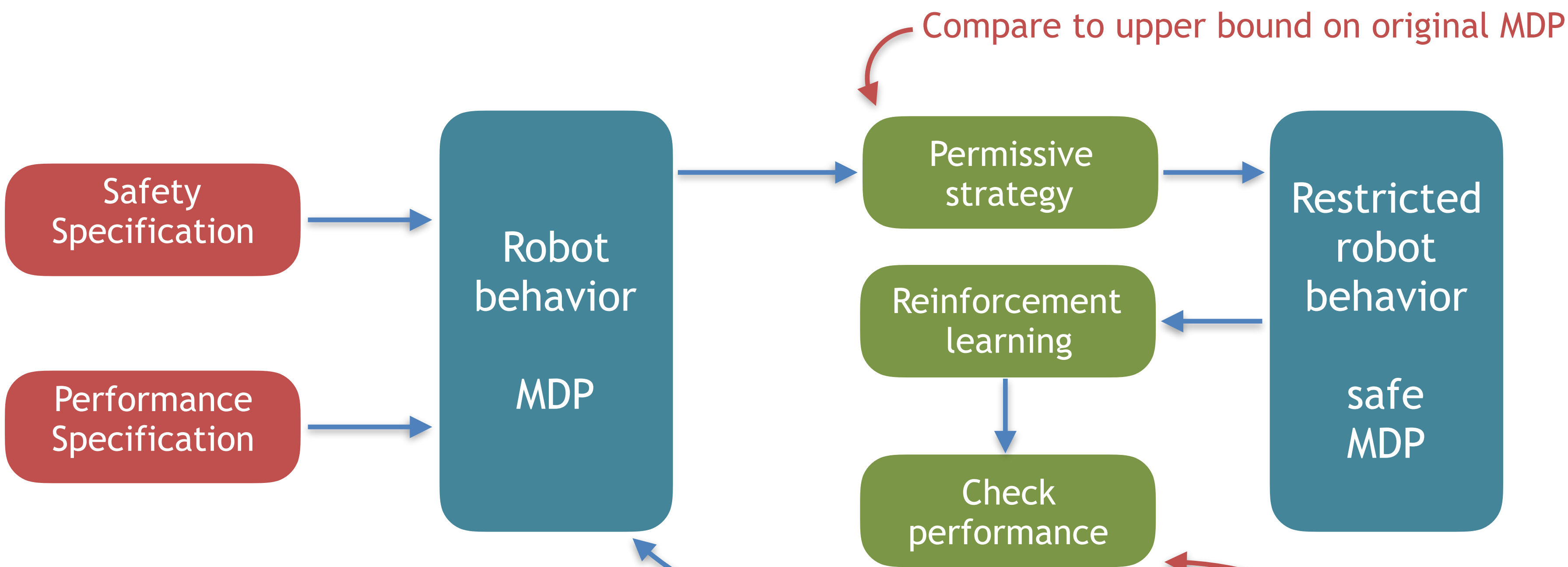


Incorporating the Bounds



Iterating means tightening the bounds

Incorporating the Bounds



exclude obtained strategy

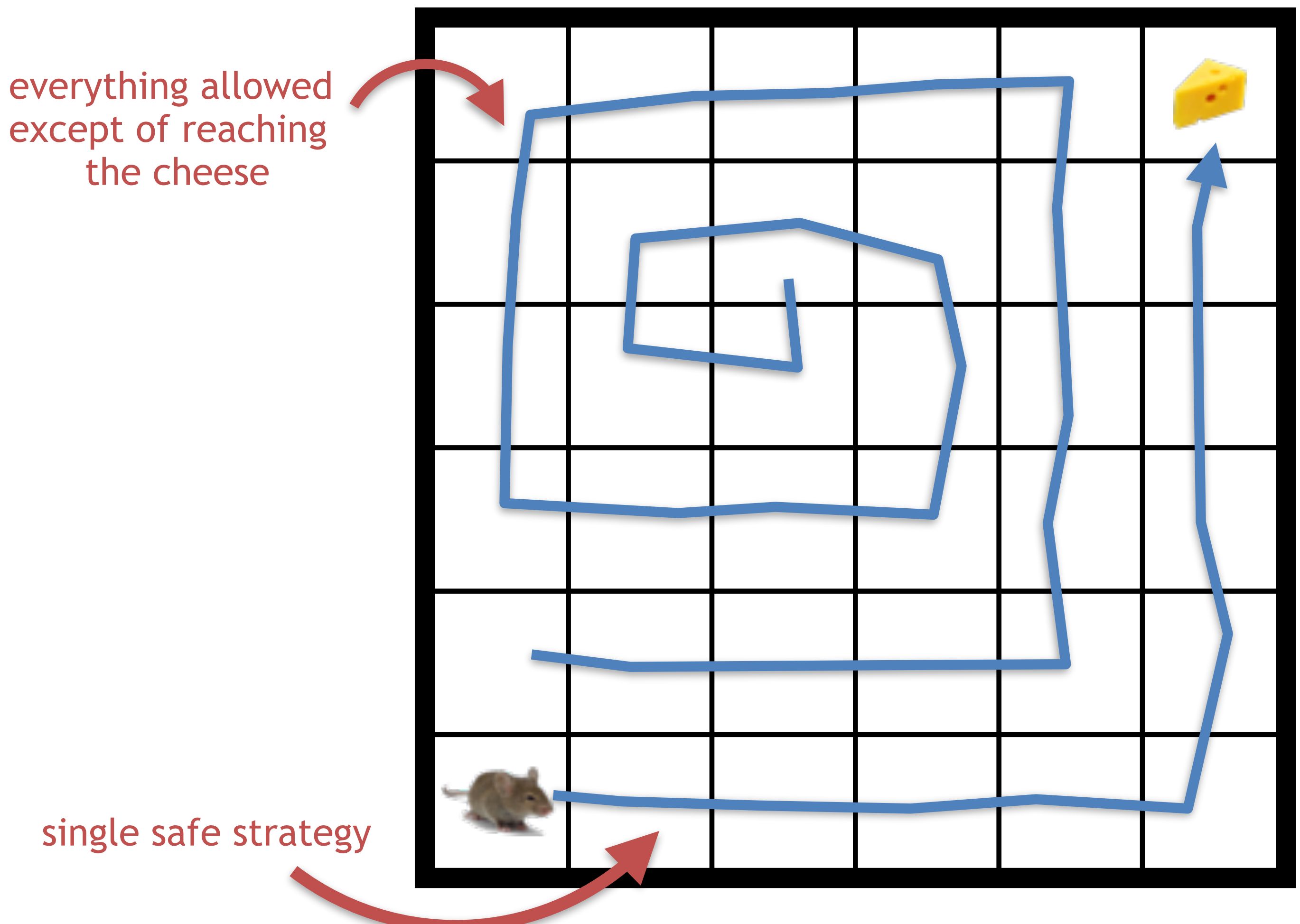
Compare to lower bound - already optimal?

Exploitation vs. Exploration

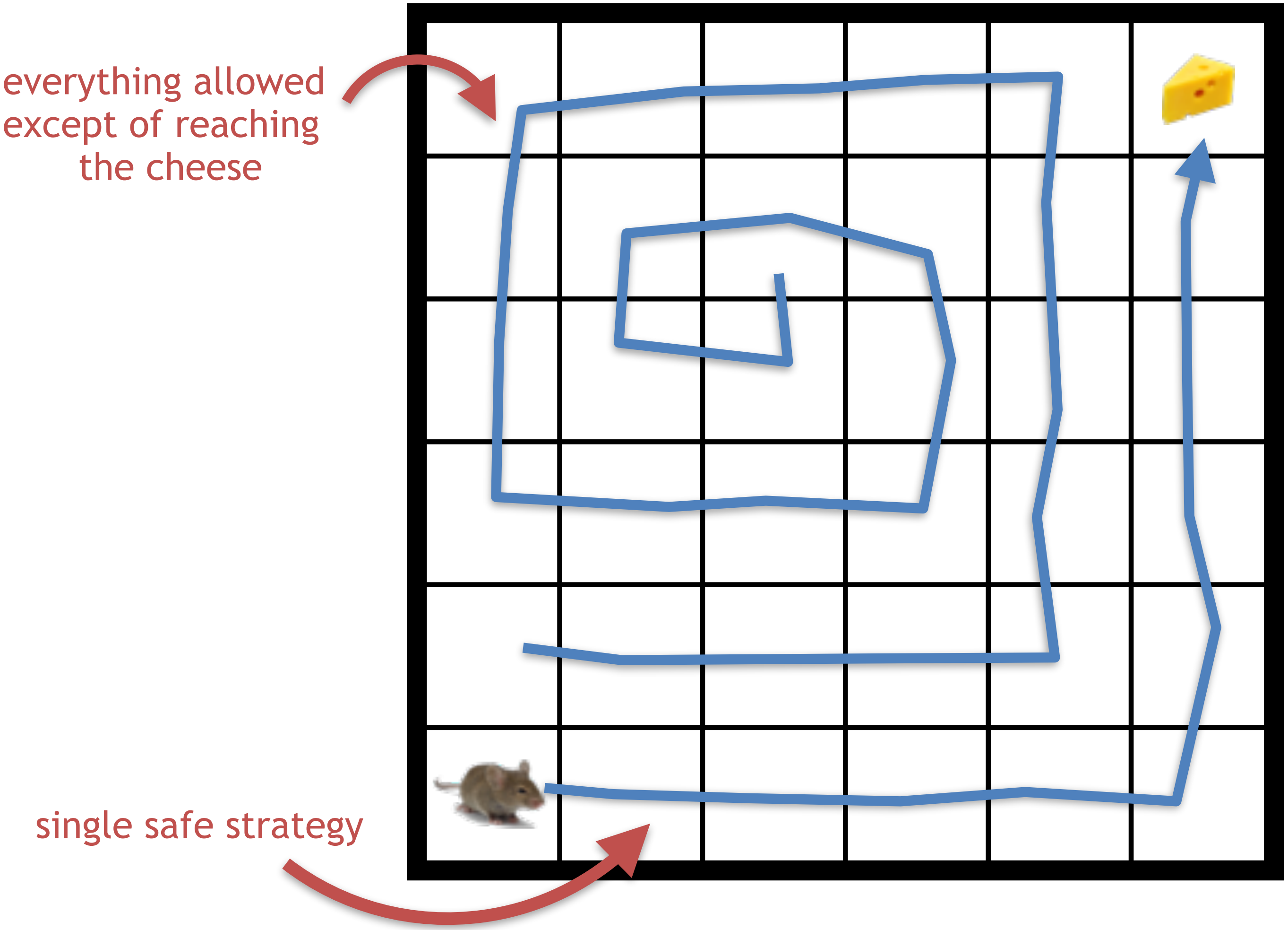
- overly optimistic expected rewards initialize Q-learning (via lower bounds) to favor unexplored parts of the MDP

Iterating means tightening the bounds

About permissiveness



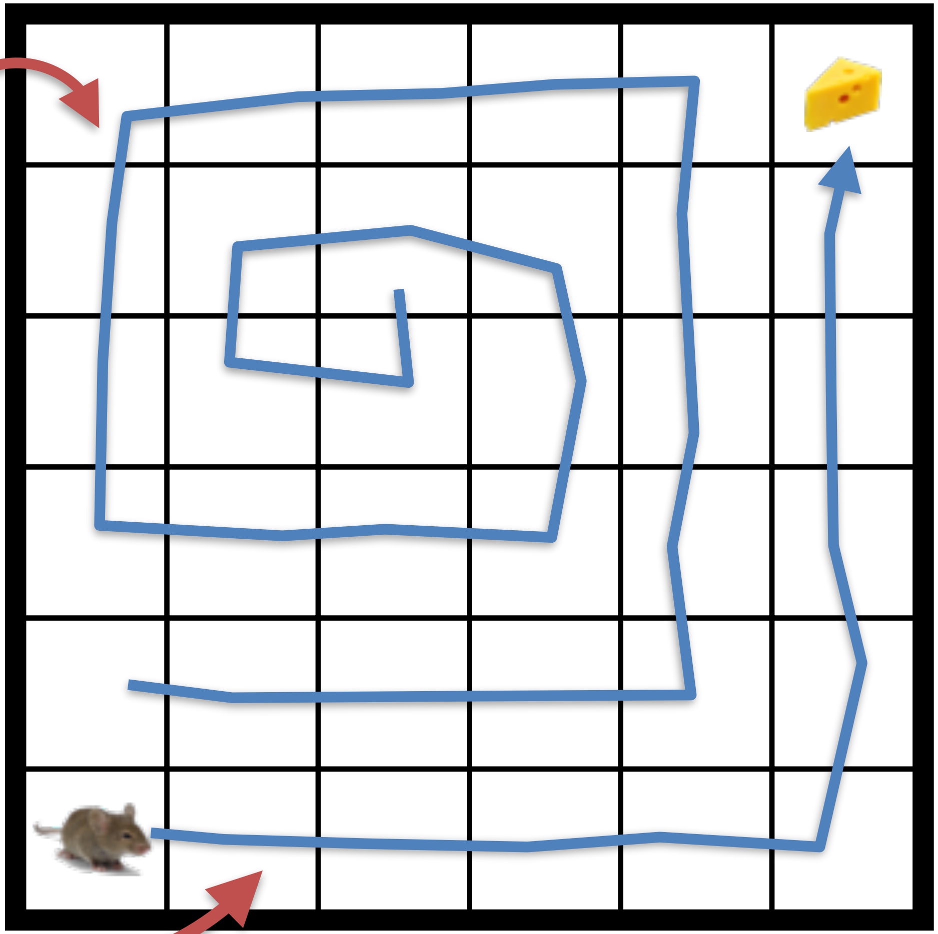
About permissiveness



Quantifying permissiveness may not be beneficent

About permissiveness

everything allowed except of reaching the cheese



single safe strategy

Quantifying permissiveness may not be beneficent

Enforce reachability of target states

very costly

number of deployments vs costly computations

Experimental Results

Benchmark		states	trans.	branch.	λ	**opt.**	i	t	lower	**upper**
Janitor	5,5	625	1125	3545	0.1	**88.6**	1	813	84	**88.6**
							2	2578	84	**88.6**
FolLine	30,15	455	1265	3693	0.01	**716.0**	1	41	715.4	**717.1**
							3	85	715.62	**716.83**
							13	306	715.9	**716.5**
	40,15	625	1775	5223	0.12	**966.0**	1	304	964.8	**968.2**
							3	420	965.4	**967.2**
							8	738	965.6	**966.7**
ComExp	6,6,6	823	2603	3726	0.08	**54.5**	1	5	0.3	**113.3**
							2	26	0.3	**74.9**
							3	105	0.3	**57.3**
	8,8,6	1495	4859	6953	0.12	**72.9**	1	15	0.42	**163.1**
							2	80	0.42	**122.0**
							3	112	0.42	**90.1**
							7	1319	0.42	**78.2**

