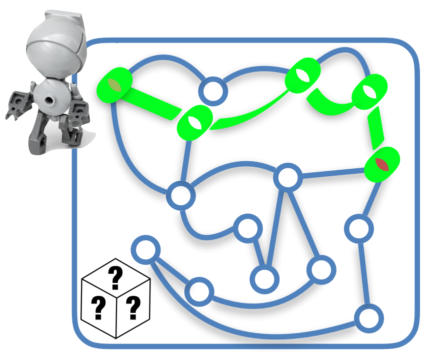
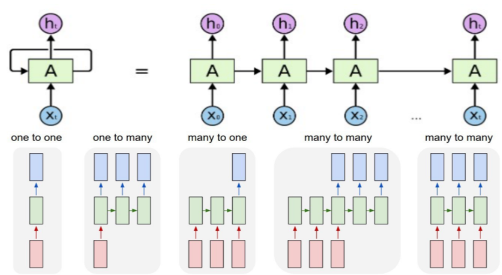
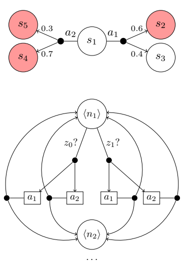
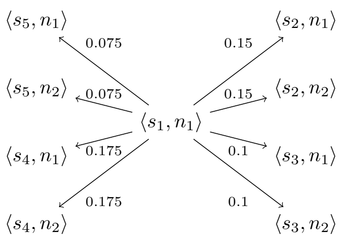
Counterexample-Guided Strategy Improvement for POMDPs Using Recurrent Neural Networks

Machine Learning and Formal Verification Join Forces?









Nils Jansen

IJCAI, August 15, 2019

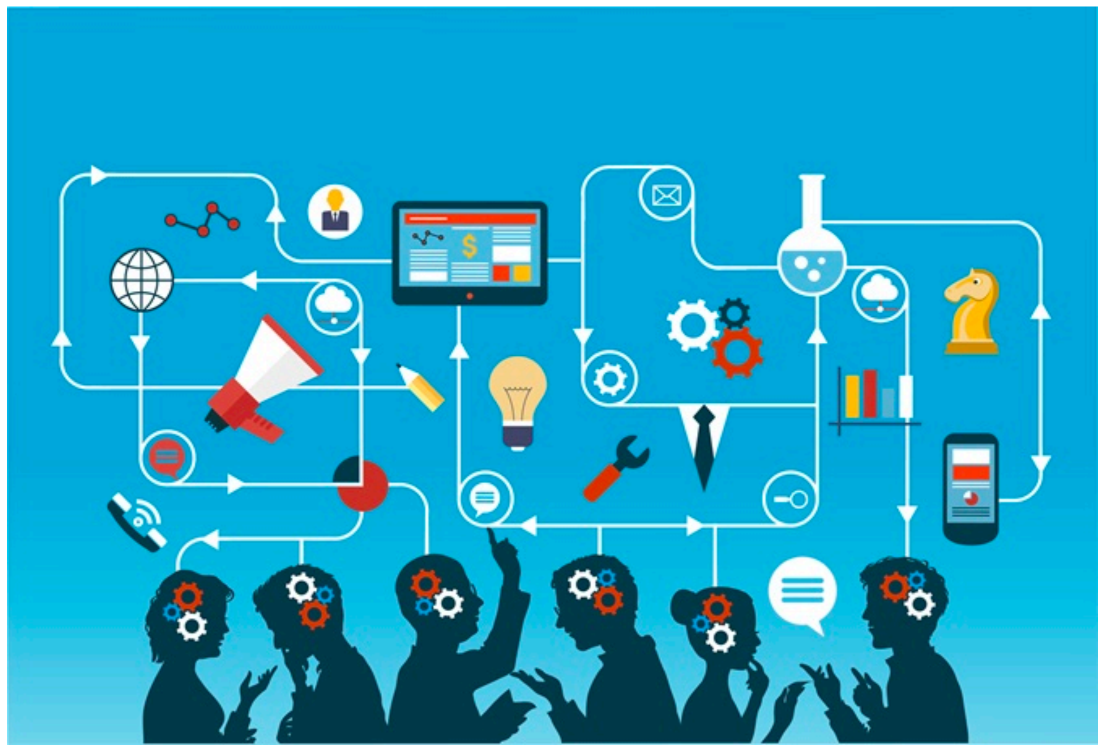
joint work with:

