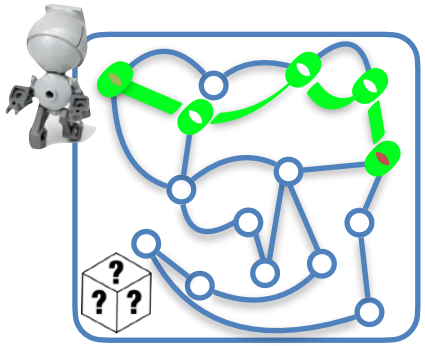
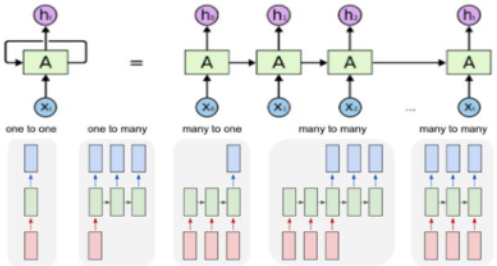
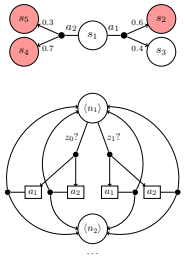
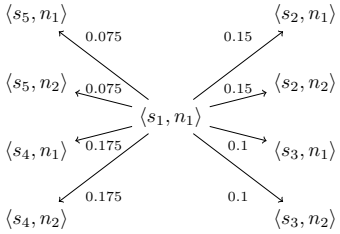
Planning under Uncertainty

AI Safety via Formal Verification









Nils Jansen

NFM AI-SAFETY, May 11, 2020

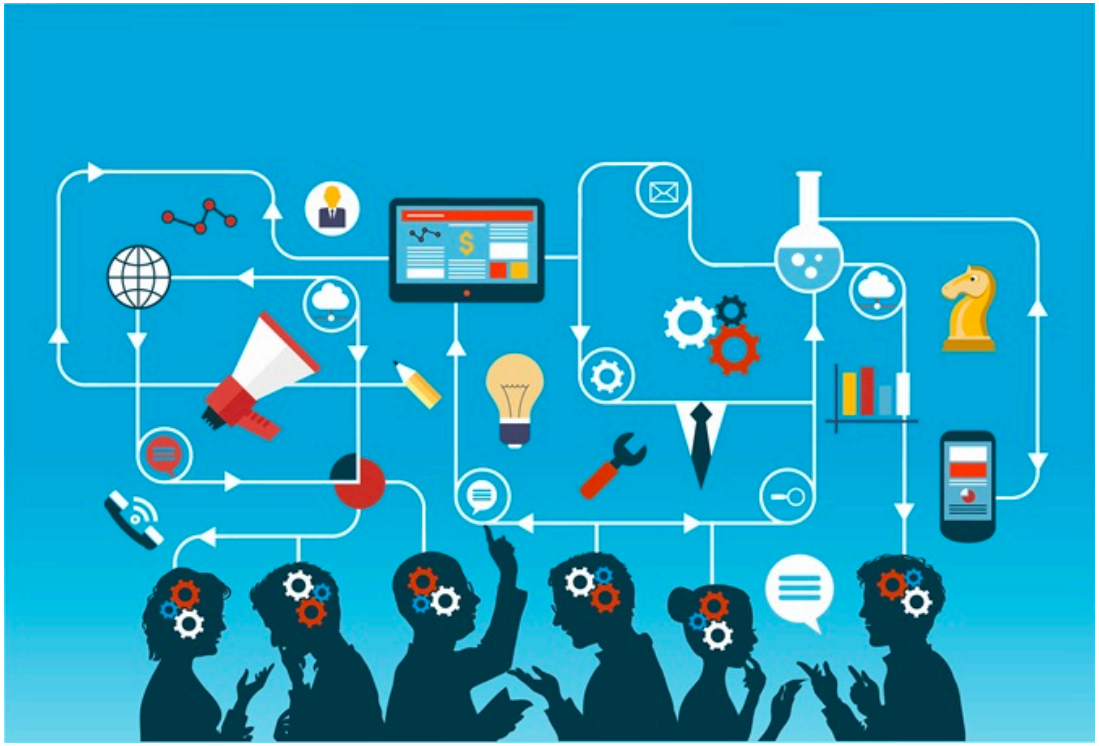
joint work with:

Bernd Becker, Steve Carr, Murat Cubuktepe,
Alexandru Serban, Marnix Suilen, Ufuk Topcu, Ralf Wimmer,

TEXAS
The University of Texas at Austin

UNI FREIBURG

Radboud University



AI Safety - A Restricted View

Neural Network Controllers

- how to ensure safe decision-making?
- what is safety?

Partial Information

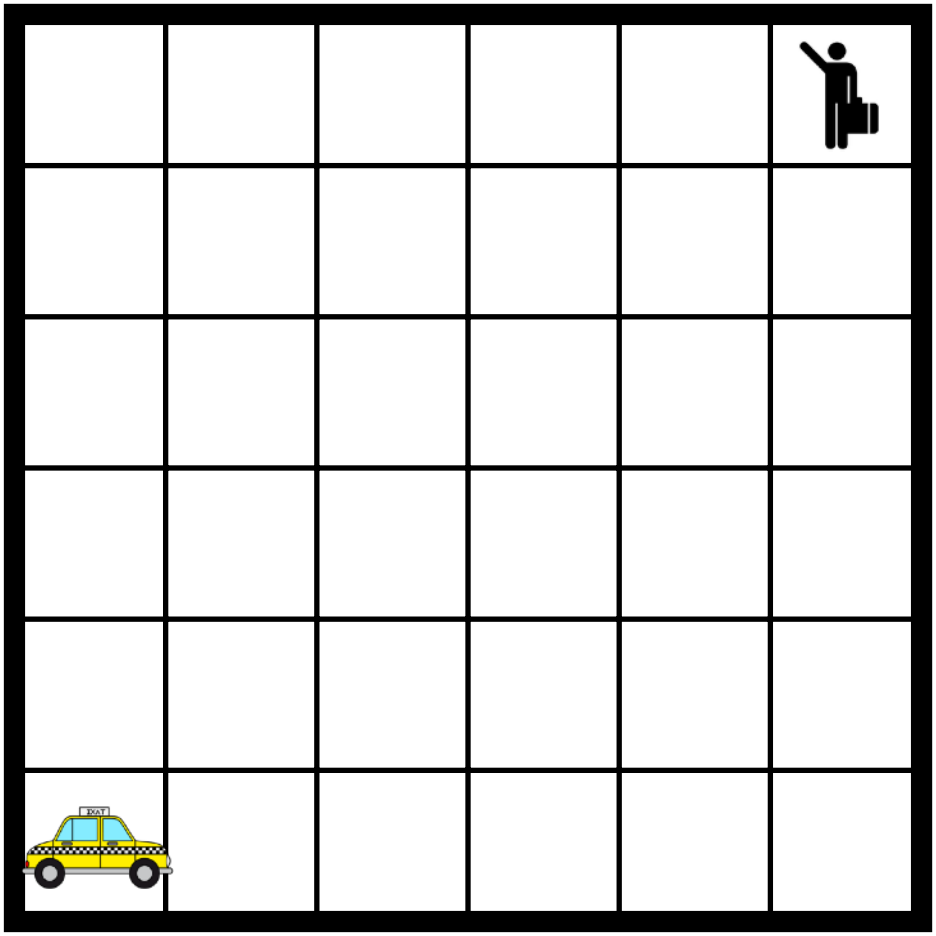
- how to operate if the system state is not fully observable?
- what guarantees can be given?

Uncertainty

- how to robustly account for imperfect sensor information?
- what is robustness?