Experimental Results

Benchmark		states	trans.	branch.	λ	opt.	i	t	lower	upper
Janitor	5,5	625	1125	3545	0.1	**88.6**	1	813	84	**88.6**
							2	2578	84	**88.6**
FolLine	30,15	455	1265	3693	0.01	**716.0**	1	41	715.4	**717.1**
							3	85	715.62	**716.83**
							13	306	715.9	**716.5**
	40,15	625	1775	5223	0.12	**966.0**	1	304	964.8	**968.2**
							3	420	965.4	**967.2**
							8	738	965.6	**966.7**
ComExp	6,6,6	823	2603	3726	0.08	**54.5**	1	5	0.3	**113.3**
							2	26	0.3	**74.9**
							3	105	0.3	**57.3**
	8,8,6	1495	4859	6953	0.12	**72.9**	1	15	0.42	**163.1**
							2	80	0.42	**122.0**
							3	112	0.42	**90.1**
							7	1319	0.42	**78.2**

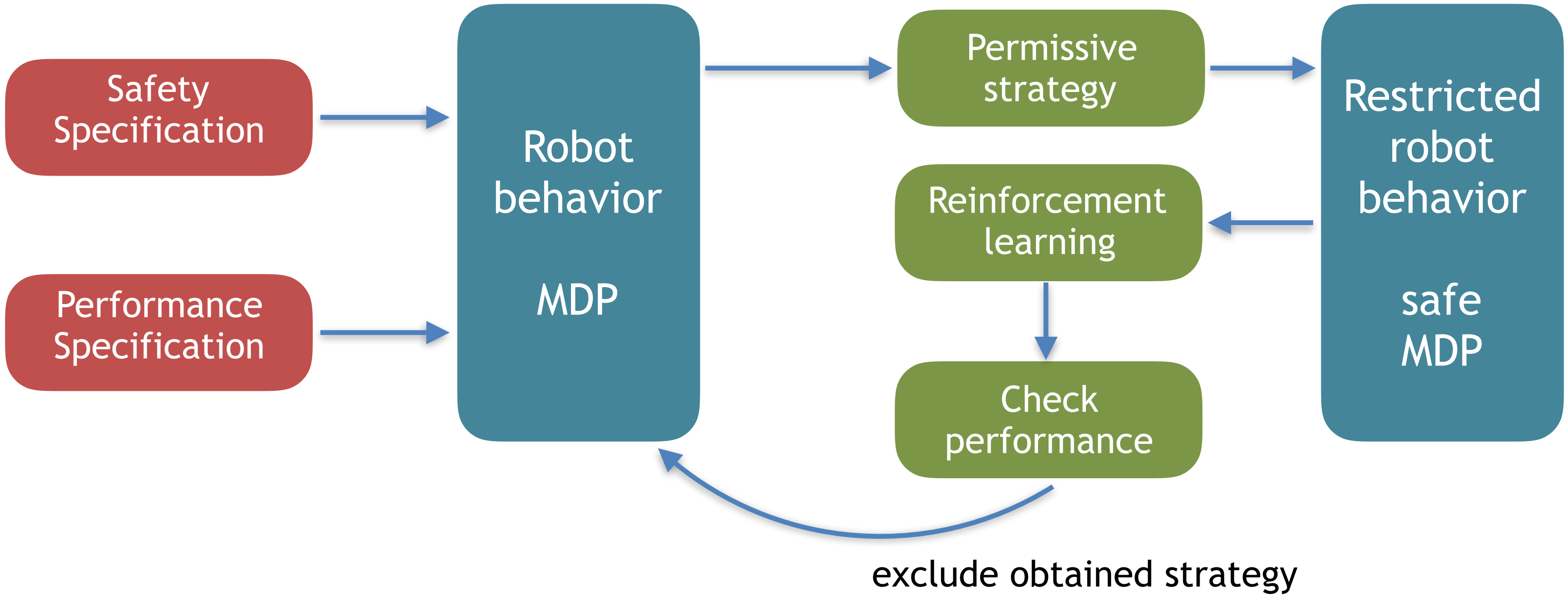
Experimental Results

Benchmark		states	trans.	branch.	λ	**opt.**	i	t	lower	**upper**
Janitor	5,5	625	1125	3545	0.1	**88.6**	1	813	84	**88.6**
							2	2578	84	**88.6**
FolLine	30,15	455	1265	3693	0.01	**716.0**	1	41	715.4	**717.1**
							3	85	715.62	**716.83**
							13	306	715.9	**716.5**
	40,15	625	1775	5223	0.12	**966.0**	1	304	964.8	**968.2**
							3	420	965.4	**967.2**
							8	738	965.6	**966.7**
ComExp	6,6,6	823	2603	3726	0.08	**54.5**	1	5	0.3	**113.3**
							2	26	0.3	**74.9**
							3	105	0.3	**57.3**
	8,8,6	1495	4859	6953	0.12	**72.9**	1	15	0.42	**163.1**
							2	80	0.42	**122.0**
							3	112	0.42	**90.1**
							7	1319	0.42	**78.2**

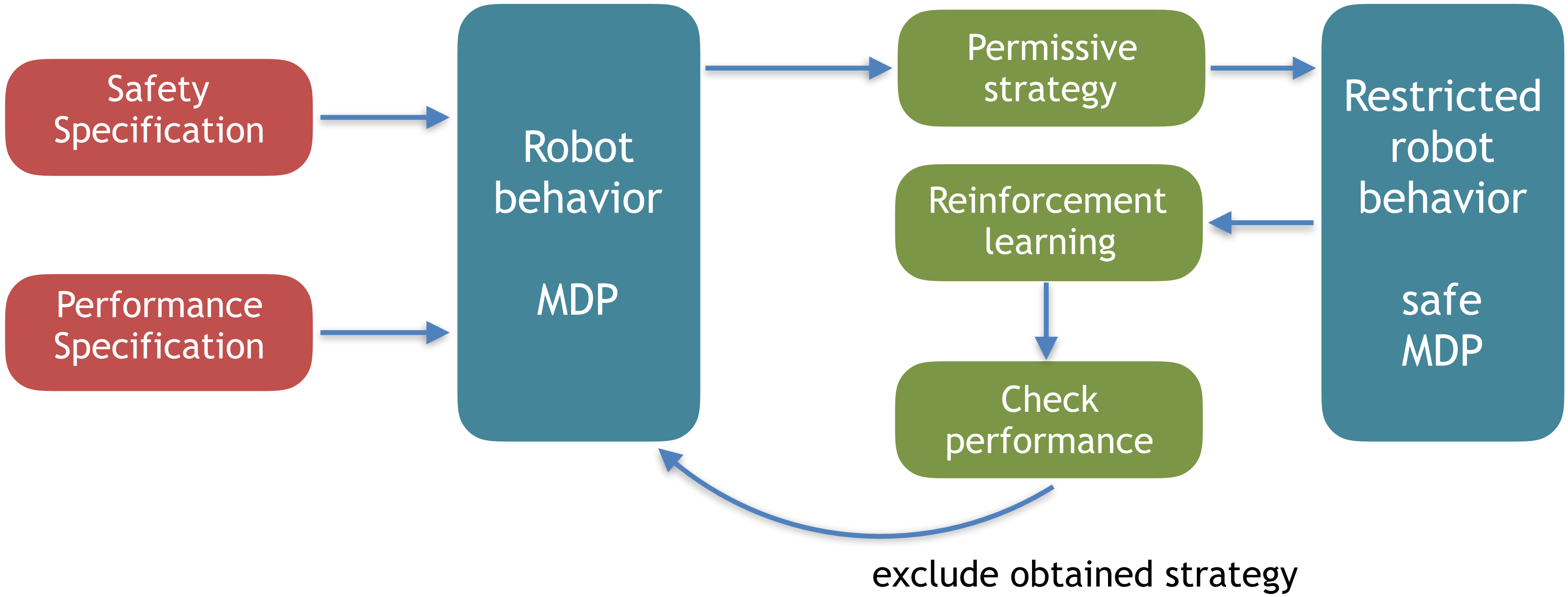
Experimental Results

Benchmark		states	trans.	branch.	λ	**opt.**	i	t	lower	**upper**
Janitor	5,5	625	1125	3545	0.1	**88.6**	1	813	84	**88.6**
							2	2578	84	**88.6**
FolLine	30,15	455	1265	3693	0.01	**716.0**	1	41	715.4	**717.1**
							3	85	715.62	**716.83**
							13	306	715.9	**716.5**
	40,15	625	1775	5223	0.12	**966.0**	1	304	964.8	**968.2**
							3	420	965.4	**967.2**
							8	738	965.6	**966.7**
ComExp	6,6,6	823	2603	3726	0.08	**54.5**	1	5	0.3	**113.3**
							2	26	0.3	**74.9**
							3	105	0.3	**57.3**
	8,8,6	1495	4859	6953	0.12	**72.9**	1	15	0.42	**163.1**
							2	80	0.42	**122.0**
							3	112	0.42	**90.1**
							7	1319	0.42	**78.2**

Results

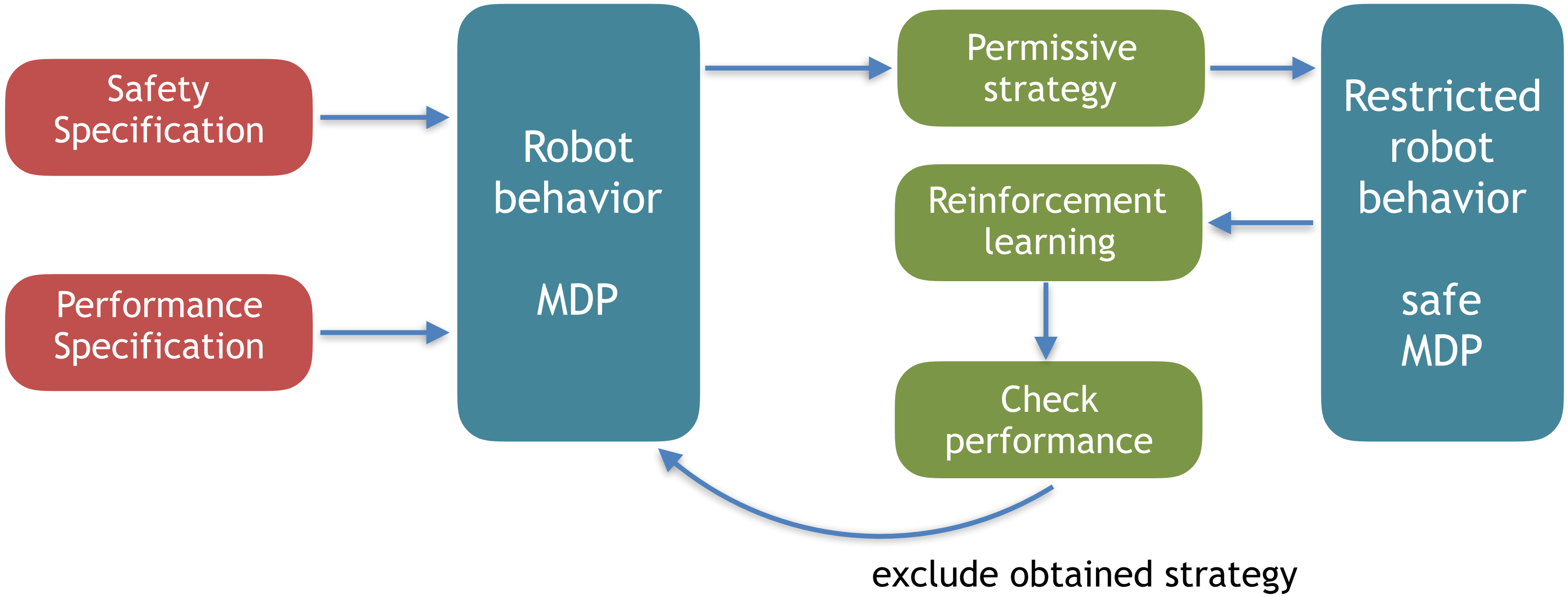

Safety Specification
Performance Specification
Robot behavior
MDP
Permissive strategy
Reinforcement learning
Check performance
Restricted robot behavior
safe MDP
exclude obtained strategy