Steve Carr, Alexandru Serban, Ralf Wimmer, Ufuk Topcu, Bernd Becker

TEXAS
The University of Texas at Austin

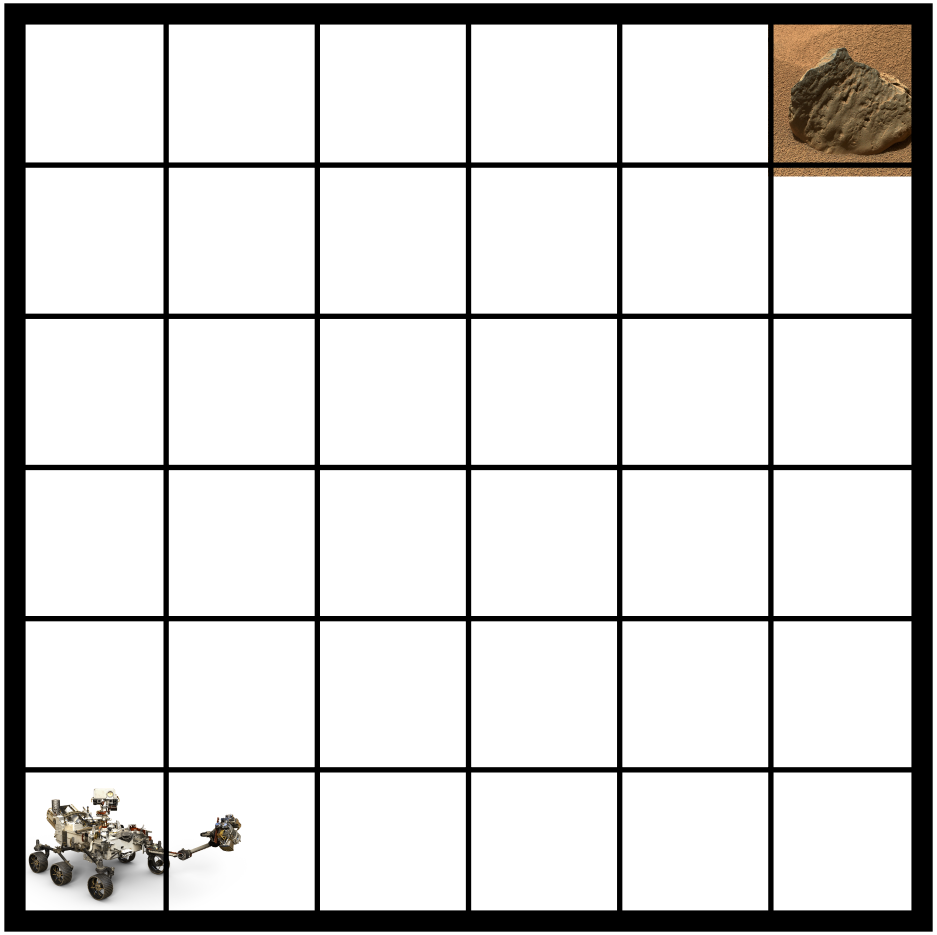
UNI FREIBURG

Radboud University



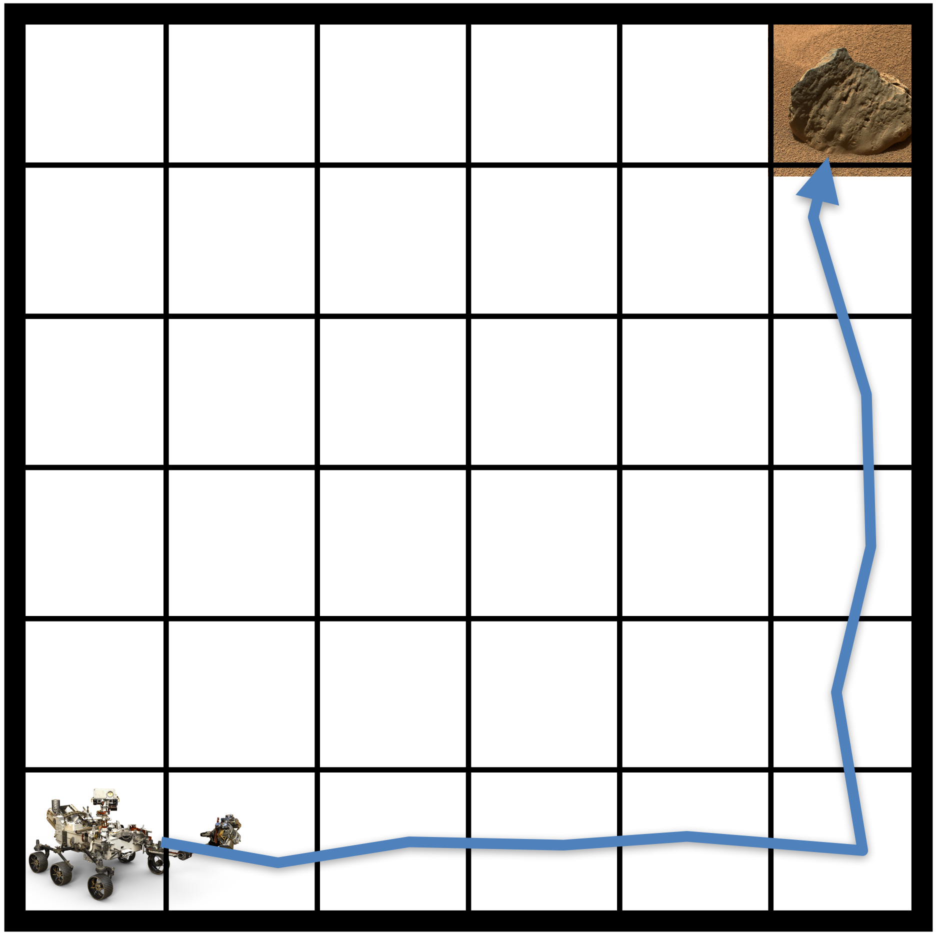
Help the Robot

Find the best way to the **rock**



Help the Robot

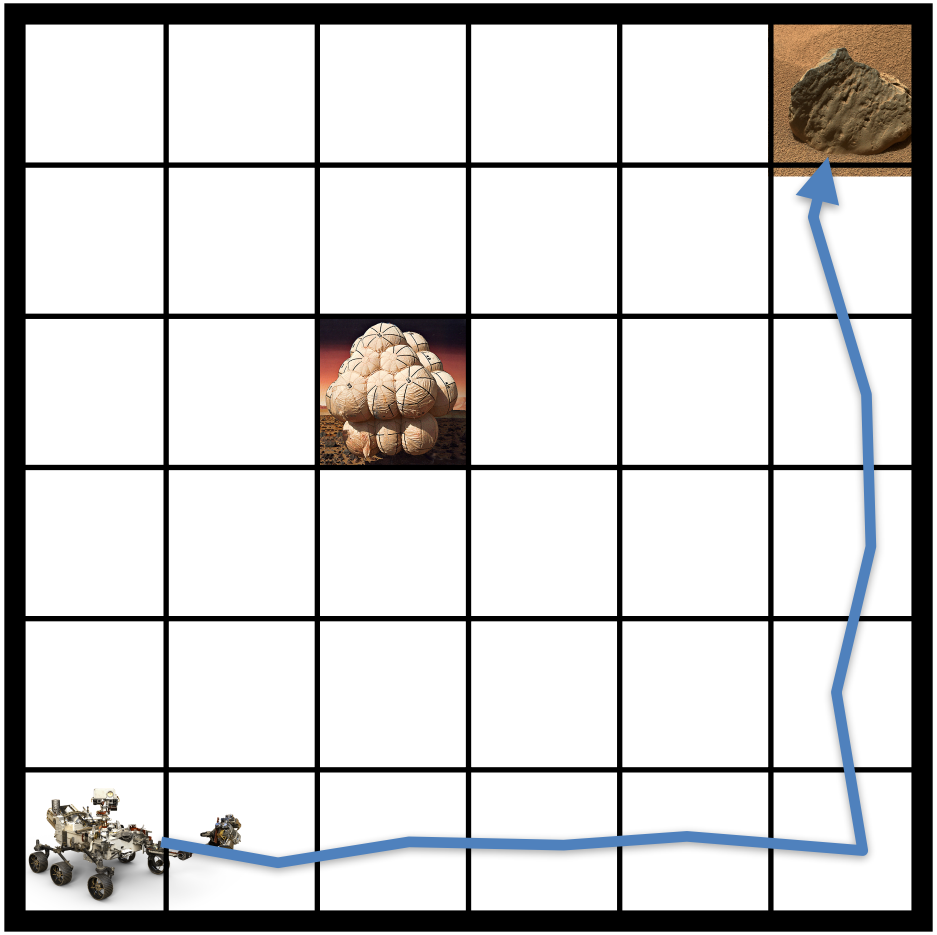
Find the best way to the **rock**



Help the Robot

Find the best way to the rock

Visit the parachute on the way

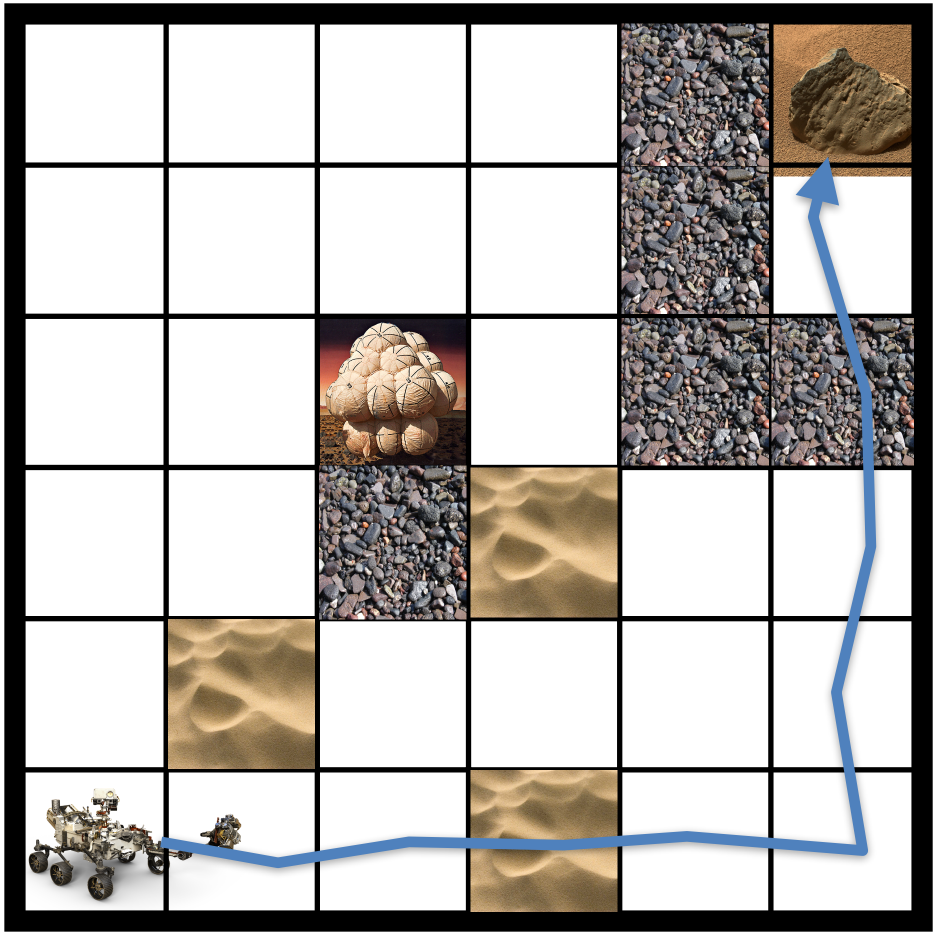


Help the Robot

Find the best way to the rock

Visit the parachute on the way

Take expensive surfaces into account



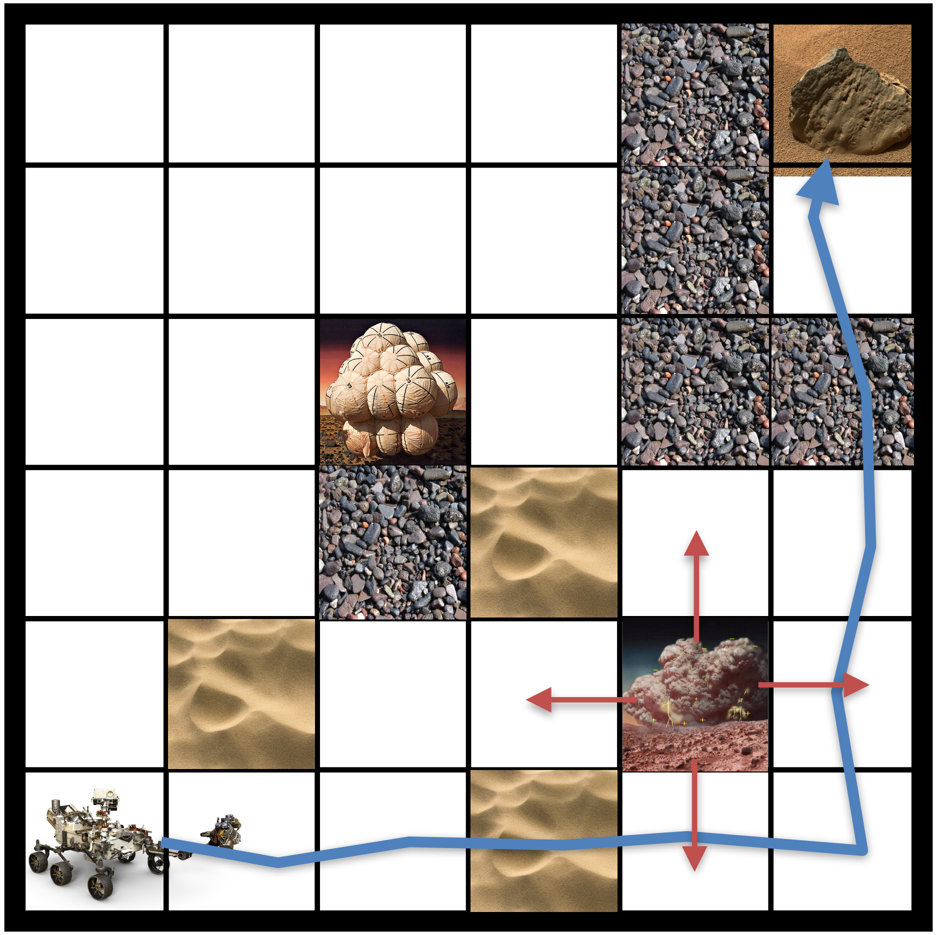
Help the Robot

Find the best way to the rock

Visit the parachute on the way

Take expensive surfaces into account

Avoid randomly moving dust storm



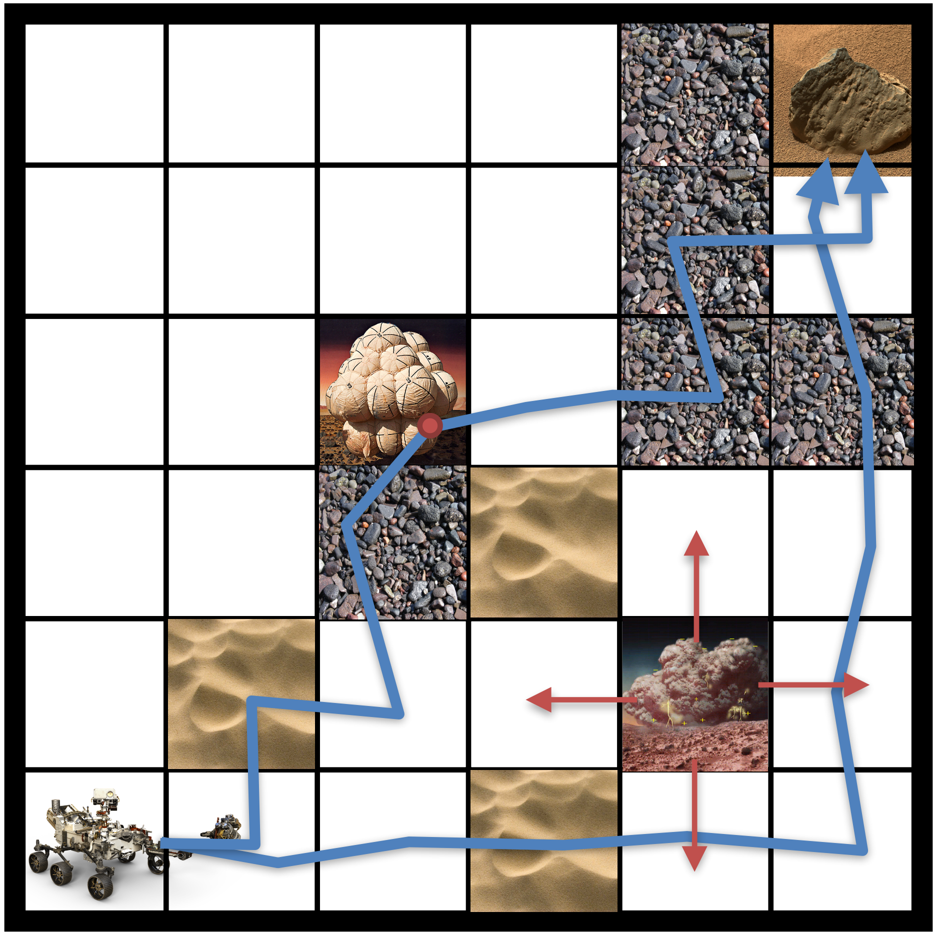
Help the Robot

Find the best way to the rock

Visit the parachute on the way

Take expensive surfaces into account

Avoid randomly moving dust storm