Help the Taxi

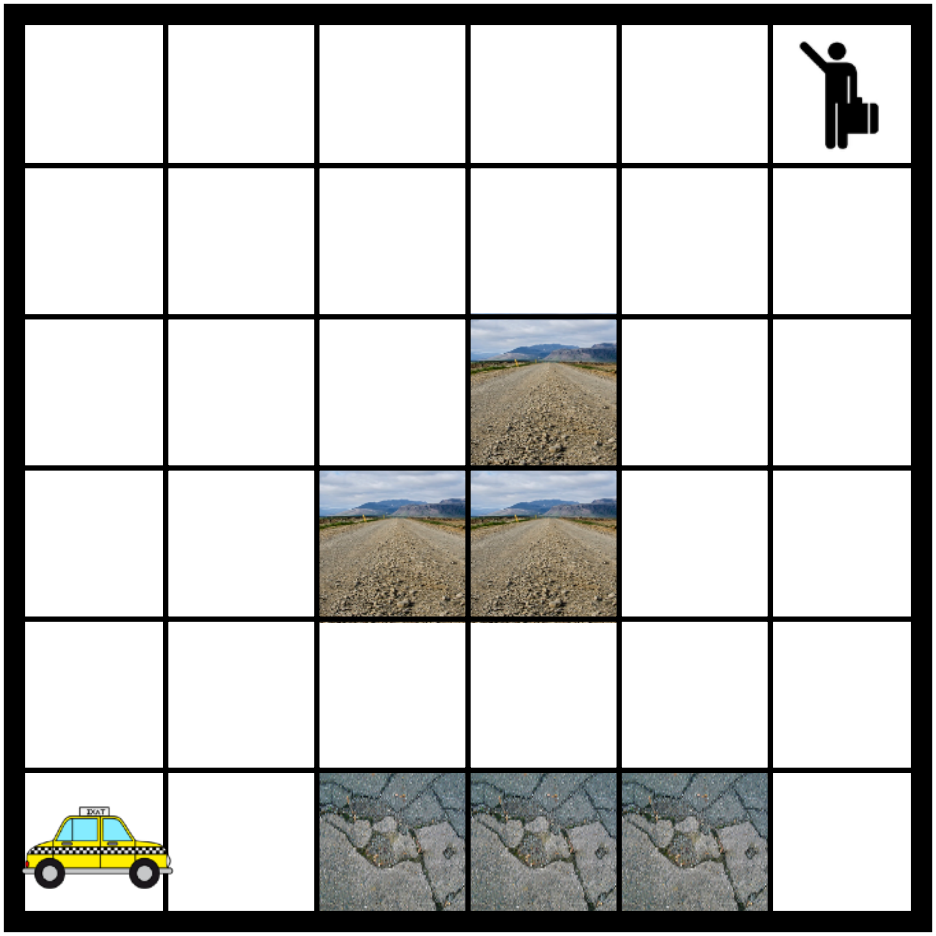
Find the best way to the passenger



Help the Taxi

Find the best way to the passenger

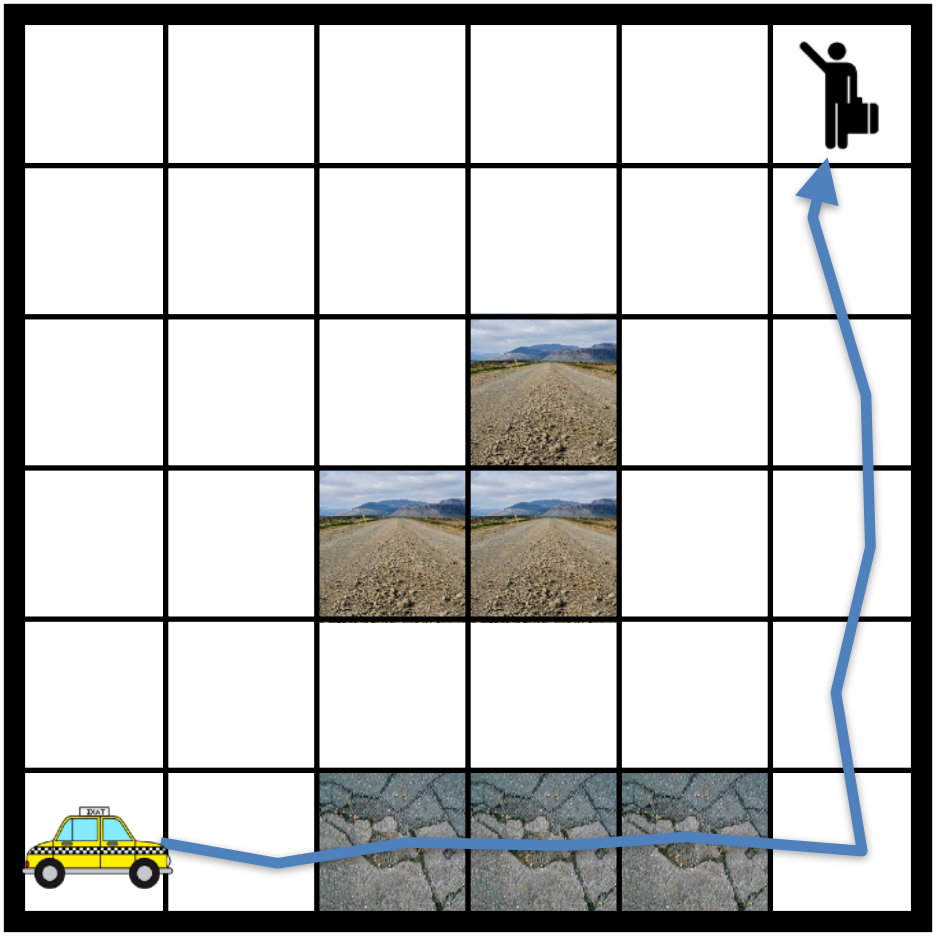
Take bad roads into account



Help the Taxi

Find the best way to the passenger

Take **bad roads** into account

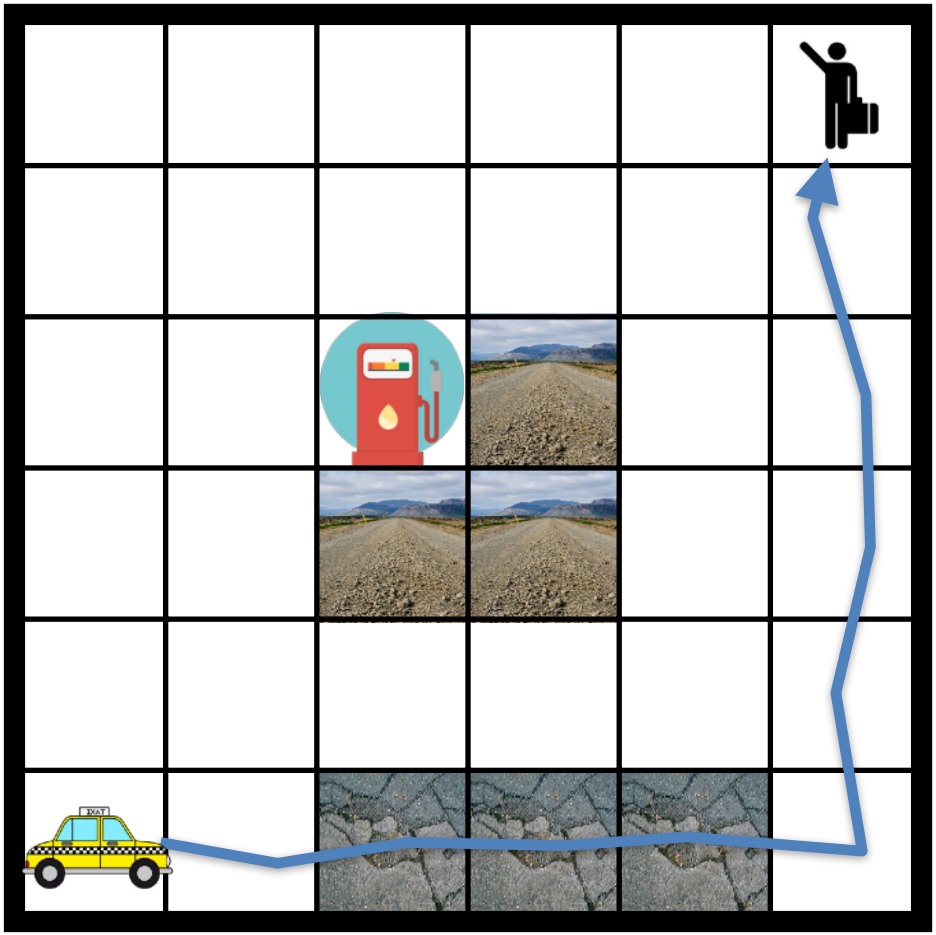


Help the Taxi

Find the best way to the passenger

Take **bad roads** into account

Refuel on the way



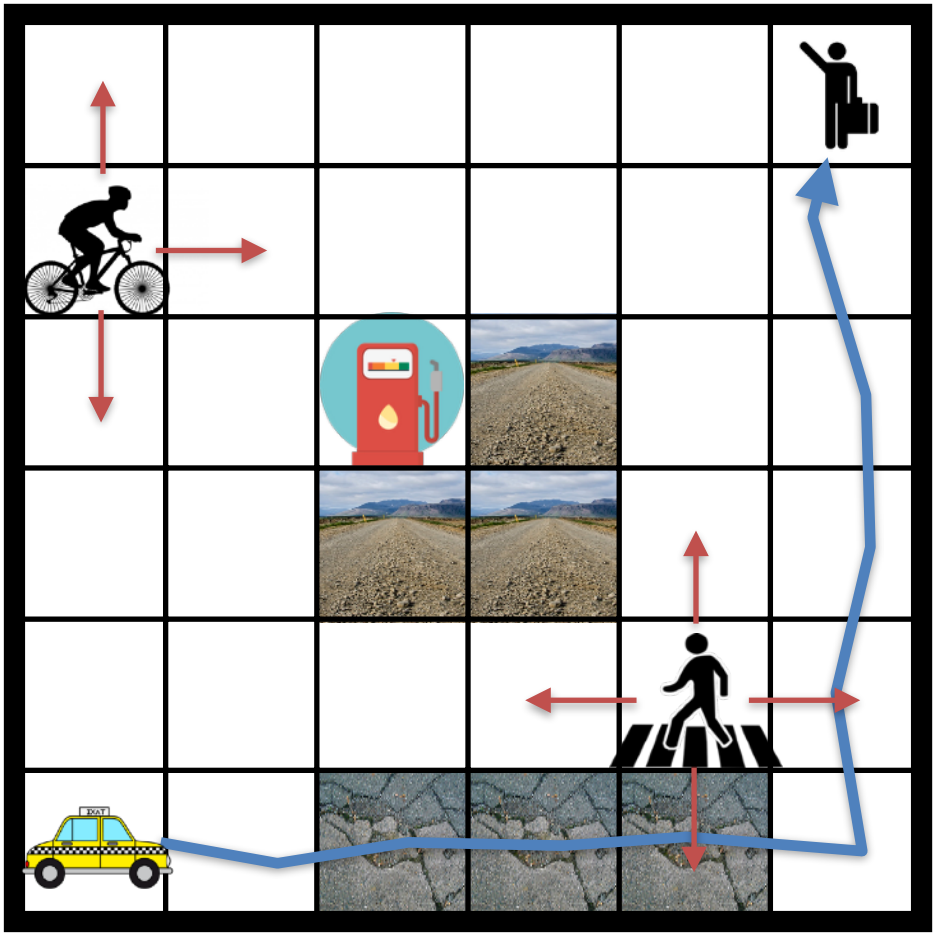
Help the Taxi

Find the best way to the passenger

Take **bad roads** into account

Refuel on the way

Avoid randomly moving **pedestrians and cyclists**



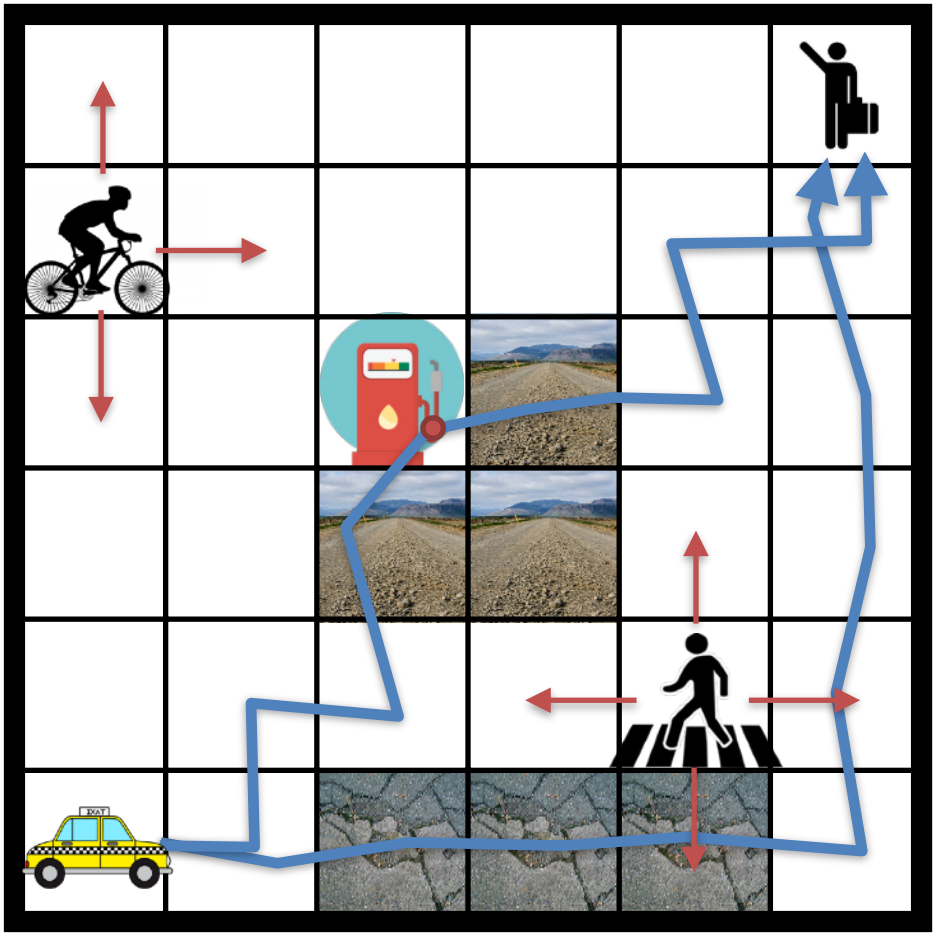
Help the Taxi

Find the best way to the passenger

Take **bad roads** into account

Refuel on the way

Avoid randomly moving **pedestrians and cyclists**



Help the Taxi

Find the best way to the passenger

Take **bad roads** into account

Refuel on the way

Avoid randomly moving **pedestrians and cyclists**

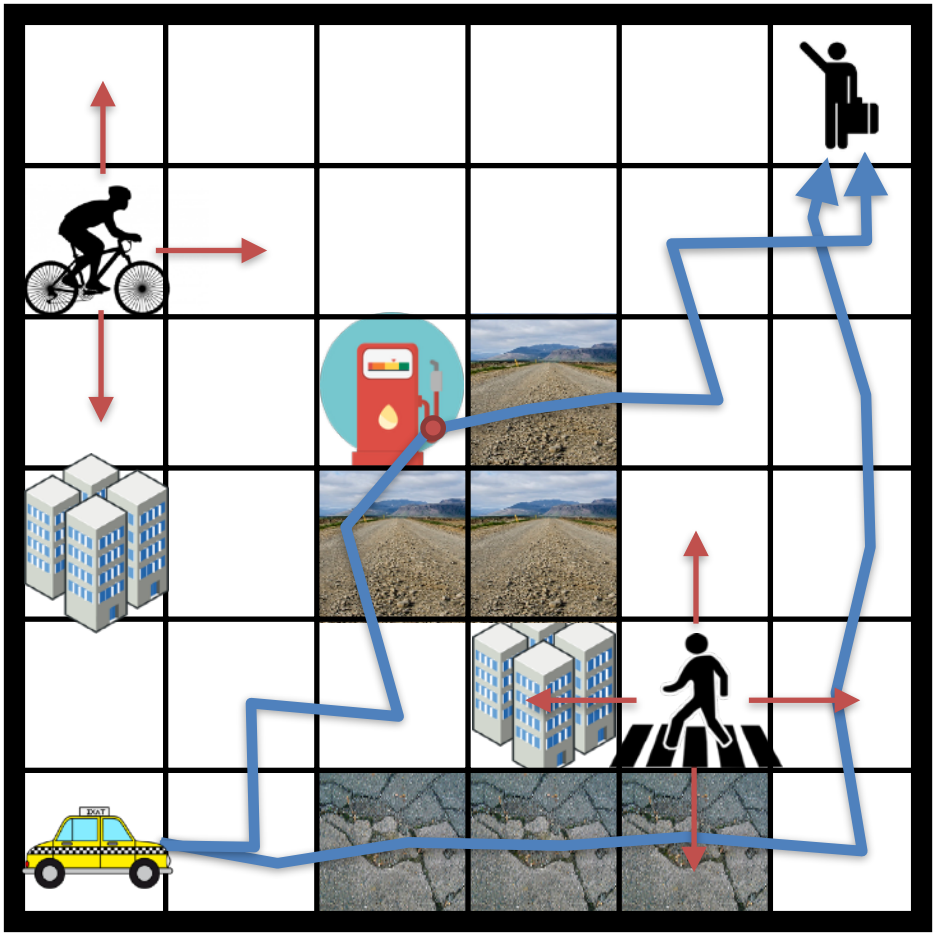
Account for **restricted view**

$Pr_{max}(\neg(\text{ped} \vee \text{cyc})\ \mathsf{U}\ (\Diamond(\text{fuel} \wedge \bigcirc\Diamond\text{pass}))$

temporal logic

$EC_{min}(\Diamond\text{passenger})$

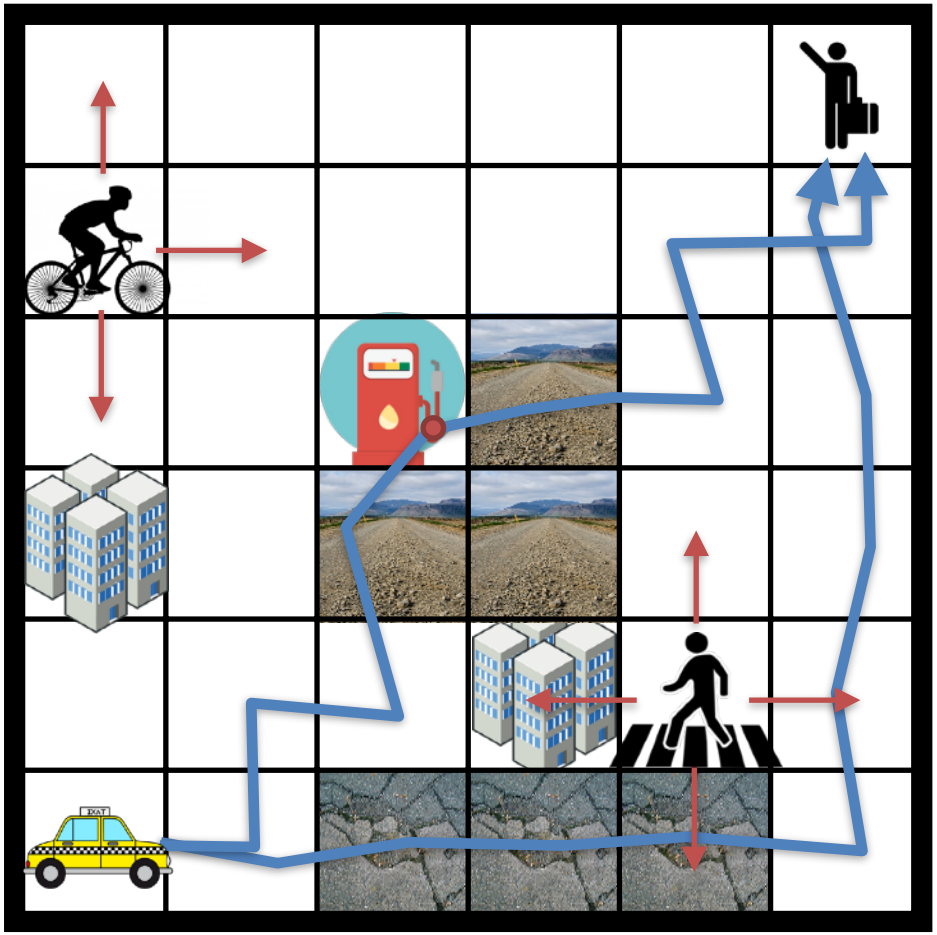
expected cost



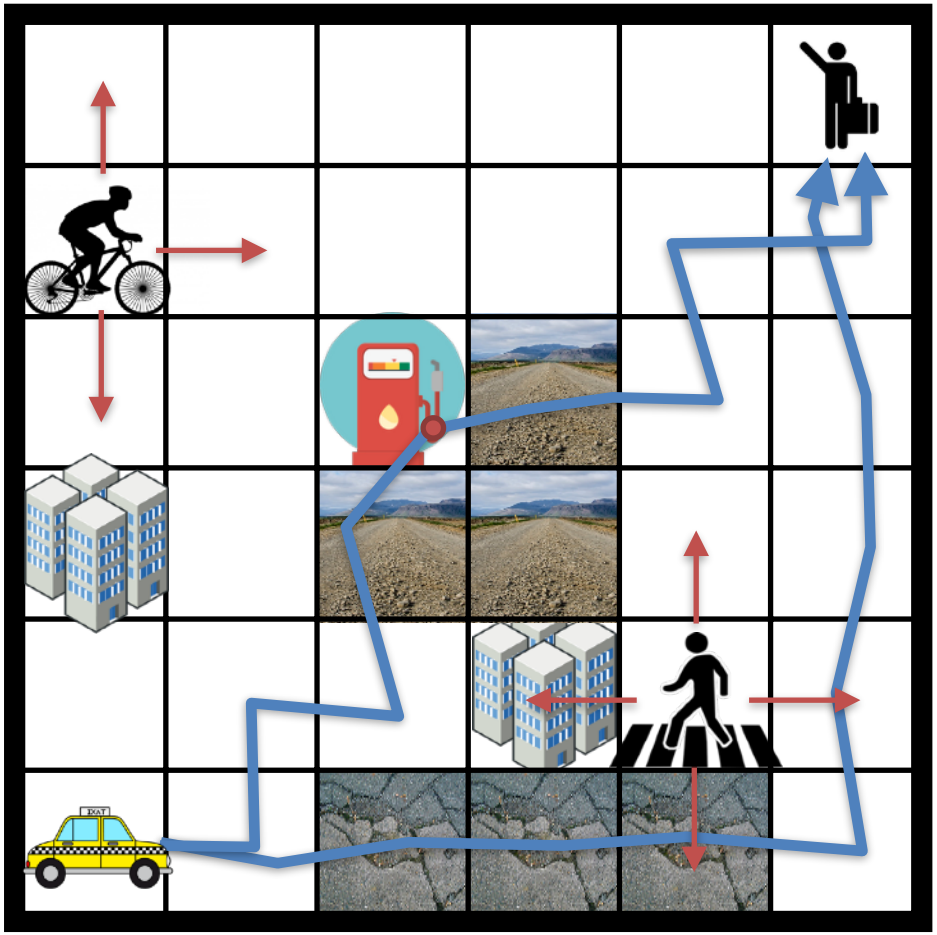
Find **safe and/or cost-optimal** policy to get to the passenger

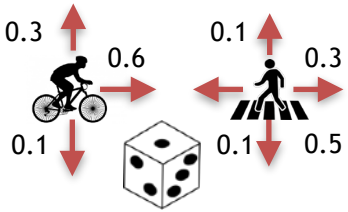
Account for **randomness and partial information**