Results



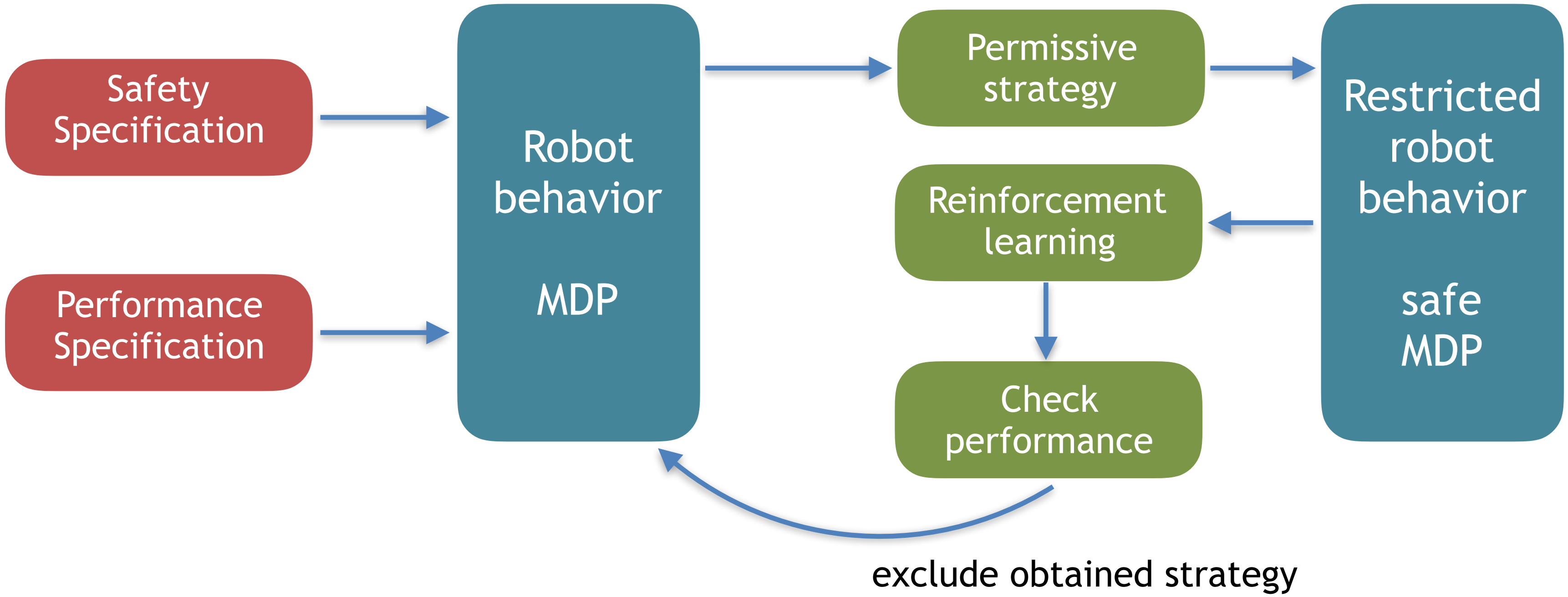
- Soundness: SMT encoding is correct

Results



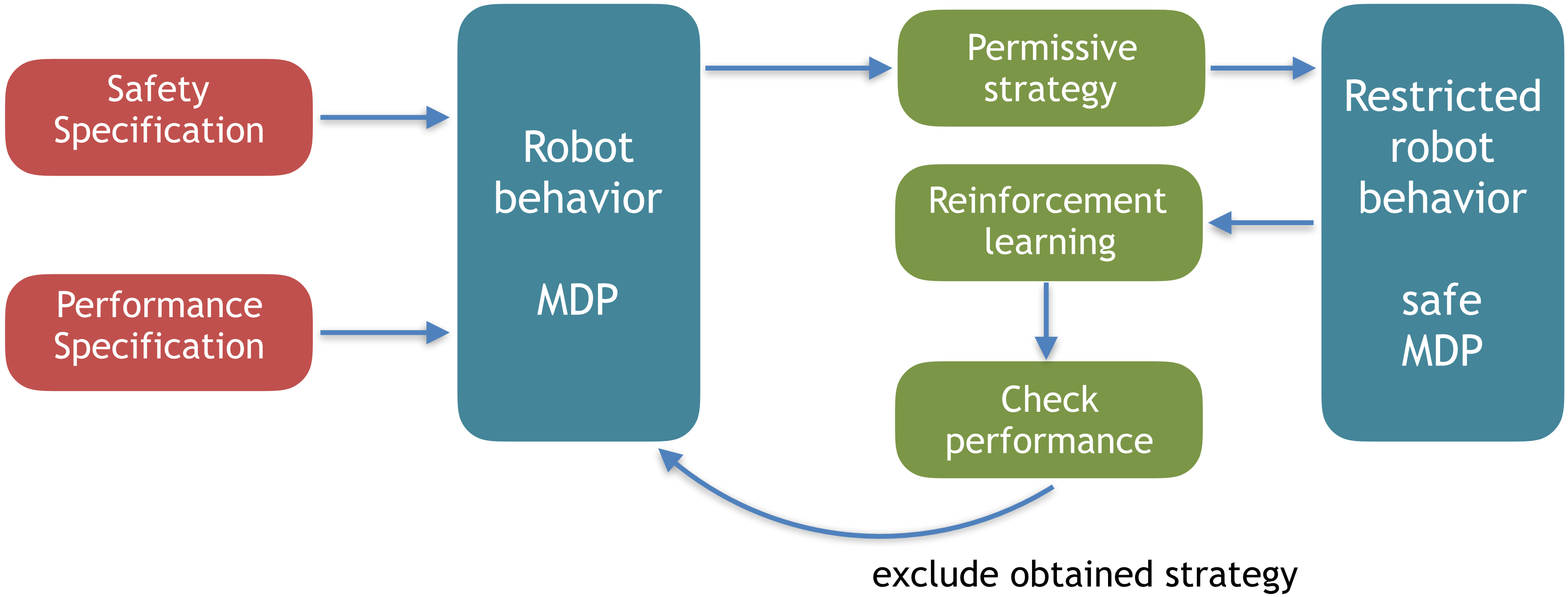
- Soundness: SMT encoding is correct
- Completeness: Optimal safe strategy is computed

Results



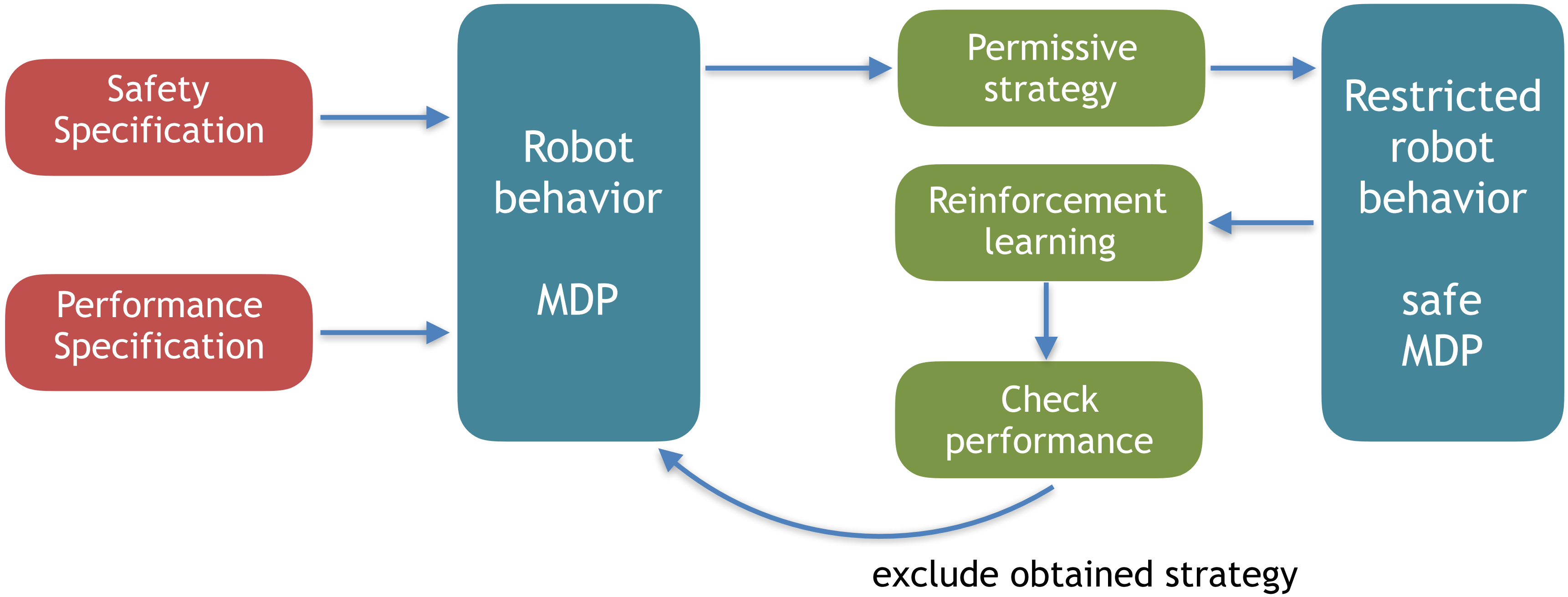
- Soundness: SMT encoding is correct
- Completeness: Optimal safe strategy is computed
- No efficient representation for 'maximally' permissive strategy

Results



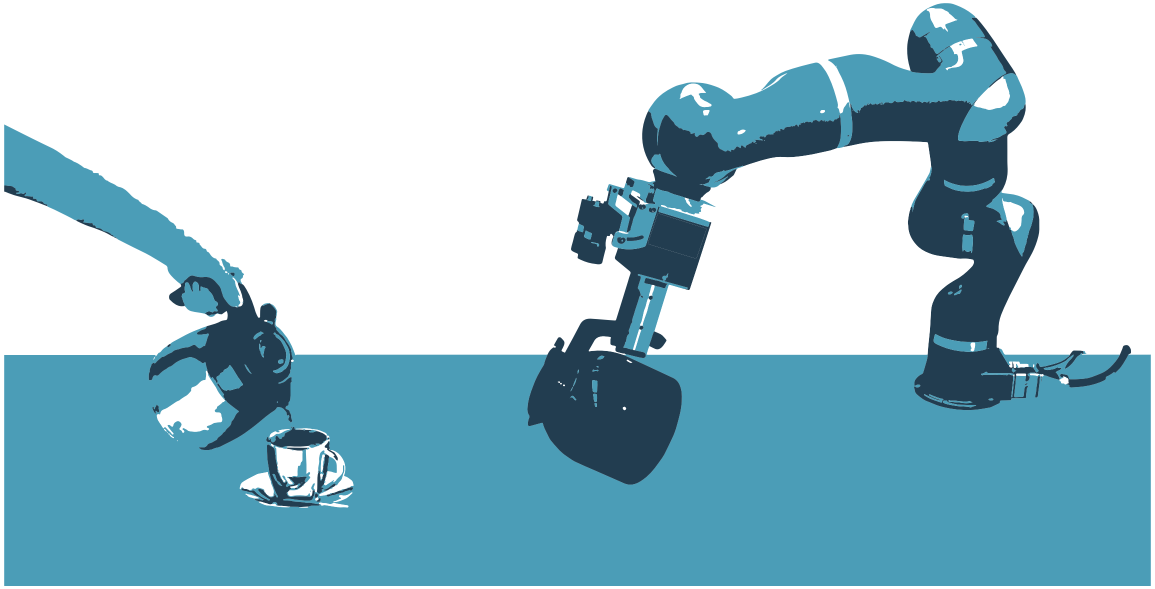
- Soundness: SMT encoding is correct
- Completeness: Optimal safe strategy is computed
- No efficient representation for ‘maximally’ permissive strategy
- Utilizing bounds: Significant speedup for costly computation