Find safe and/or cost-optimal strategy to get to the airbag

Help the Robot

Find the best way to the rock

Visit the parachute on the way

Take expensive surfaces into account

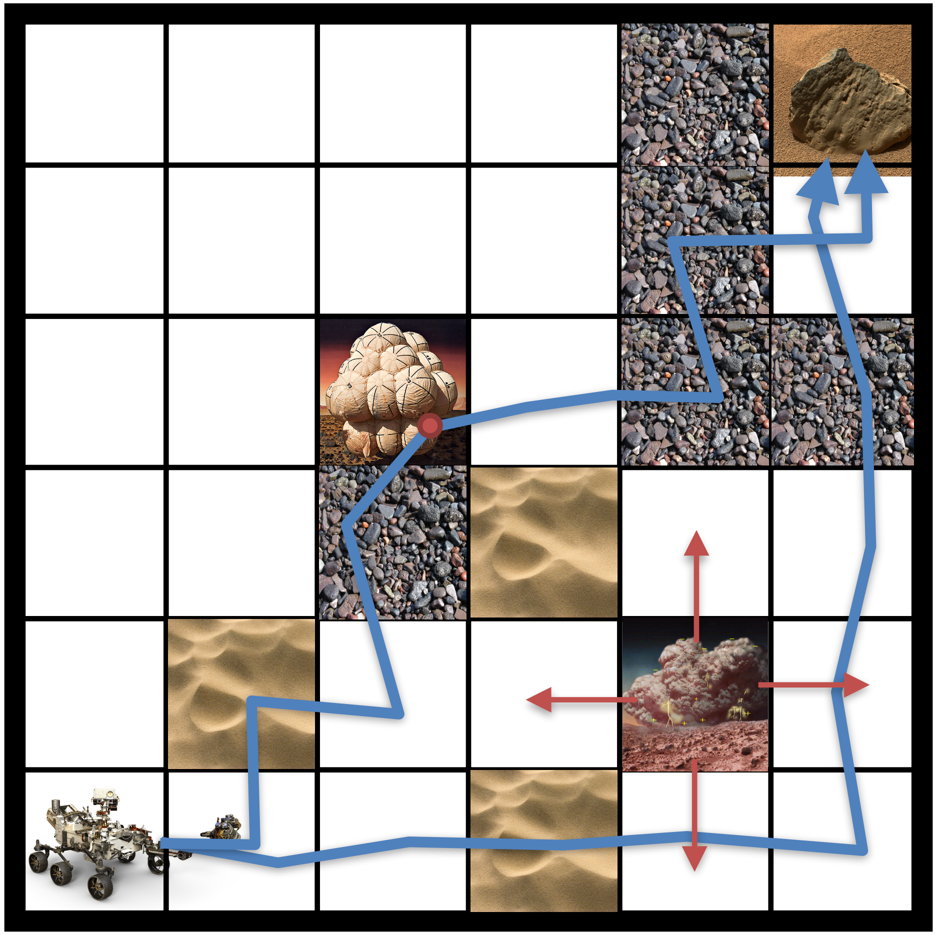
Avoid randomly moving dust storm

$$Pr_{max}(\neg s\ U\ (\Diamond(p \wedge \bigcirc \Diamond r))$$

$$EC_{min}(\Diamond r)$$

temporal logic

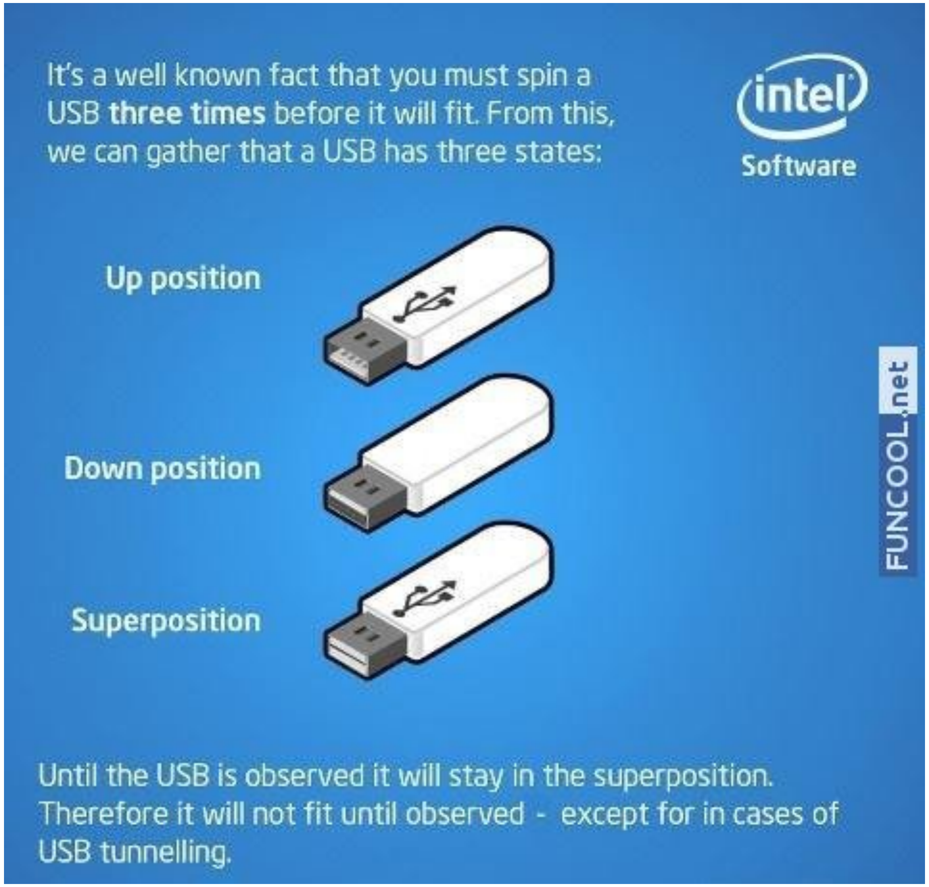
expected cost



Find safe and/or cost-optimal strategy to get to the airbag

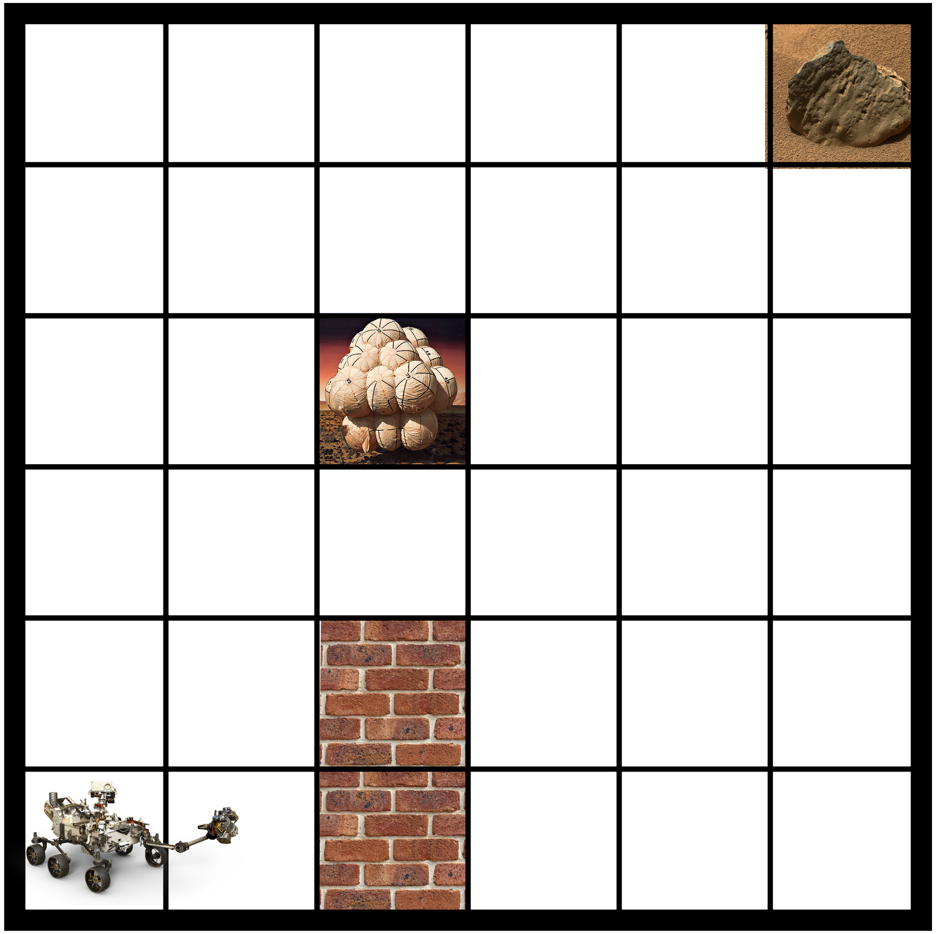
Underlying Model: Markov Decision Process

Partial Observability



Help the Robot with Partial Observability

Robot has restricted range of vision

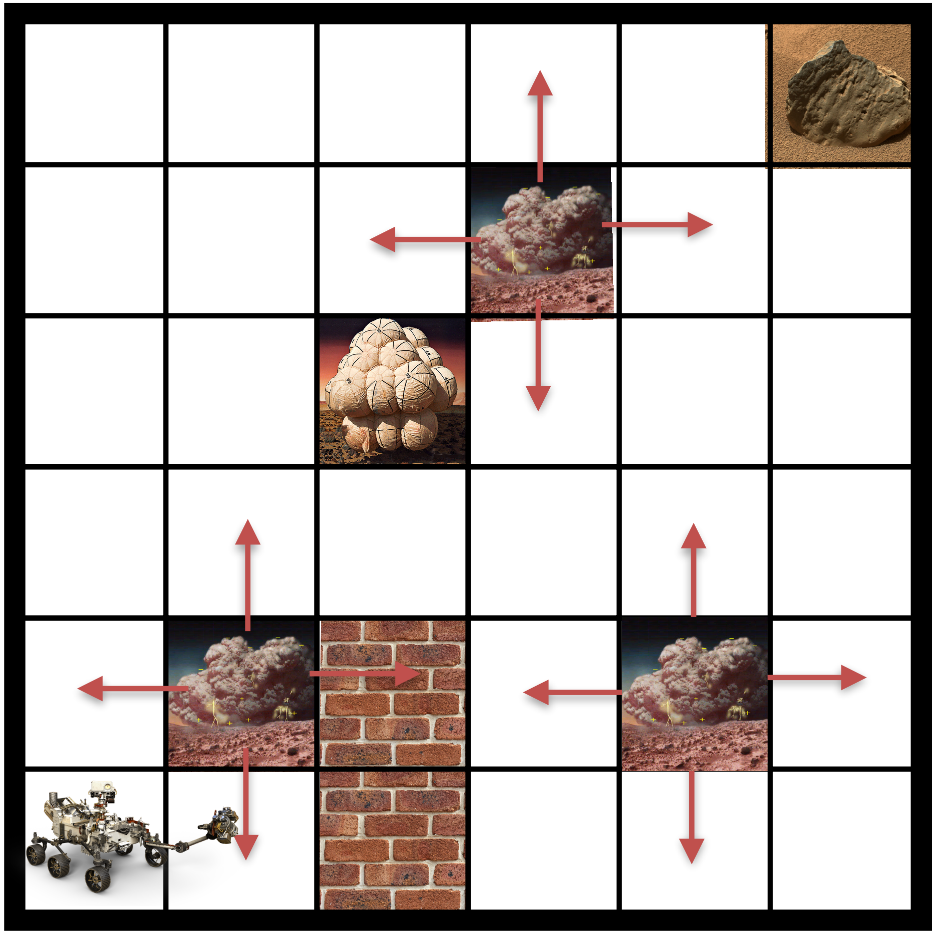


Find **safe** and/or **cost-optimal** strategy to get to the airbag

Help the Robot with Partial Observability

Robot has restricted range of vision

Storm is only **observable** when near



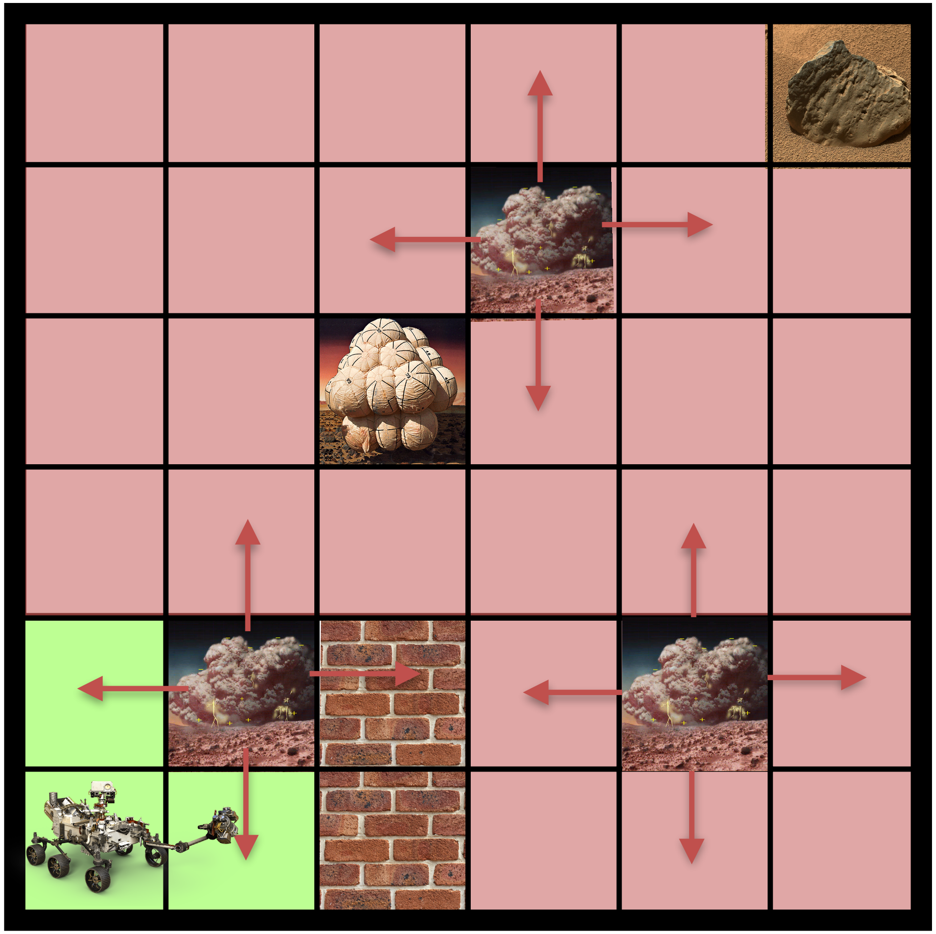
Find safe and/or cost-optimal strategy to get to the airbag