Help the Taxi: A Closer Look



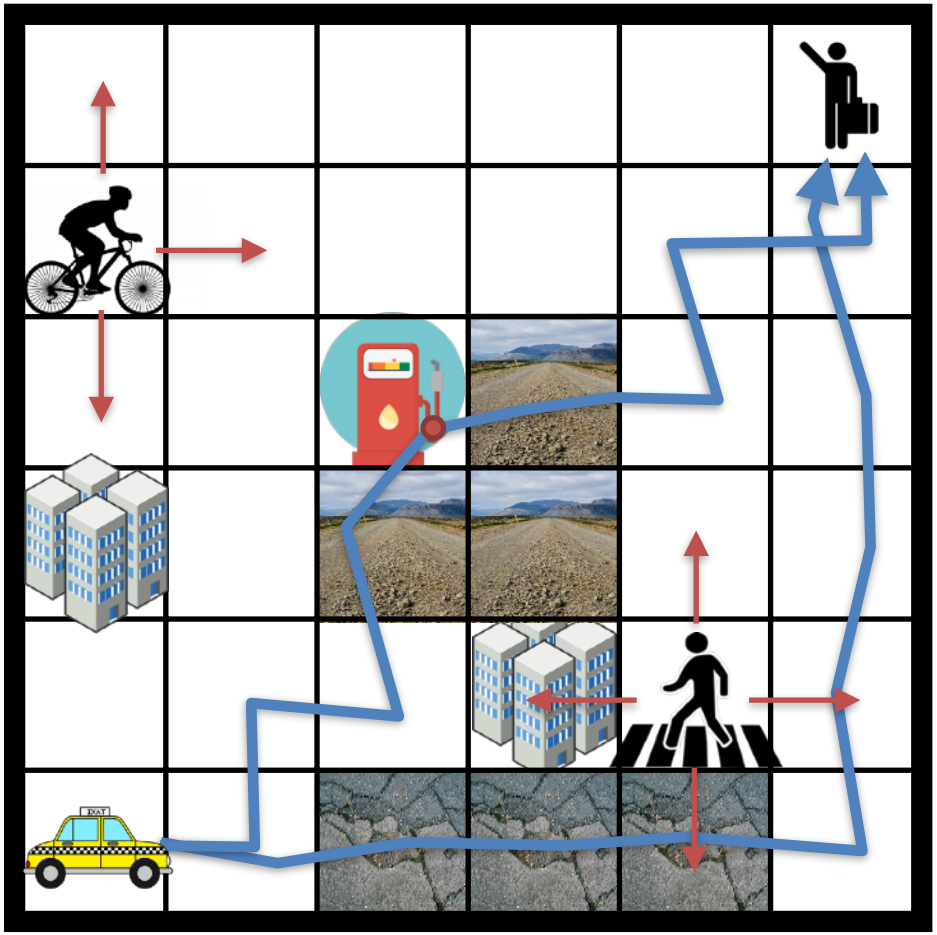
Help the Taxi: A Closer Look

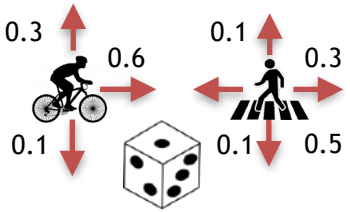




probability distributions on movements

Help the Taxi: A Closer Look

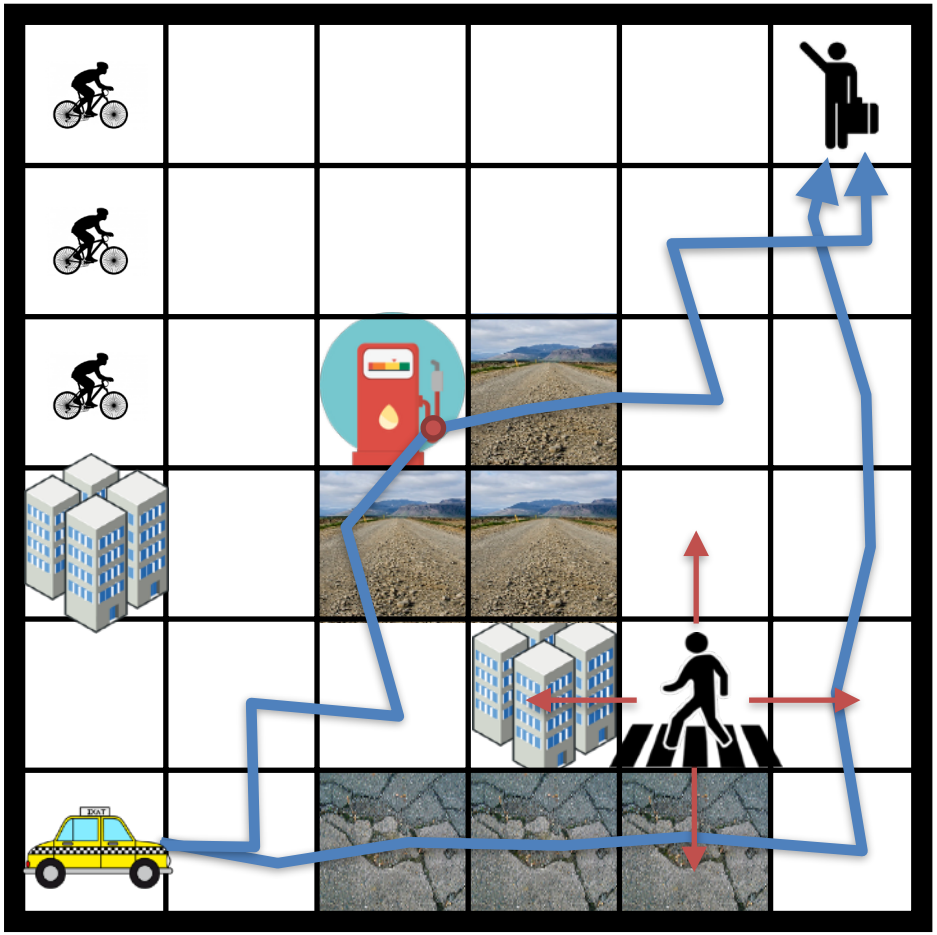


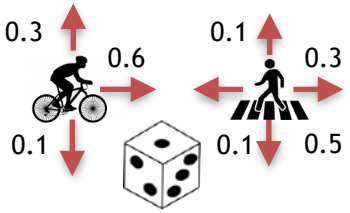


probability distributions on movements

efficient control via neural network

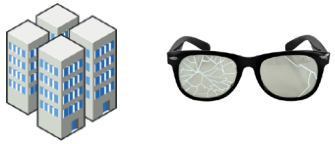
Help the Taxi: A Closer Look





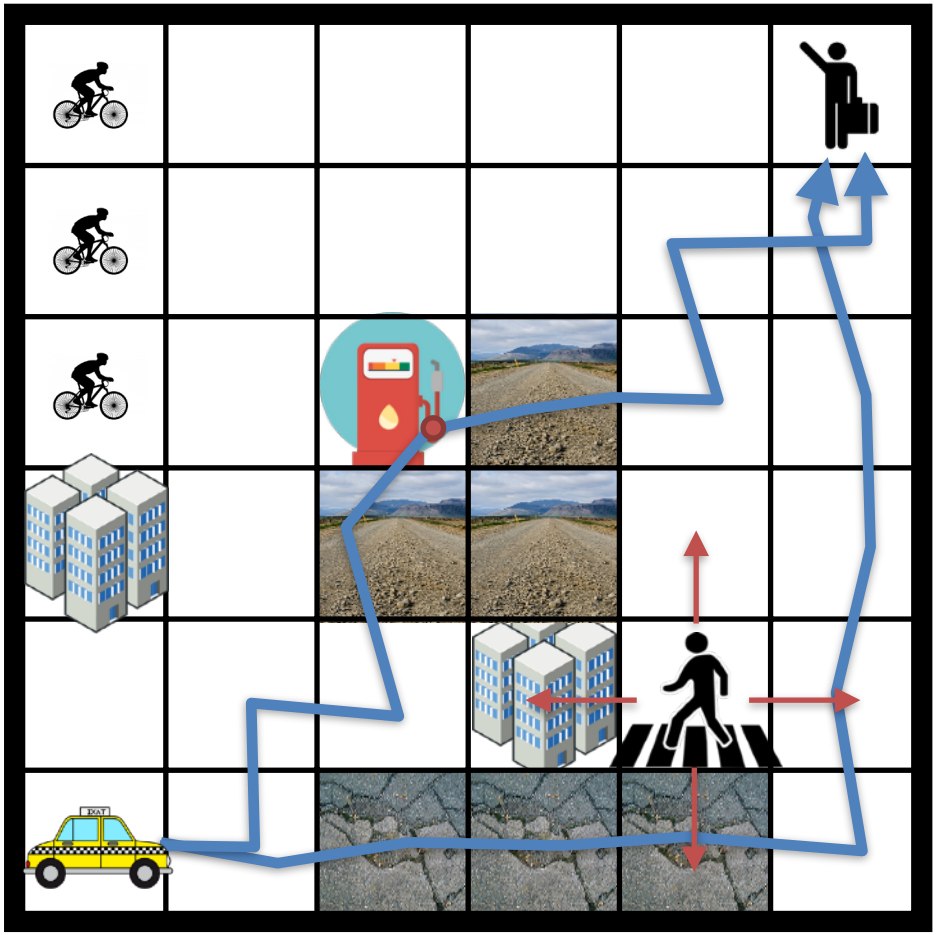
probability distributions on movements

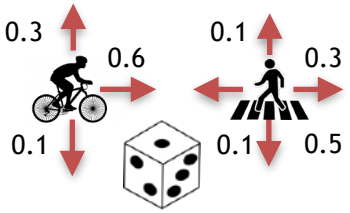
efficient control via neural network



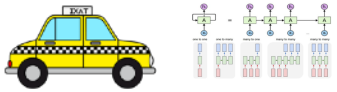
partial observability induces unknown system state

Help the Taxi: A Closer Look

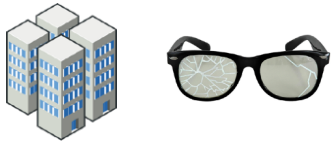




probability distributions on movements



efficient control via neural network



partial observability induces unknown system state

Underlying formal problem:
Find a provably correct Neural Network controller for a partially observable Markov Decision Process

Two Notoriously Hard Problems

Underlying formal problem:
Find a provably correct Neural Network controller for a partially observable Markov Decision Process

1. Verification of Neural Networks
2. POMDP synthesis

Two Notoriously Hard Problems

Underlying formal problem:
Find a provably correct Neural Network controller for a partially observable Markov Decision Process

1. Verification of Neural Networks
2. POMDP synthesis

- Randomized with infinite memory: undecidable, optimal results.

Two Notoriously Hard Problems

Underlying formal problem:
Find a provably correct Neural Network controller for a partially observable Markov Decision Process

1. Verification of Neural Networks
2. POMDP synthesis

- Randomized with **infinite memory**: undecidable, optimal results.
- Randomized with **finite memory**: NP-hard, SQRT-SUM-hard, in PSPACE, not optimal in general, but sufficient for many applications.

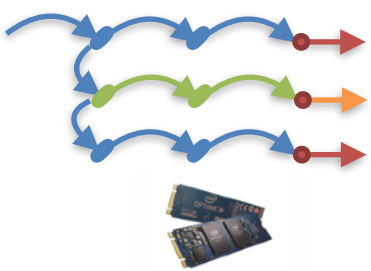
Two Notoriously Hard Problems

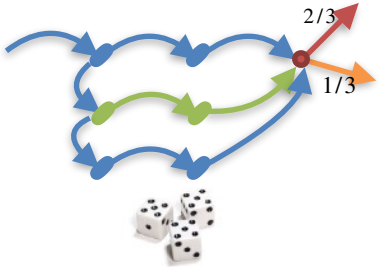
Underlying formal problem:
Find a provably correct Neural Network controller for a partially observable Markov Decision Process

1. Verification of Neural Networks
2. POMDP synthesis

- Randomized with **infinite memory**: undecidable, optimal results.

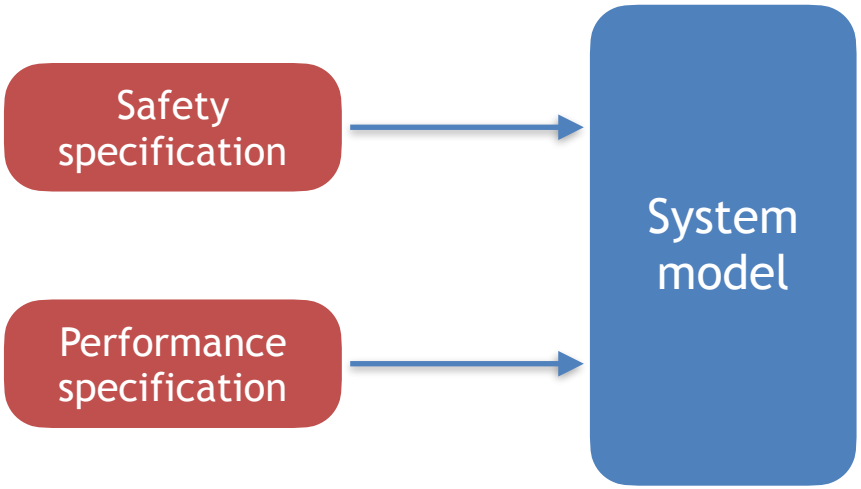
- Randomized with **finite memory**: NP-hard, SQRT-SUM-hard, in PSPACE, not optimal in general, but sufficient for many applications.
- Intuitively: Randomization can often trade off memory.



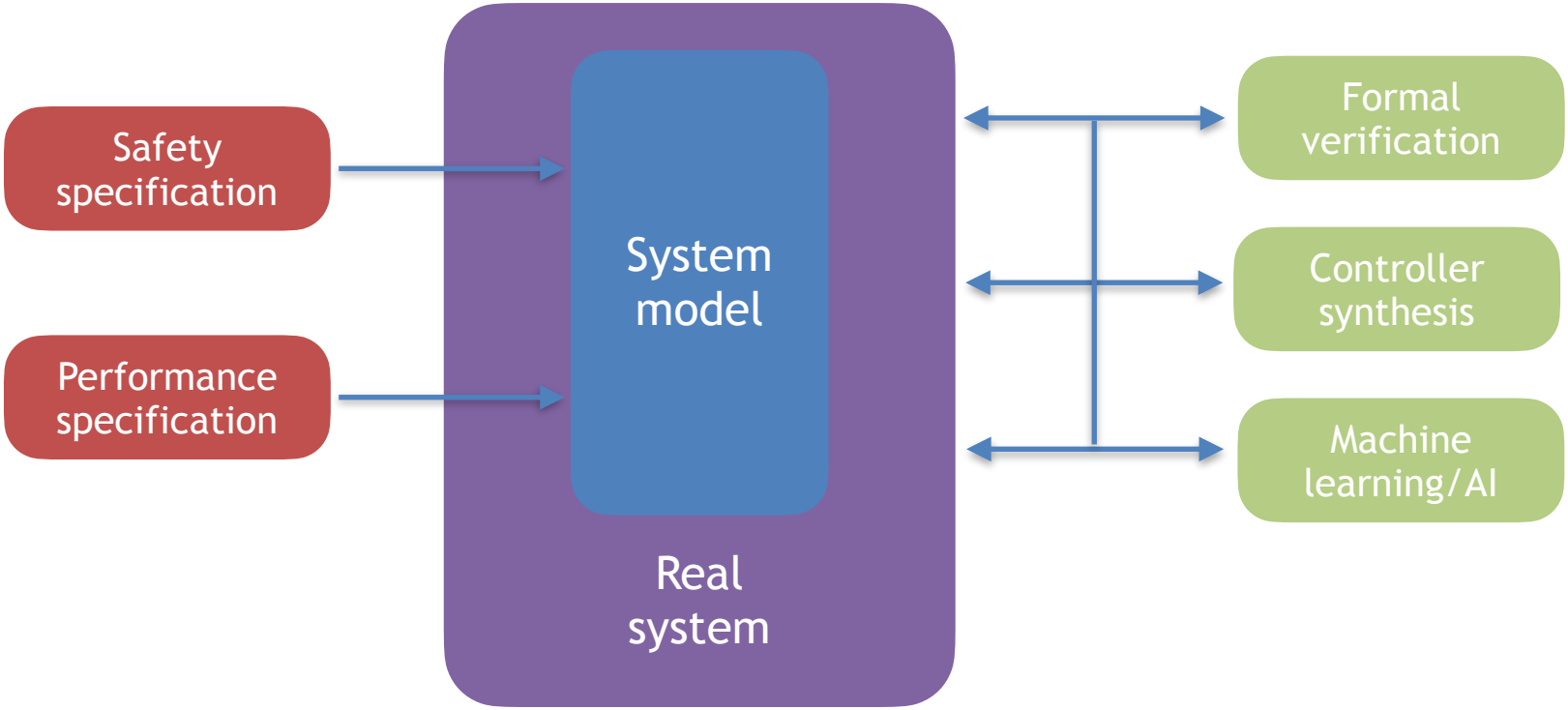


$\sigma : ObsSeq_{fin} \rightarrow Distr(Act)$ $\sigma : Obs \rightarrow Distr(Act)$

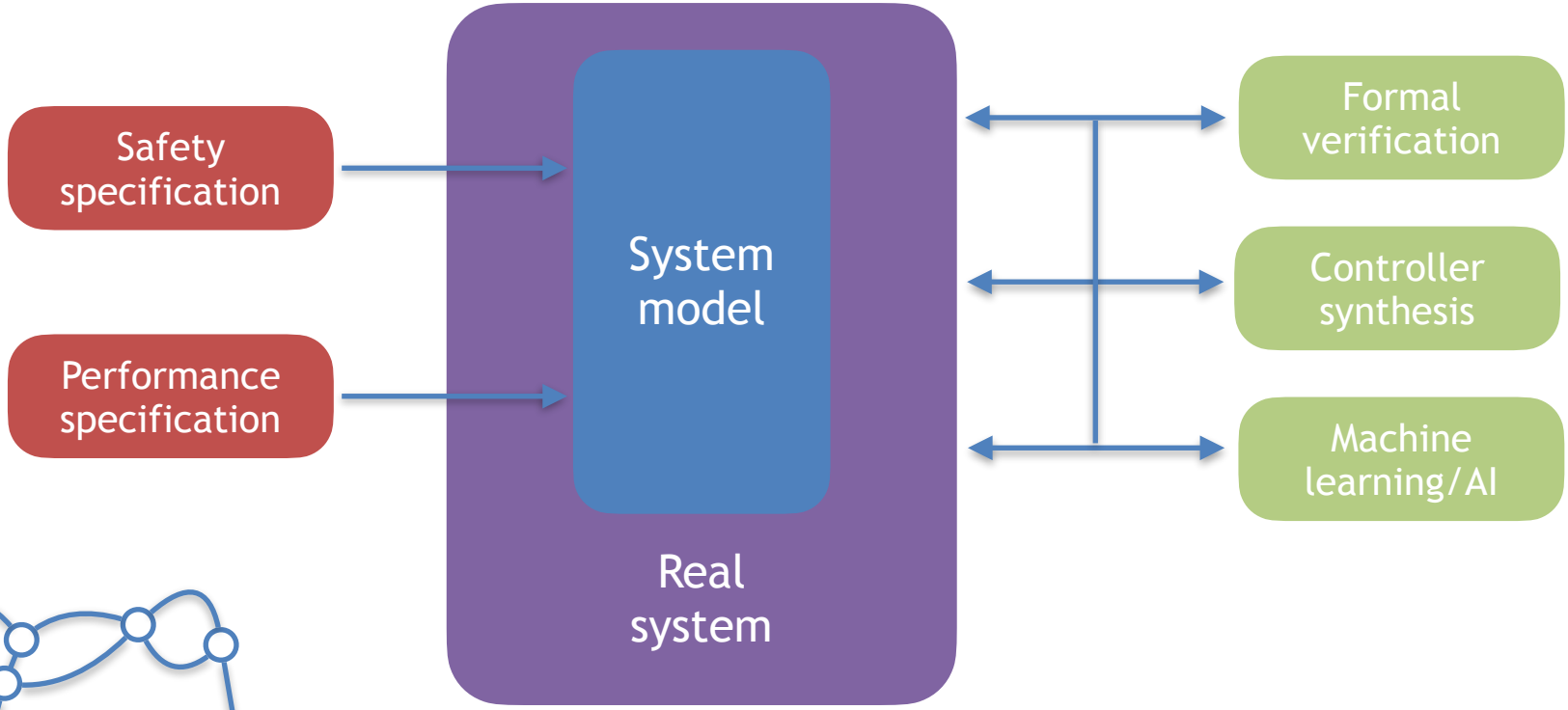
Decision-Making under Uncertainty



Decision-Making under Uncertainty



Decision-Making under Uncertainty



Model-based approaches: Probabilistic Models (POMDPs)