Results



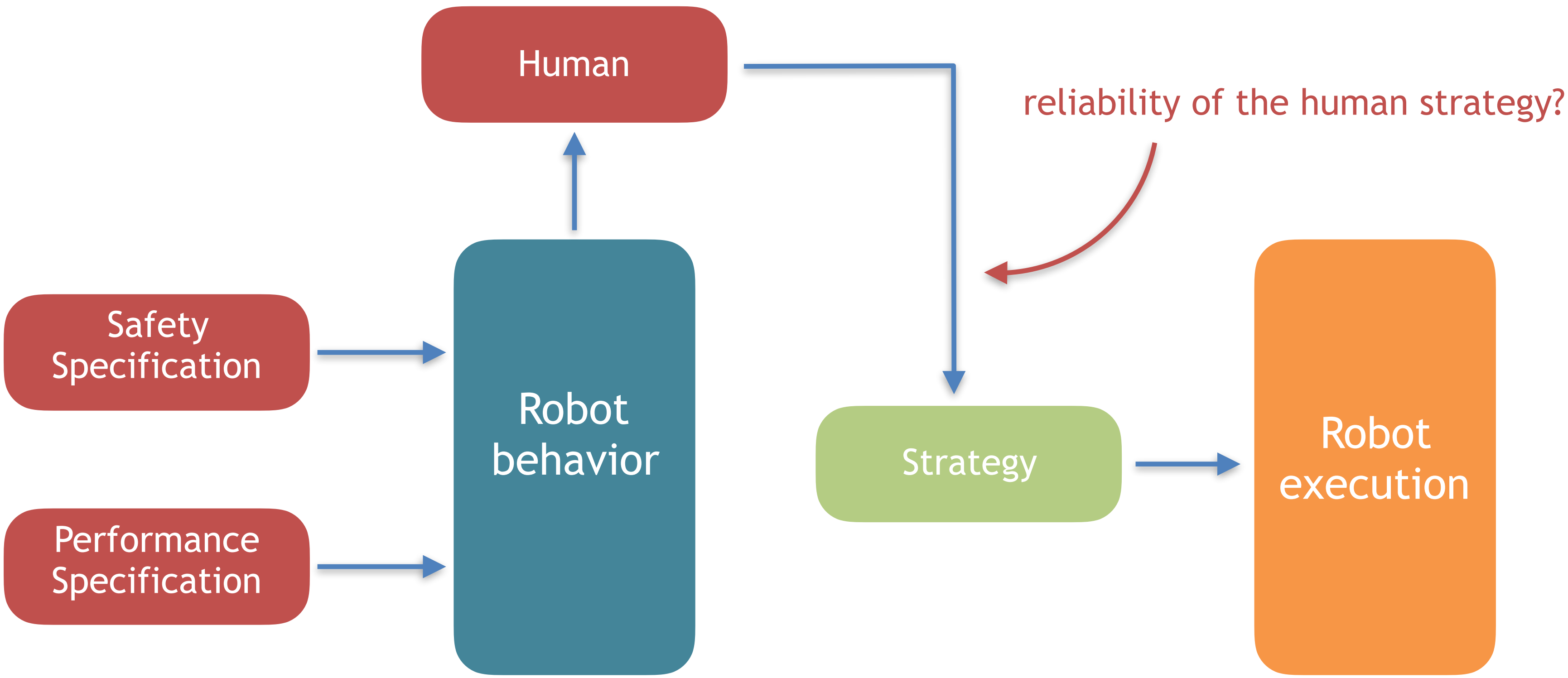
- Soundness: SMT encoding is correct
- Completeness: Optimal safe strategy is computed
- No efficient representation for ‘maximally’ permissive strategy
- Utilizing bounds: Significant speedup for costly computation
- Extension to randomized schedulers (non-linear)

Synthesis of Shared Control Protocols



Nils Jansen and Ufuk Topcu

Shared Control

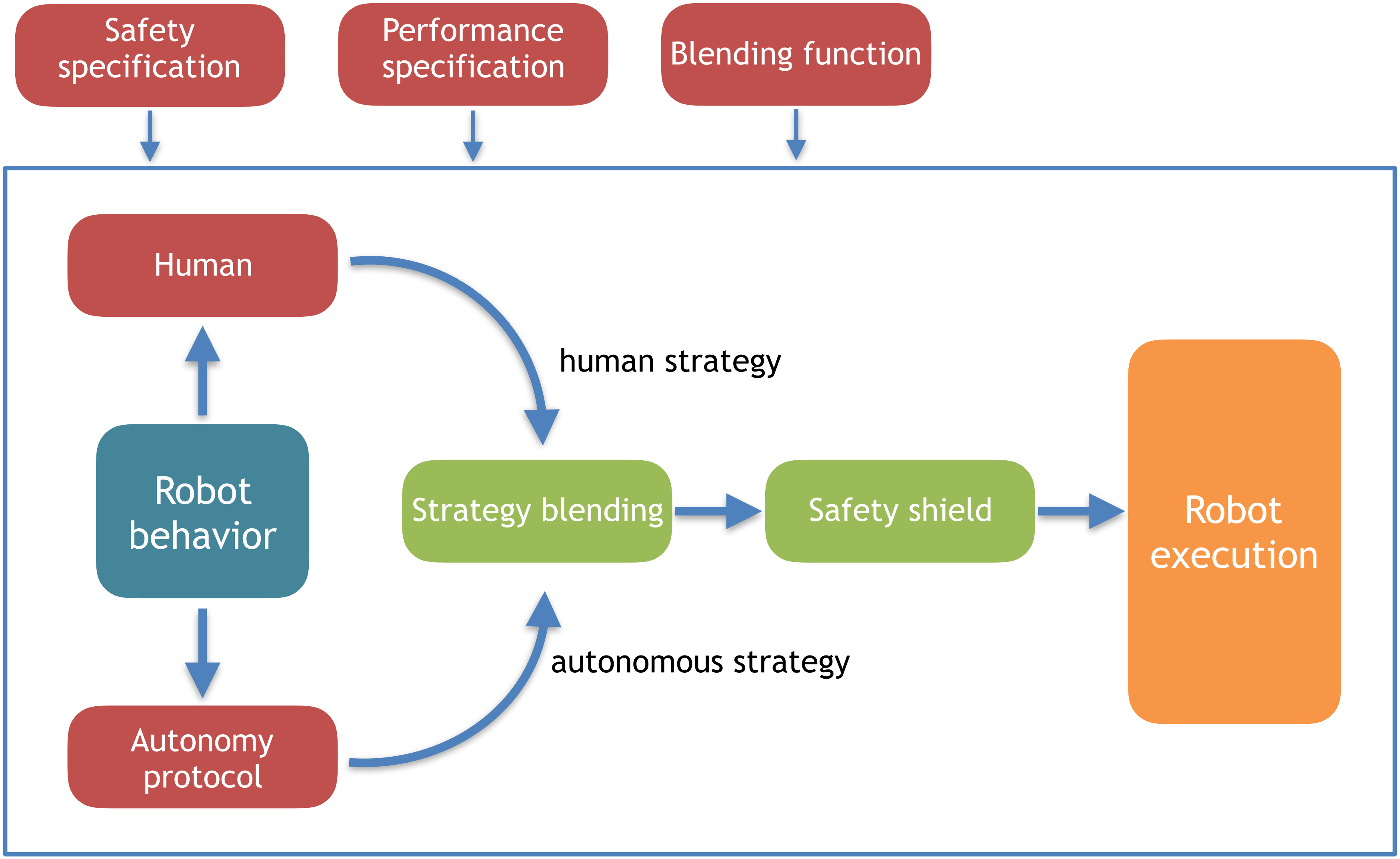


Galán et al.

