

# *Learning to Identify Irrelevant State Variables*

Nicholas K. Jong and Peter Stone

{nkj,pstone}@cs.utexas.edu.

Department of Computer Sciences

The University of Texas at Austin

## Introduction

---

- Choice of **state representation** is critical for AI algorithms
  - If too **coarse**, suboptimal behavior
  - If too **fine**, poor efficiency and generalization

## Introduction

---

- Choice of **state representation** is critical for AI algorithms
  - If too **coarse**, suboptimal behavior
  - If too **fine**, poor efficiency and generalization
- **Abstractions** can adjust representations to the right granularity
  - Removes **irrelevant** distinctions
  - Alleviates **curse of dimensionality**
  - Usually determined manually

## Introduction

- Choice of **state representation** is critical for AI algorithms
  - If too **coarse**, suboptimal behavior
  - If too **fine**, poor efficiency and generalization
- **Abstractions** can adjust representations to the right granularity
  - Removes **irrelevant** distinctions
  - Alleviates **curse of dimensionality**
  - Usually determined manually

*How can we obtain state  
abstractions automatically?*

## Some premises

- Learn only **safe** state abstractions
  - Preserves optimal policy

## Some premises

- Learn only **safe** state abstractions
  - Preserves optimal policy
- Learn from analysis of **solved** tasks
  - Necessary to preserve safety
  - Allows **transfer** to similar tasks

## Some premises

---

- Learn only **safe** state abstractions
  - Preserves optimal policy
- Learn from analysis of **solved** tasks
  - Necessary to preserve safety
  - Allows **transfer** to similar tasks
- Learn about the relevance of state variables
  - For ease of presentation
  - Generalizes to arbitrary state merging

## Defining irrelevance

---

A state variable is **irrelevant** iff **there exists** an optimal policy that ignores that variable.

## Defining irrelevance

A state variable is **irrelevant** iff **there exists** an optimal policy that ignores that variable.

- Let  $\pi(\vec{x}, y)$  be the action that policy  $\pi$  specifies for the state with values  $\vec{x}$  and  $y$  for state variables  $X_1, \dots, X_n$  and  $Y$

## Defining irrelevance

A state variable is **irrelevant** iff **there exists** an optimal policy that ignores that variable.

- Let  $\pi(\vec{x}, y)$  be the action that policy  $\pi$  specifies for the state with values  $\vec{x}$  and  $y$  for state variables  $X_1, \dots, X_n$  and  $Y$
- A policy  $\pi$  **ignores** a variable  $Y$  iff  $\pi(\vec{x}, y_1) = \pi(\vec{x}, y_2)$  for all  $\vec{x}$ ,  $y_1$ , and  $y_2$

## Determining irrelevance

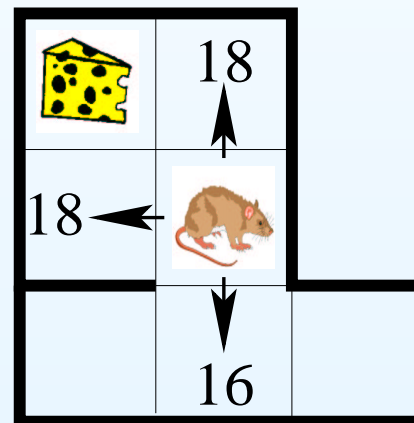
---

- Consider a rat learning a maze
  - $\vec{X}$  is the rat's **location**
  - $Y$  is irrelevant (perhaps **dosage of an experimental drug**)

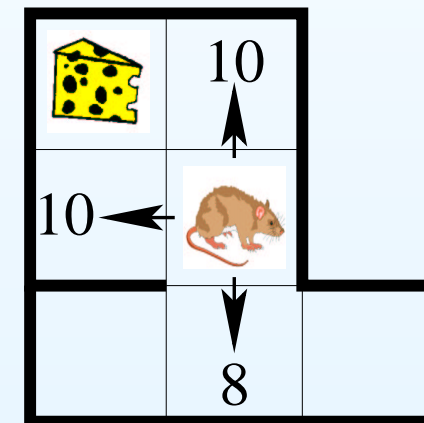
## Determining irrelevance

- Consider a rat learning a maze
  - $\vec{X}$  is the rat's **location**
  - $Y$  is irrelevant (perhaps **dosage of an experimental drug**)
- Two optimal policies for this  $\vec{x}$  **ignore**  $Y$