Solutions at the interfaces of domains

POMDPs - Applications

Stock Market

Surveying Threatened Species

Health Care

Wireless Sensor Networks

Autonomous Systems

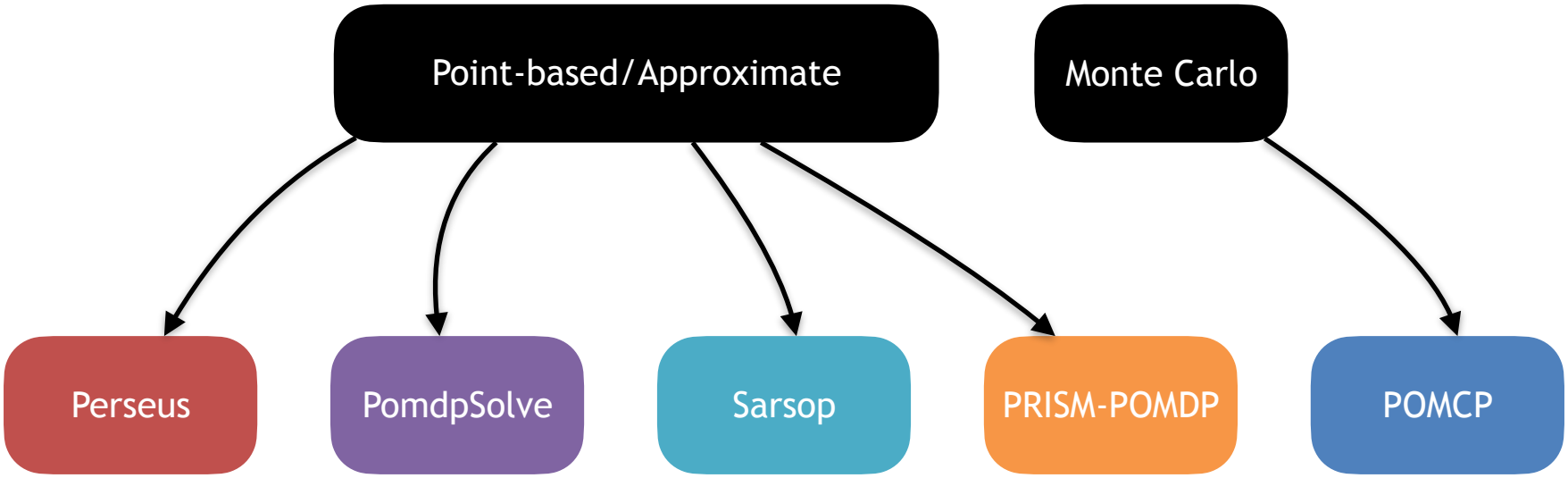
Machine Vision

POMDP Solving - State of The Art

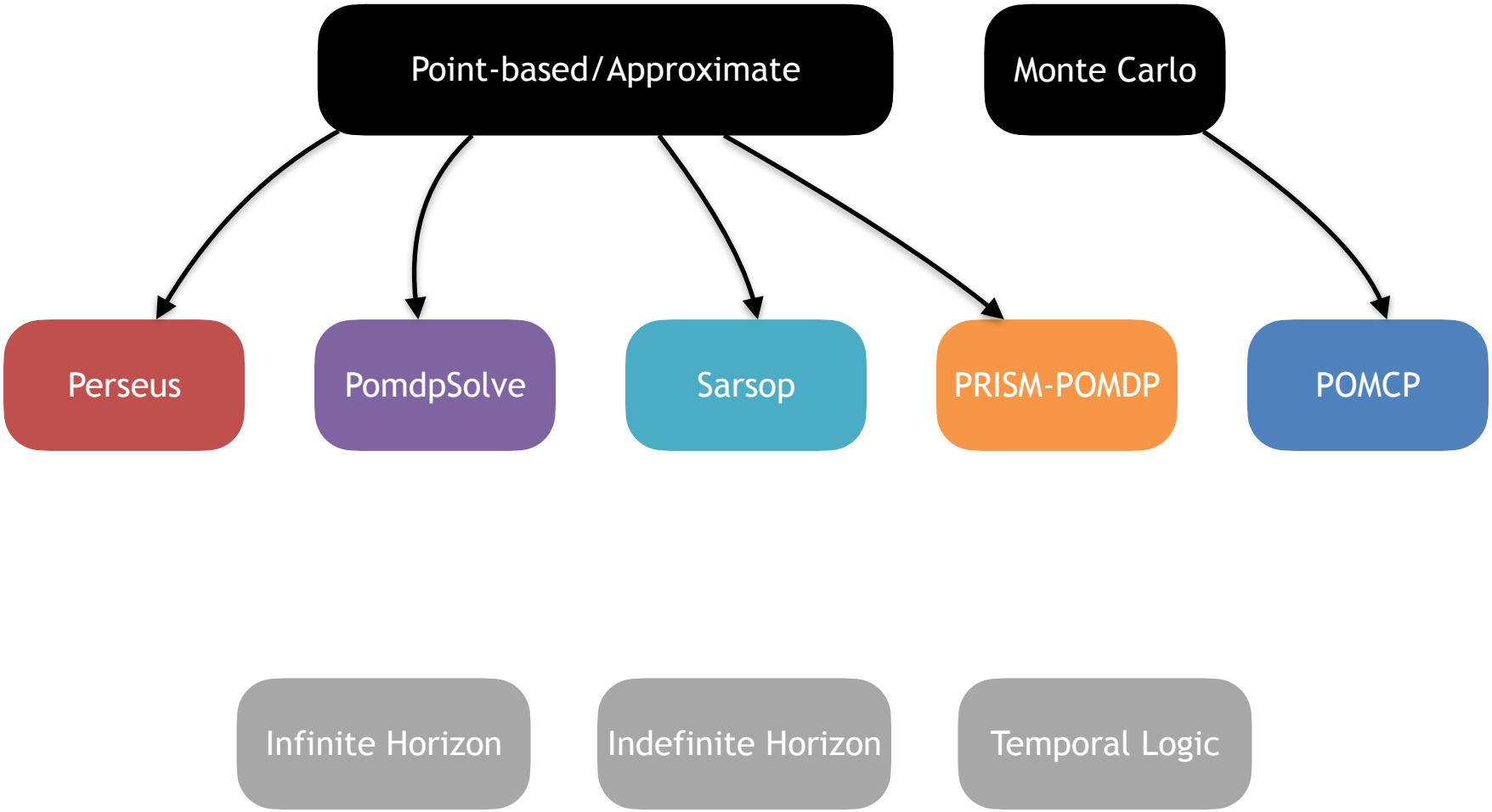
Point-based/Approximate

Monte Carlo

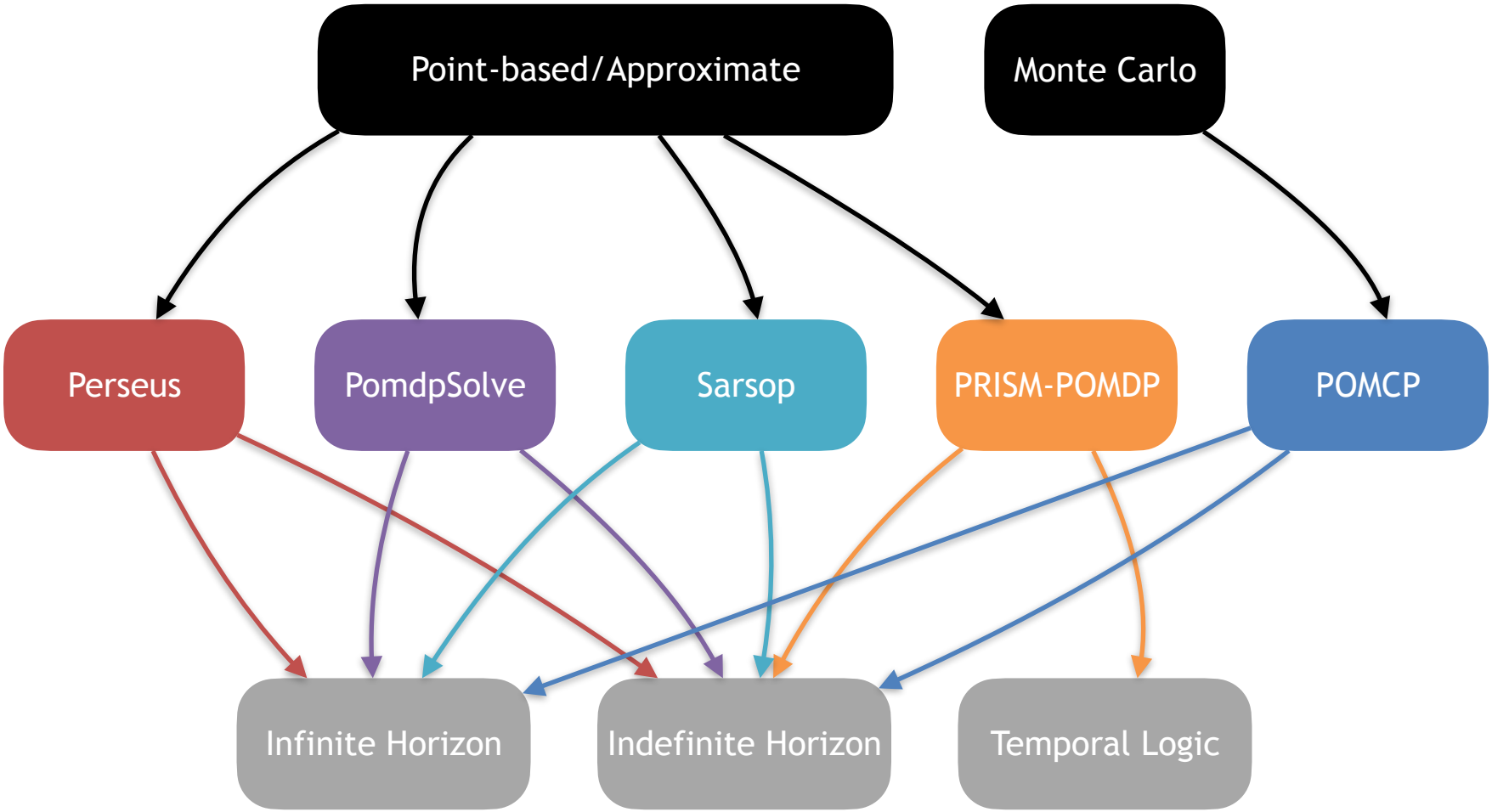
POMDP Solving - State of The Art



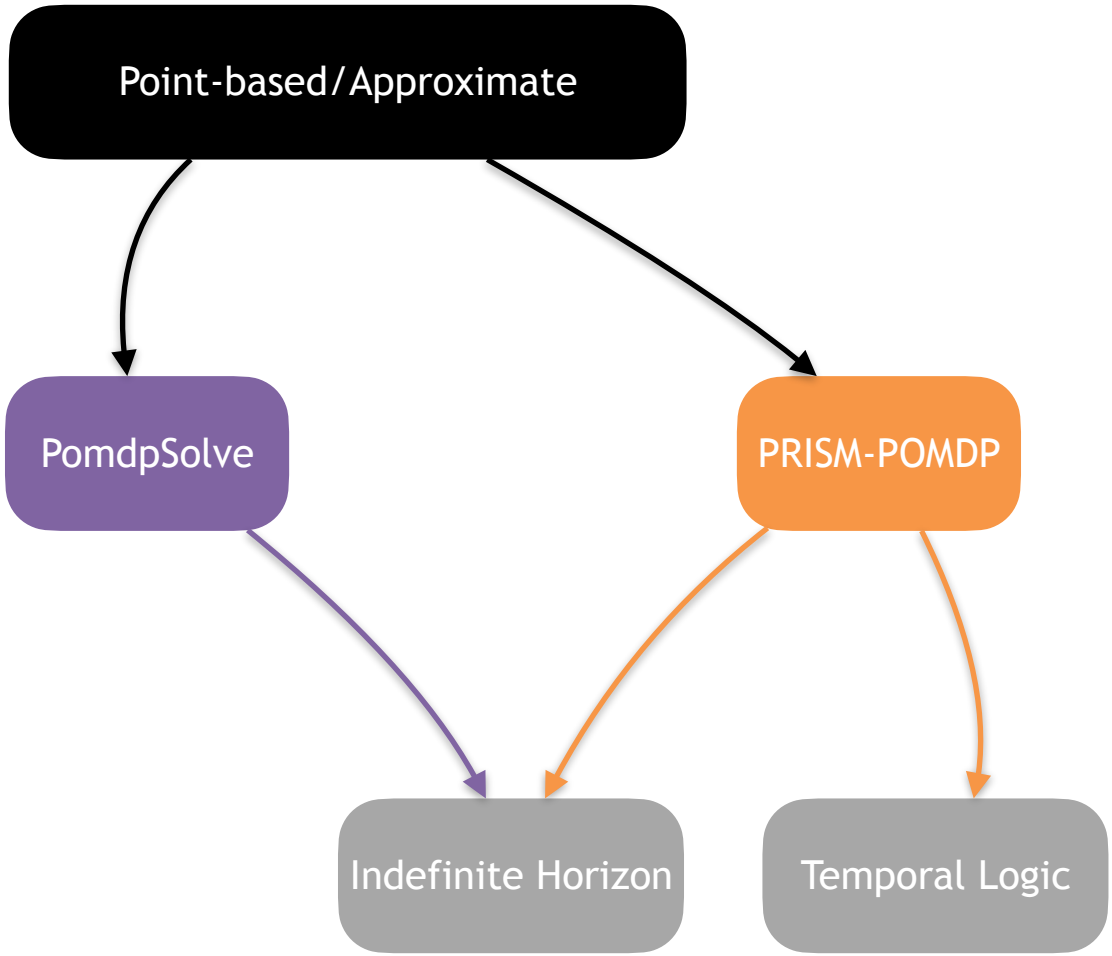
POMDP Solving - State of The Art



POMDP Solving - State of The Art

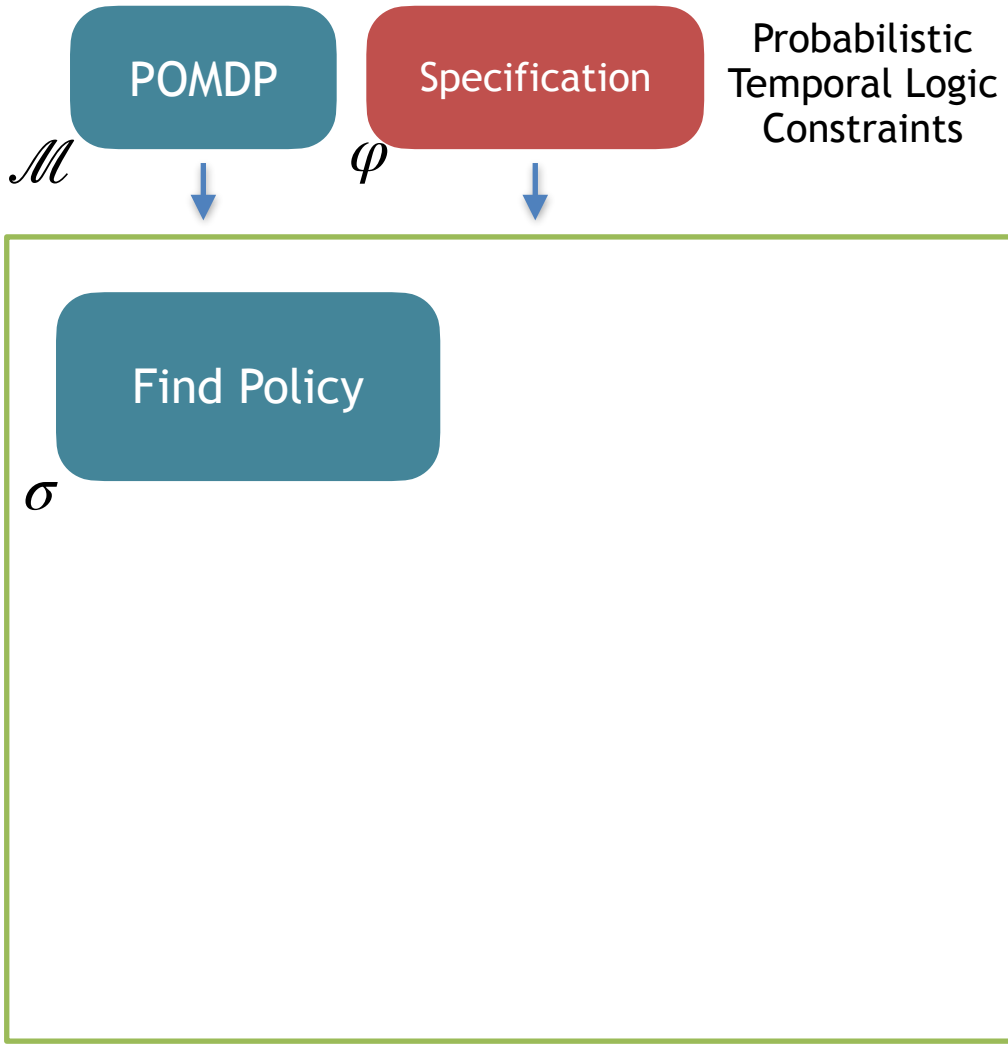


POMDP Solving - State of The Art