Brain-actuated Wheelchair

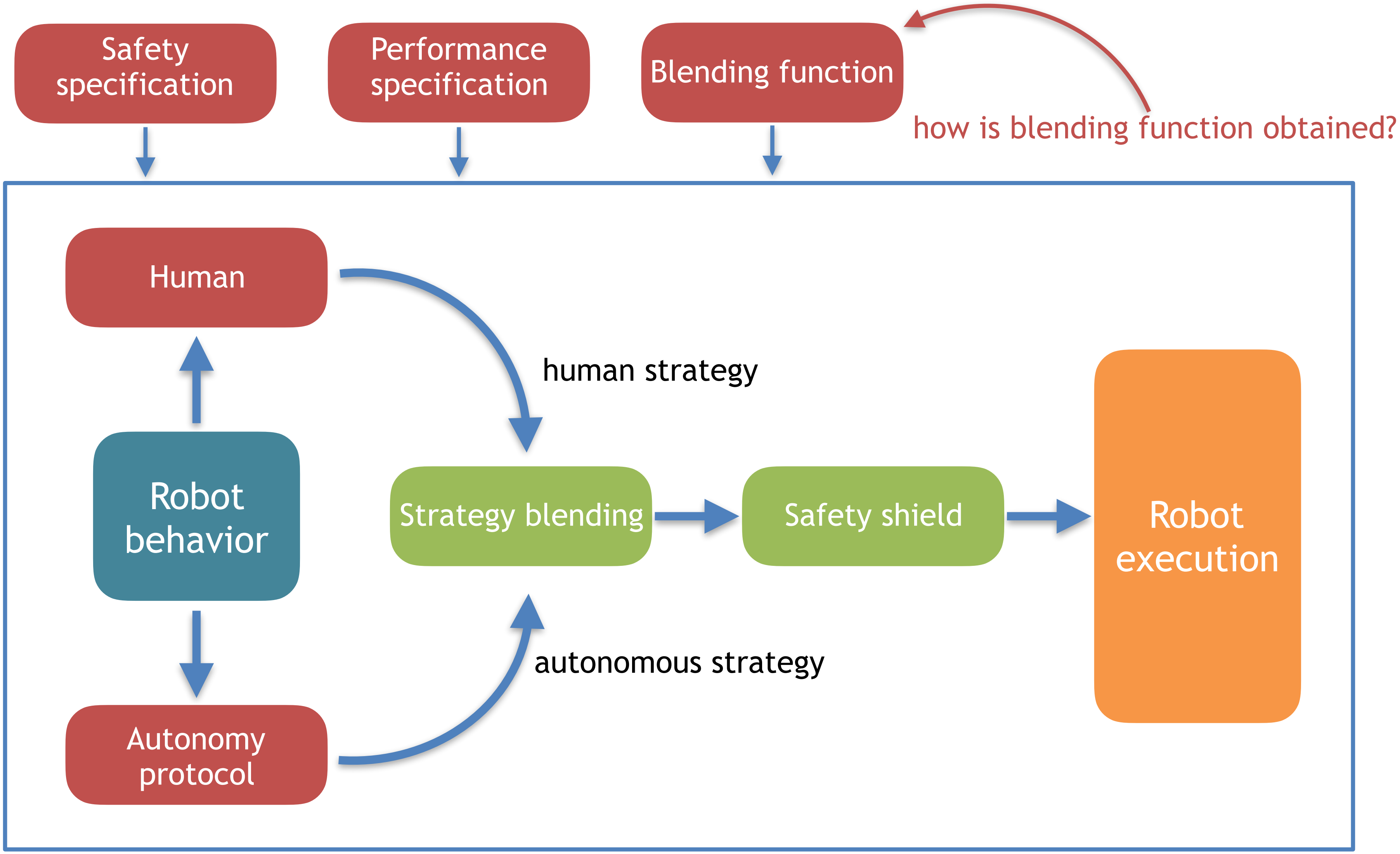
Anca D. Dragan et al., Peter Trautman

Shared Control Protocol



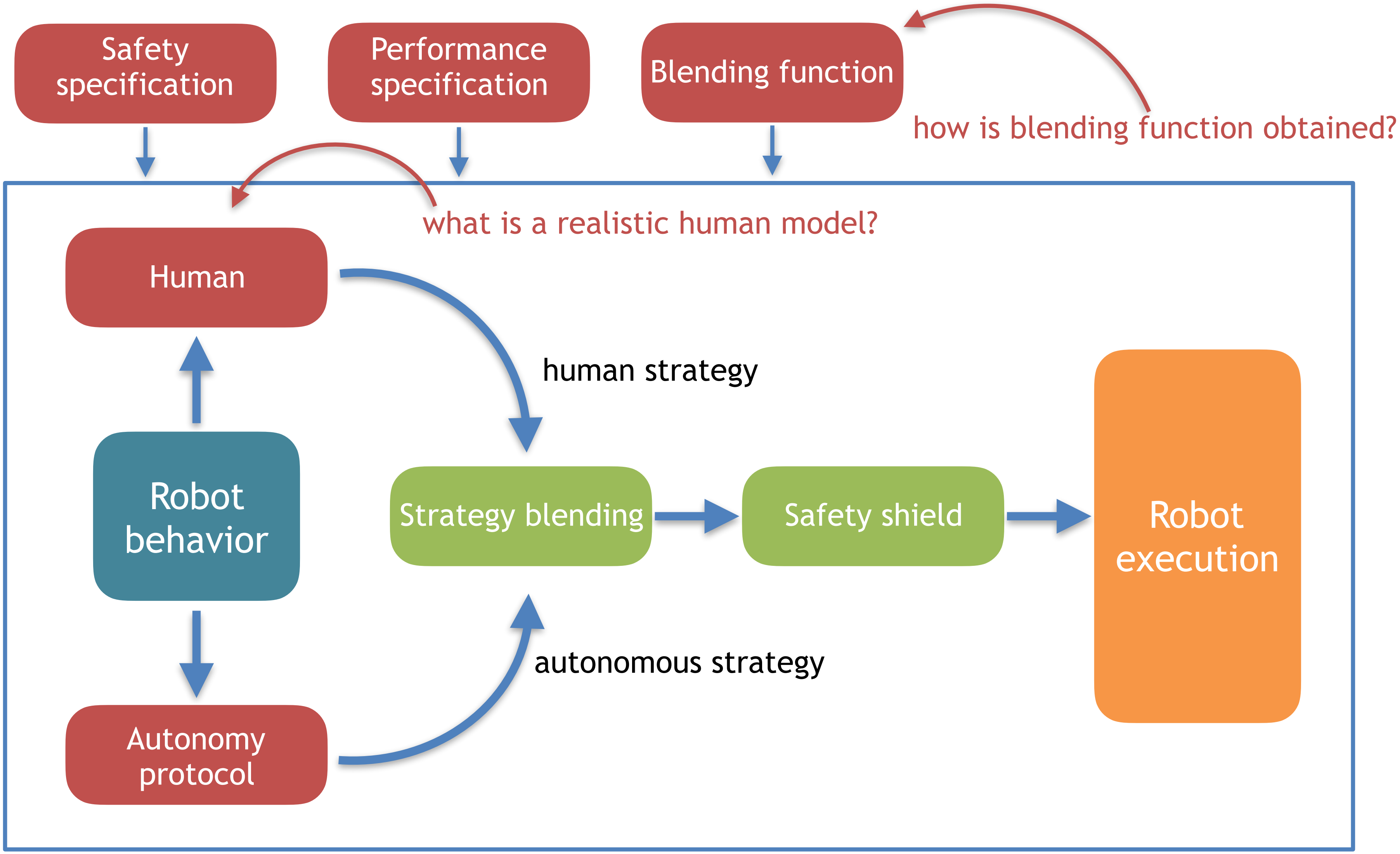
Anca D. Dragan et al., Peter Trautman

Shared Control Protocol



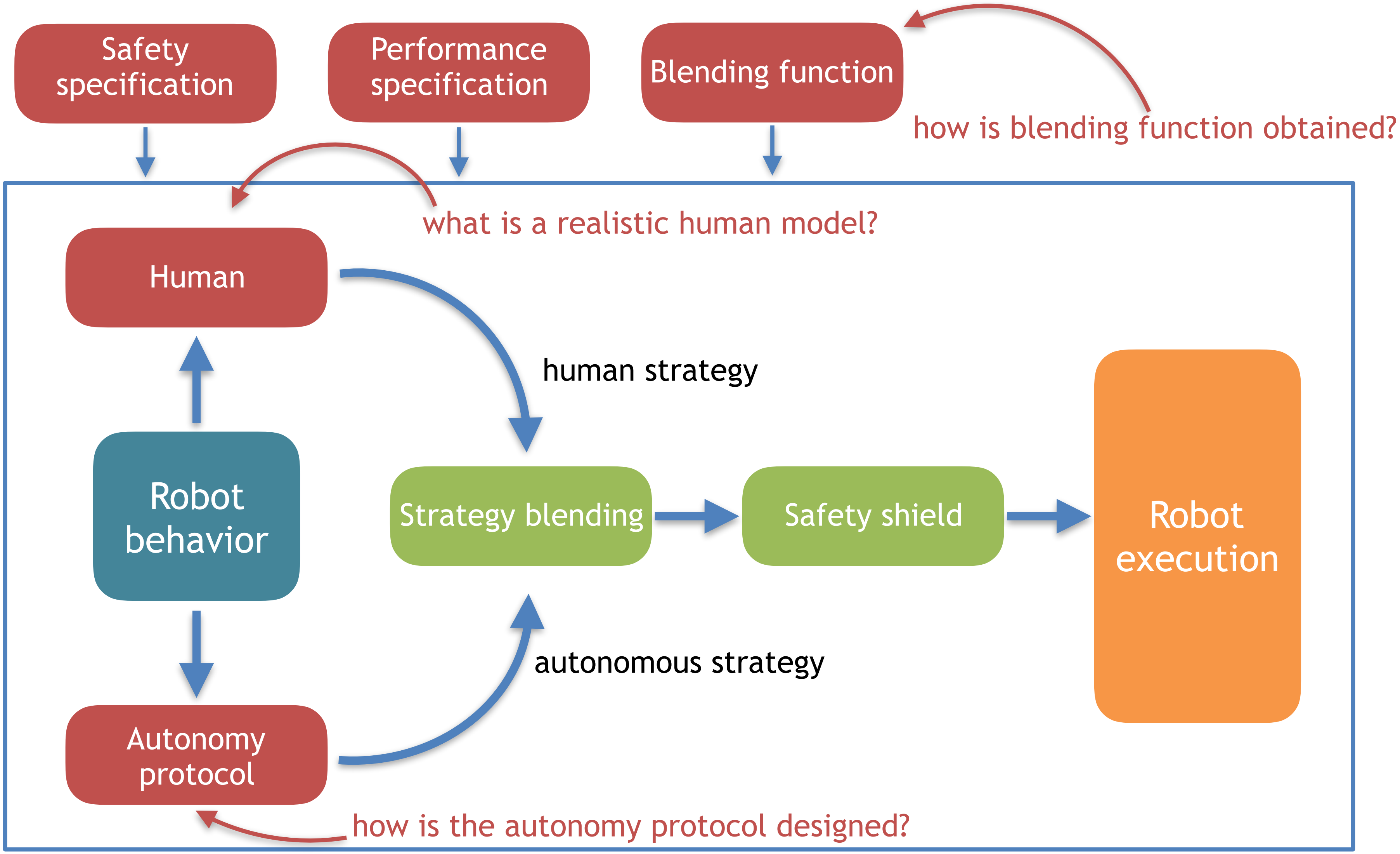
Anca D. Dragan et al., Peter Trautman

Shared Control Protocol



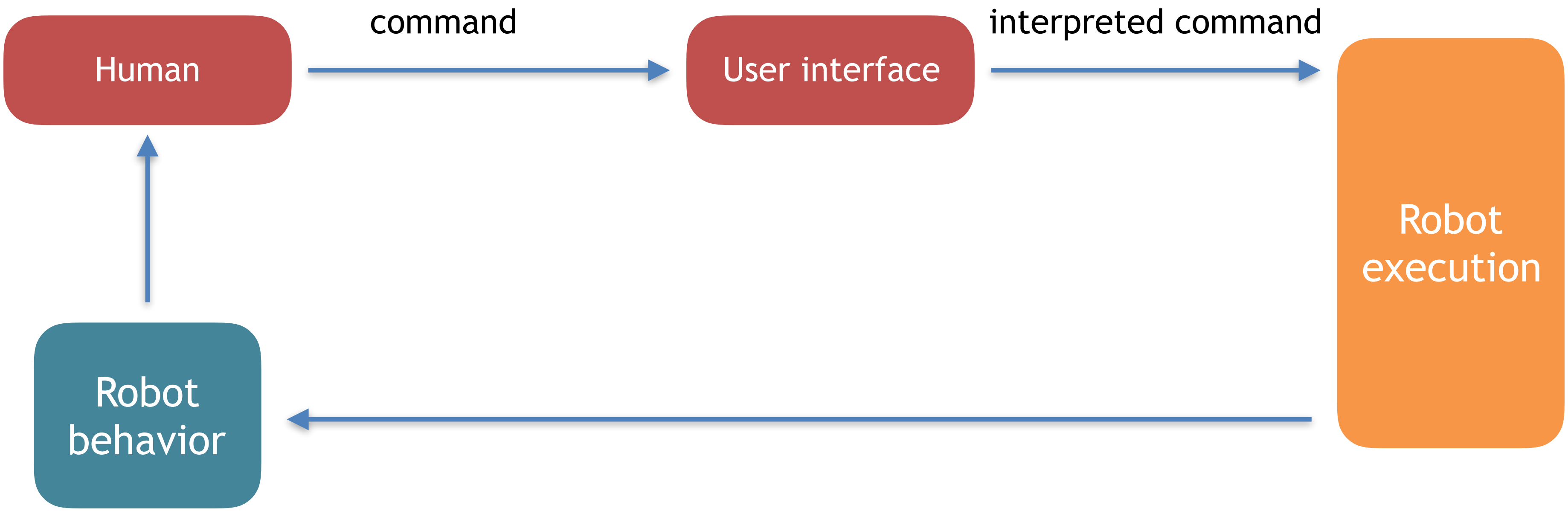
Anca D. Dragan et al., Peter Trautman

Shared Control Protocol

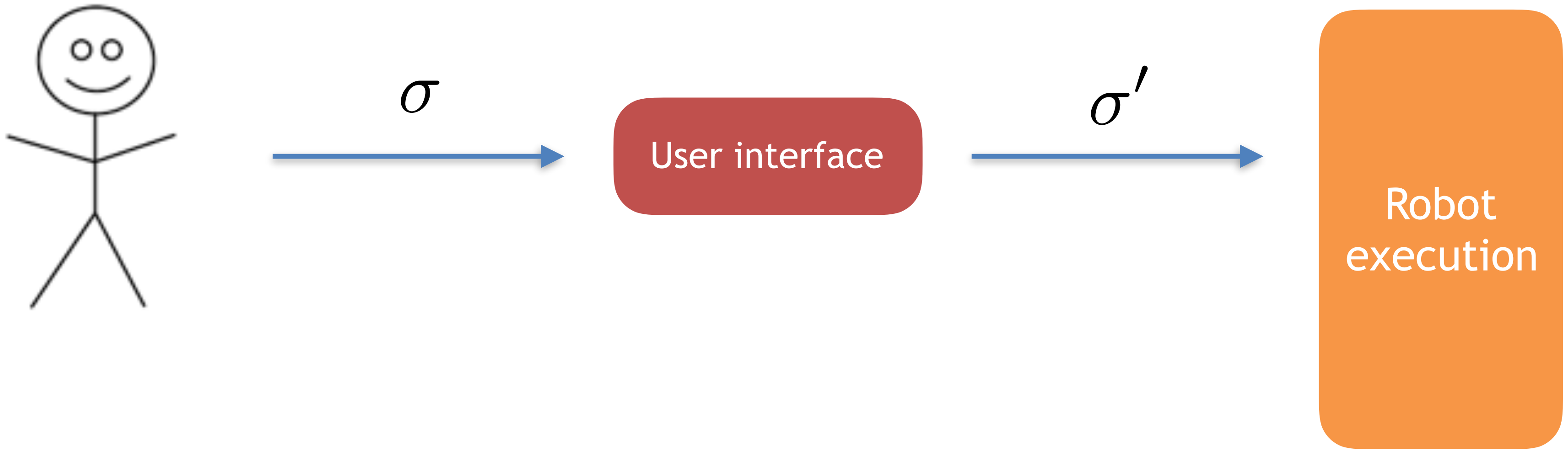


Human Model

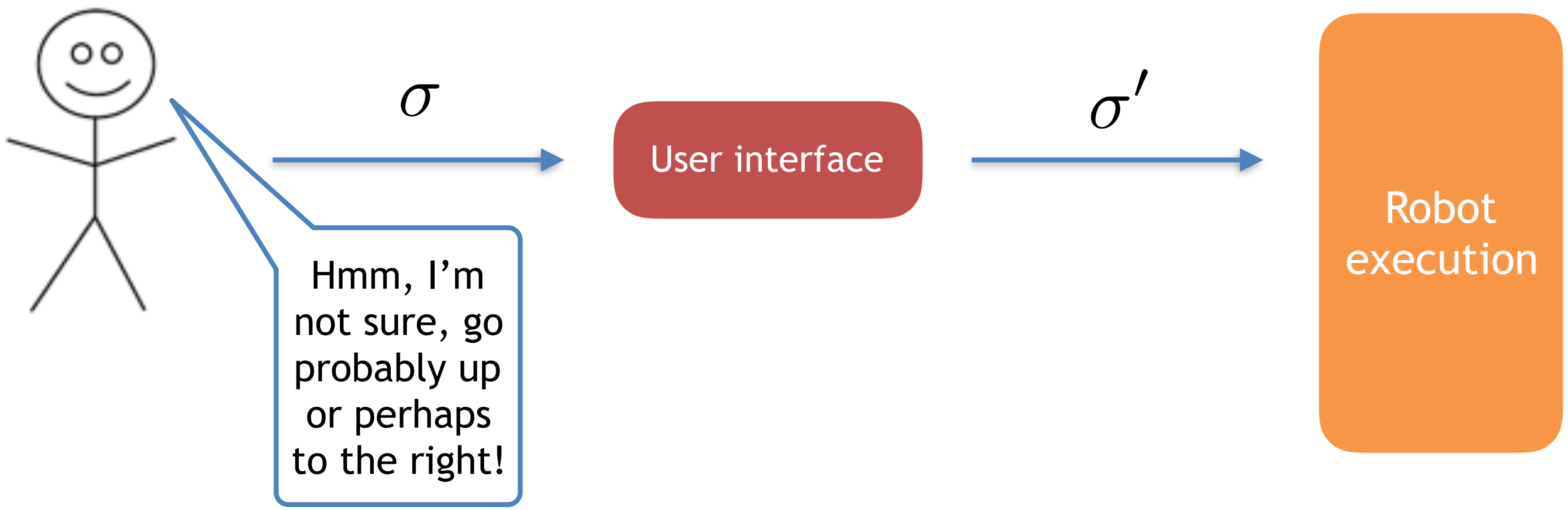