True Q function for optimal policies



$$Y = y_1$$

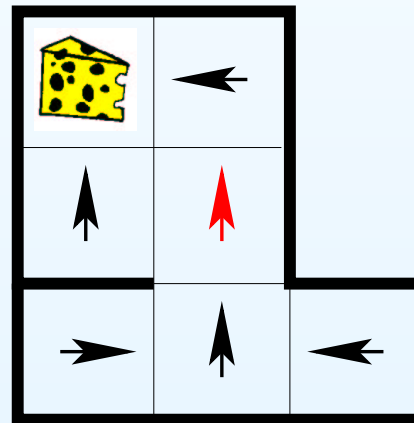


$$Y = y_2$$

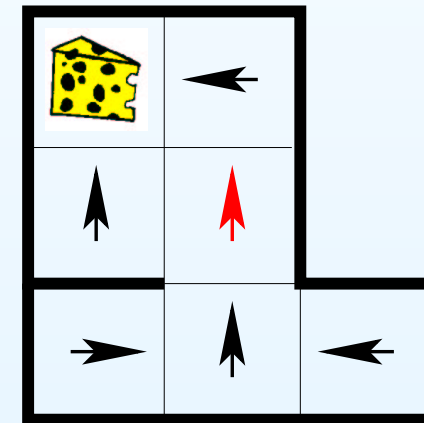
## Determining irrelevance

- Consider a rat learning a maze
  - $\vec{X}$  is the rat's **location**
  - $Y$  is irrelevant (perhaps **dosage of an experimental drug**)
- Two optimal policies for this  $\vec{x}$  **ignore**  $Y$

One optimal policy  
ignoring  $Y$



$$Y = y_1$$

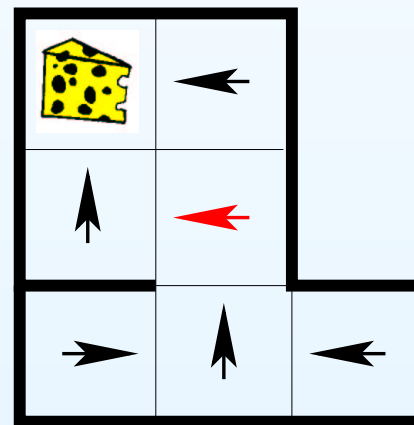


$$Y = y_2$$

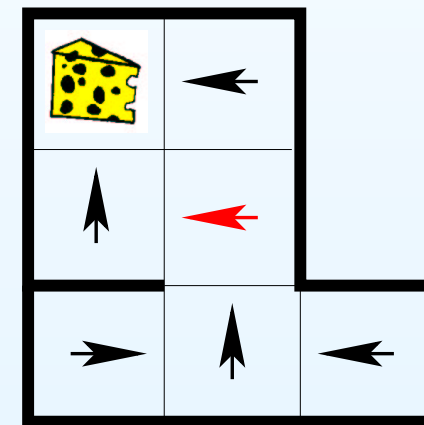
## Determining irrelevance

- Consider a rat learning a maze
  - $\vec{X}$  is the rat's **location**
  - $Y$  is irrelevant (perhaps **dosage of an experimental drug**)
- Two optimal policies for this  $\vec{x}$  **ignore**  $Y$

Another optimal policy  
ignoring  $Y$



$$Y = y_1$$

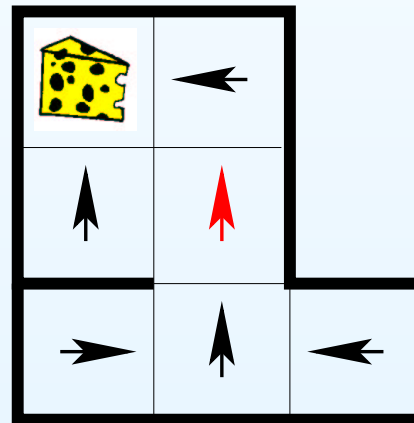


$$Y = y_2$$

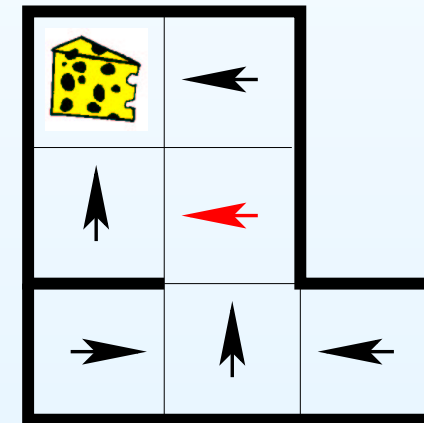
## Determining irrelevance

- Consider a rat learning a maze
  - $\vec{X}$  is the rat's **location**
  - $Y$  is irrelevant (perhaps **dosage of an experimental drug**)
- Two optimal policies for this  $\vec{x}$  **ignore**  $Y$