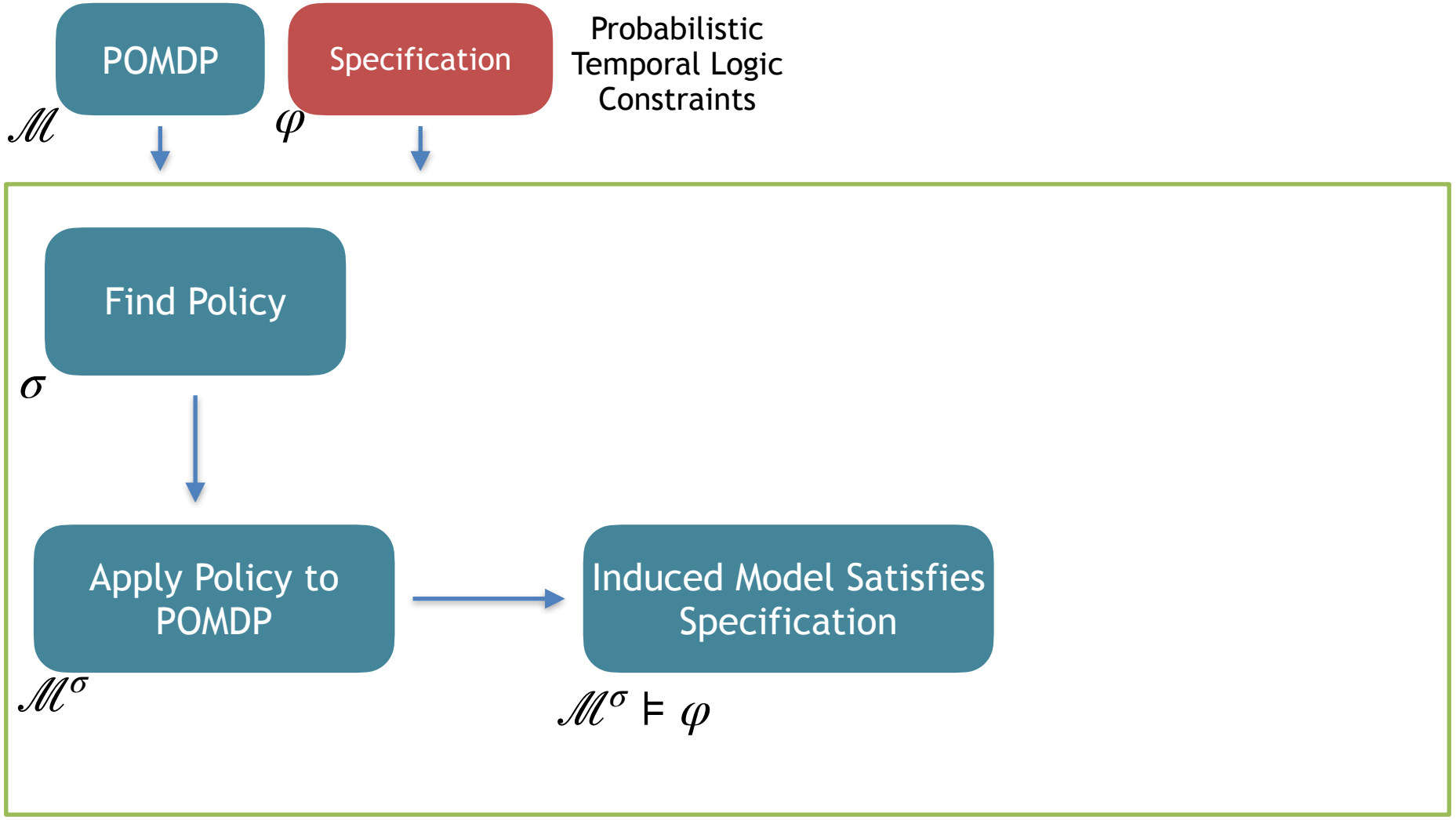


The Problem: Policy Synthesis

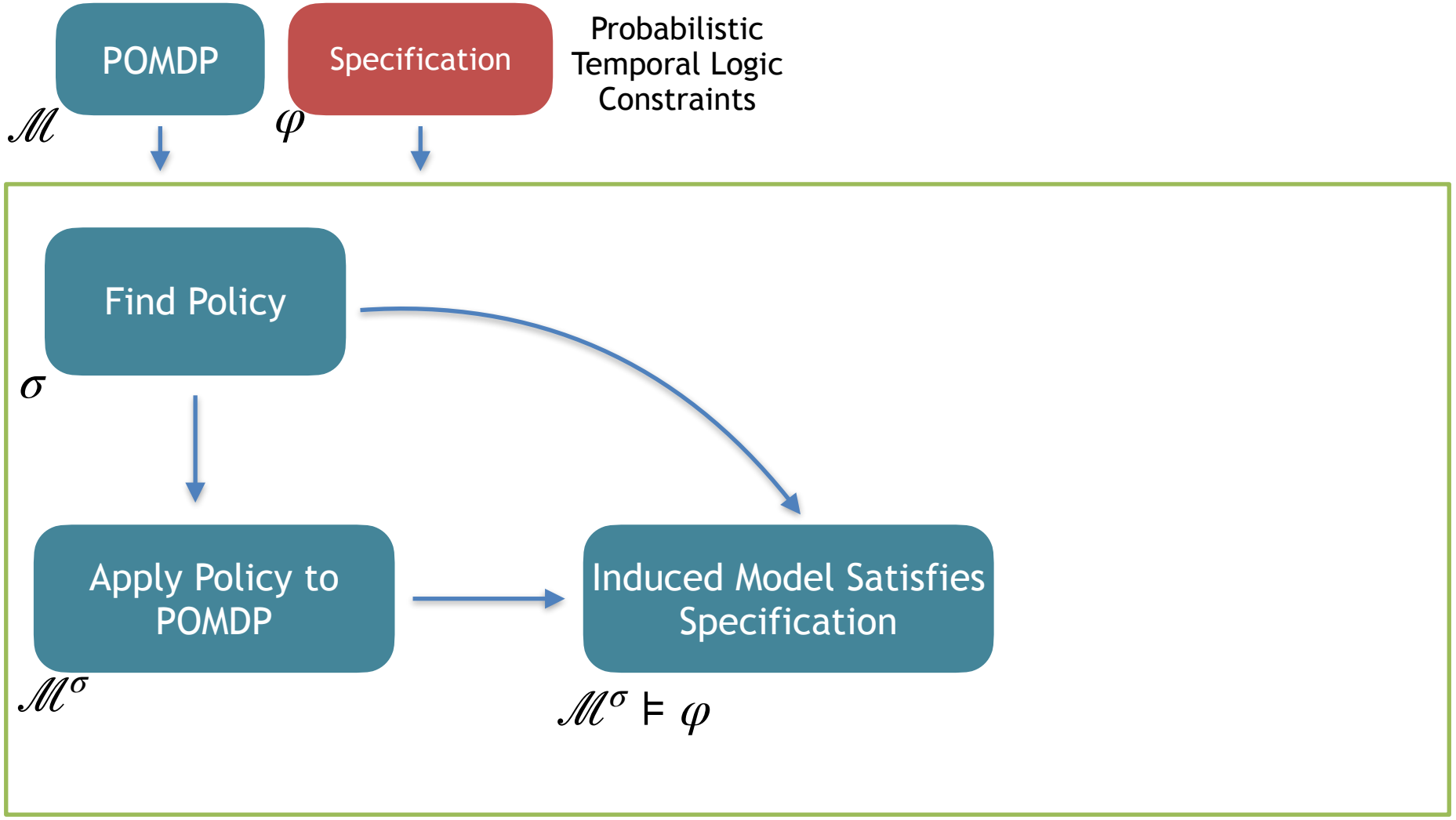
The Problem: Policy Synthesis



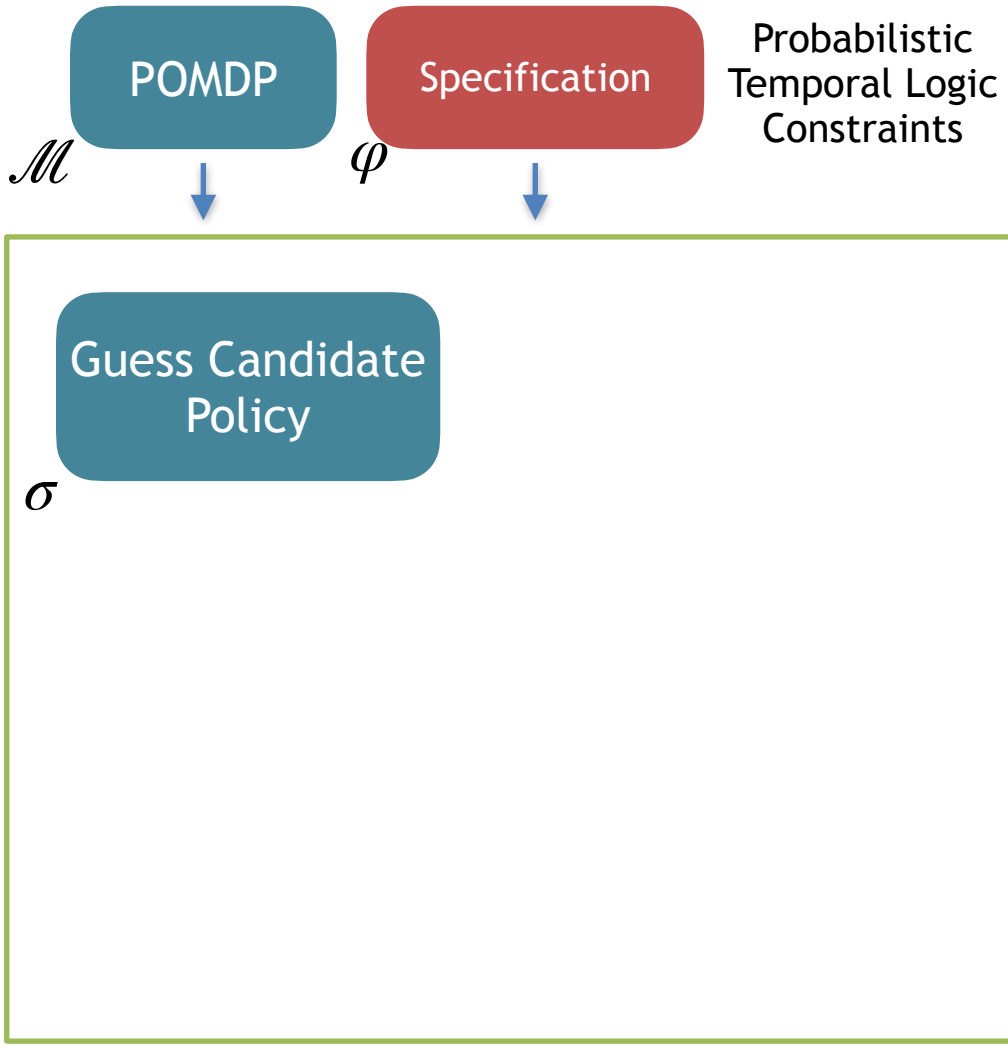
The Problem: Policy Synthesis



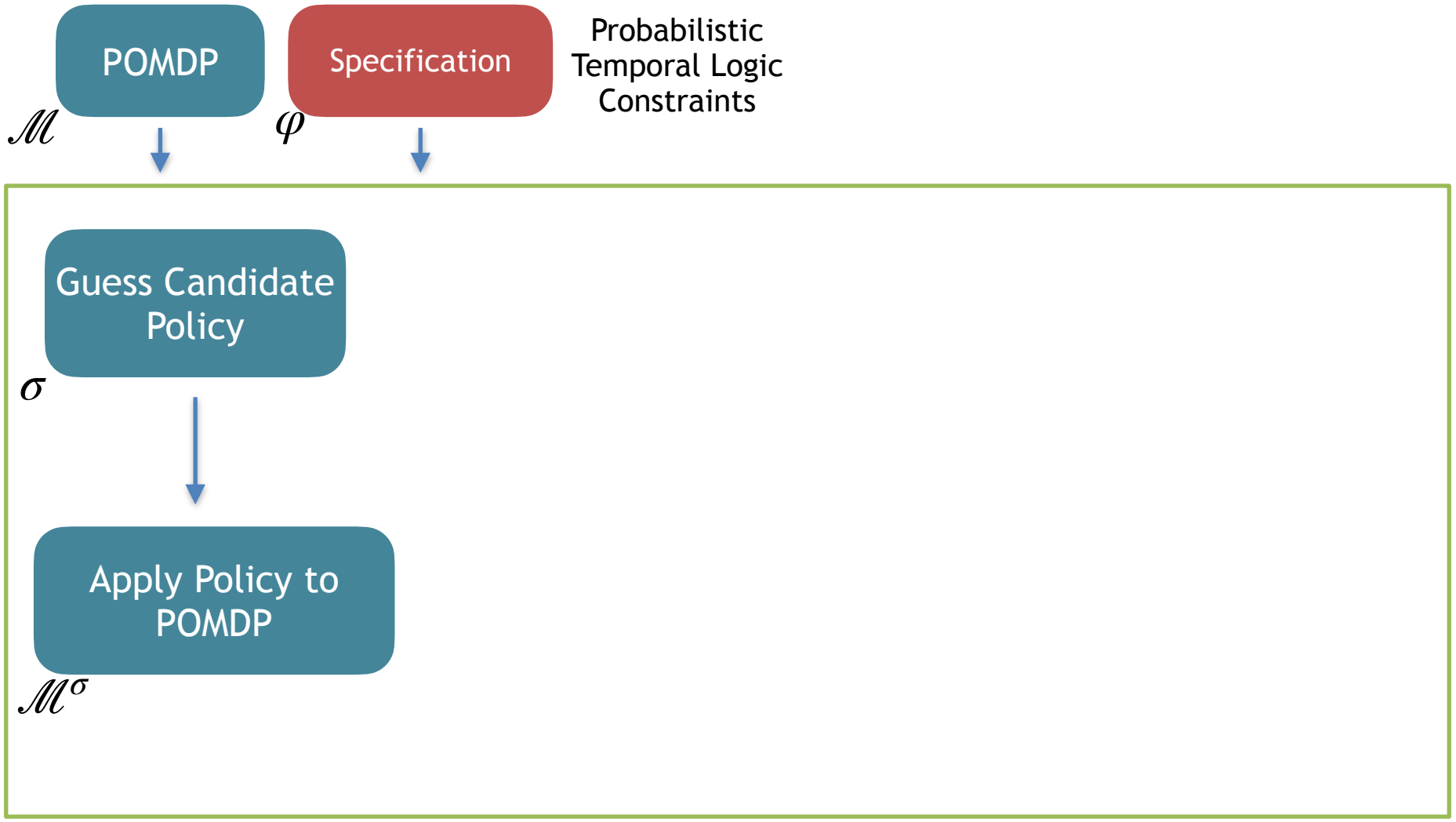
The Problem: Policy Synthesis



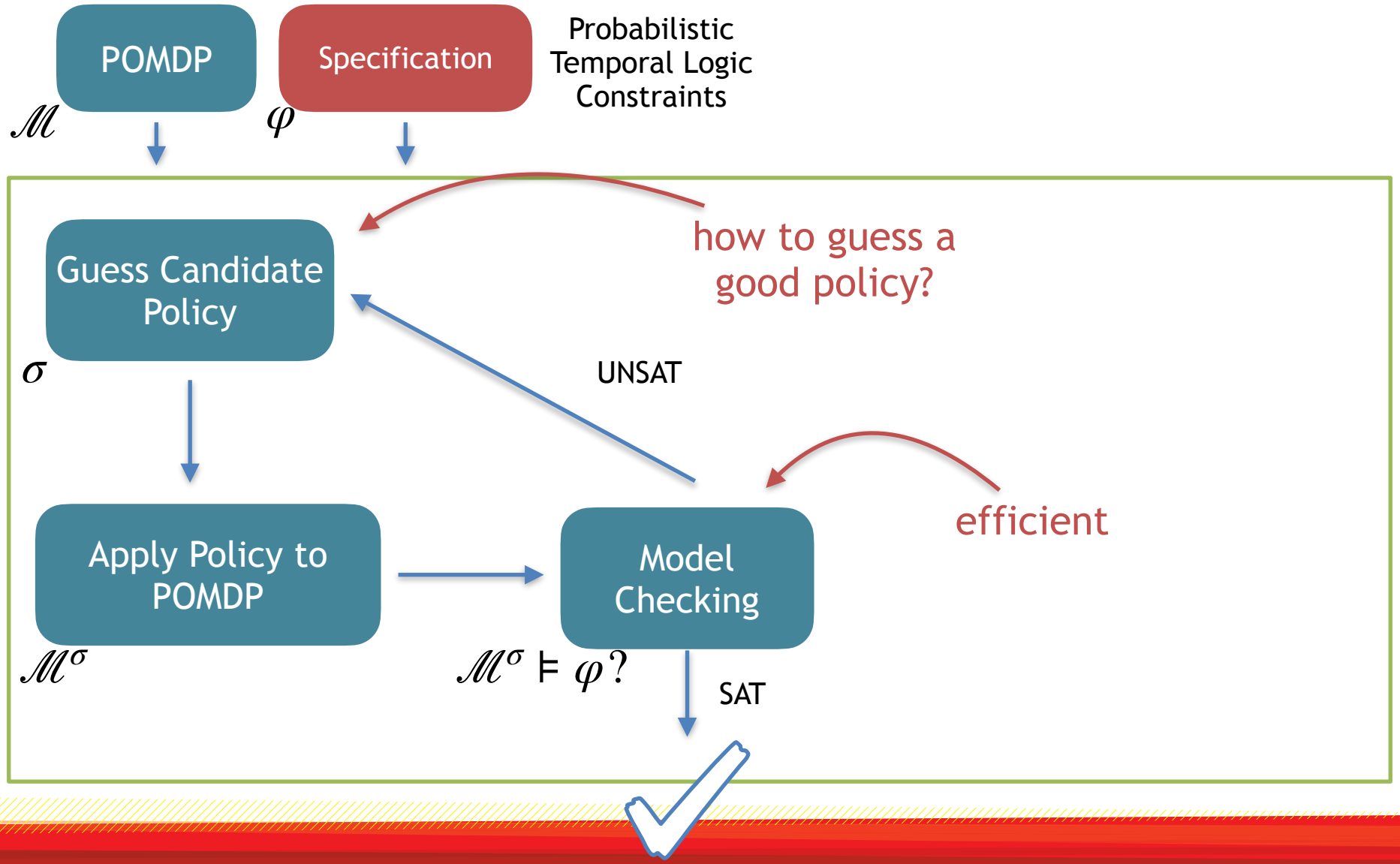
Be Lazy: Guess a Policy and Verify!



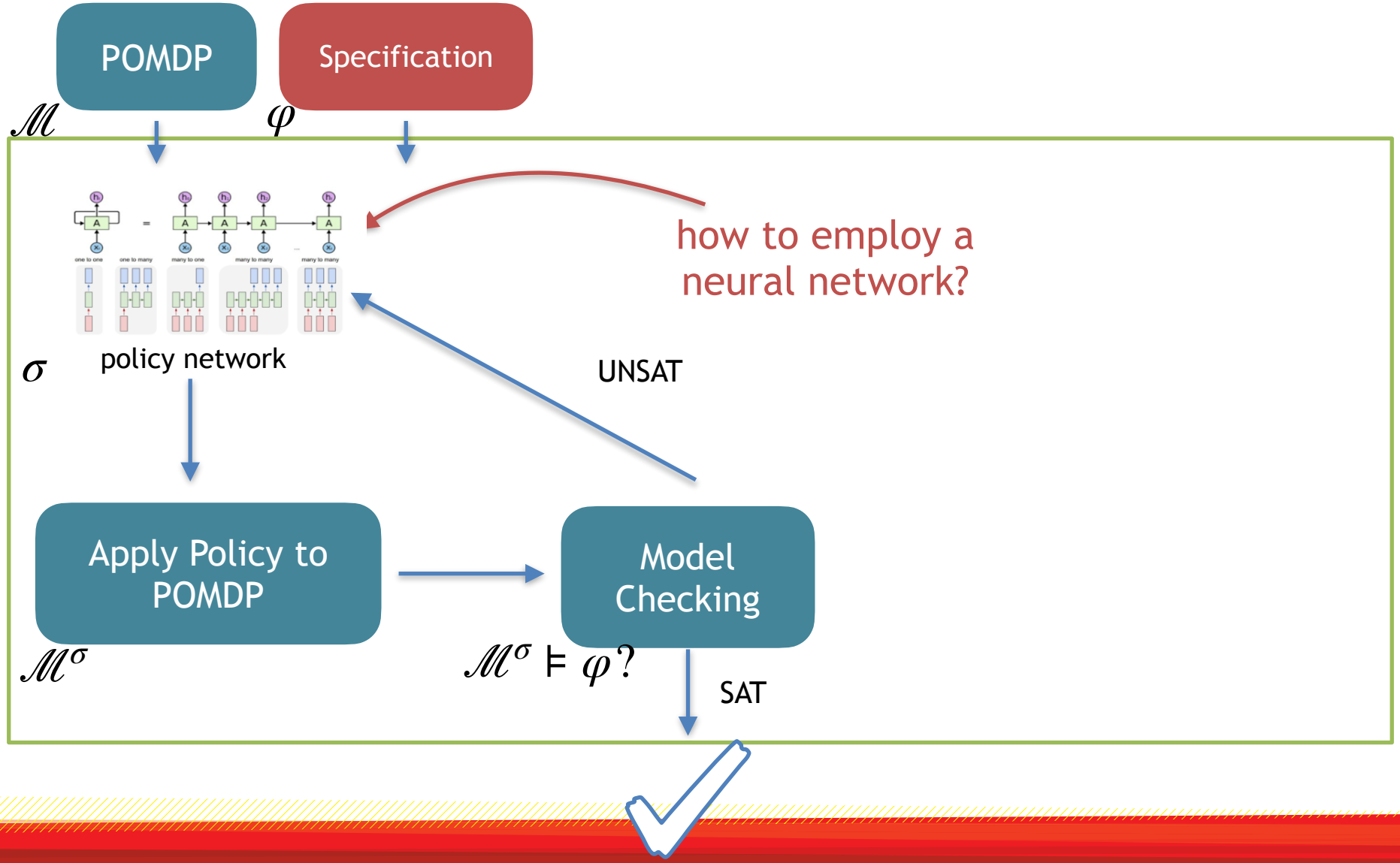
Be Lazy: Guess a Policy and Verify!



Be Lazy: Guess a Policy and Verify!



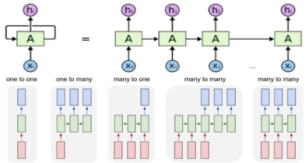
Let Machine Learning do the Guessing?



RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ

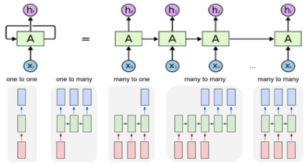


policy network

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ



policy network

σ

Extract Policy

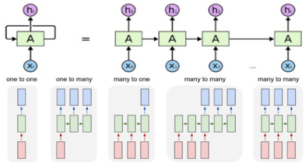
$\mathcal{M}^{\sigma}$

Apply Policy to POMDP

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ



policy network

σ

Extract Policy

$\mathcal{M}^\sigma$

Apply Policy to POMDP

$\mathcal{M}^\sigma \models \varphi ?$

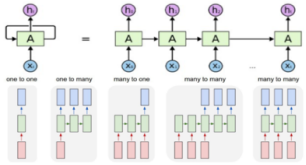
Model Checking

SAT

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ



policy network

σ

Extract Policy

$\mathcal{M}^\sigma$

Apply Policy to POMDP

$\mathcal{M}^\sigma \models \varphi ?$

Model Checking

SAT

UNSAT

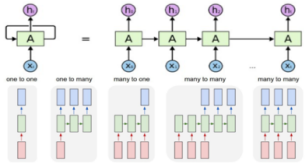
$S' \subseteq S$

Counterexamples

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ



policy network

σ

Extract Policy

$\mathcal{M}^\sigma$

Apply Policy to POMDP

$\mathcal{M}^\sigma \models \varphi ?$

Model Checking

SAT

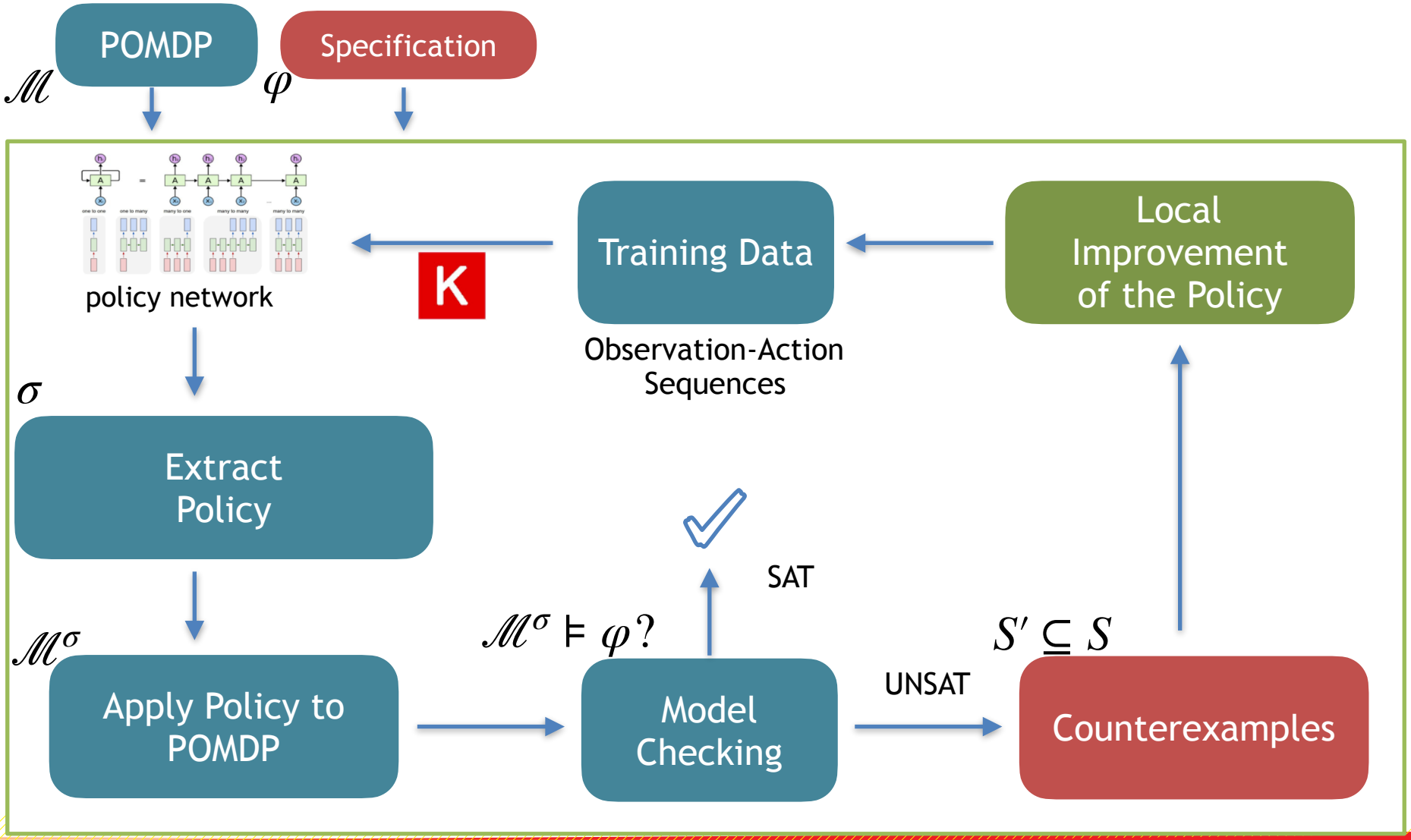
UNSAT

$S' \subseteq S$

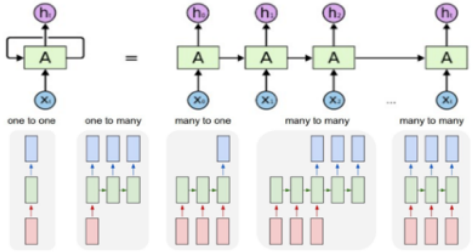
Counterexamples

Local Improvement of the Policy

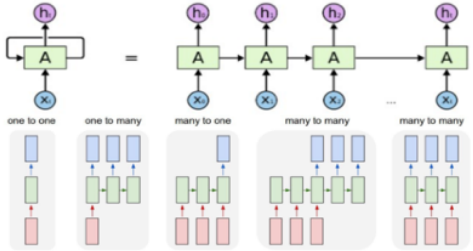
RNN Policy Improvement



Learning Strategies with RNNs



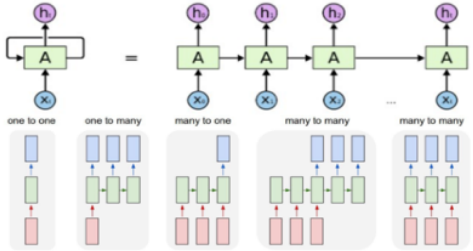
Learning Strategies with RNNs



Recurrent Neural Network

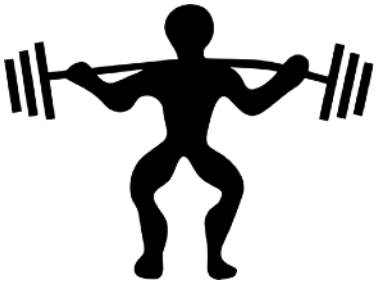
- long short-term memory (LSTM) architecture to learn dependencies in sequential data
- trained with observation-action sequences $ObsSeq_{fin}$
- policy network $\sigma: ObsSeq_{fin} \rightarrow Distr(Act)$
- observations are input labels, actions are output labels

Learning Strategies with RNNs



Recurrent Neural Network

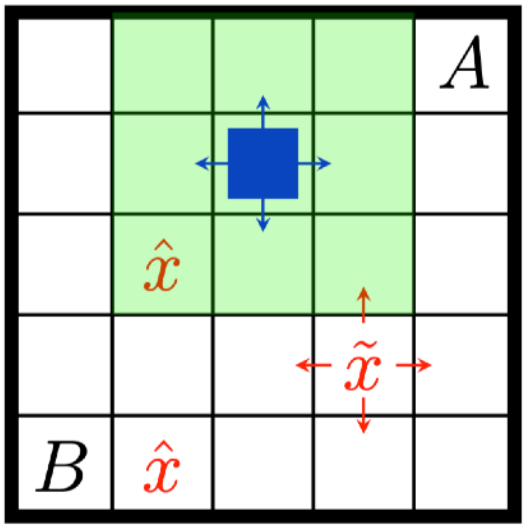
- long short-term memory (LSTM) architecture to learn dependencies in sequential data
- trained with observation-action sequences $ObsSeq_{fin}$
- policy network $\sigma : ObsSeq_{fin} \rightarrow Distr(Act)$
- observations are input labels, actions are output labels



Initial Training

- compute optimal MDP policy
- generate (possible) observation-action sequences

Experiments - LTL



Problem	$\lvert S\rvert$	$\lvert Act\rvert$	$\lvert Z\rvert$
Navigation (c)	c^4	4	256
Delivery (c)	c^2	4	256
Slippery (c)	c^2	4	256
Maze(c)	$3c+8$	4	7
Grid(c)	c^2	4	2
RockSample[4, 4]	257	9	2
RockSample[5, 5]	801	10	2
RockSample[7, 8]	12545	13	2

Problem	States	Type, φ	RNN-based Synthesis Res.	RNN-based Synthesis Time (s)	PRISM-POMDP Res.	PRISM-POMDP Time (s)
Navigation (3)	333	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.74	**14.16**	**0.84**	73.88
Navigation (4)	1088	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.82	**22.67**	**0.93**	1034.64
Navigation (4) [2-FSC]	13373	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.91	47.26	–	–
Navigation (4) [4-FSC]	26741	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.92	59.42	–	–
Navigation (4) [8-FSC]	53477	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.92**	85.26	–	–
Navigation (5)	2725	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.91	**34.34**	MO	MO
Navigation (5) [2-FSC]	33357	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.92	115.16	–	–
Navigation (5) [4-FSC]	66709	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.92	159.61	–	–
Navigation (5) [8-FSC]	133413	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.92**	250.91	–	–
Navigation (10)	49060	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.79	**822.87**	MO	MO
Navigation (10) [2-FSC]	475053	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.83	1185.41	–	–
Navigation (10) [4-FSC]	950101	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.85**	1488.77	–	–
Navigation (10) [8-FSC]	1900197	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	0.81	1805.22	–	–
Navigation (15)	251965	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.91**	**1271.80***	MO	MO
Navigation (20)	798040	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.96**	**4712.25***	MO	MO
Navigation (30)	4045840	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	**0.95**	**25191.05***	MO	MO
Navigation (40)	–	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_1$	TO	TO	MO	MO
Delivery (4) [2-FSC]	80	$\mathbb{E}^{\mathcal{M}}_{\min}, \varphi_2$	6.02	35.35	**6.0**	**28.53**
Delivery (5) [2-FSC]	125	$\mathbb{E}^{\mathcal{M}}_{\min}, \varphi_2$	8.11	**78.32**	**8.0**	102.41
Delivery (10) [2-FSC]	500	$\mathbb{E}^{\mathcal{M}}_{\min}, \varphi_2$	**18.13**	**120.34**	MO	MO
Slippery (4) [2-FSC]	460	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	0.78	67.51	**0.90**	**5.10**
Slippery (5) [2-FSC]	730	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	0.89	84.32	**0.93**	**83.24**
Slippery (10) [2-FSC]	2980	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	**0.98**	**119.14**	MO	MO
Slippery (20) [2-FSC]	11980	$\mathbb{P}^{\mathcal{M}}_{\max}, \varphi_3$	**0.99**	**1580.42**	MO	MO

Experiments - Standard POMDPs

Problem	Type	RNN-based Synthesis States	RNN-based Synthesis Res	RNN-based Synthesis Time (s)	PRISM-POMDP Res	PRISM-POMDP Time (s)	pomdpSolve Res	pomdpSolve Time (s)
Maze (1)	$\mathbb{E}^{\mathcal{M}}_{\min}$	68	4.31	31.70	**4.30**	**0.09**	4.30	0.30
Maze (2)	$\mathbb{E}^{\mathcal{M}}_{\min}$	83	5.31	46.65	5.23	2.176	**5.23**	**0.67**
Maze (3)	$\mathbb{E}^{\mathcal{M}}_{\min}$	98	8.10	58.75	7.13	38.82	**7.13**	**2.39**
Maze (4)	$\mathbb{E}^{\mathcal{M}}_{\min}$	113	11.53	58.09	8.58	543.06	**8.58**	**7.15**
Maze (5)	$\mathbb{E}^{\mathcal{M}}_{\min}$	128	14.40	**68.09**	13.00	4110.50	**12.04**	132.12
Maze (6)	$\mathbb{E}^{\mathcal{M}}_{\min}$	143	22.34	**71.89**	MO	MO	**18.52**	1546.02
Maze (10)	$\mathbb{E}^{\mathcal{M}}_{\min}$	203	100.21	**158.33**	MO	MO	MO	MO
Grid (3)	$\mathbb{E}^{\mathcal{M}}_{\min}$	165	2.90	38.94	2.88	2.332	**2.88**	**0.07**
Grid (4)	$\mathbb{E}^{\mathcal{M}}_{\min}$	381	4.32	79.99	4.13	1032.53	**4.13**	**0.77**
Grid (5)	$\mathbb{E}^{\mathcal{M}}_{\min}$	727	6.623	91.42	MO	MO	**5.42**	**1.94**
Grid (10)	$\mathbb{E}^{\mathcal{M}}_{\min}$	5457	**13.630**	**268.40**	MO	MO	MO	MO
RockSample[4, 4]	$\mathbb{E}^{\mathcal{M}}_{\max}$	2432	17.71	35.35	N/A	N/A	**18.04**	**0.43**
RockSample[5, 5]	$\mathbb{E}^{\mathcal{M}}_{\max}$	8320	18.40	**43.74**	N/A	N/A	**19.23**	621.28
RockSample[7, 8]	$\mathbb{E}^{\mathcal{M}}_{\max}$	166656	20.32	**860.53**	N/A	N/A	**21.64**	20458.41

Improving the Policy

Improving the Policy

- Identify **critical decisions** that lead to states with high probability of **violating** the specification.

Improving the Policy

- Identify **critical decisions** that lead to states with high probability of **violating** the specification.
- For each observation with critical decision, minimize the number of different critical actions.

Local linear program

$$\max_{\gamma(z)(a), a \in Act} \min_{s \in S} p_s \qquad (1)$$

subject to

$$\forall s \in O^{-1}(z). \quad p_s = \sum_{a \in Act} \gamma(z)(a) \cdot \sum_{s' \in S} \mathscr{P}(s,a,s') \cdot p^*(s')$$

Improving the Policy

- Identify **critical decisions** that lead to states with high probability of **violating** the specification.
- For each observation with critical decision, minimize the number of different critical actions.
- **Retrain** with the new (locally improved) policy.

Local linear program

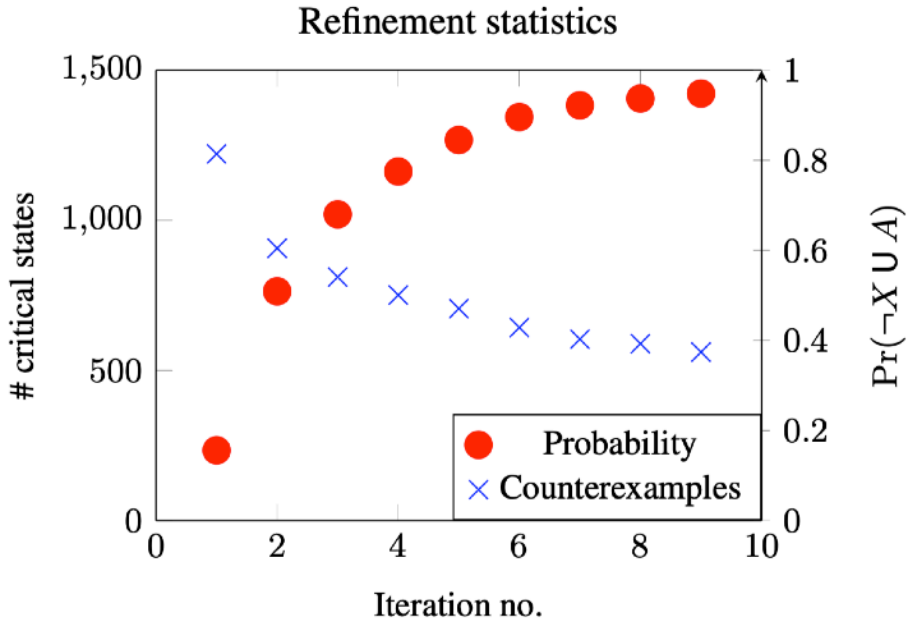
$$\max_{\gamma(z)(a), a \in Act} \min_{s \in S} p_s \qquad (1)$$

subject to

$$\forall s \in O^{-1}(z). \quad p_s = \sum_{a \in Act} \gamma(z)(a) \cdot \sum_{s' \in S} \mathscr{P}(s,a,s') \cdot p^*(s')$$

Improving the Policy

- Identify **critical decisions** that lead to states with high probability of **violating** the specification.
- For each observation with critical decision, **minimize** the number of different critical actions.
- **Retrain** with the new (locally improved) policy.



Local linear program

$$\max_{\gamma(z)(a),a\in Act} \min_{s\in S} p_s \qquad (1)$$

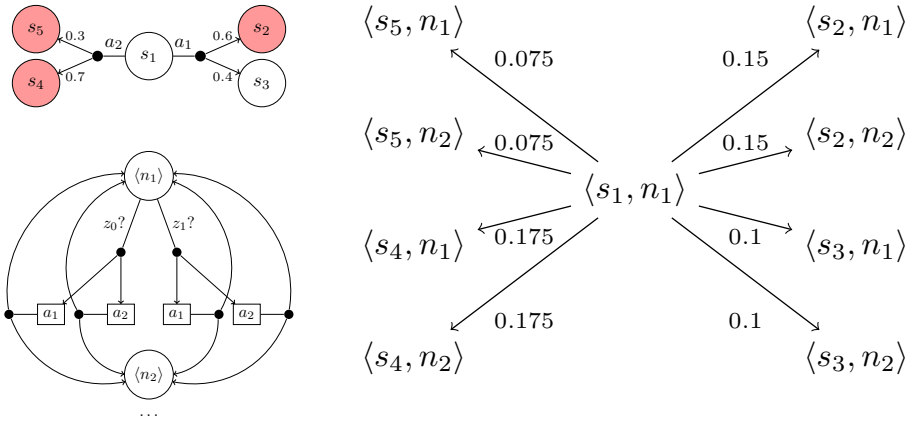
subject to

$$\forall s \in O^{-1}(z). \quad p_s = \sum_{a\in Act} \gamma(z)(a) \cdot \sum_{s'\in S} \mathscr{P}(s,a,s') \cdot p^*(s')$$

Even if specification is satisfied, there may be critical states and decisions!

Finite-memory Strategies (FSC)

- Encode finite memory directly into the state space:

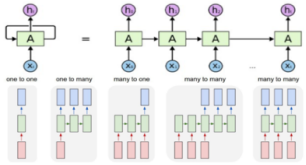


- So far: Predefined memory update, internal memory state of the RNN largely ignored
- How to infer a memory-update function to construct an FSC from the RNN?

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ

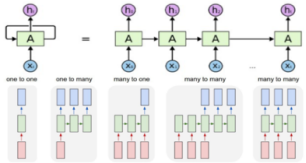


trained policy network

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ



trained policy network

$\mathcal{A}^{\sigma}$

Extract FSC

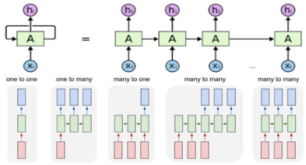
$\mathcal{M}^{\mathcal{A}^{\sigma}}$

Apply FSC to POMDP

RNN Policy Improvement

POMDP $\mathcal{M}$

Specification φ



trained policy network

$\mathcal{A}^\sigma$

Extract FSC

$\mathcal{M}^{\mathcal{A}^\sigma}$

Apply FSC to POMDP

$\mathcal{M}^{\mathcal{A}^\sigma} \models \varphi ?$

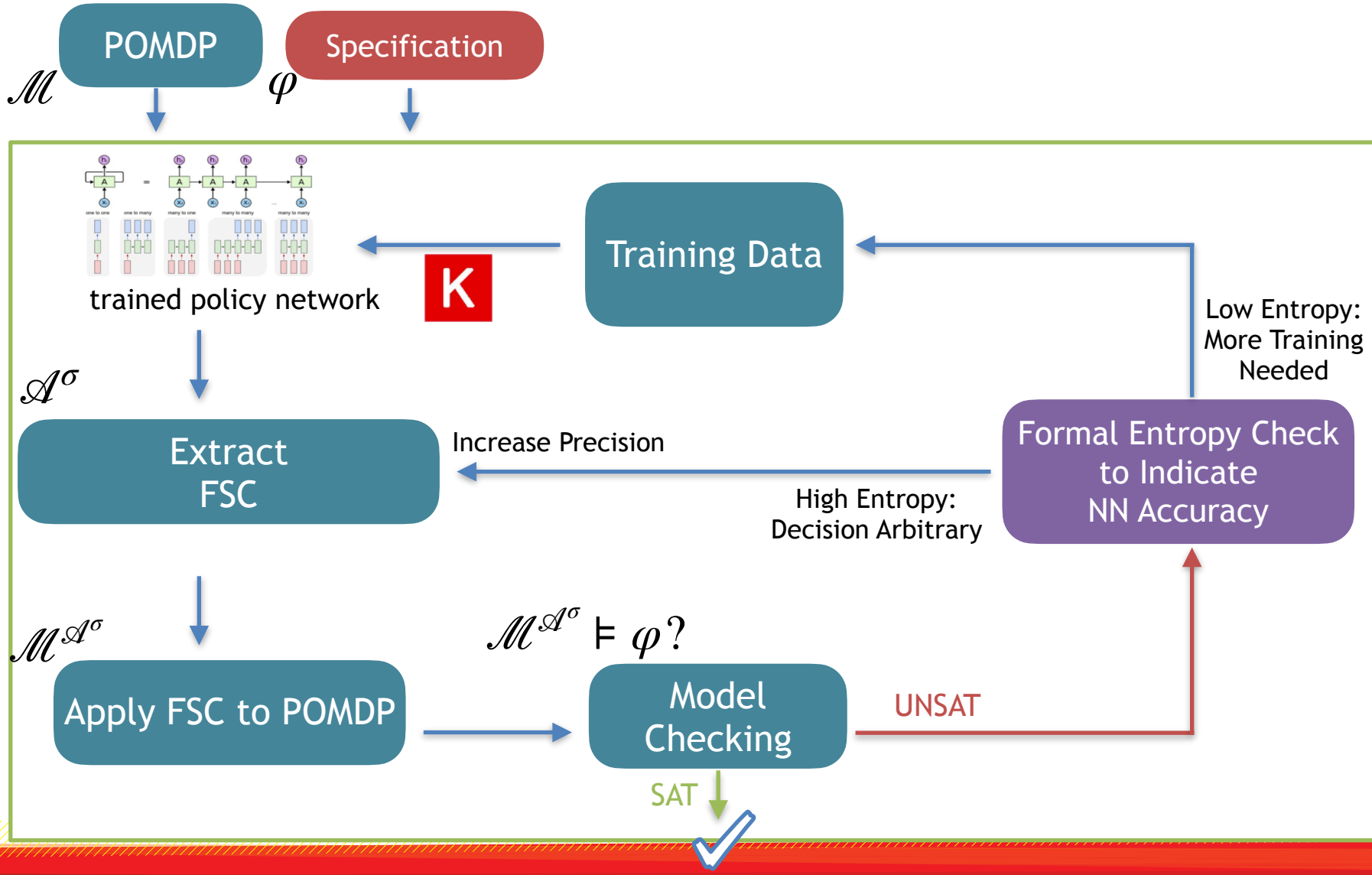
Model Checking

UNSAT

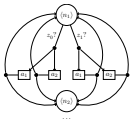
Formal Entropy Check to Indicate NN Accuracy