Help the Robot with Partial Observability

Robot has restricted range of vision

Storm is only observable when near

For robot, storm is either near or far



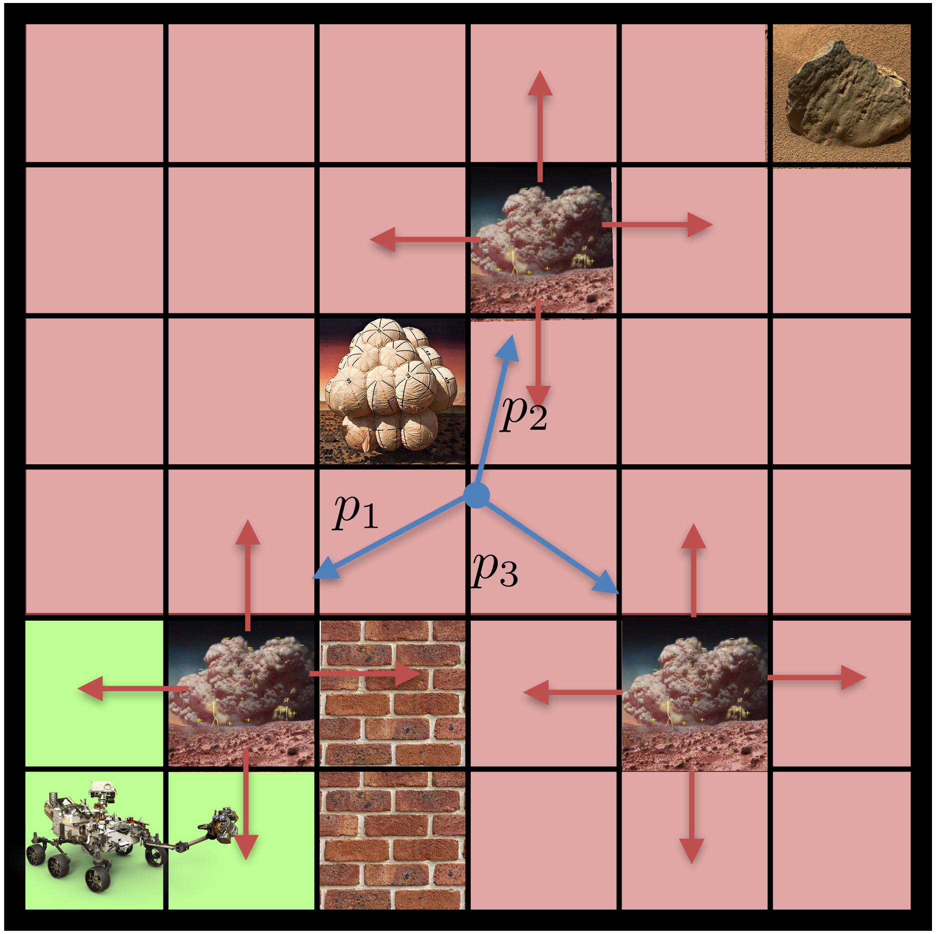
Find safe and/or cost-optimal strategy to get to the airbag

Help the Robot with Partial Observability

Robot has restricted range of vision

Storm is only observable when near

For robot, storm is either near or far



Find safe and/or cost-optimal strategy to get to the airbag

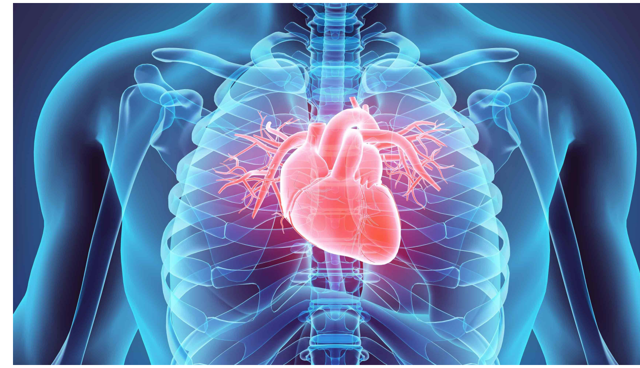
Belief state: Likelihood of the actual position of the storm

infinite belief MDP

POMDPs - Applications

Stock Market

Surveying Threatened Species



Health Care

Wireless Sensor Networks

Autonomous Systems

Machine Vision

Computing Strategies for POMDPs

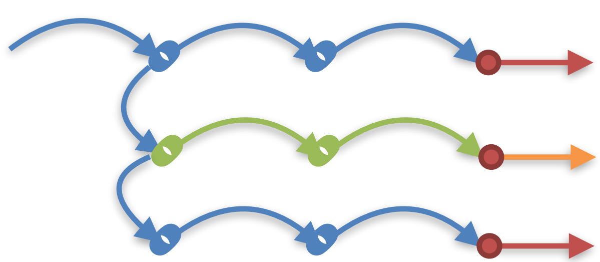
- Randomized with infinite memory: undecidable, optimal results.

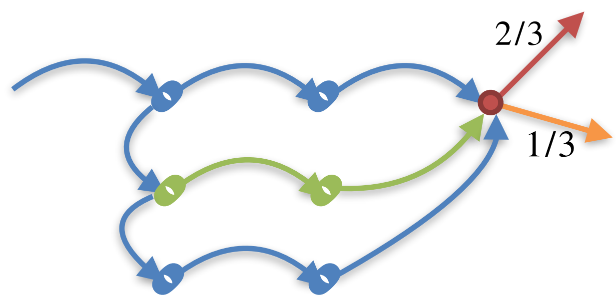
Computing Strategies for POMDPs

- 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.

Computing Strategies for POMDPs

- 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.





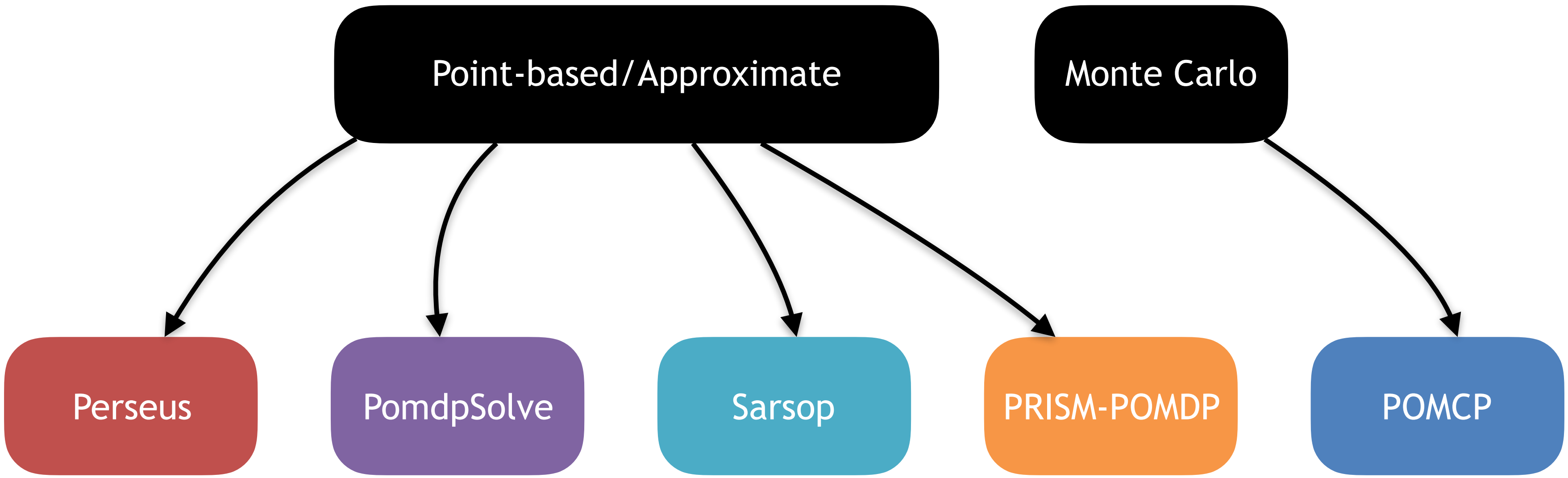
$\sigma : ObsSeq_{fin} \to Distr(Act)$ $\sigma : Obs \to Distr(Act)$