An optimal policy **not** ignoring  $Y$



$Y = y_1$

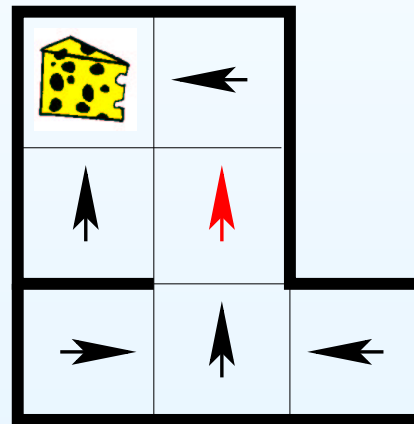


$Y = y_2$

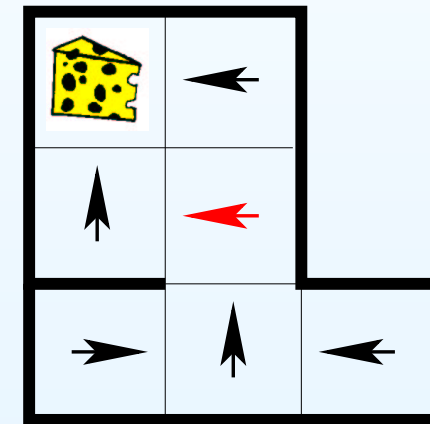
## Analyzing learned behavior

- The **learned** optimal policy may not ignore an **irrelevant** state variable

The learned policy



$Y = y_1$

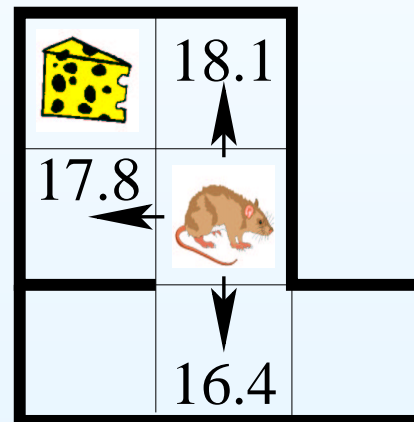


$Y = y_2$

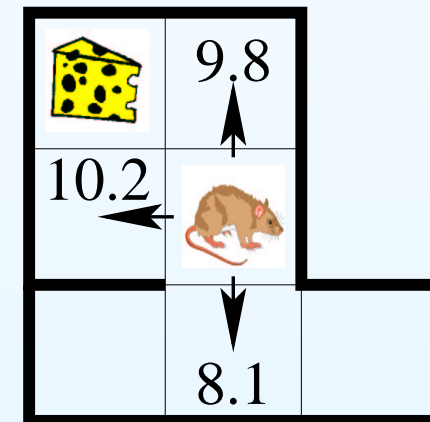
## Analyzing learned behavior

- The **learned** optimal policy may not ignore an **irrelevant** state variable
- The state-action values are only **estimates!**

The learned Q function



$$Y = y_1$$



$$Y = y_2$$

## Comparing action values

---

- Action  $a$  is **better** than action  $a'$  for value  $y$  of  $Y$  at  $\vec{x}$ .

$$Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$$

## Comparing action values

---

- Action  $a$  is **better** than action  $a'$  for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** for value  $y$  of  $Y$  at  $\vec{x}$ .

$$\forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$$

## Comparing action values

---

- Action  $a$  is **better** than action  $a'$  for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** at  $\vec{x}$ .

$$\forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$$

## Comparing action values

---

- Action  $a$  is **better** than action  $a'$  for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** at  $\vec{x}$ .
- Some action is **optimal** at  $\vec{x}$ .

$$\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$$

## Comparing action values

---

- Action  $a$  is **better** than action  $a'$  for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** for value  $y$  of  $Y$  at  $\vec{x}$ .
- Action  $a$  is **optimal** at  $\vec{x}$ .
- Some action is **optimal** at  $\vec{x}$ .
- State variable  $Y$  is **irrelevant** at  $\vec{x}$ !

$$\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$$

## Statistical hypothesis testing

---

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

## Statistical hypothesis testing

---

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

- Collect a sizable **sample** of each Q-value  $Q(\vec{x}, y, a)$

## Statistical hypothesis testing

---

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

- Collect a sizable **sample** of each Q-value  $Q(\vec{x}, y, a)$
- Statistically **test** each null hypothesis:  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

## Statistical hypothesis testing

---

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

- Collect a sizable **sample** of each Q-value  $Q(\vec{x}, y, a)$
- Statistically **test** each null hypothesis:  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$
- Call the resulting *p*-value  $p_{\vec{x}, y, a, a'}$