SAT

RNN Policy Improvement



Extracting and Improving FSCs

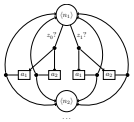


Quantized Bottleneck Insertion [1]

- Discretization of the continuous memory structure of the RNN
- Mapping to a pre-defined number of finite memory state

[1] Anurag Koul, Alan Fern, and Sam Greydanus. Learning finite state representations of recurrent policy networks. In ICLR, 2019

Extracting and Improving FSCs



Quantized Bottleneck Insertion [1]

- Discretization of the continuous memory structure of the RNN
- Mapping to a pre-defined number of finite memory state

Entropy Check

$$H(Crit_{\mathcal{A}}^{\mathcal{M}}) = \frac{1}{|Crit_{\mathcal{A}}^{\mathcal{M}}|} \sum_{(n,s) \in Crit_{\mathcal{A}}^{\mathcal{M}}} H(n,s)$$

- Localized to critical states
- Entropy high: NN is likely extrapolating
- Entropy low: NN (may be) well trained, increase precision of FSC extraction

[1] Anurag Koul, Alan Fern, and Sam Greydanus. Learning finite state representations of recurrent policy networks. In ICLR, 2019

Experiments

Problem	$\|S\|$	$\|Z\|$	Type	Extraction Approach Memory	Extraction Approach Value	Extraction Approach Time (s)	Handcrafted Memory	Handcrafted Value	Handcrafted Time (s)	PRISM-POMDP Value	PRISM-POMDP Time (s)	SolvePOMDP Value	SolvePOMDP Time (s)
Maze(1)	11	7	Min	2	4.33	80.31	2	4.31	30.70	**4.30**	**0.09**	4.30	0.30
Maze(2)	14	7	Min	3	5.34	114.23	3	5.31	46.65	5.23	2.176	**5.23**	**0.67**
Maze(5)	23	7	Min	3	13.29	160.12	6	14.40	68.09	13.00*	4110.50	**12.04**	**134.46**
Maze(10)	38	7	Min	5	**23.02**	210.01	11	100.21	**158.33**	MO	MO	MO	MO
Grid(3)	9	2	Min	3	2.90	87.31	2	2.90	38.94	2.88	2.332	**2.88**	**0.06**
Grid(4)	16	2	Min	7	4.20	124.31	3	4.32	79.99	4.13	1032.53	**4.13**	**0.73**
Grid(5)	25	2	Min	9	5.91	250.14	4	6.623	91.42	MO	MO	**5.42**	**1.97**
Grid(10)	100	2	Min	9	**12.92**	1031.21	9	13.63	**268.40**	MO	MO	MO	MO
Grid(25)	625	2	Min	16	**35.32**	6514.30	24	531.05	**622.31**	MO	MO	MO	MO
Navigation (4)	256	256	Max	8	0.92	160.32	8	0.92	80.26	**0.93***	**1034.64**	NA	NA
Navigation (5)	625	256	Max	8	**0.95**	311.65	8	0.92	**253.11**	MO	MO	NA	NA
Navigation (10)	10^4	256	Max	8	**0.90**	2561.02	4	0.85	**1471.17**	MO	MO	NA	NA
Navigation (20)	1.6×10^5	256	Max	9	**0.98**	8173.03	4	0.96	**7068.24**	MO	MO	NA	NA

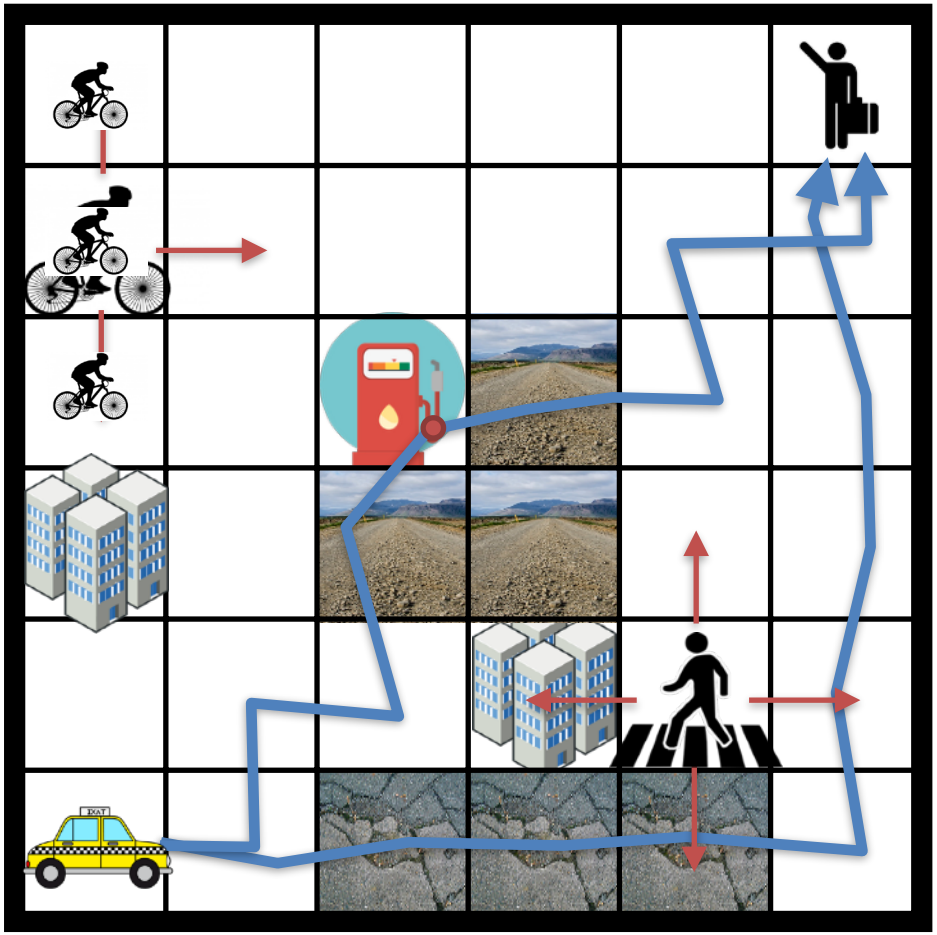
- scales to larger examples than other solvers (but not optimal!)
- higher-quality than handcrafted memory structure

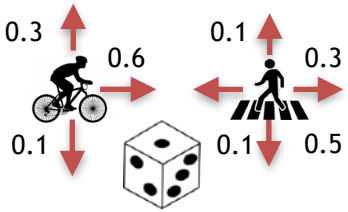
Experiments

Problem	$\|S\|$	$\|Z\|$	Type	Extraction Approach Memory	Extraction Approach Value	Extraction Approach Time (s)	Handcrafted Memory	Handcrafted Value	Handcrafted Time (s)	PRISM-POMDP Value	PRISM-POMDP Time (s)	SolvePOMDP Value	SolvePOMDP Time (s)
Maze(1)	11	7	Min	2	4.33	80.31	2	4.31	30.70	**4.30**	**0.09**	4.30	0.30
Maze(2)	14	7	Min	3	5.34	114.23	3	5.31	46.65	5.23	2.176	**5.23**	**0.67**
Maze(5)	23	7	Min	3	13.29	160.12	6	14.40	68.09	13.00*	4110.50	**12.04**	**134.46**
Maze(10)	38	7	Min	5	**23.02**	210.01	11	100.21	**158.33**	MO	MO	MO	MO
Grid(3)	9	2	Min	3	2.90	87.31	2	2.90	38.94	2.88	2.332	**2.88**	**0.06**
Grid(4)	16	2	Min	7	4.20	124.31	3	4.32	79.99	4.13	1032.53	**4.13**	**0.73**
Grid(5)	25	2	Min	9	5.91	250.14	4	6.623	91.42	MO	MO	**5.42**	**1.97**
Grid(10)	100	2	Min	9	**12.92**	1031.21	9	13.63	**268.40**	MO	MO	MO	MO
Grid(25)	625	2	Min	16	**35.32**	6514.30	24	531.05	**622.31**	MO	MO	MO	MO
Navigation (4)	256	256	Max	8	0.92	160.32	8	0.92	80.26	**0.93***	**1034.64**	NA	NA
Navigation (5)	625	256	Max	8	**0.95**	311.65	8	0.92	**253.11**	MO	MO	NA	NA
Navigation (10)	10^4	256	Max	8	**0.90**	2561.02	4	0.85	**1471.17**	MO	MO	NA	NA
Navigation (20)	1.6×10^5	256	Max	9	**0.98**	8173.03	4	0.96	**7068.24**	MO	MO	NA	NA

- scales to larger examples than other solvers (but not optimal!)
- higher-quality than handcrafted memory structure

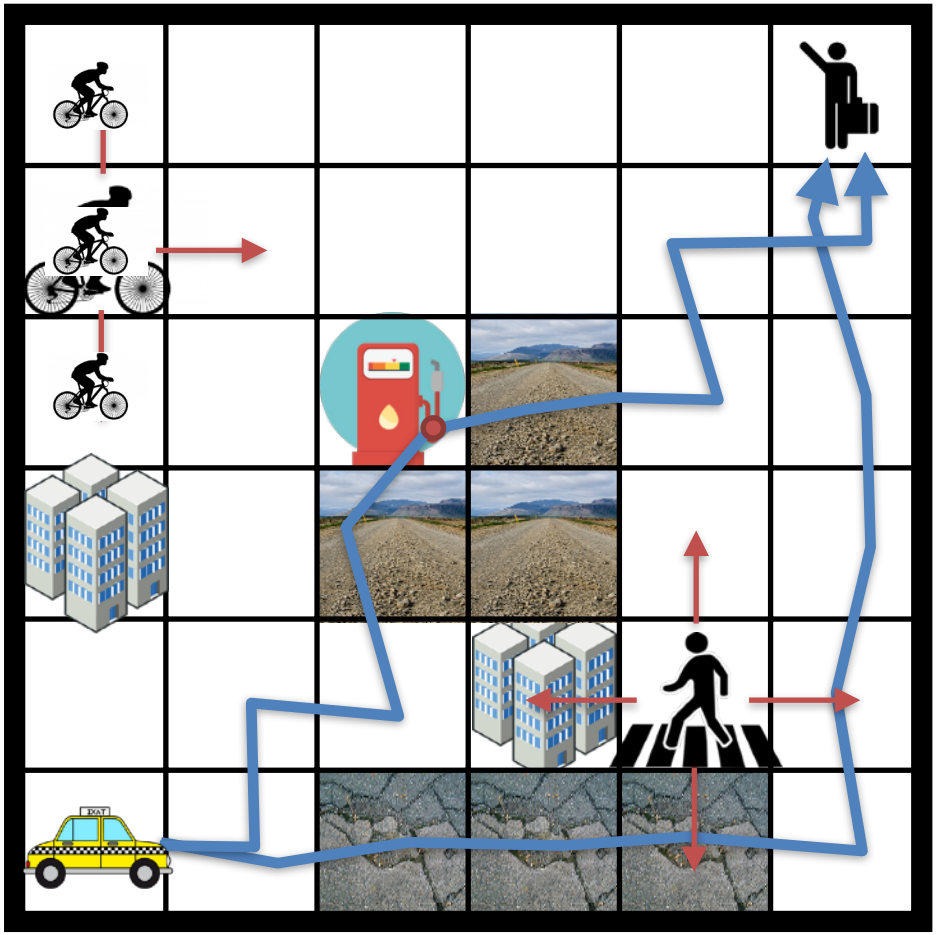
Recall: Help the Taxi: A Closer Look

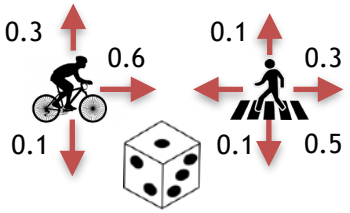




probability distributions on movements

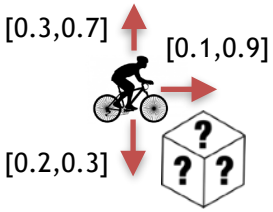
Recall: Help the Taxi: A Closer Look





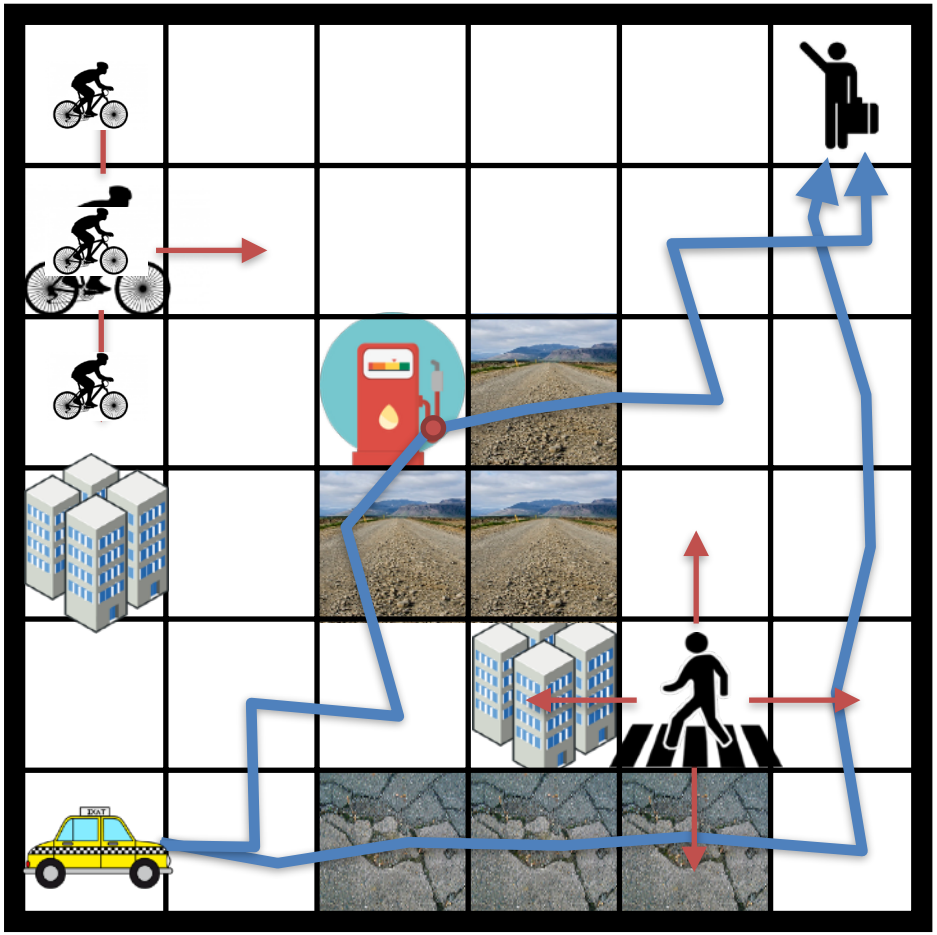
probability distributions on movements

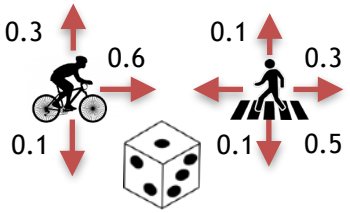
What if the distributions are not exactly known?



uncertain probability distributions

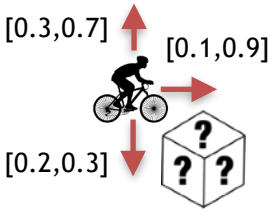
Recall: Help the Taxi: A Closer Look





probability distributions on movements

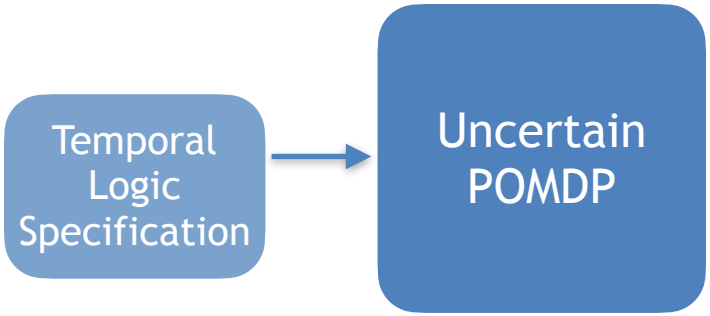
What if the distributions are not exactly known?