POMDP Solving - State of The Art

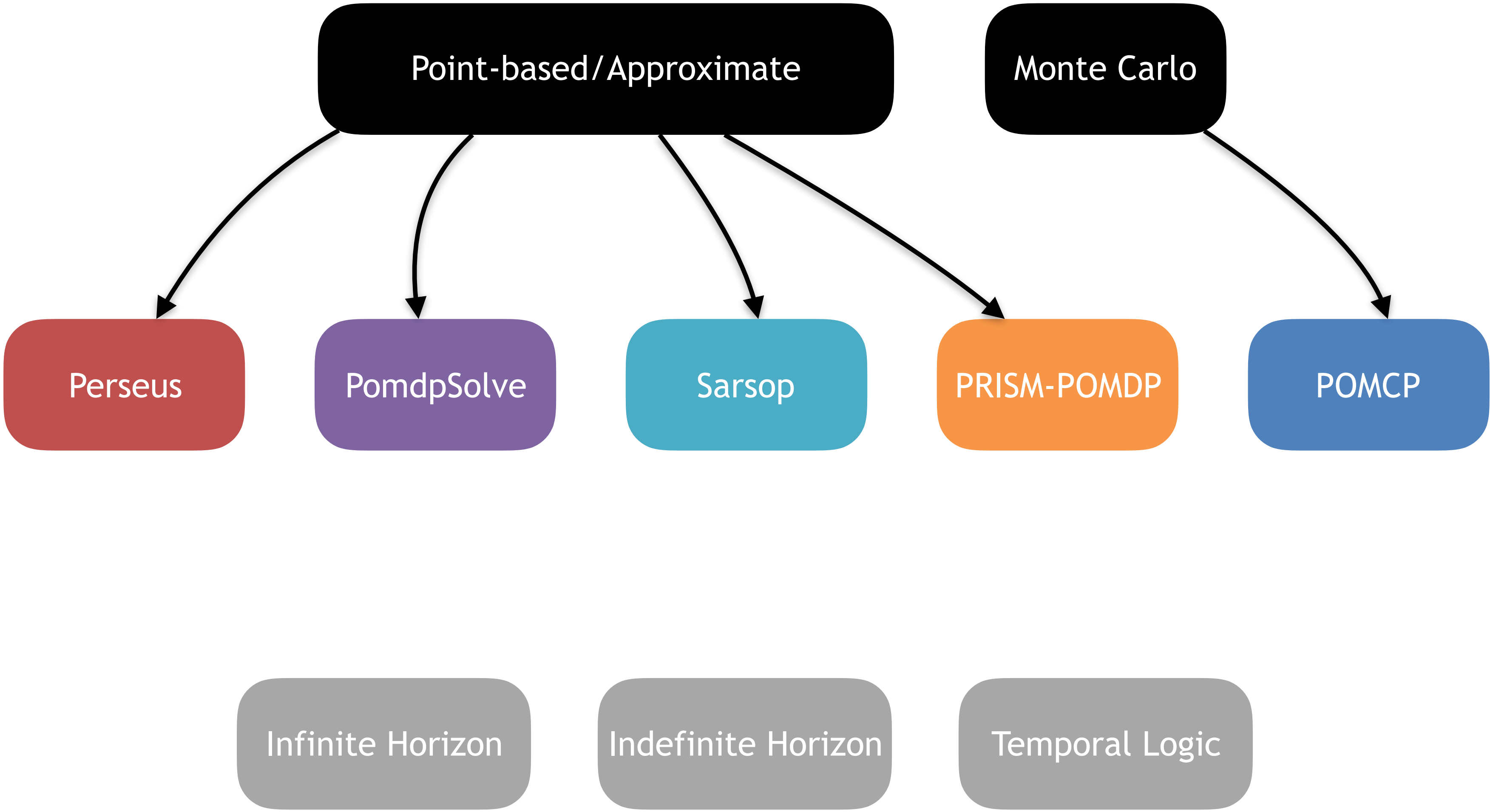
Point-based/Approximate

Monte Carlo

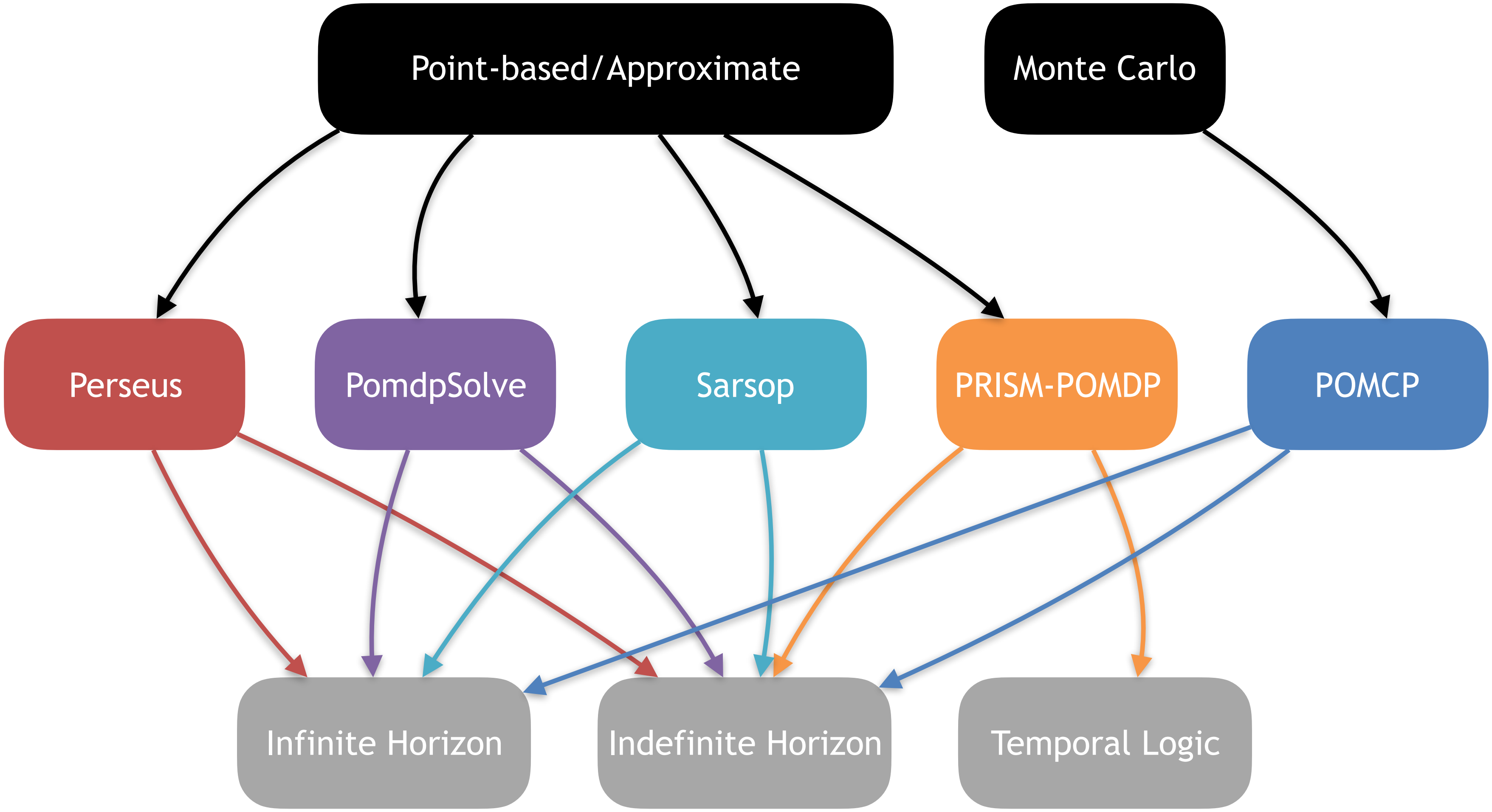
POMDP Solving - State of The Art



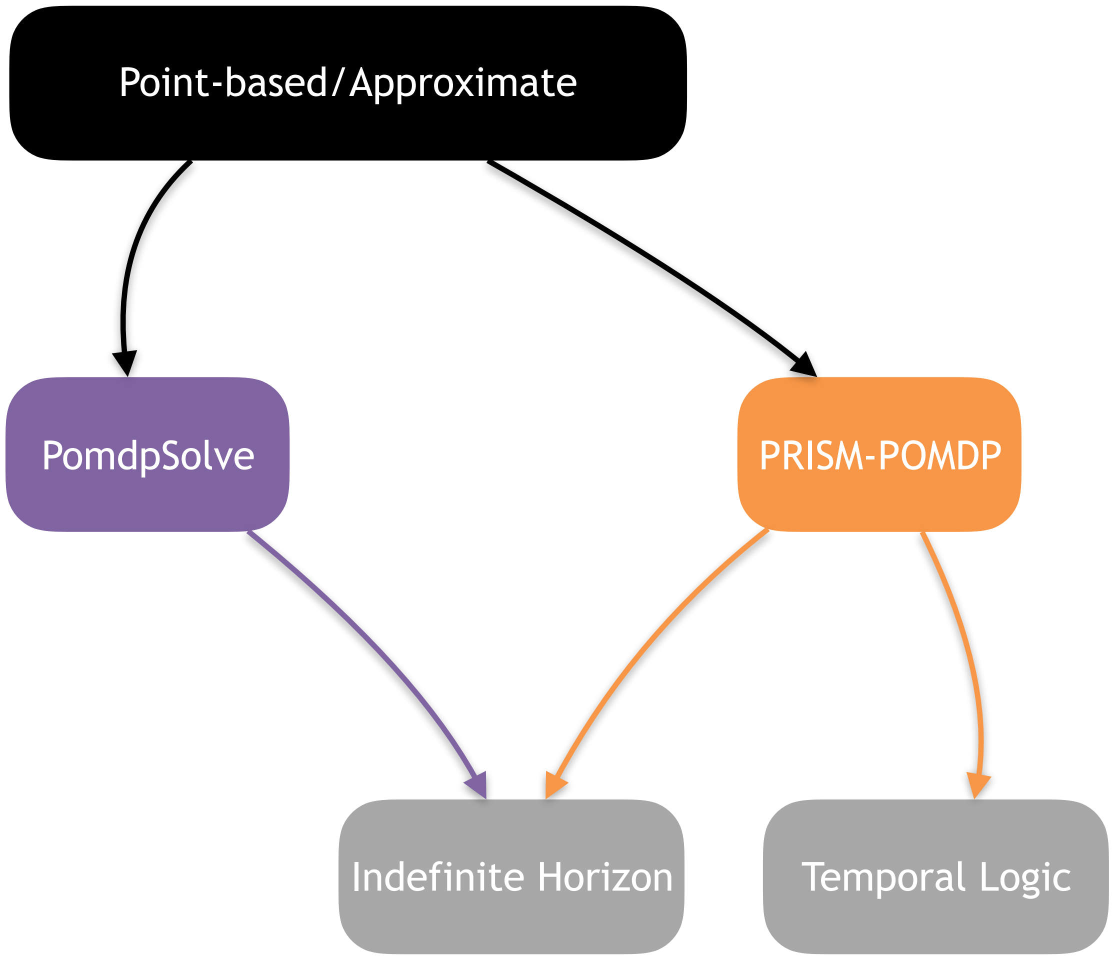
POMDP Solving - State of The Art



POMDP Solving - State of The Art



POMDP Solving - State of The Art



The Problem: Strategy Synthesis

$\mathscr{M}$ POMDP

φ Specification

Probabilistic
Temporal Logic
Constraints

The Problem: Strategy Synthesis

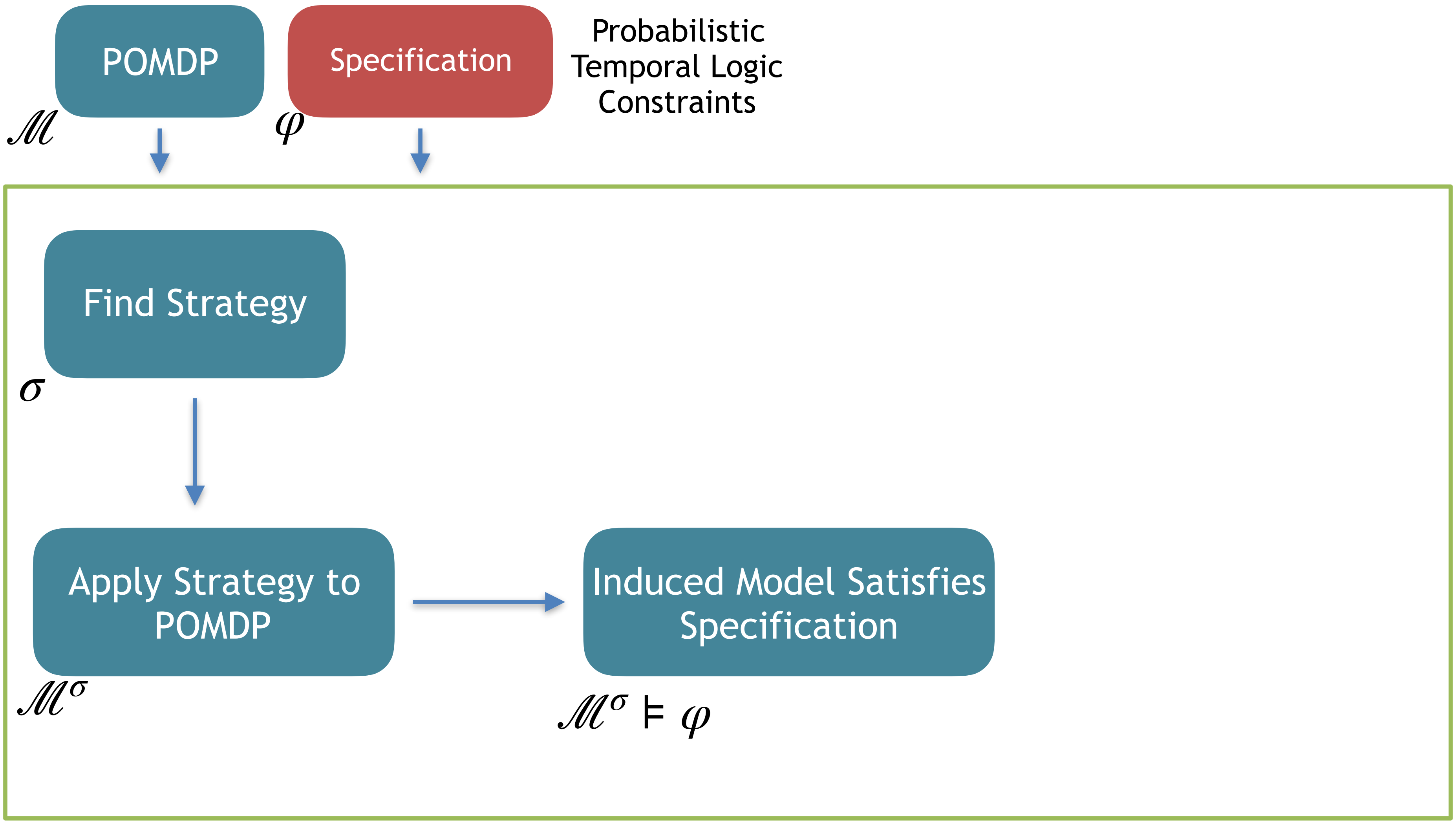
POMDP $\mathscr{M}$

Specification φ

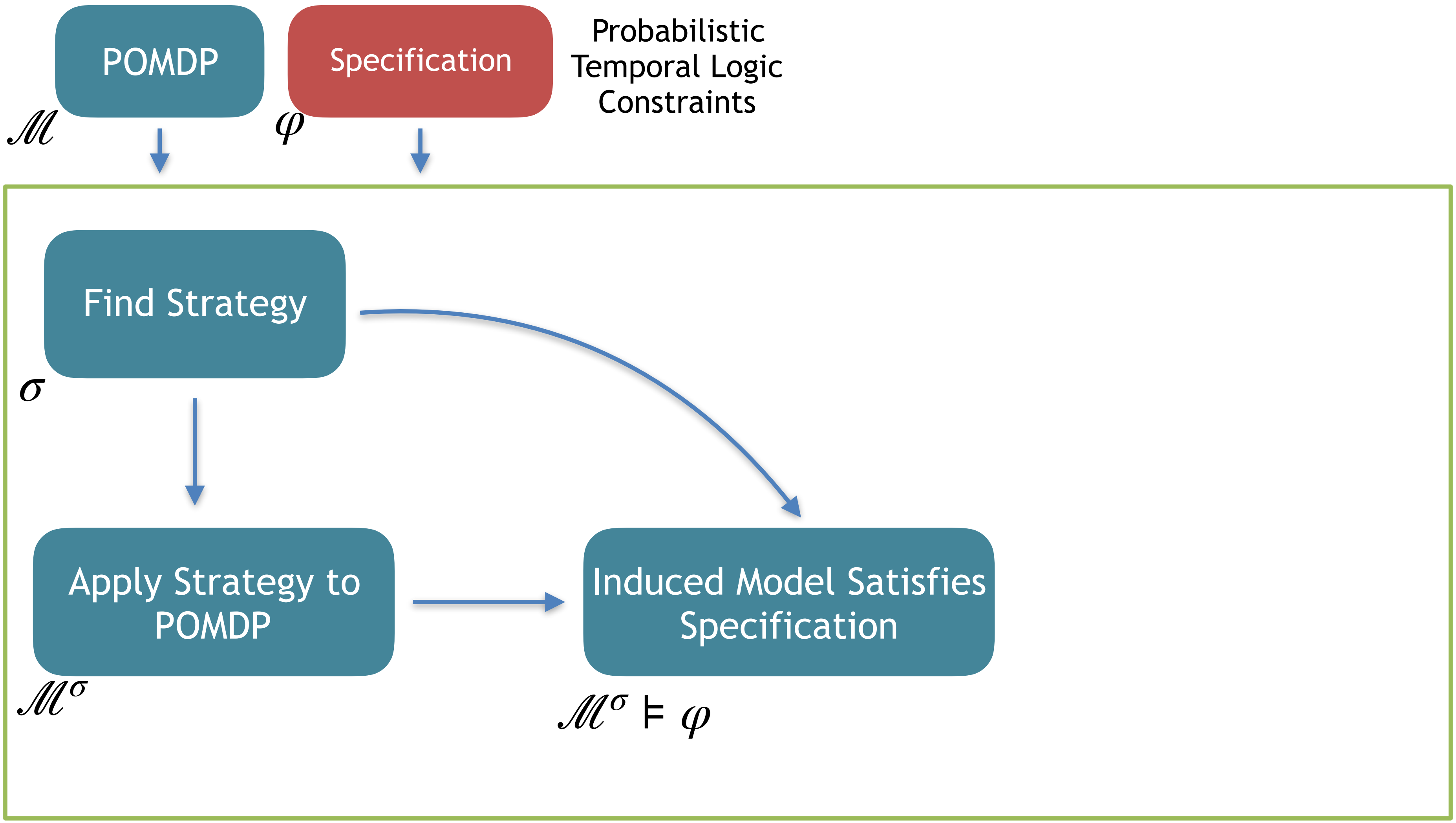
Probabilistic Temporal Logic Constraints