## Statistical hypothesis testing

---

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

- Collect a sizable **sample** of each Q-value  $Q(\vec{x}, y, a)$
- Statistically **test** each null hypothesis:  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$
- Call the resulting *p*-value  $p_{\vec{x}, y, a, a'}$
- Accept  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$  if  $p_{\vec{x}, y, a, a'}$  exceeds some threshold

## Statistical hypothesis testing

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

- Collect a sizable **sample** of each Q-value  $Q(\vec{x}, y, a)$
- Statistically **test** each null hypothesis:  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$
- Call the resulting *p*-value  $p_{\vec{x},y,a,a'}$
- Accept  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$  if  $p_{\vec{x},y,a,a'}$  exceeds some threshold
- Accept  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$  if  $\max_a \min_y \min_{a'} p_{\vec{x},y,a,a'}$  exceeds some threshold

## Statistical hypothesis testing

How to determine whether  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$

- Collect a sizable **sample** of each Q-value  $Q(\vec{x}, y, a)$
- Statistically **test** each null hypothesis:  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$
- Call the resulting *p*-value  $p_{\vec{x},y,a,a'}$
- Accept  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$  if  $p_{\vec{x},y,a,a'}$  exceeds some threshold
- Accept  $\exists a \forall y \forall a' Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$  if  $\max_a \min_y \min_{a'} p_{\vec{x},y,a,a'}$  exceeds some threshold

Computationally efficient but requires a lot of data!



## A model-based alternative

---

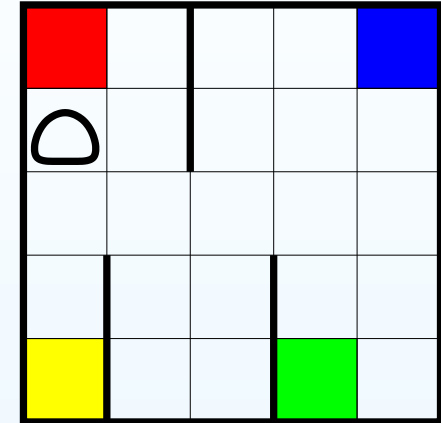
- Build a **Bayesian model** of the underlying MDP using all the data
  - Assume each state-action pair **independent**
  - For each state-action pair, maintain a **belief state** over the parameters of the successor state distribution
  - **Condition** each belief state on the data
  - Obtain a **joint distribution** over MDP parameters from the belief states
- Estimate  $\Pr(Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a'))$  using **Monte Carlo simulation**

## A model-based alternative

- Build a **Bayesian model** of the underlying MDP using all the data
  - Assume each state-action pair **independent**
  - For each state-action pair, maintain a **belief state** over the parameters of the successor state distribution
  - **Condition** each belief state on the data
  - Obtain a **joint distribution** over MDP parameters from the belief states
- Estimate  $\Pr(Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a'))$  using **Monte Carlo simulation**
  - **Generate** random MDPs from the Bayesian model
  - **Solve** each generated MDP
  - Count the **fraction** of MDPs for which  $Q(\vec{x}, y, a) \geq Q(\vec{x}, y, a')$
  - Call this value  $p_{\vec{x}, y, a, a'}$

## A familiar test domain

- Four state variables
  - Taxi  $x$  coordinate
  - Taxi  $y$  coordinate
  - Current passenger location
  - Passenger destination
- Six actions: North, South, East, West, Pick Up, Put Down
- Optimal policy:
  - Navigate to the passenger's location
  - Pick up the passenger
  - Navigate to the passenger's destination
  - Put down the passenger