Human Model


Human
command
User interface
interpreted command
Robot execution
Robot behavior

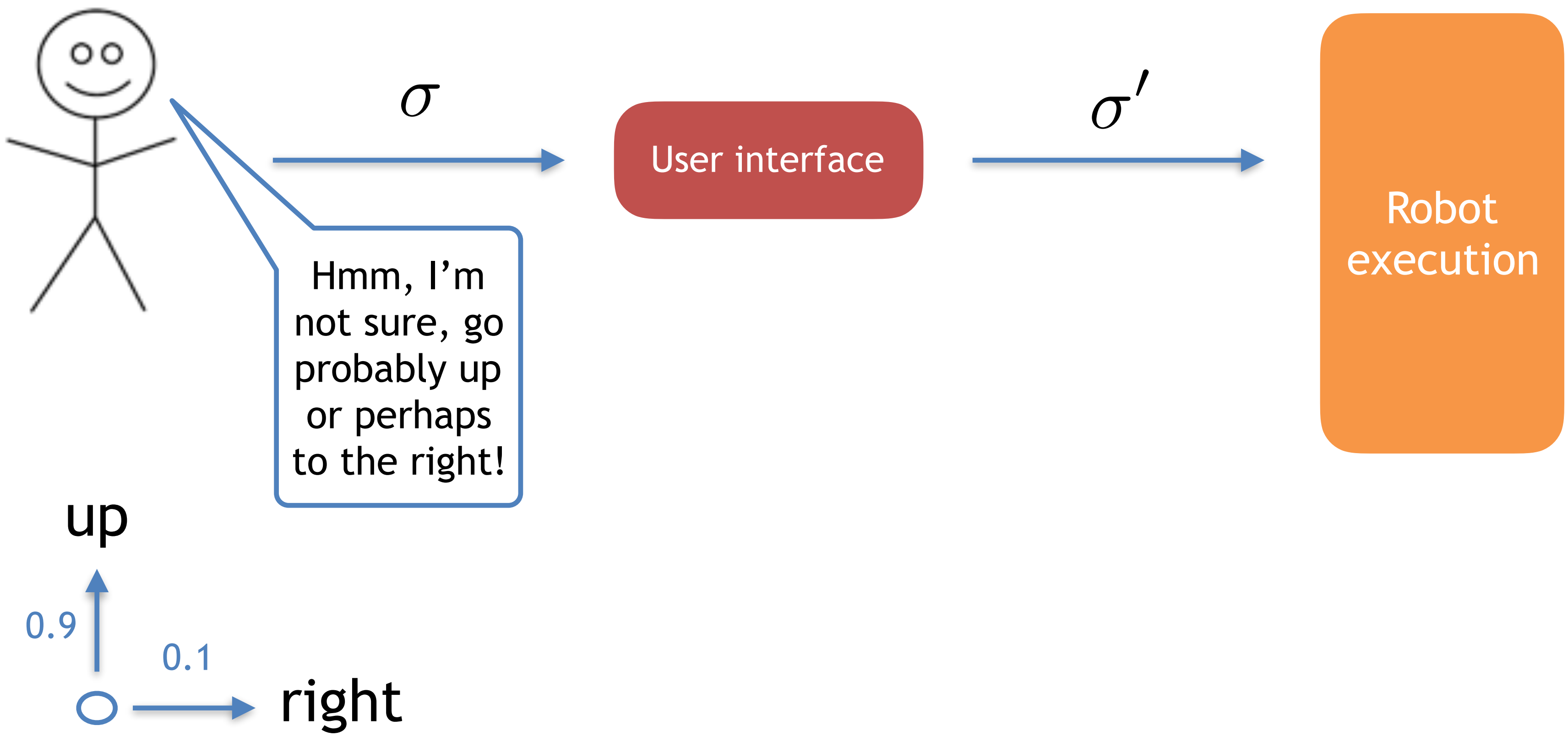
Human Model - Randomization



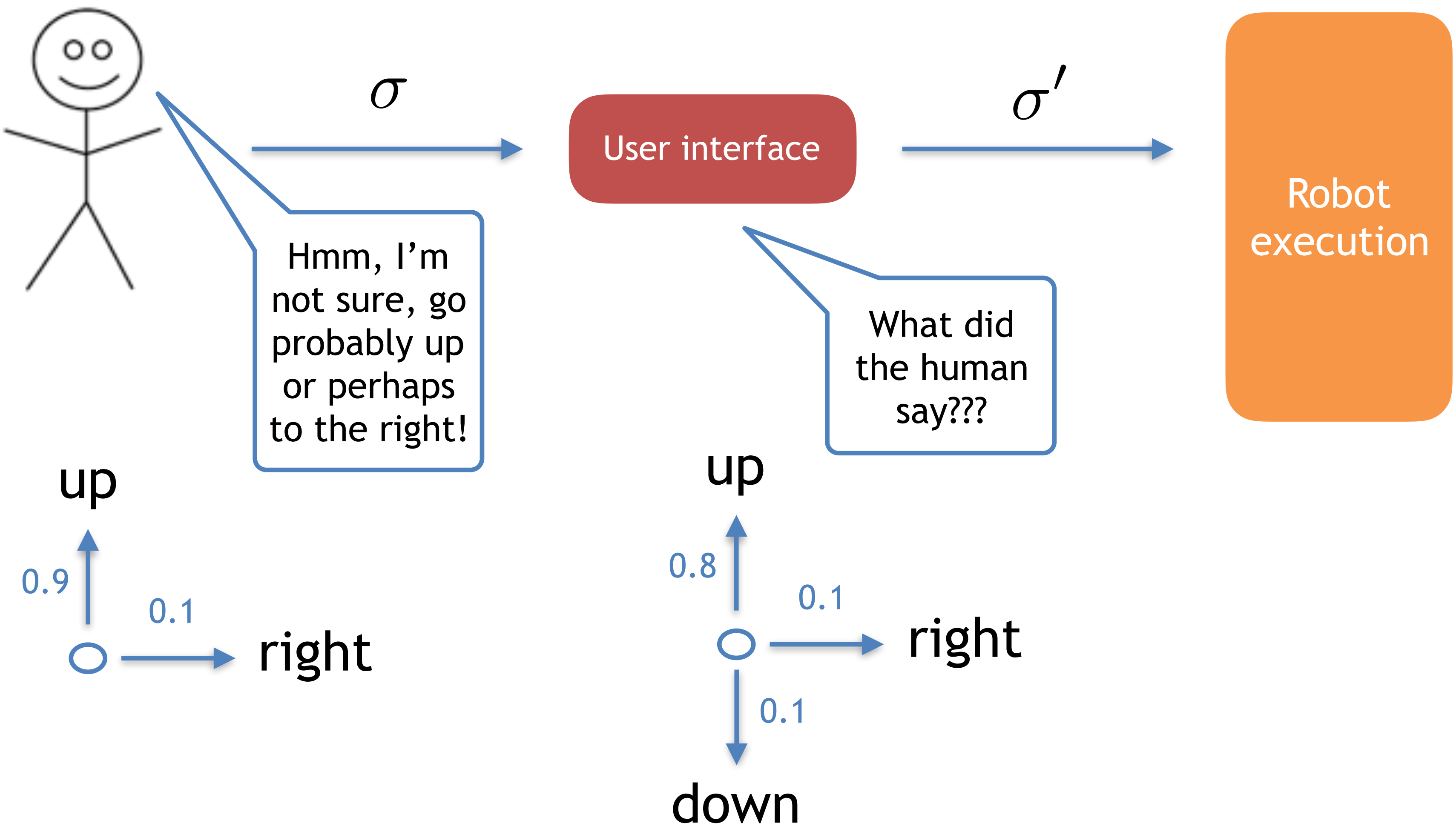
Human Model - Randomization



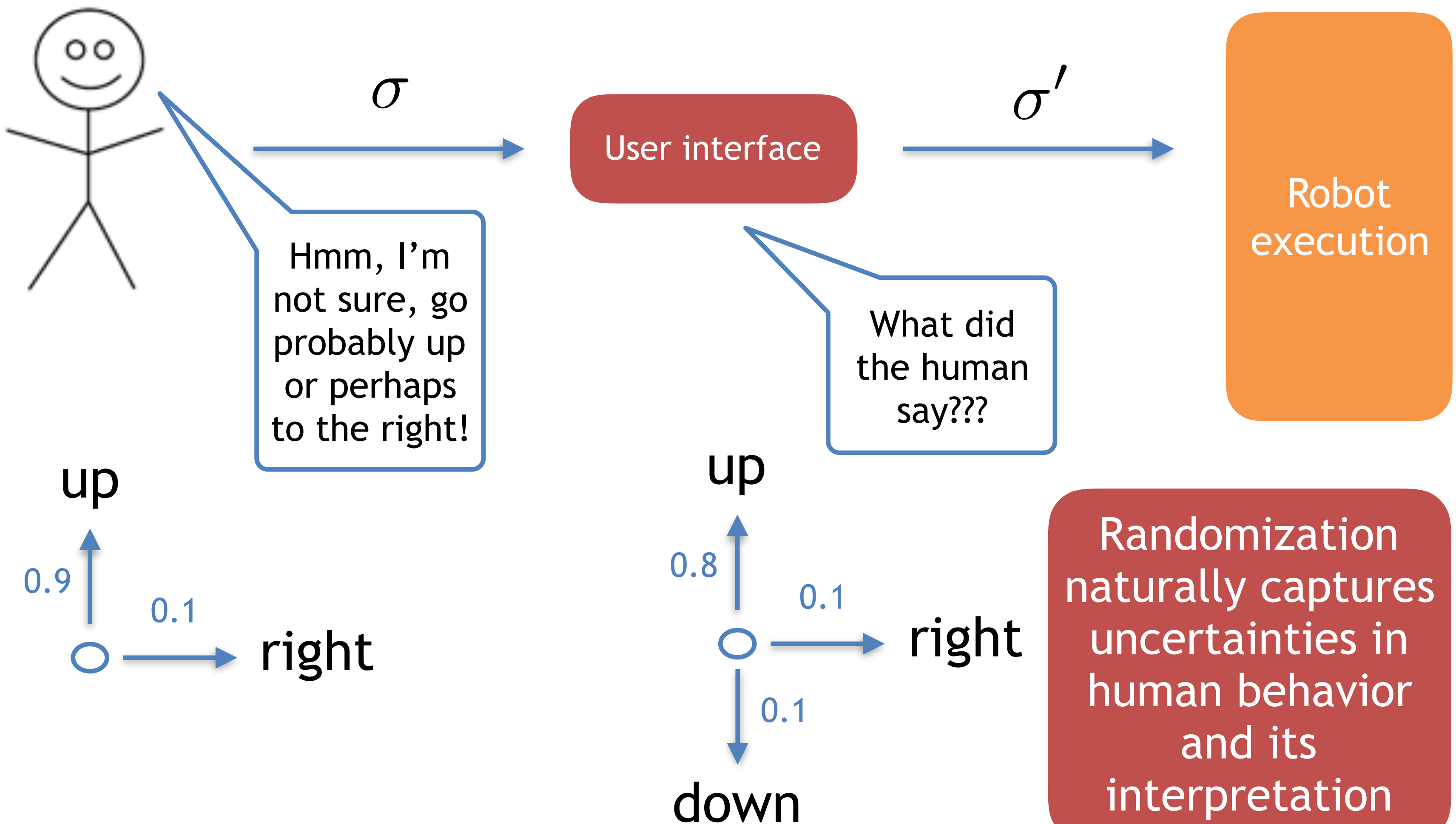
Human Model - Randomization



Human Model - Randomization

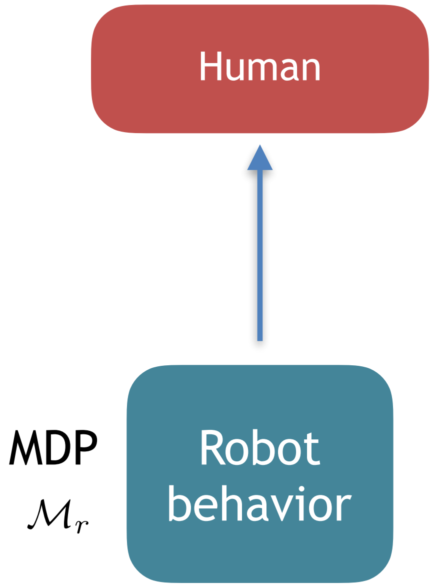


Human Model - Randomization

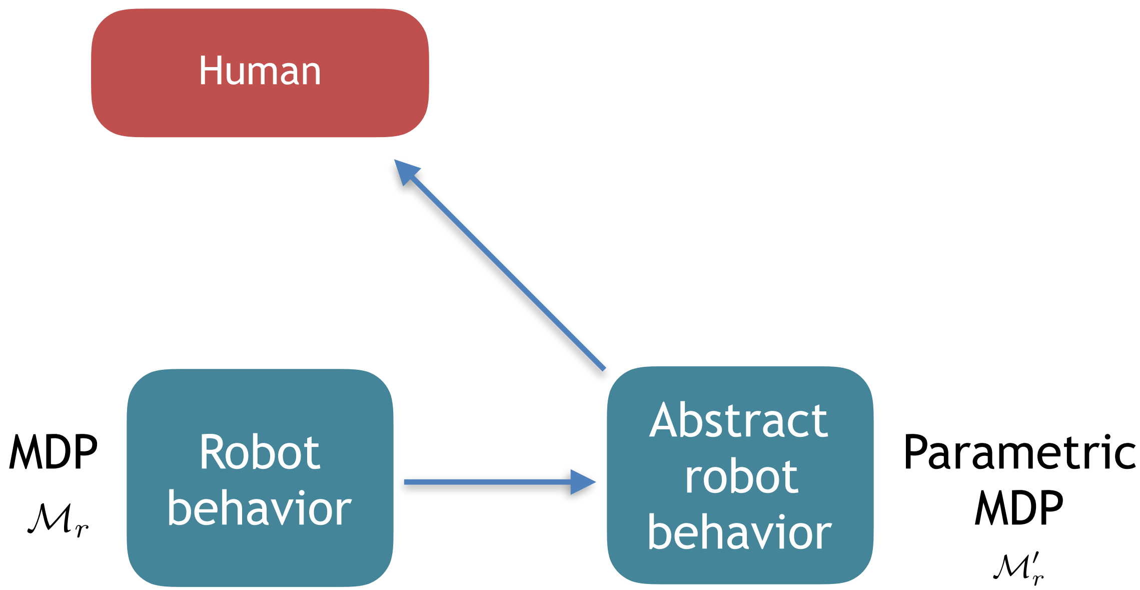


Randomization naturally captures uncertainties in human behavior and its interpretation