Find Strategy σ

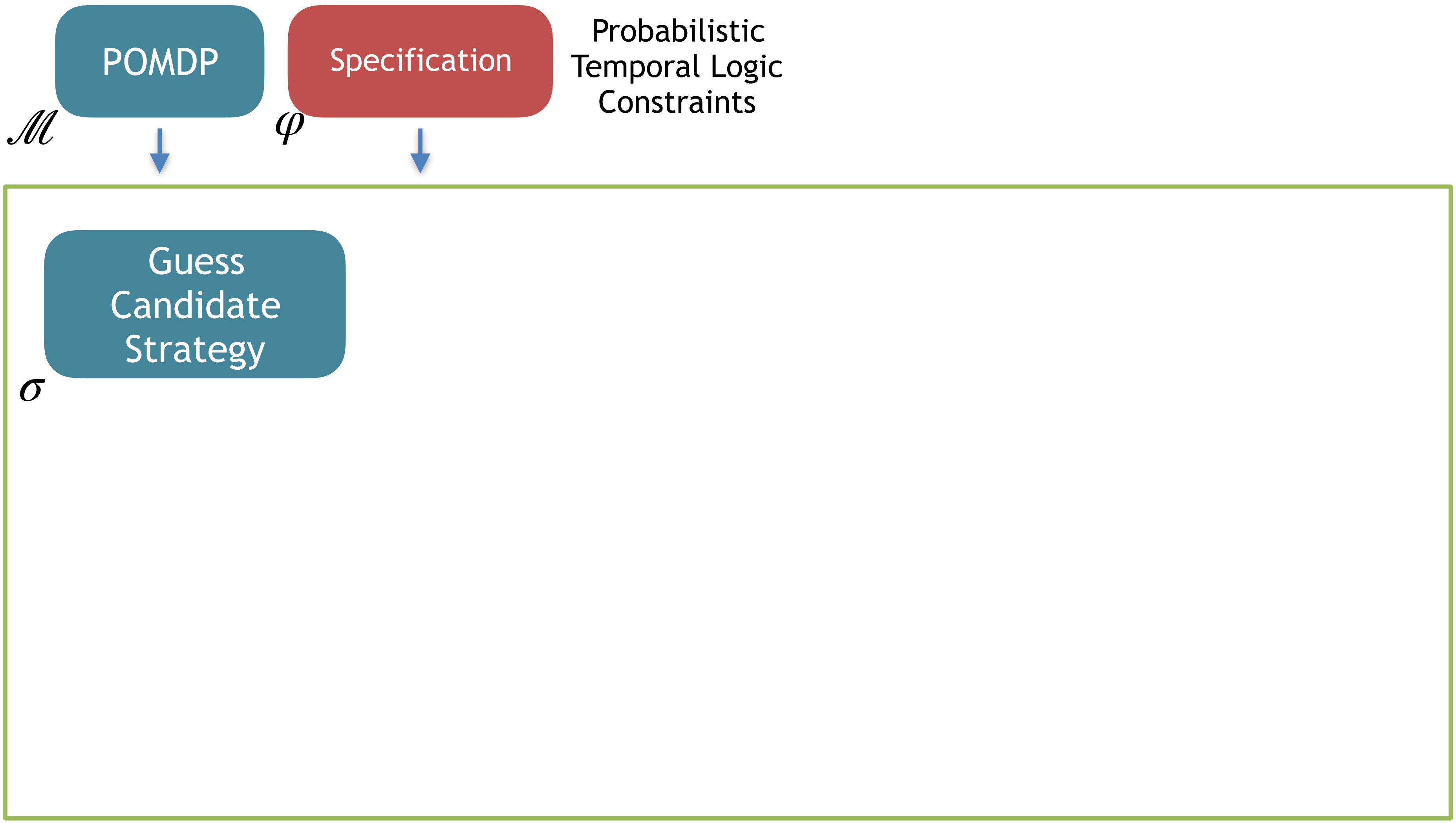
The Problem: Strategy Synthesis



The Problem: Strategy Synthesis



Be Lazy: Guess a Strategy and Verify!



Be Lazy: Guess a Strategy and Verify!

POMDP

$\mathscr{M}$

Specification

φ

Probabilistic Temporal Logic Constraints

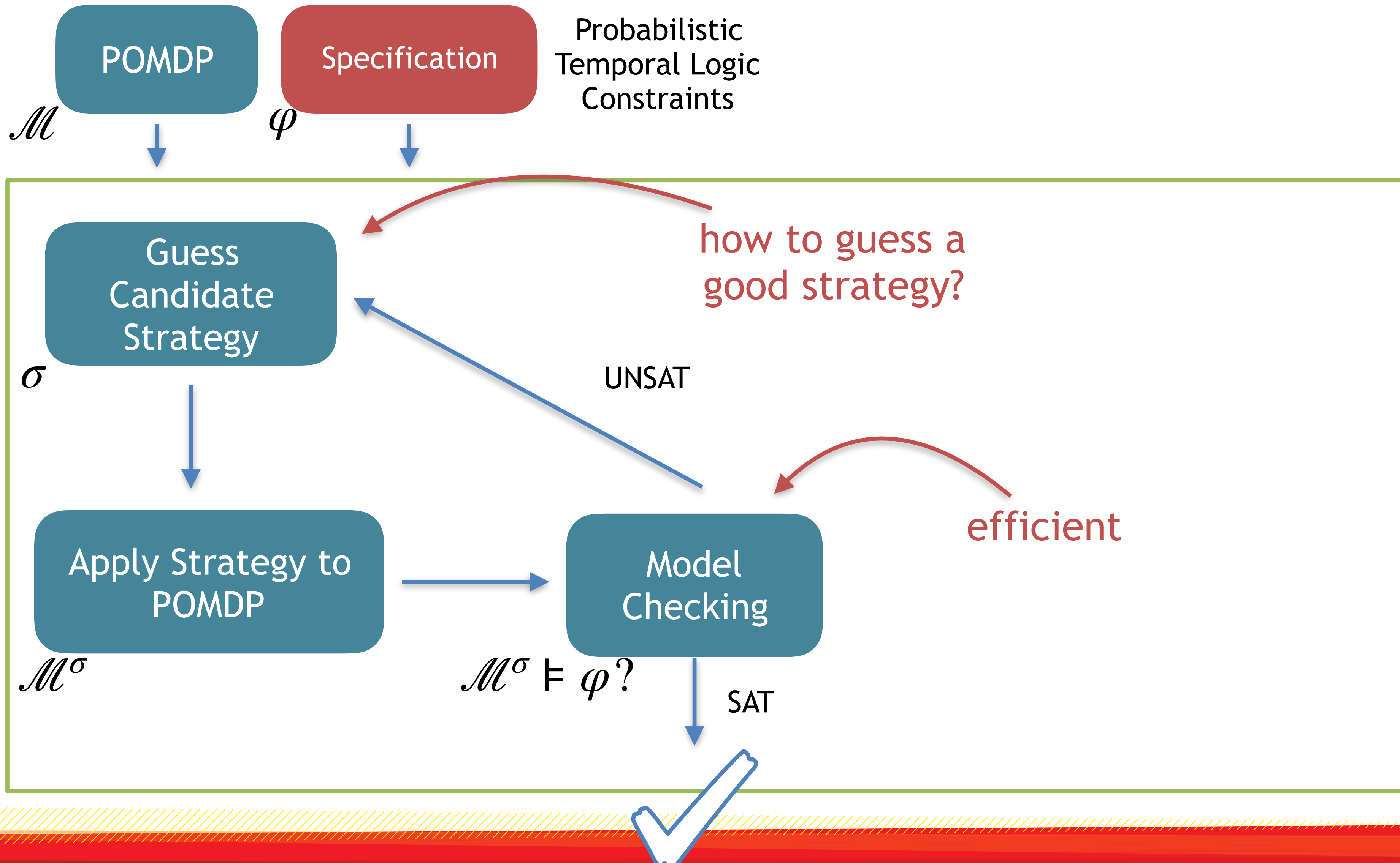
Guess Candidate Strategy

σ

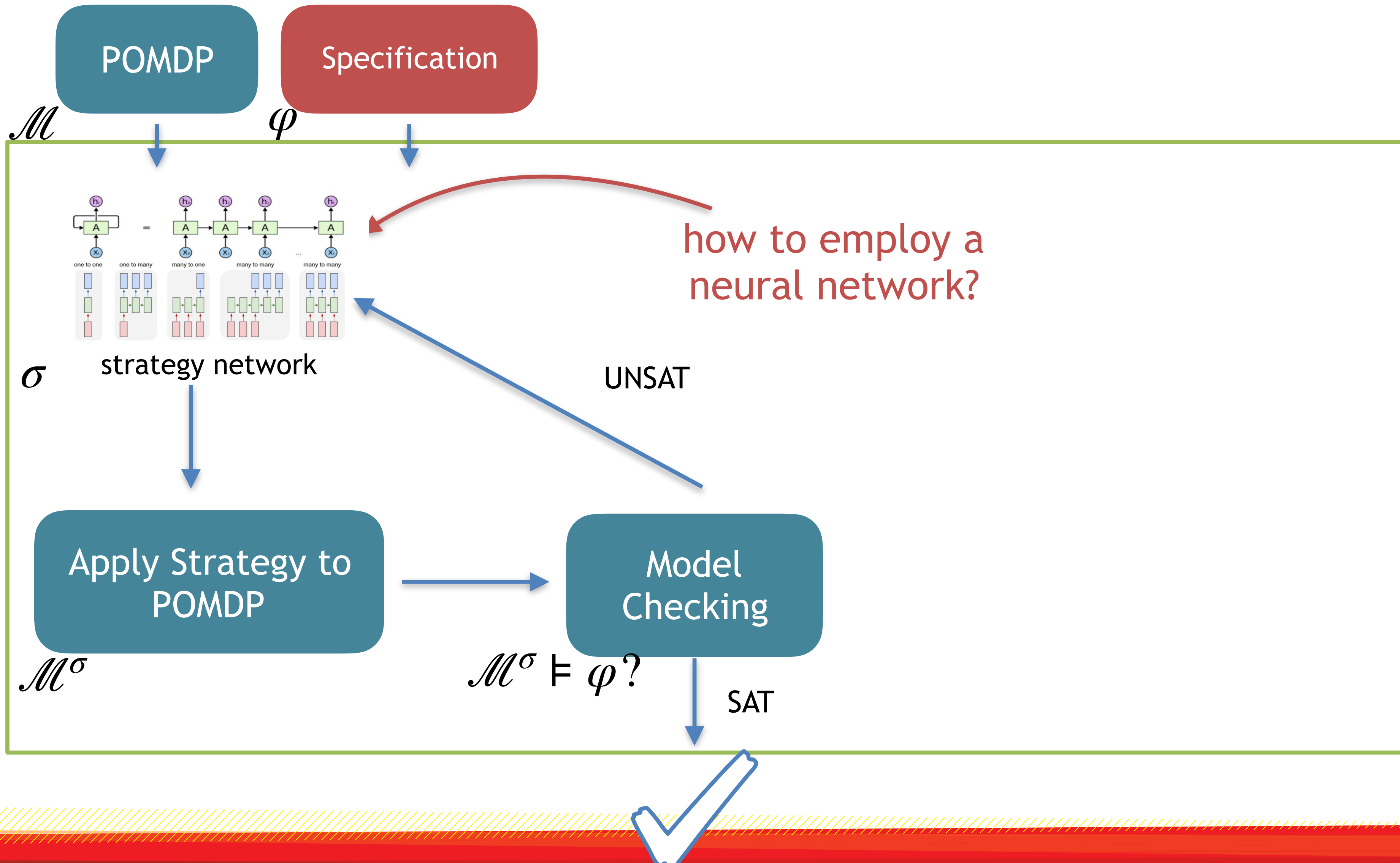
Apply Strategy to POMDP

$\mathscr{M}^{\sigma}$

Be Lazy: Guess a Strategy and Verify!