## Irrelevancy in the Taxi domain

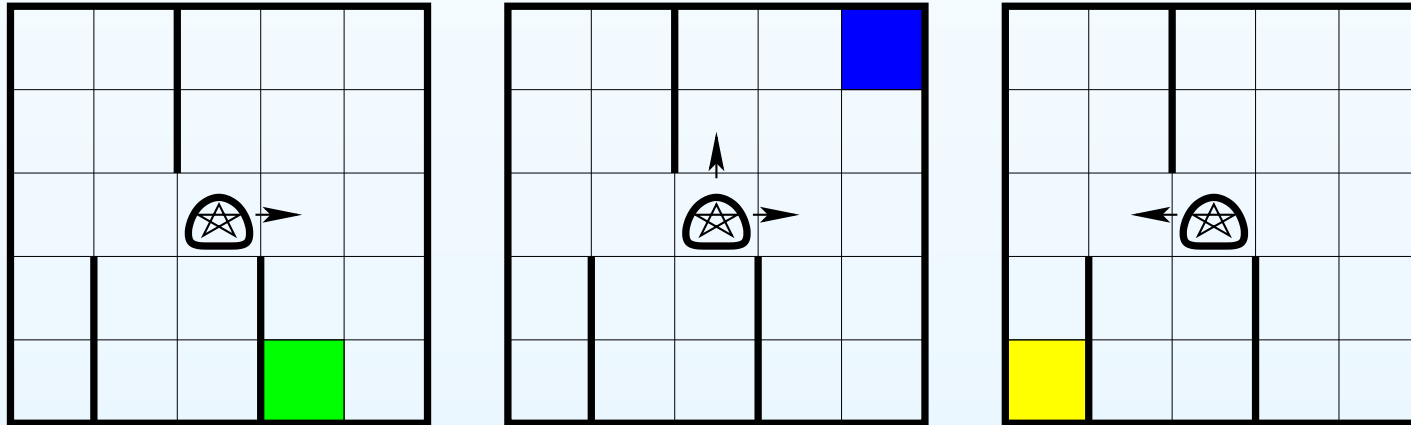
---

Relevance of the **passenger destination**...

## Irrelevancy in the Taxi domain

Relevance of the **passenger destination**...

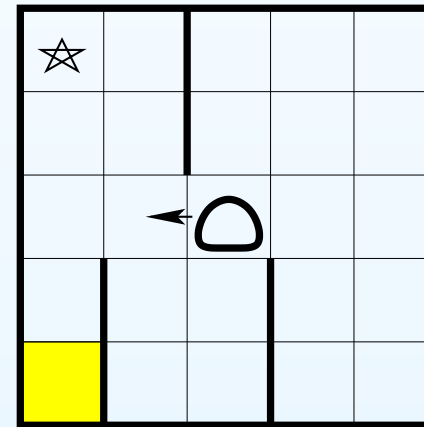
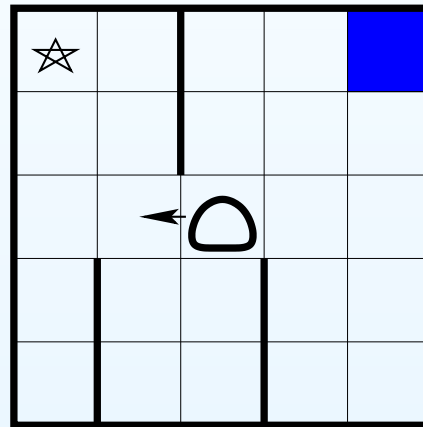
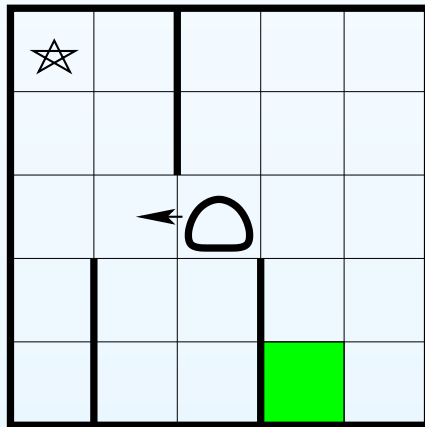
- When the passenger is **inside** the taxi



## Irrelevancy in the Taxi domain

Relevance of the **passenger destination**...

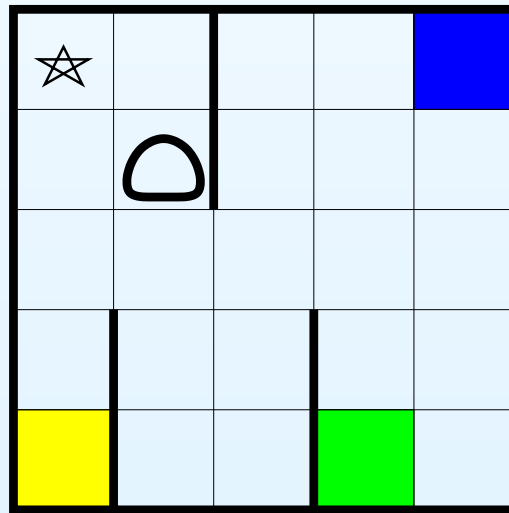
- When the passenger is **inside** the taxi
- When the passenger is **not** inside the taxi