Human Model - Abstraction



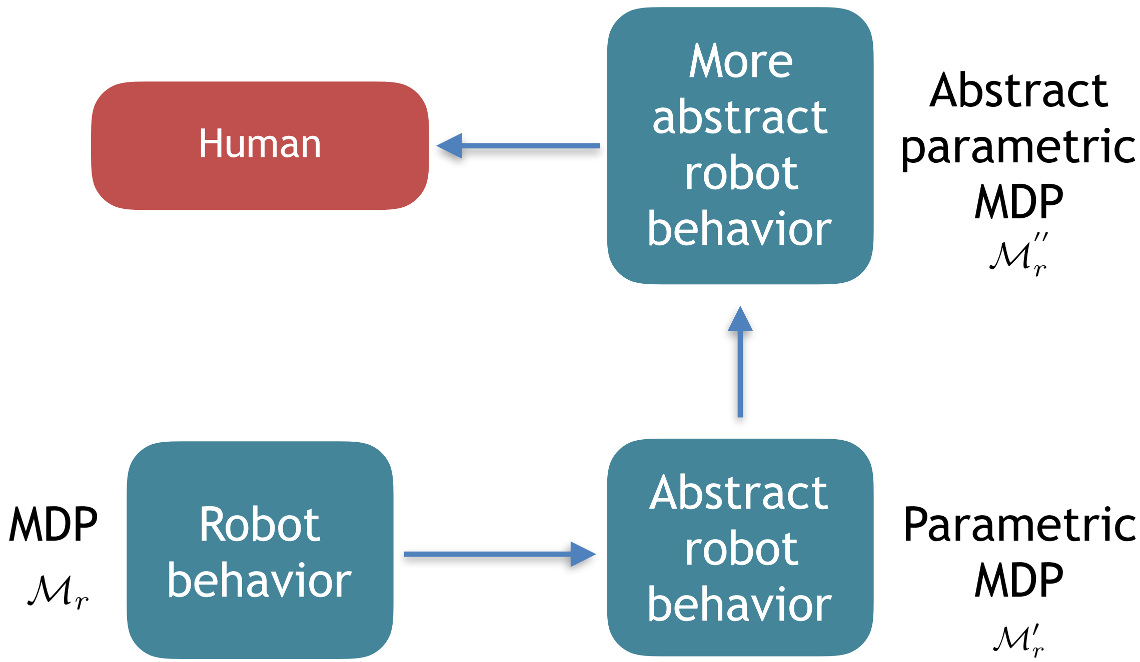
Human Model - Abstraction



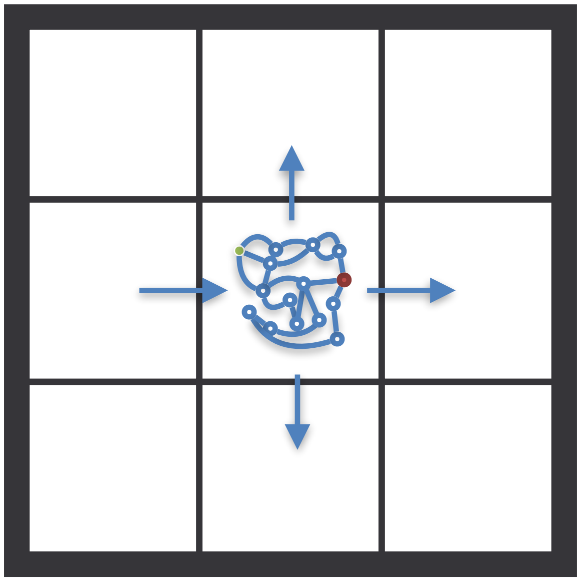
Abstract concrete probabilities by parameters

$$p_{high} \text{ if } \mathcal{P}_h(s, \alpha, s') > y$$
$$p_{low} \text{ if } \mathcal{P}_h(s, \alpha, s') \leq y$$

Human Model - Abstraction



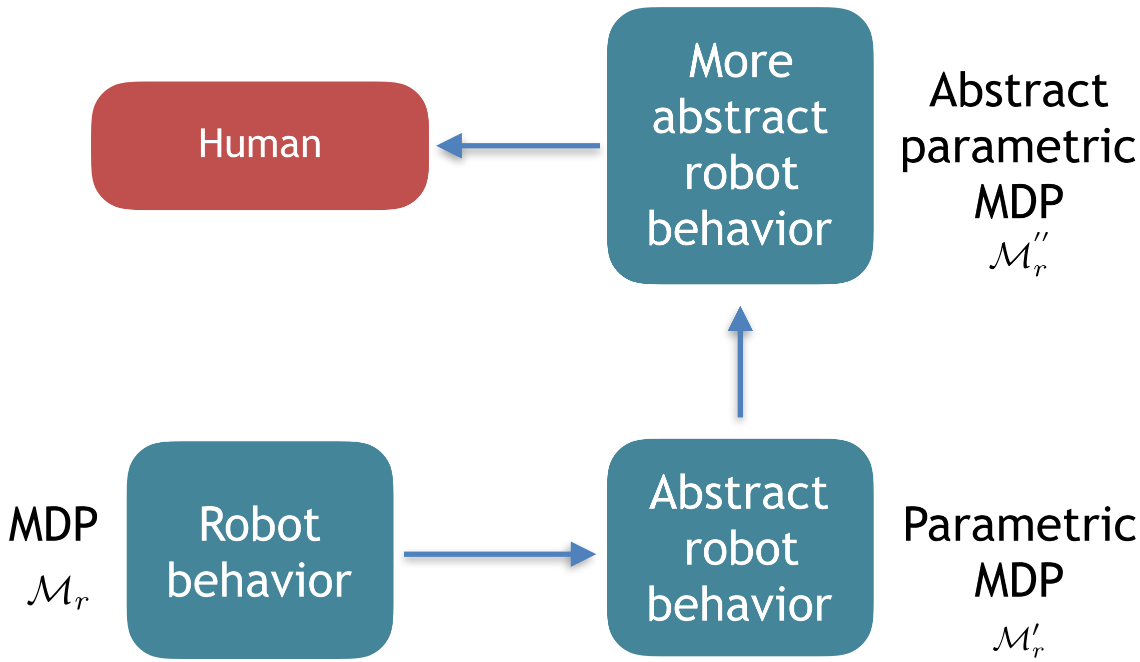
Abstract state space



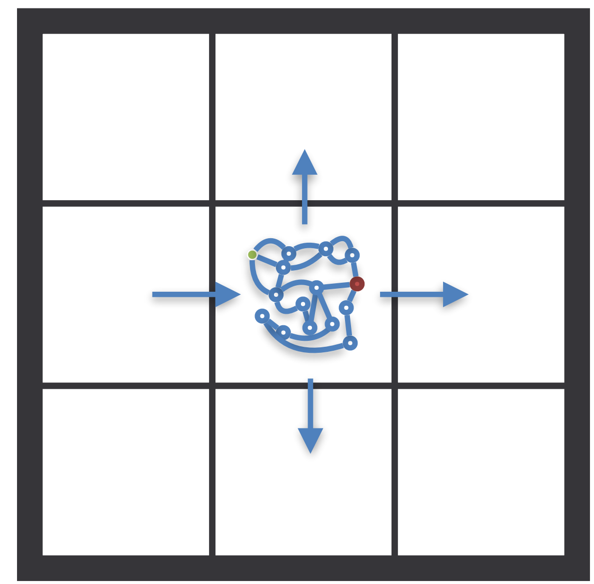
Abstract concrete probabilities by parameters

$$p_{high} \text{ if } \mathcal{P}_h(s, \alpha, s') > y$$
$$p_{low} \text{ if } \mathcal{P}_h(s, \alpha, s') \leq y$$