Let Machine Learning do the Guessing?



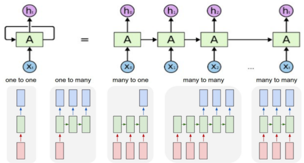
RNN Strategy Improvement

POMDP

$\mathscr{M}$

Specification

φ



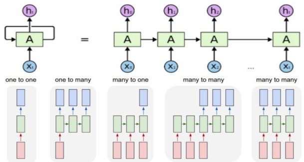
strategy network

RNN Strategy Improvement

POMDP

Specification

$\mathscr{M}$ φ



strategy network

σ

Extract Strategy

$\mathscr{M}^{\sigma}$

Apply Strategy to POMDP

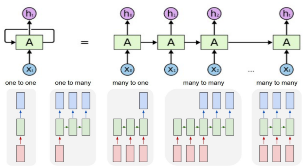
RNN Strategy Improvement

POMDP

$\mathscr{M}$

Specification

φ



strategy network

σ

Extract Strategy

$\mathscr{M}^\sigma$

Apply Strategy to POMDP

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

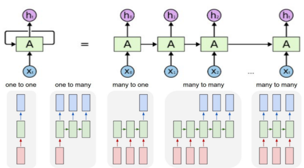
Model Checking

SAT

RNN Strategy Improvement

POMDP $\mathscr{M}$

Specification φ



strategy network

σ

Extract Strategy

$\mathscr{M}^\sigma$

Apply Strategy to POMDP

$\mathscr{M}^\sigma \vDash \varphi$?

Model Checking

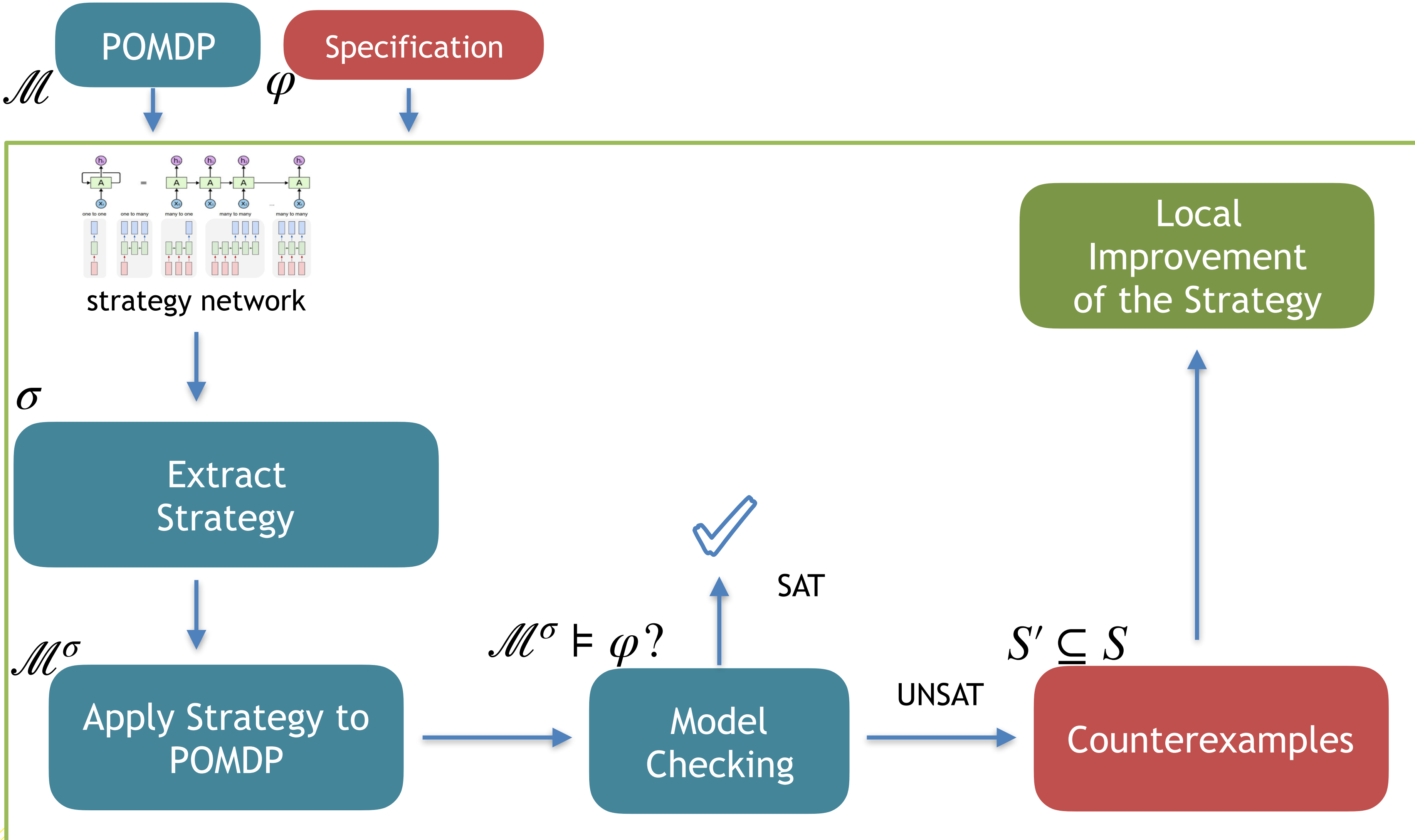
SAT ✓

UNSAT

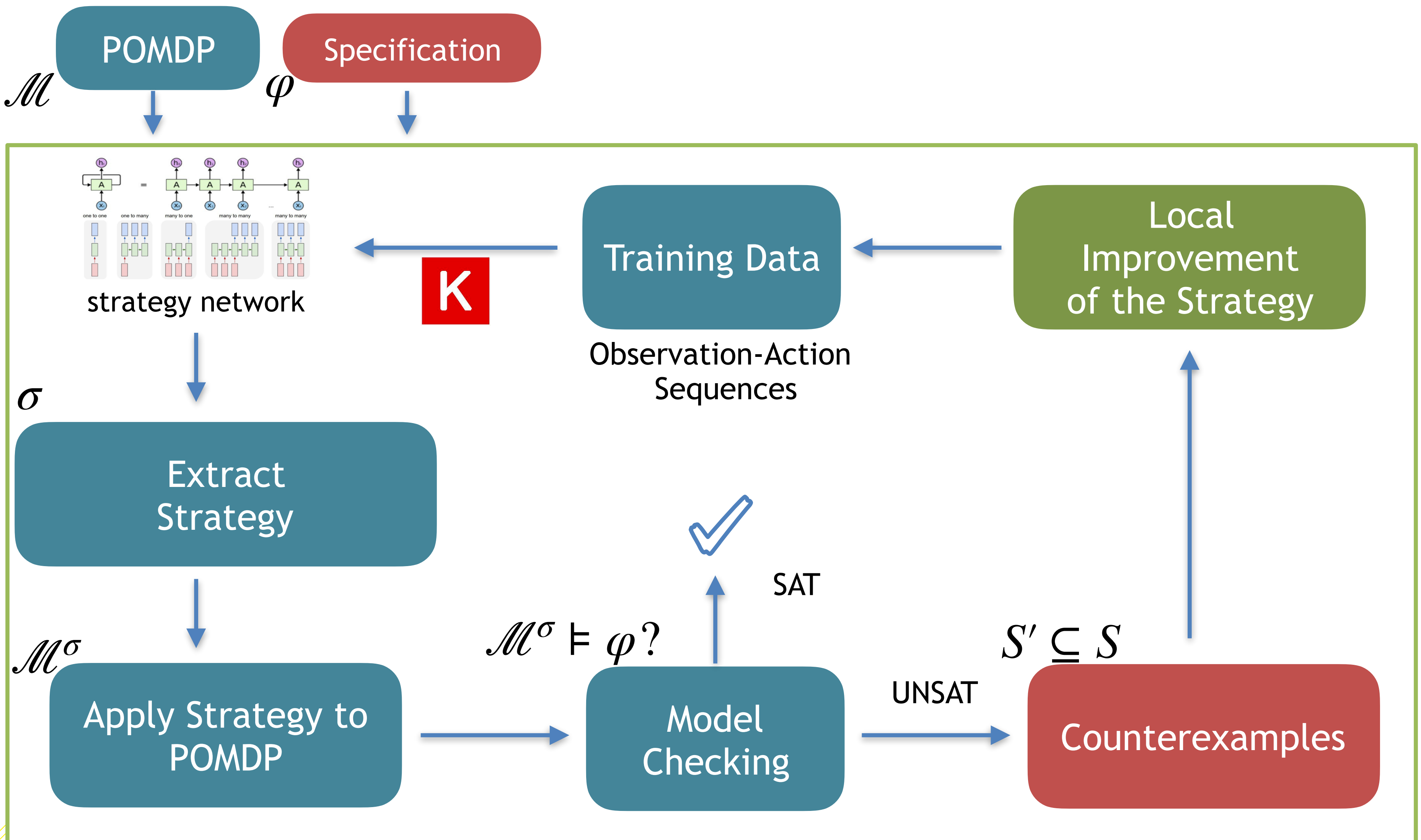
$S' \subseteq S$

Counterexamples

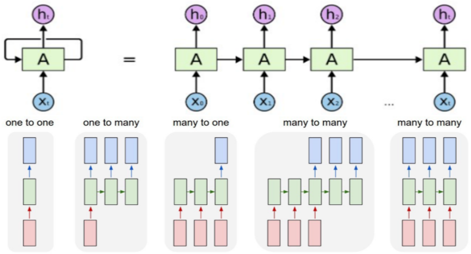
RNN Strategy Improvement



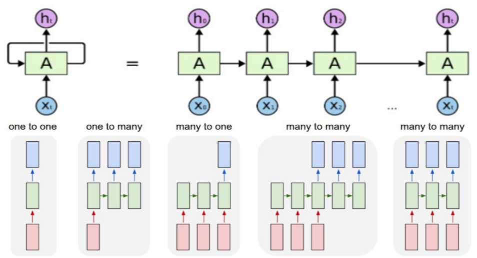
RNN Strategy Improvement



Learning Strategies with RNNs



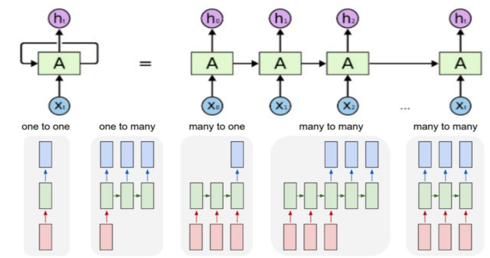
Learning Strategies with RNNs



Recurrent Neural Network

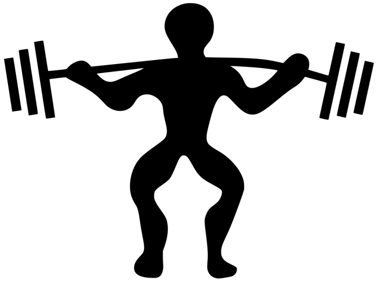
- long short-term memory (LSTM) architecture to learn dependencies in sequential data
- trained with observation-action sequences $ObsSeq_{fin}$
- strategy network $\sigma\colon ObsSeq_{fin} \rightarrow Distr(Act)$
- observations are input labels, actions are output labels

Learning Strategies with RNNs



Recurrent Neural Network

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



Initial Training

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

Improving the Strategy

Improving the Strategy

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

Improving the Strategy

- 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 Strategy

- 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) strategy.