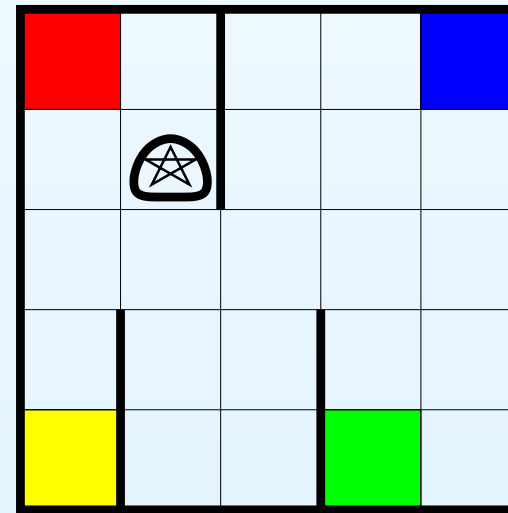
## Typical results

- Exploration data from one run of **Prioritized Sweeping**
- 100 MDPs sampled from the learned Bayesian model and solved using **value iteration**
- $Y =$  **passenger destination**

passenger at red landmark



passenger in taxi

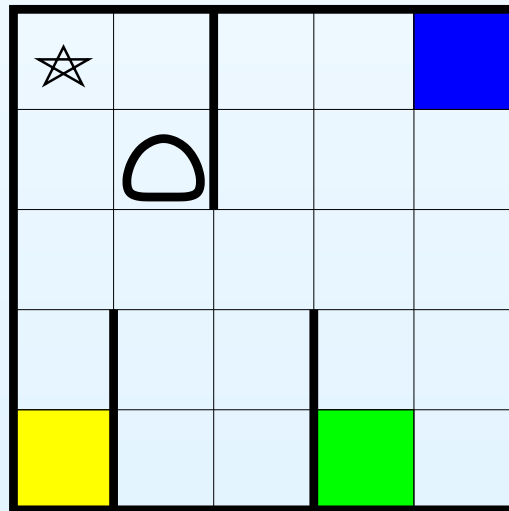


## Typical results

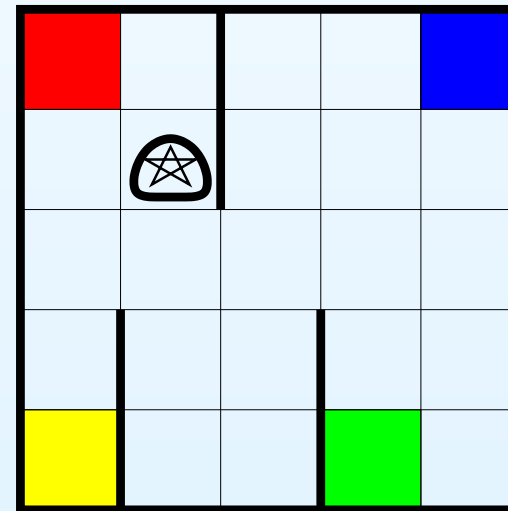
- Exploration data from one run of **Prioritized Sweeping**
- 100 MDPs sampled from the learned Bayesian model and solved using **value iteration**
- $Y =$  **passenger destination**

$$\max_a \min_y \min_{a'} P_{\vec{x}, y, a, a'}$$

passenger at red landmark



passenger in taxi

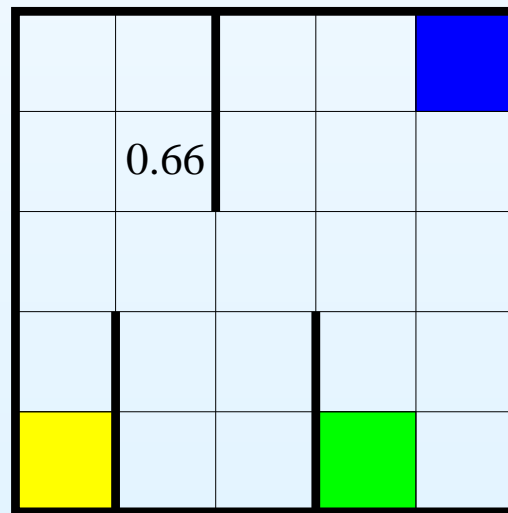


## Typical results

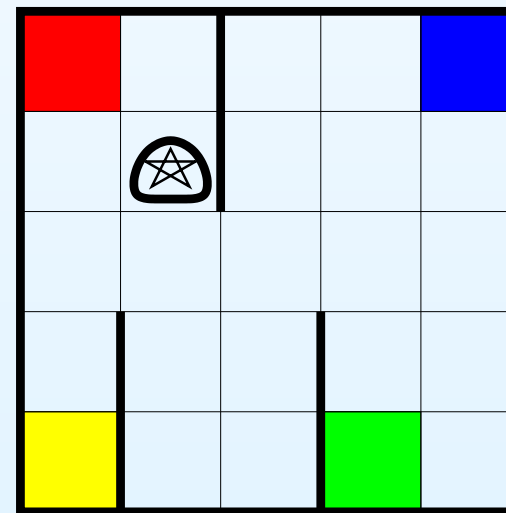
- Exploration data from one run of **Prioritized Sweeping**
- 100 MDPs sampled from the learned Bayesian model and solved using **value iteration**
- $Y =$  **passenger destination**