Human Model - Abstraction



Abstract state space

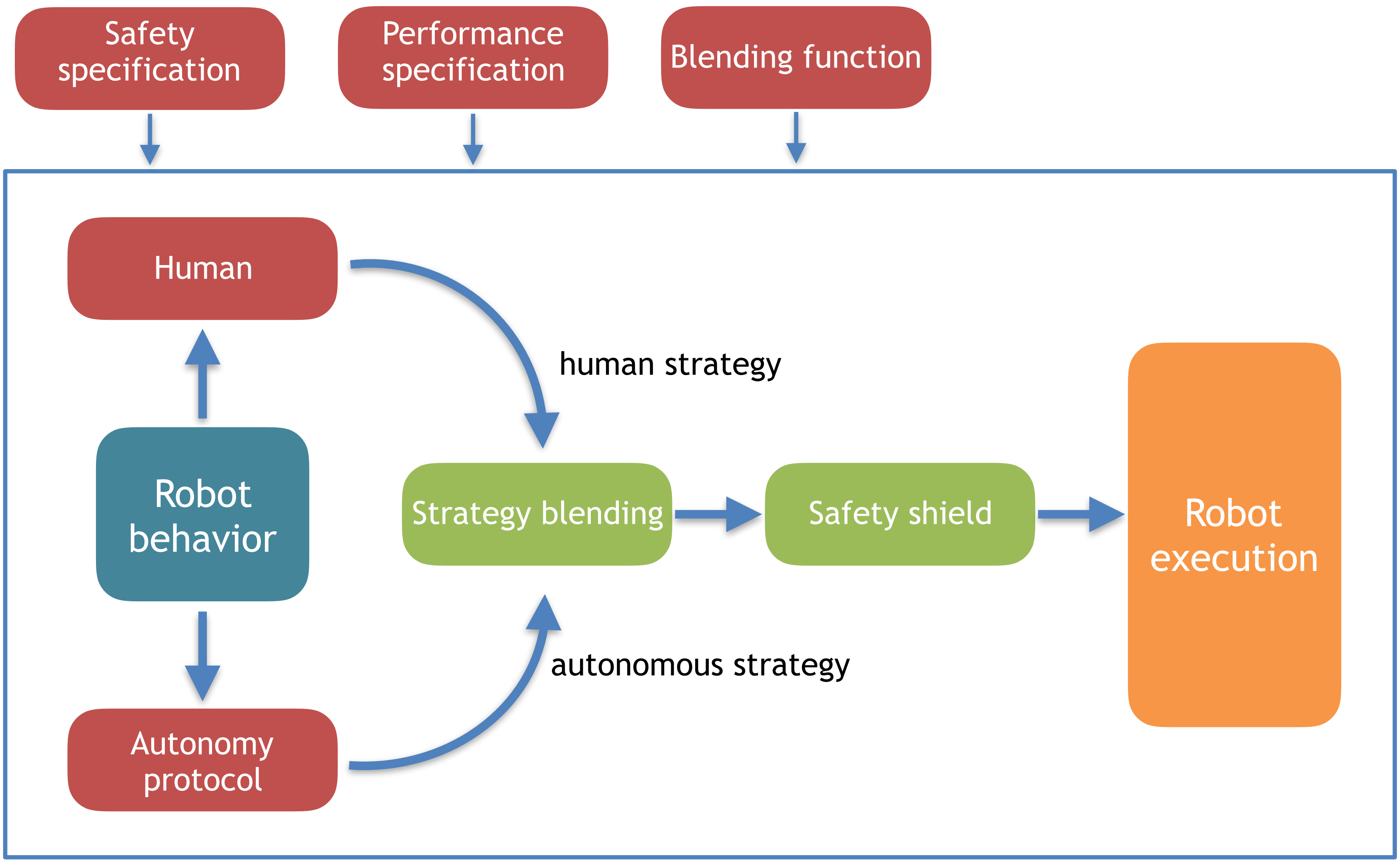


Abstract concrete probabilities by parameters

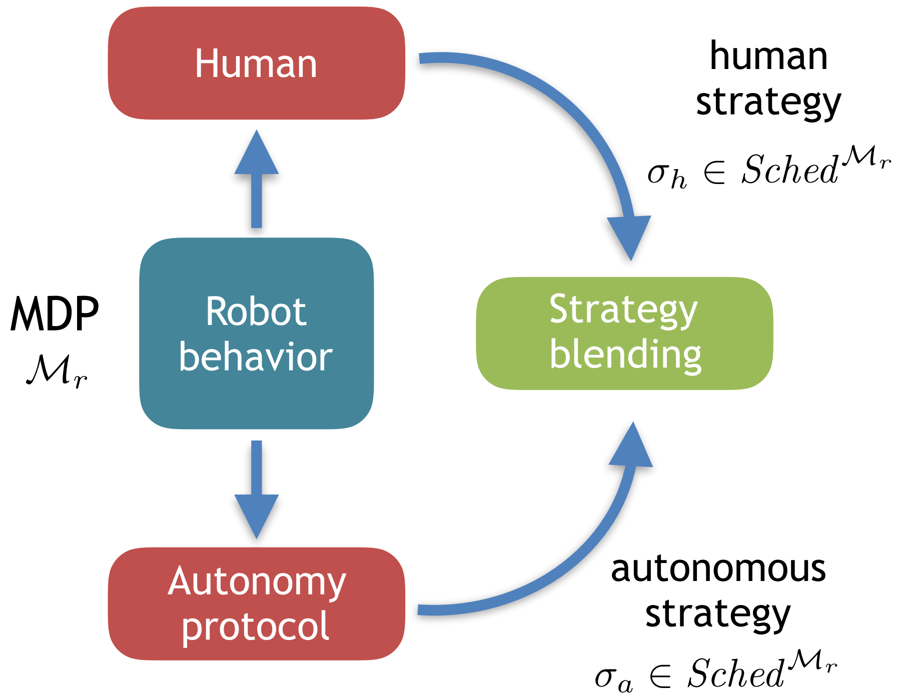
$$p_{high} \text{ if } \mathcal{P}_h(s, \alpha, s') > y$$
$$p_{low} \text{ if } \mathcal{P}_h(s, \alpha, s') \leq y$$

abstract strategy needs to be compatible with concrete state space

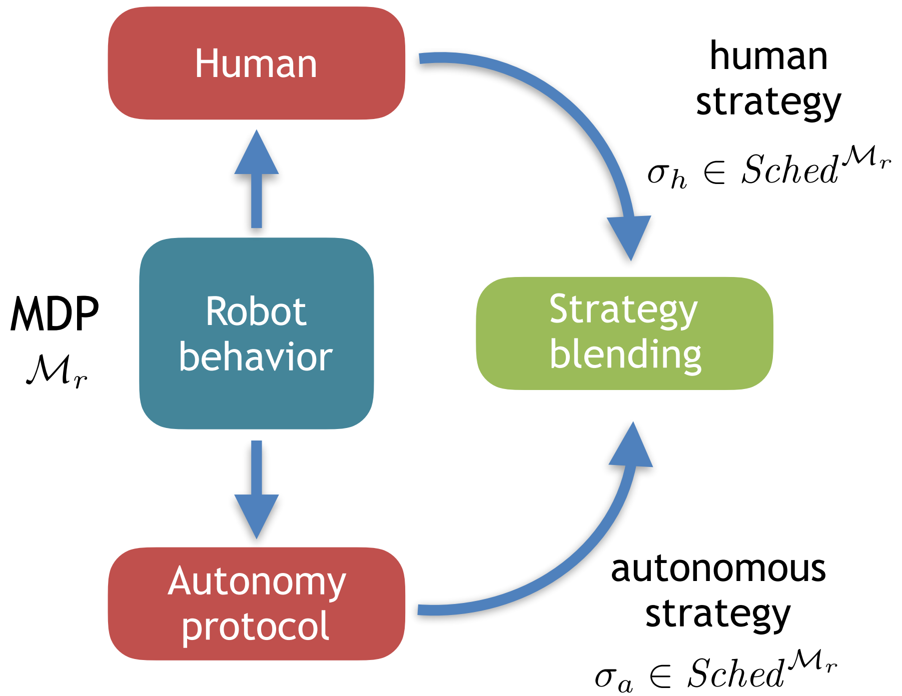
Shared Control Protocol



Strategy Blending



Strategy Blending

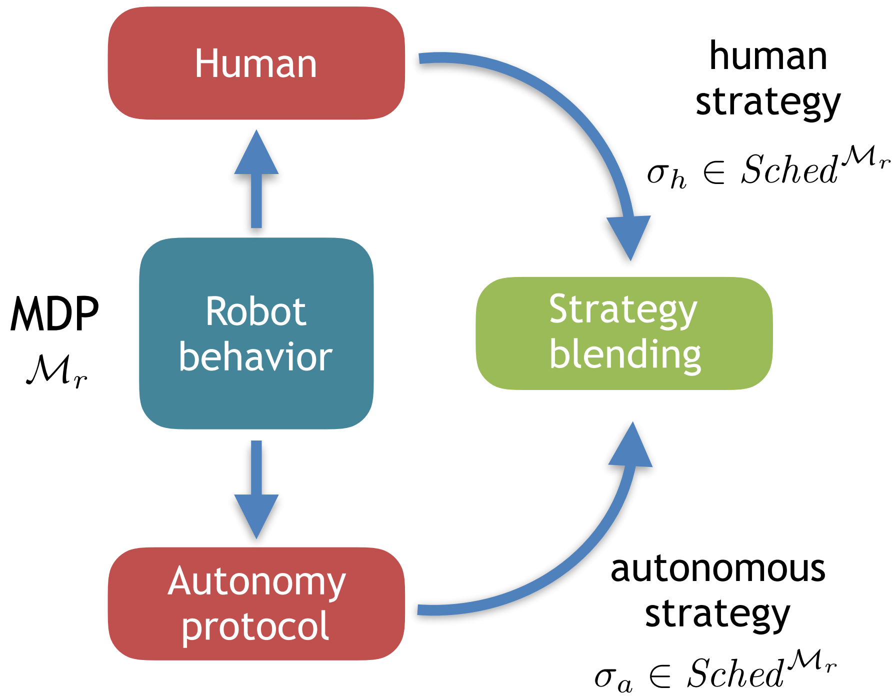


Blending function

$$f_a : S \to [0, 1]$$

- confidence in human
- level of abstraction for human
- accuracy of autonomy protocol

Strategy Blending



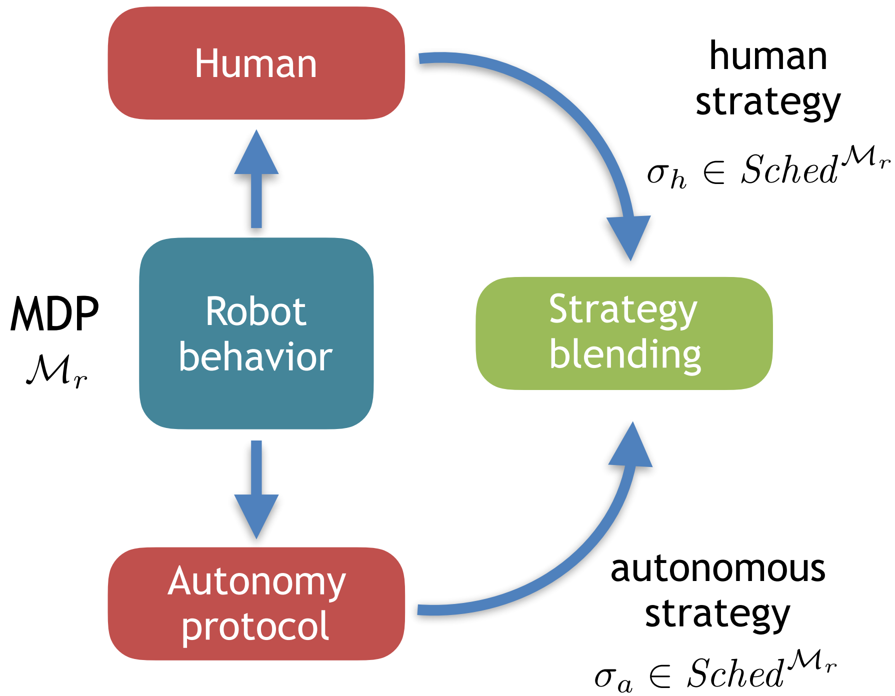
Blending function

$$f_a : S \to [0, 1]$$

- confidence in human
- level of abstraction for human
- accuracy of autonomy protocol

$$\sigma_{ah}(s, \alpha) = f_a(s) \cdot \sigma_h(s, \alpha) + (1 - f_a(s)) \cdot \sigma_a(s, \alpha)$$

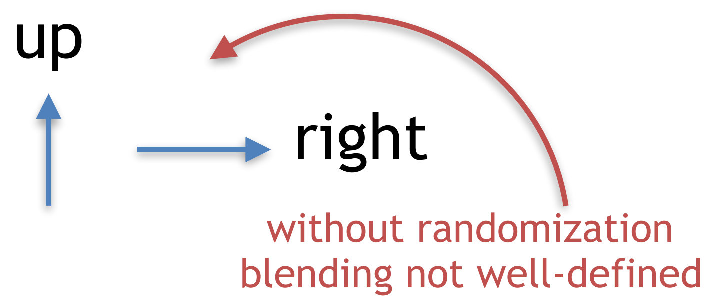
Strategy Blending



Blending function

$$f_a : S \to [0, 1]$$

- confidence in human
- level of abstraction for human
- accuracy of autonomy protocol



$$\sigma_{ah}(s, \alpha) = f_a(s) \cdot \sigma_h(s, \alpha) + (1 - f_a(s)) \cdot \sigma_a(s, \alpha)$$

Design of Autonomy Protocol

Compute autonomous strategy such that

Design of Autonomy Protocol

Compute autonomous strategy such that

- blended strategy deviates minimally from human strategy

Design of Autonomy Protocol

Compute autonomous strategy such that

- blended strategy deviates minimally from human strategy

- safety and performance specs are satisfied

Design of Autonomy Protocol

Compute autonomous strategy such that

- blended strategy deviates minimally from human strategy

- safety and performance specs are satisfied

- if not feasible, obtain new blending function

Design of Autonomy Protocol

Compute autonomous strategy such that

- blended strategy deviates minimally from human strategy

minimal additive perturbation of human strategy

- safety and performance specs are satisfied

model checking

- if not feasible, obtain new blending function

minimal deviation from given blending function or safety shield

Design of Autonomy Protocol

$$
\begin{aligned}
\text{minimize} \quad & \|(\delta^{s\alpha} \mid s \in S, \alpha \in Act)\| \\
\text{such that} \quad & \\
& p_{s_I} \leq \lambda \\
\forall s \in T. \quad & p_s = 1 \\
\forall s \in S. \quad & \sum_{\alpha \in Act} \sigma_a^{s,\alpha} = \sum_{\alpha \in Act} \sigma_{ah}^{s,\alpha} = 1 \\
\forall s \in S. \forall \alpha \in Act. \quad & \sigma_{ah}^{s,\alpha} = \sigma_h(s)(\alpha) + \delta^{s,\alpha} \\
\forall s \in S. \quad & \sum_{\alpha \in Act} \delta^{s,\alpha} = 0 \\
\forall s \in S. \forall \alpha \in Act. \quad & \sigma_{ah}^{s,\alpha} = f_a(s) \cdot \sigma_h(s)(\alpha) + (1 - f_a(s)) \cdot \sigma_a^{s,\alpha} \\
\forall s \in S. \quad & p_s = \sum_{\alpha \in Act} \sigma_{ah}^{s,\alpha} \cdot \sum_{s' \in S} \mathcal{P}(s,\alpha)(s') \cdot p_{s'}
\end{aligned}
$$

Design of Autonomy Protocol

$$
\begin{aligned}
\text{minimize} \quad & \|(\delta^{s\alpha} \mid s \in S, \alpha \in Act)\| \\
\text{such that} \quad & \\
& p_{s_I} \leq \lambda \\
\forall s \in T. \quad & p_s = 1 \\
\forall s \in S. \quad & \sum_{\alpha \in Act} \sigma_a^{s,\alpha} = \sum_{\alpha \in Act} \sigma_{ah}^{s,\alpha} = 1 \\
\forall s \in S. \forall \alpha \in Act. \quad & \sigma_{ah}^{s,\alpha} = \sigma_h(s)(\alpha) + \delta^{s,\alpha} \\
\forall s \in S. \quad & \sum_{\alpha \in Act} \delta^{s,\alpha} = 0 \\
\forall s \in S. \forall \alpha \in Act. \quad & \sigma_{ah}^{s,\alpha} = f_a(s) \cdot \sigma_h(s)(\alpha) + (1 - f_a(s)) \cdot \sigma_a^{s,\alpha} \\
\forall s \in S. \quad & p_s = \sum_{\alpha \in Act} \sigma_{ah}^{s,\alpha} \cdot \sum_{s' \in S} \mathcal{P}(s,\alpha)(s') \cdot p_{s'}
\end{aligned}
$$

non-linear programming

Current/Future Work

- avoid non-linear program
- model repair
- convex form
- include permissiveness into autonomy
- start conducting real case studies

Thank you for your attention!