Local linear program

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

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 Strategy

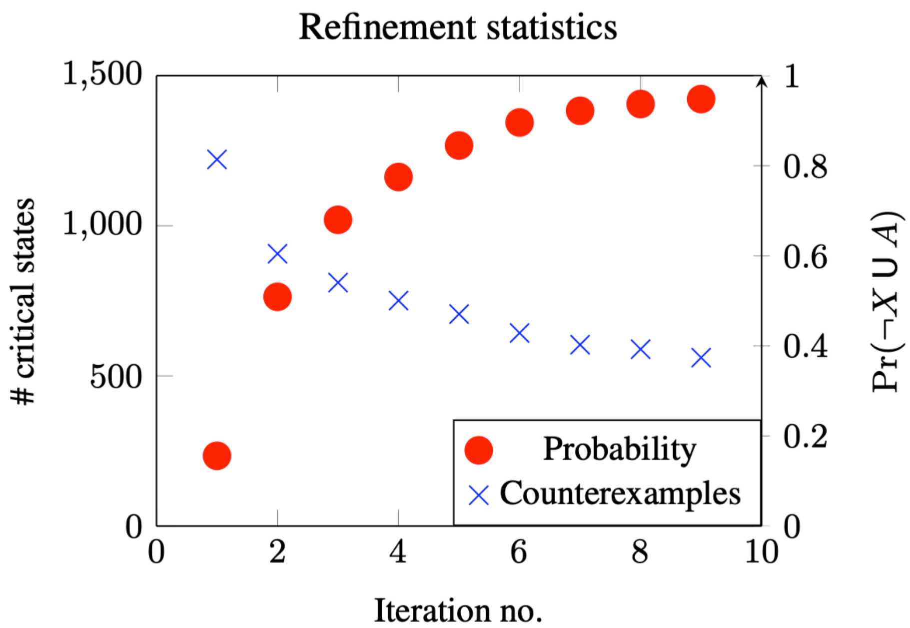
- Identify **critical decisions** that lead to states with high probability of **violating** the specification.
- For each observation with critical decision, **minimize** the number of different critical actions.
- **Retrain** with the new (locally improved) strategy.

Local linear program

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

subject to

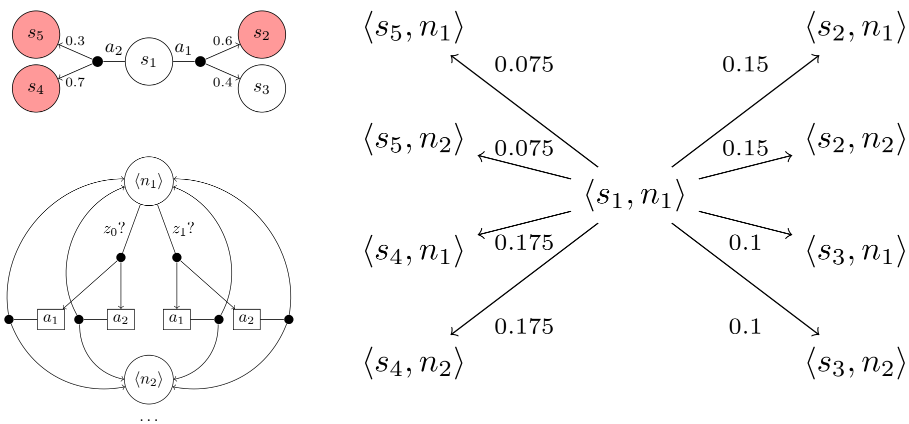
$$\forall s \in O^{-1}(z). \quad p_s = \sum_{a \in Act} \gamma(z)(a) \cdot \sum_{s' \in S} \mathscr{P}(s,a,s') \cdot p^*(s')$$



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

Finite-memory Strategies (FSC)

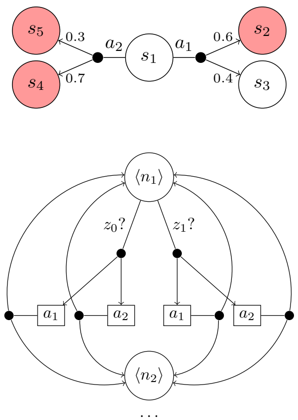
- Encode finite memory directly into the state space:

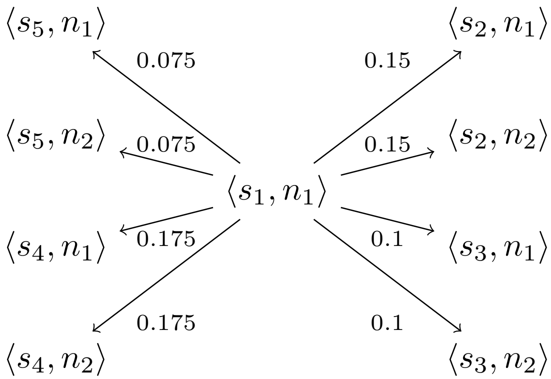


- Strategy network is of the form $\sigma\colon ObsSeq_{fin} \rightarrow Distr(Act)$
- But: How to infer a memory-update function to construct an FSC?

Finite-memory Strategies (FSC)

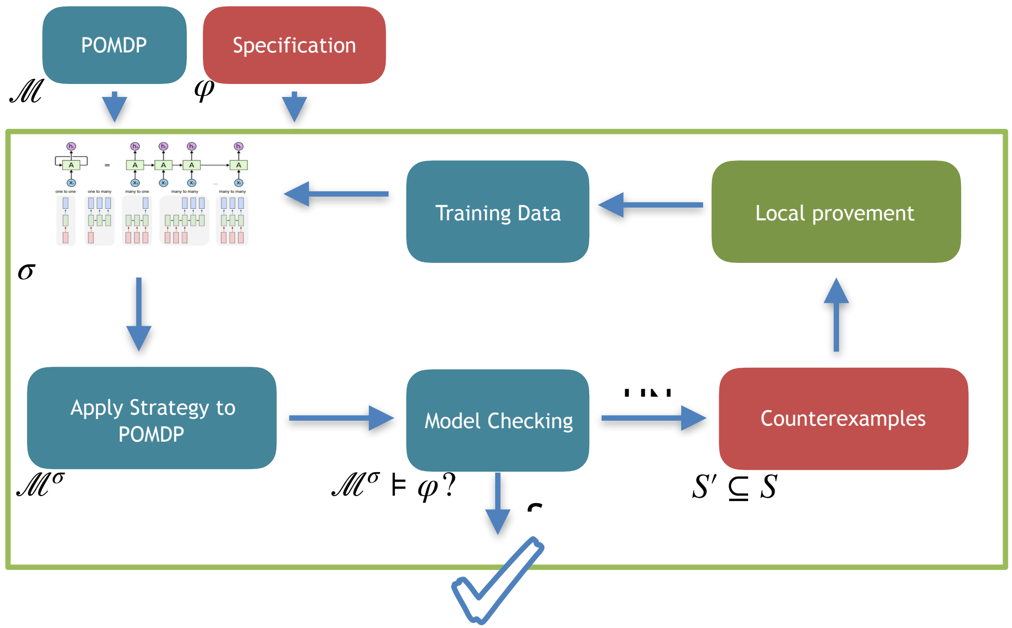
- Encode finite memory directly into the state space:



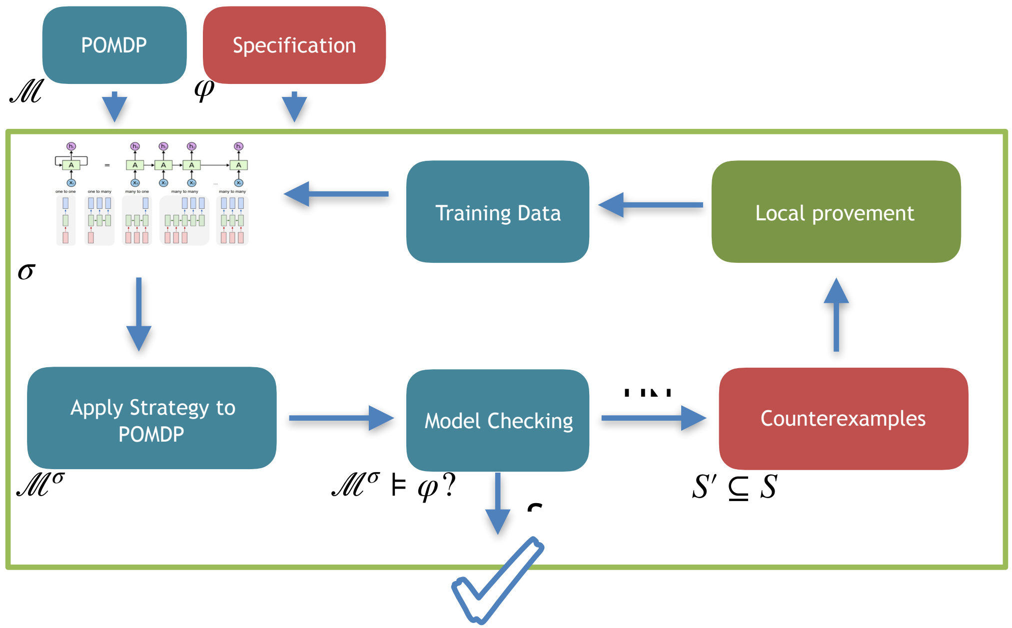


- Strategy network is of the form $\sigma\colon ObsSeq_{fin} \rightarrow Distr(Act)$
- But: How to infer a **memory-update function** to construct an FSC?
- **Predefine memory update**, for instance (deterministic) transition upon repetition of an observation.
- Compute product of FSC and POMDP and compute memoryless strategy.

Correctness and Completeness?

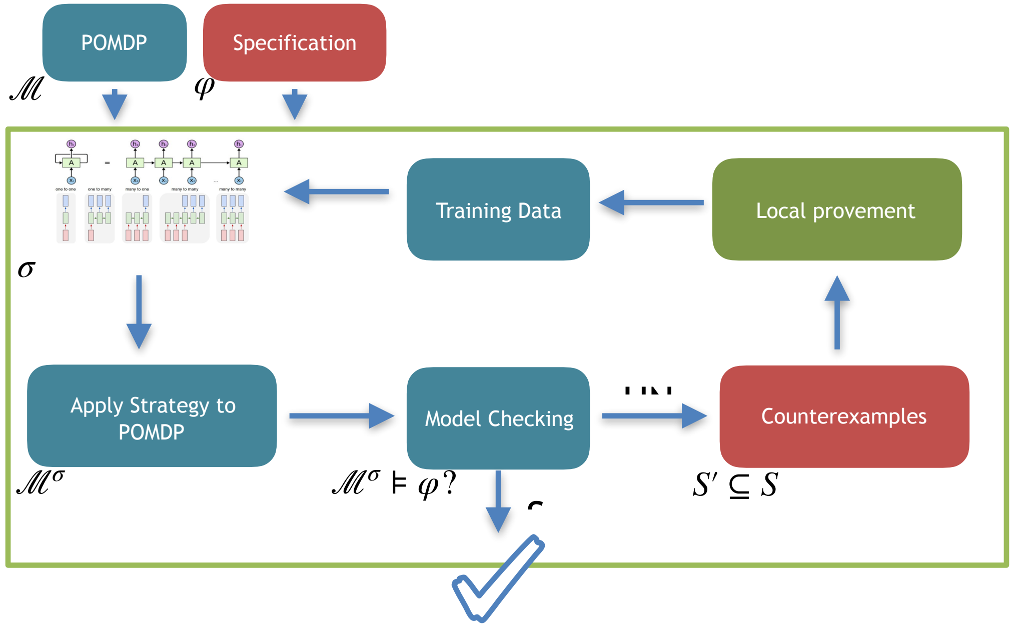


Correctness and Completeness?



Correct, as each strategy prediction is evaluated using model checking.

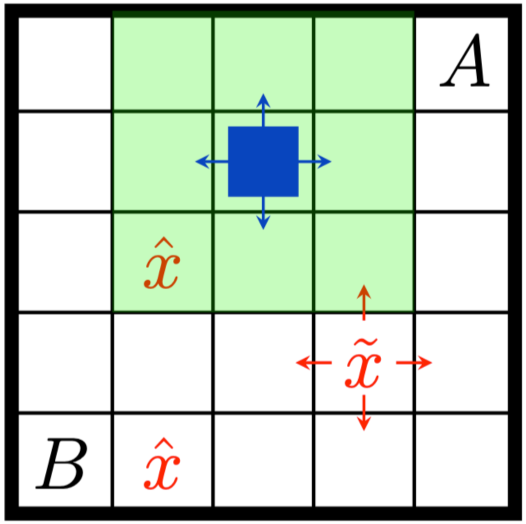
Correctness and Completeness?



Correct, as each strategy prediction is evaluated using model checking.

Not complete, as we may never find a feasible strategy. Also, problem is undecidable (or hard) anyways :).

Experiments - LTL



Problem	$\|S\|$	$\|Act\|$	$\|Z\|$
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

			RNN-based Synthesis		PRISM-POMDP	
Problem	States	Type, φ	Res.	Time (s)	Res.	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	Res	Time (s)	PRISM-POMDP Res	Time (s)	pomdpSolve Res	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

Conclusion

- Novel way to generate provably correct POMDP strategies
- Good scalability, not optimal
- Results transferrable
- Future work: More principled approach to finite-memory strategies —> Extract FSC directly from RNN