$$\max_a \min_y \min_{a'} P_{\vec{x}, y, a, a'}$$

passenger at red landmark



passenger in taxi

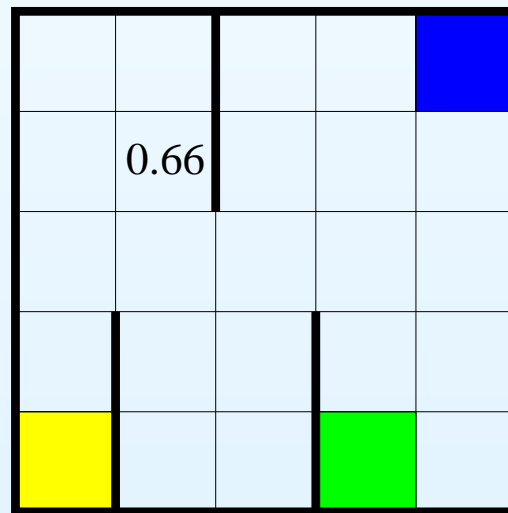


## Typical results

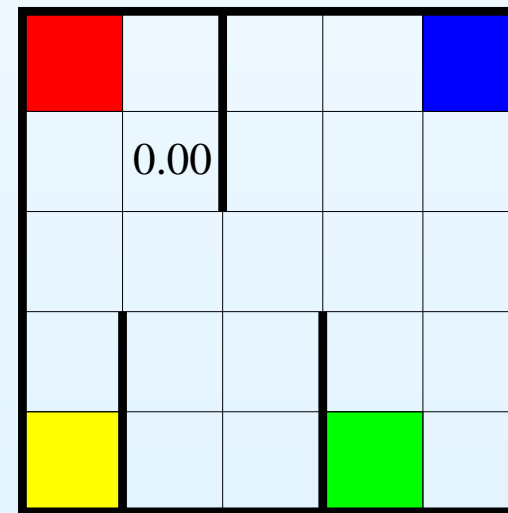
- Exploration data from one run of **Prioritized Sweeping**
- 100 MDPs sampled from the learned Bayesian model and solved using **value iteration**
- $Y =$  **passenger destination**

$$\max_a \min_y \min_{a'} P_{\vec{x}, y, a, a'}$$

passenger at red landmark



passenger in taxi



## Typical results

- Exploration data from one run of **Prioritized Sweeping**
- 100 MDPs sampled from the learned Bayesian model and solved using **value iteration**
- $Y =$  **passenger destination**
- Each cell contains  $\max_a \min_y \min_{a'} P_{\vec{x},y,a,a'}$

passenger at red landmark

1.00	1.00	1.00	0.20	0.77
1.00	0.66	1.00	0.21	0.47
0.99	0.81	1.00	1.00	1.00
1.00	1.00	0.34	1.00	0.67
1.00	1.00	0.28	0.99	0.56

passenger in taxi

0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00
0.00	0.23	0.74	0.00	0.00
0.00	0.79	0.64	0.00	0.00