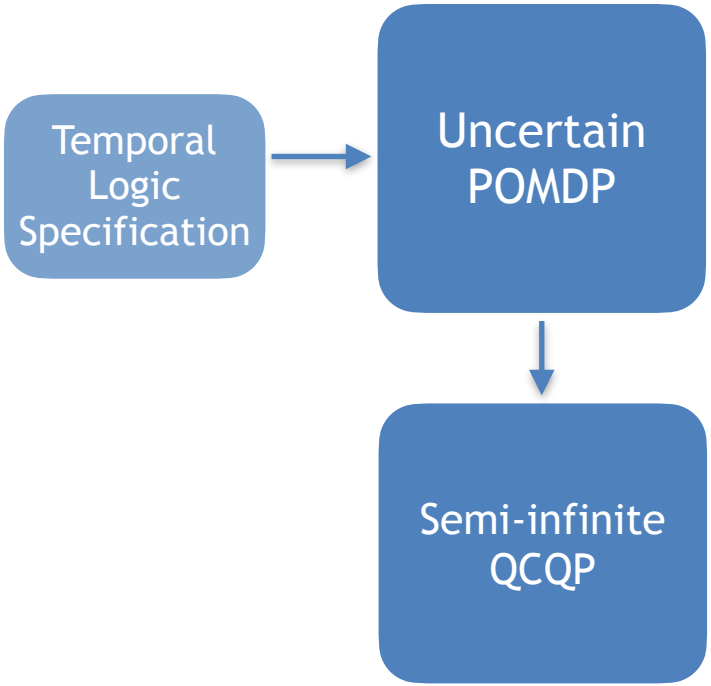
uncertain probability distributions

Underlying formal problem: Find a robust policy that accounts for uncertainty for a partially observable Markov Decision Process

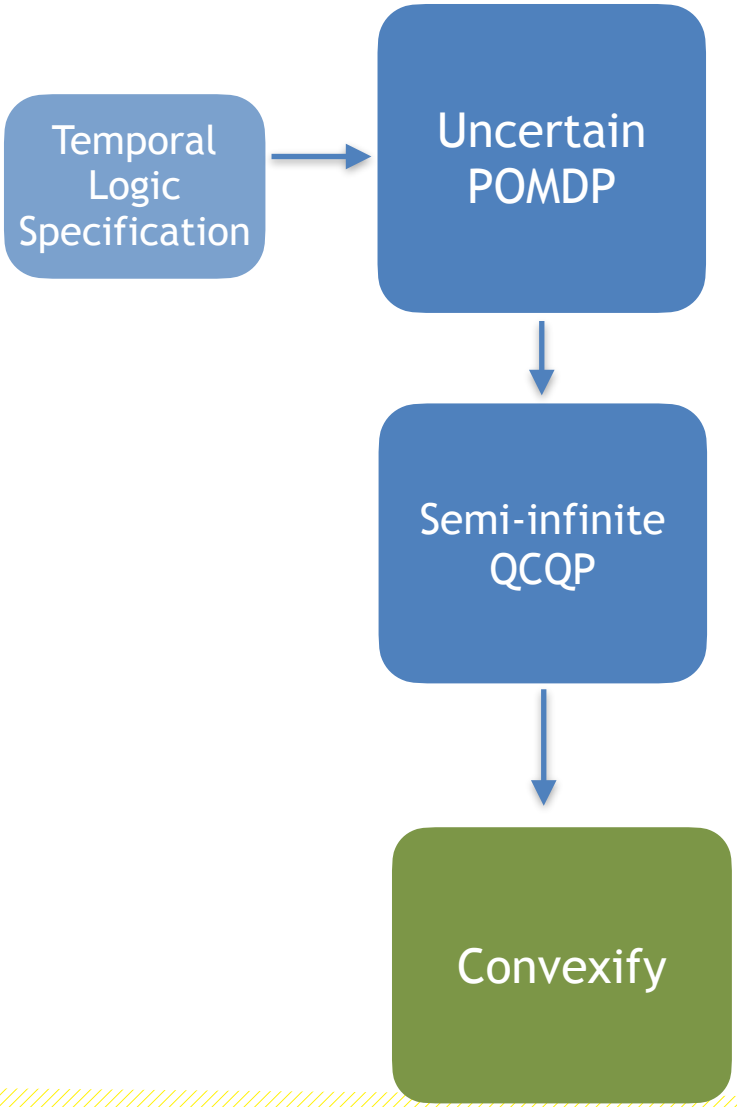
Story - Convex-concave Procedure



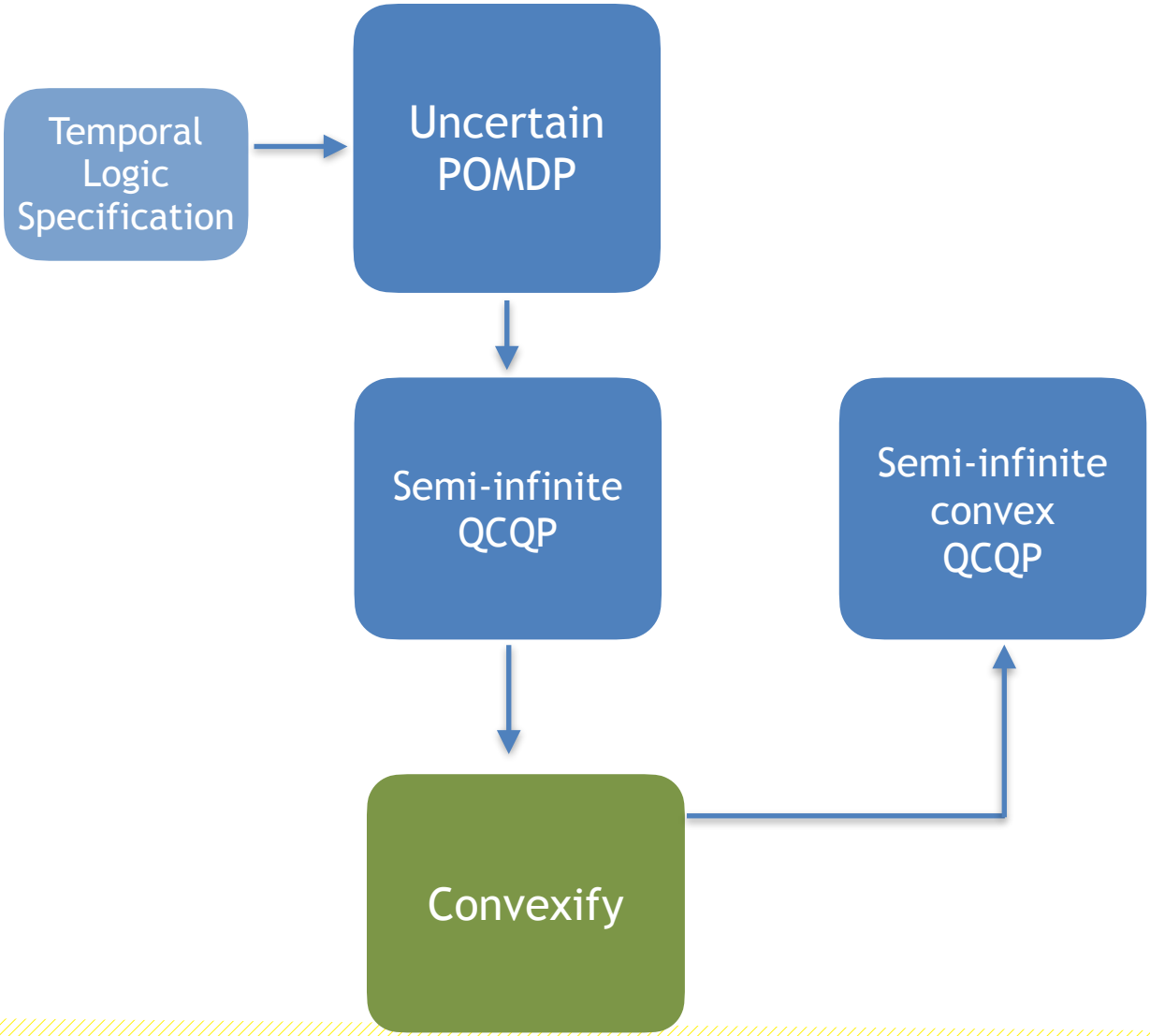
Story - Convex-concave Procedure



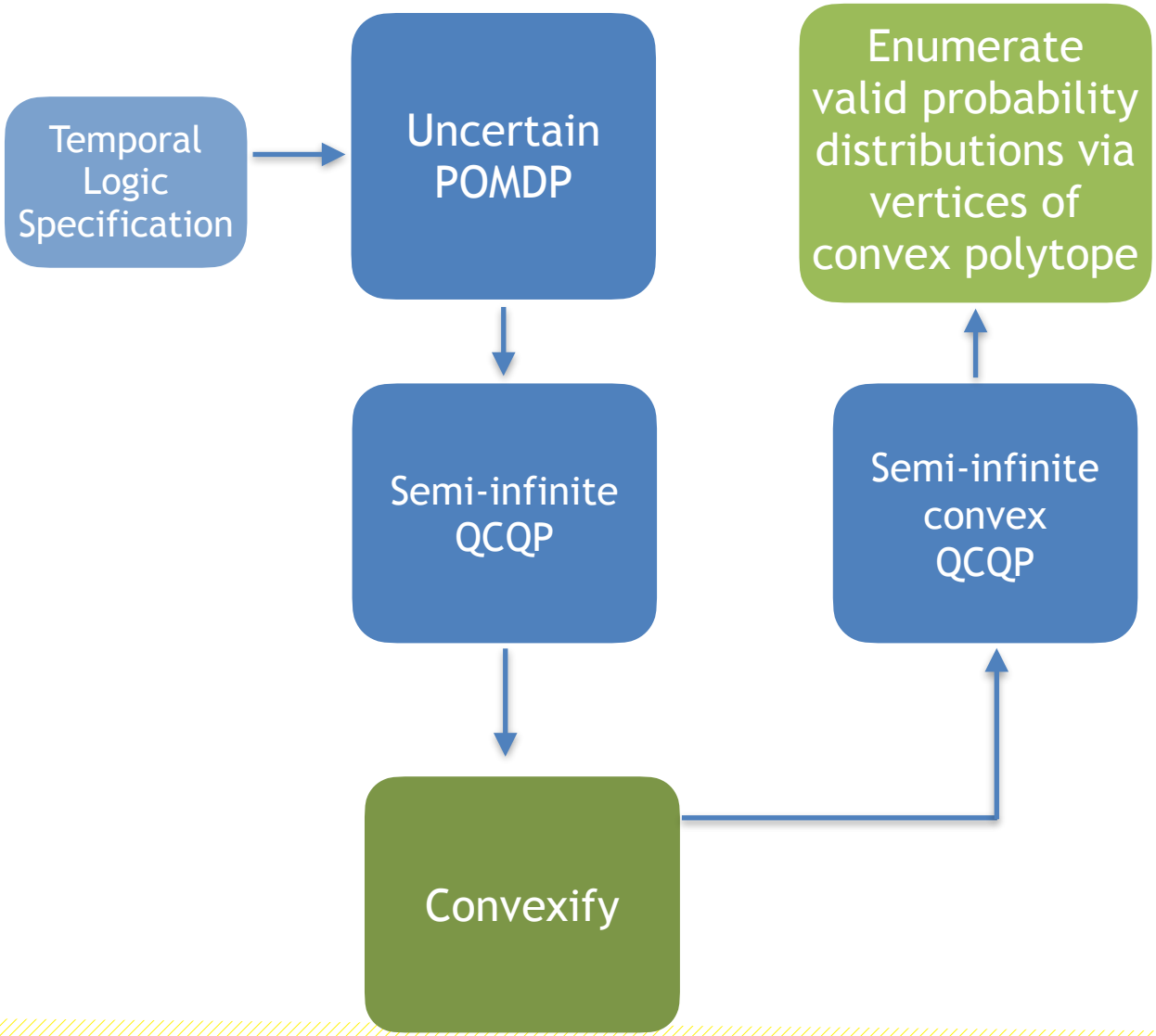
Story - Convex-concave Procedure



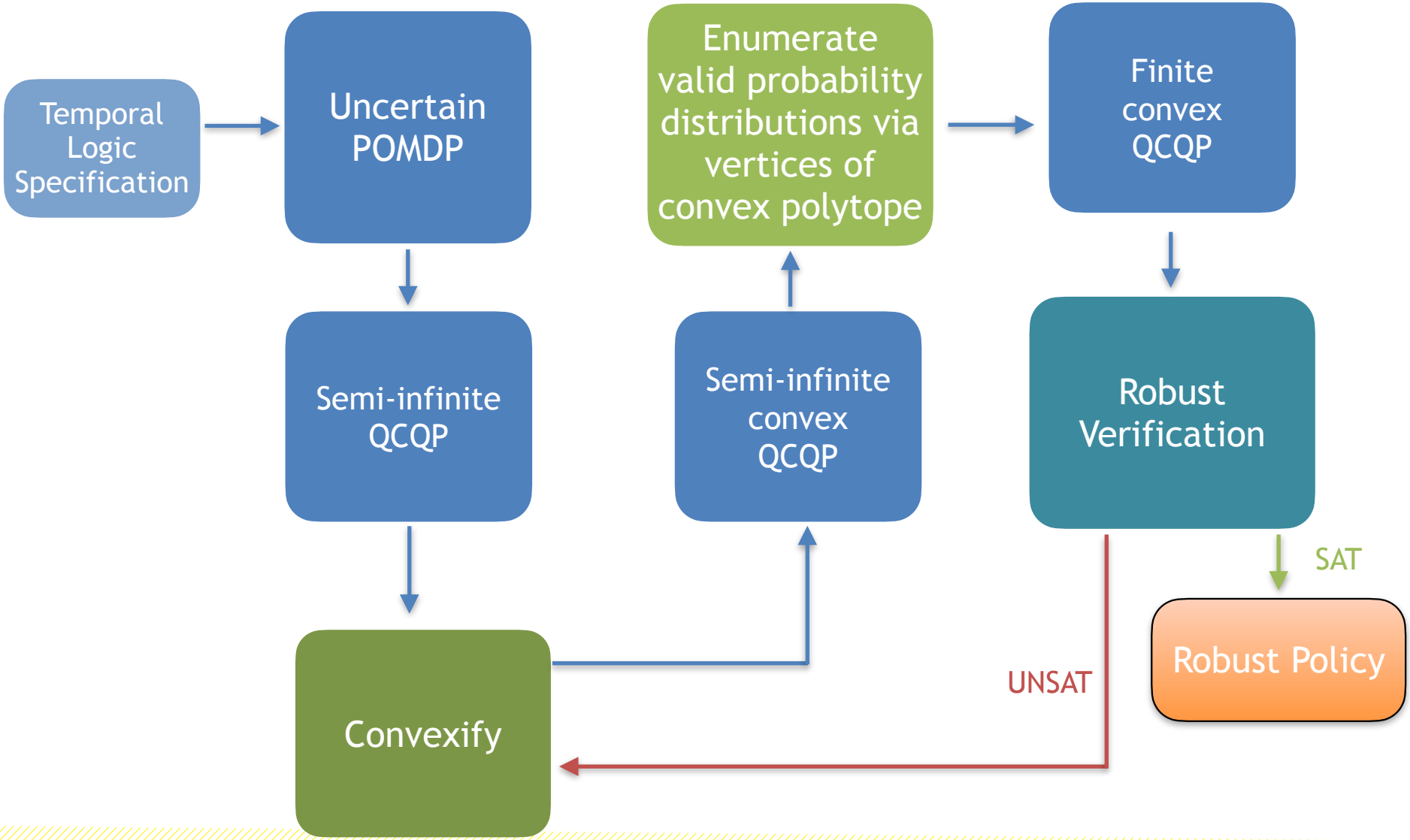
Story - Convex-concave Procedure



Story - Convex-concave Procedure



Story - Convex-concave Procedure



Interested?

Steven Carr, Nils Jansen, Ralf Wimmer, Alexandru C. Serban, Bernd Becker, Ufuk Topcu. **Counterexample-Guided Strategy Improvement for POMDPs Using Recurrent Neural Networks.** In IJCAI 2019, pages 5532-5539.

Steven Carr, Nils Jansen, Ufuk Topcu. **Verifiable RNN-Based Policies for POMDPs Under Temporal Logic Constraints.** In IJCAI 2020, to appear.

Marnix Suilen, Nils Jansen, Murat Cubuktepe, Ufuk Topcu. **Robust Policy Synthesis for Uncertain POMDPs via Convex Optimization.** In IJCAI 2020, to appear.

Leonore Winterer, Sebastian Junges, Ralf Wimmer, Nils Jansen, Ufuk Topcu, Joost-Pieter Katoen, Bernd Becker. **Strategy Synthesis for POMDPs in Robot Planning via Game-Based Abstractions.** In Transactions of Automatic Control, 2020, to appear.

Leonore Winterer, Ralf Wimmer, Nils Jansen, Bernd Becker. **Strengthening Deterministic Policies for POMDPs.** In NFM, 2020, to appear.

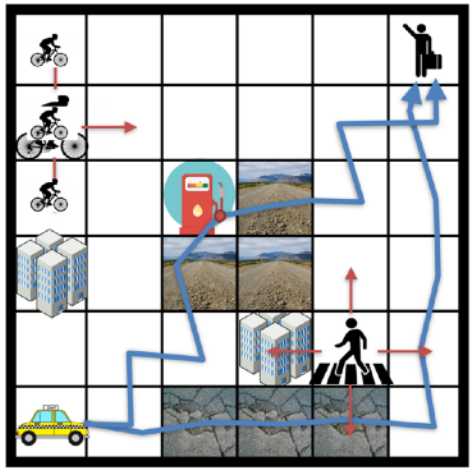
Murat Cubuktepe, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen, Ufuk Topcu. **Scenario-Based Verification of Uncertain MDPs.** In TACAS, 2020, to appear.

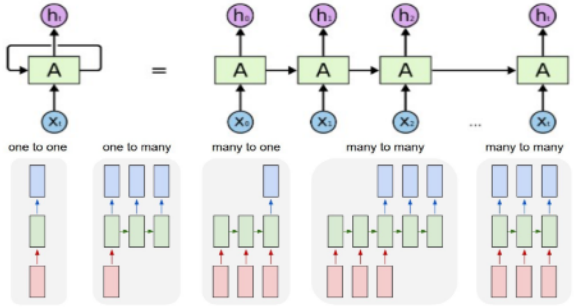
Sebastian Junges, Nils Jansen, Ralf Wimmer, Tim Quatmann, Leonore Winterer, Joost-Pieter Katoen, Bernd Becker. **Finite-state Controllers of POMDPs via Parameter Synthesis.** In Conference on Uncertainty in Artificial Intelligence (UAI), pages 519-529, 2018.

http://nilsjansen.org

Conclusion

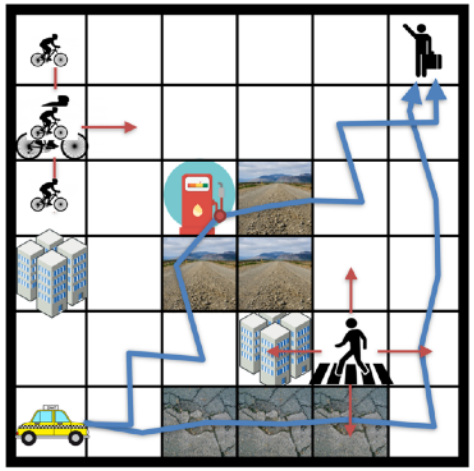
- Novel way to generate provably correct POMDP strategies
- Good scalability, not optimal
- Results transferrable
- First result on robust policies for uncertain POMDPs for temporal logic constraints



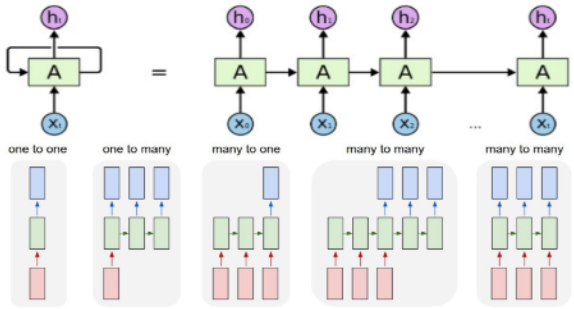


Conclusion

- Novel way to generate provably correct POMDP strategies
- Good scalability, not optimal
- Results transferrable
- First result on robust policies for uncertain POMDPs for temporal logic constraints



$Pr_{max}(\neg(\text{ped} \vee \text{cyc})\ \mathsf{U}\ (\Diamond(\text{fuel} \wedge \bigcirc\Diamond\text{pass}))$



$EC_{min}(\Diamond\text{passenger})$