## Applying learned state abstractions

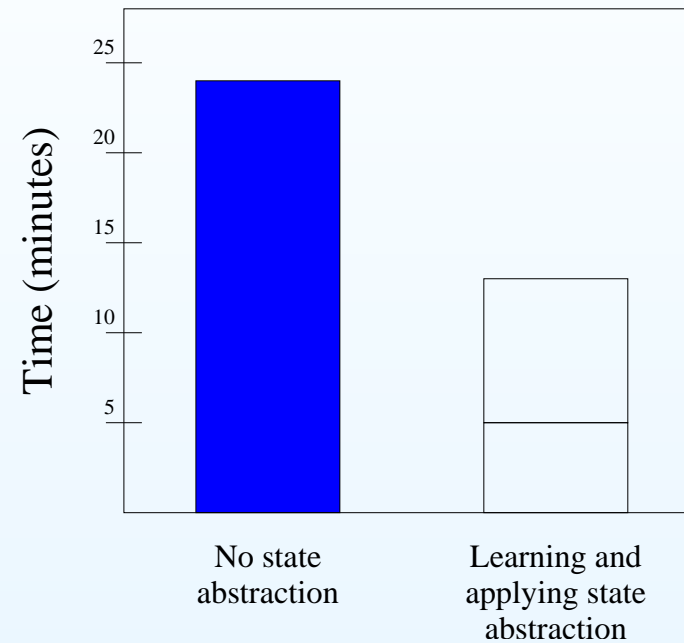
---

Solving **randomly generated**  $15 \times 15$  Taxi domains

# Applying learned state abstractions

Solving randomly generated  $15 \times 15$  Taxi domains

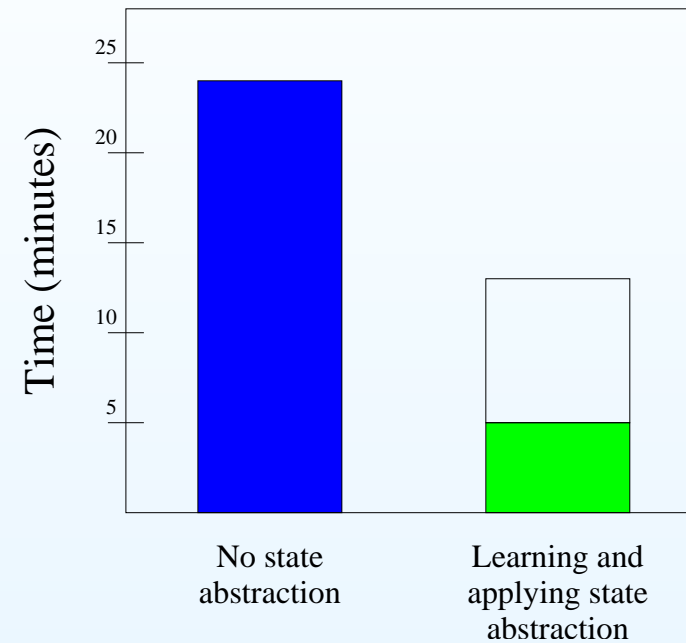
- Prioritized sweeping on original state space: 24 minutes (330000 steps)



# Applying learned state abstractions

## Solving randomly generated $15 \times 15$ Taxi domains

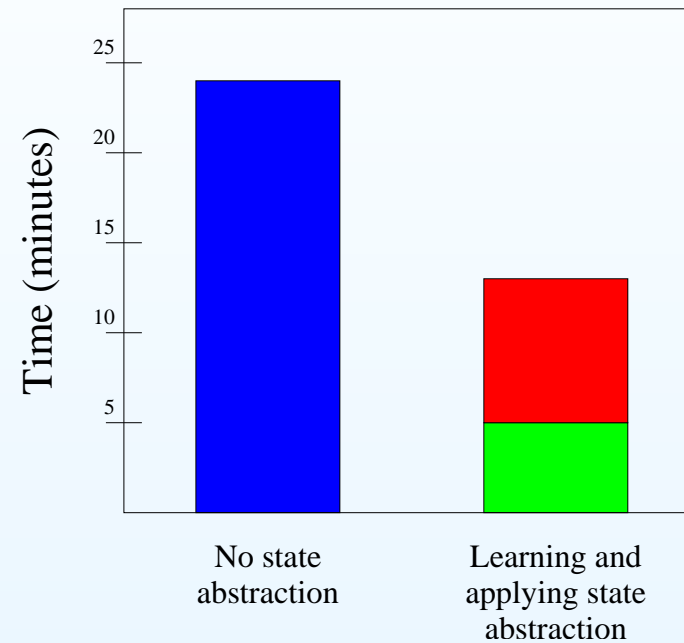
- Prioritized sweeping on original state space: 24 minutes (330000 steps)
- Prioritized sweeping in abstracted state space:
  - Monte Carlo simulation: 5 minutes



# Applying learned state abstractions

## Solving randomly generated $15 \times 15$ Taxi domains

- Prioritized sweeping on original state space: 24 minutes (330000 steps)
- Prioritized sweeping in abstracted state space: 12.5 minutes (200000 steps total)
  - Monte Carlo simulation: 5 minutes
  - Solving large domain with abstract state space: 7 minutes



## Summary

---

- We can transform the question of state abstraction discovery into one of action value comparisons

## Summary

---

- We can transform the question of state abstraction discovery into one of action value comparisons
- We can achieve robust action value comparisons by building Bayesian models and applying a Monte Carlo approach

## Summary

---

- We can transform the question of state abstraction discovery into one of action value comparisons
- We can achieve robust action value comparisons by building Bayesian models and applying a Monte Carlo approach
- Qualitative knowledge learned from solving an easy task can speed learning for a more difficult but related task

## Related work

---

### Previous approaches to automated state abstraction

- Derivation from other domain knowledge
  - Structured policy iteration (Boutilier, 1995)
- U-Tree algorithm (McCallum, 1995)
  - Uses statistical tests to build a state representation
  - Top down approach
  - No convergence guarantees

## Future work

---

- How do we know **what state variables to test** for relevance under what conditions?
- How can we apply this approach **before converging** to an optimal policy?
- What threshold **minimizes** the number of false positives and negatives?
- How can we **generalize** to other value function representations?
- In what ways can we **apply** the learned abstractions?