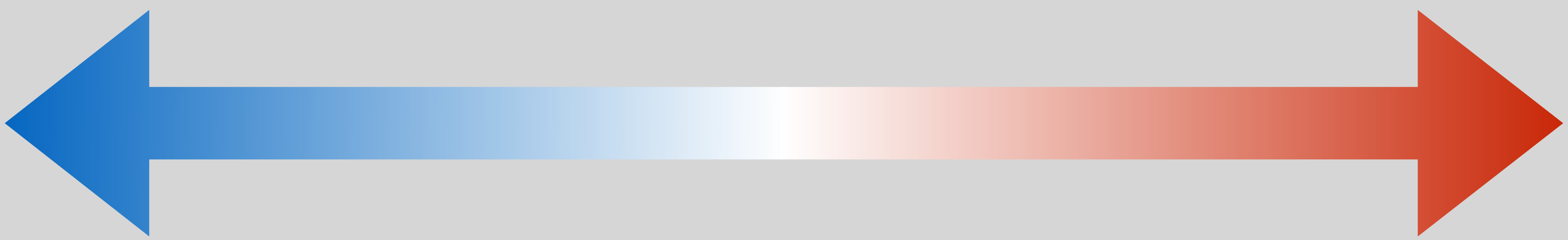
Escaping Groundhog Day

James MacGlashan
Stefanie Tellex
Michael Littman

BROWN

Key Message



Ground

everything is constant

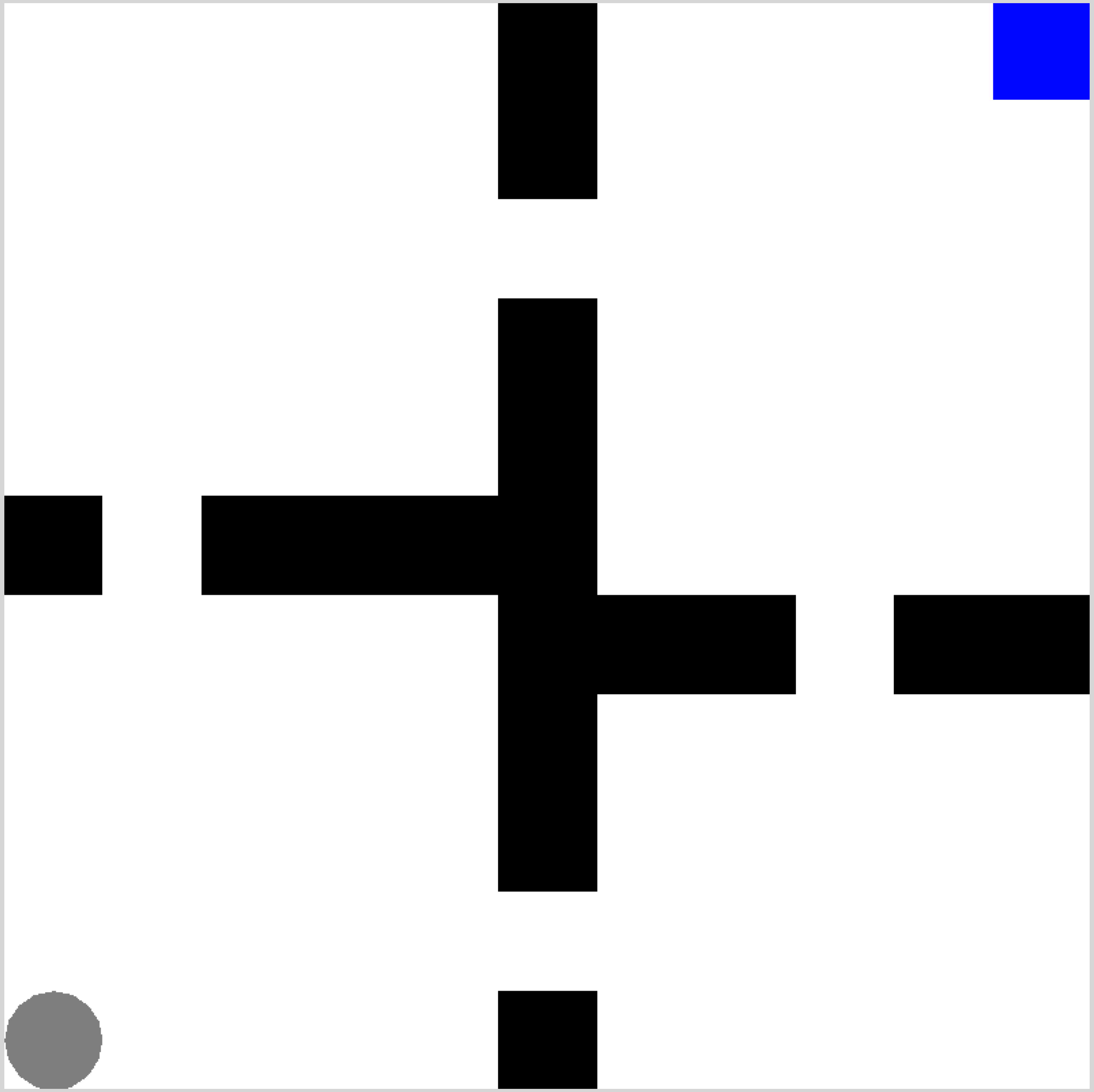
Problem Generators

some constants, some variables

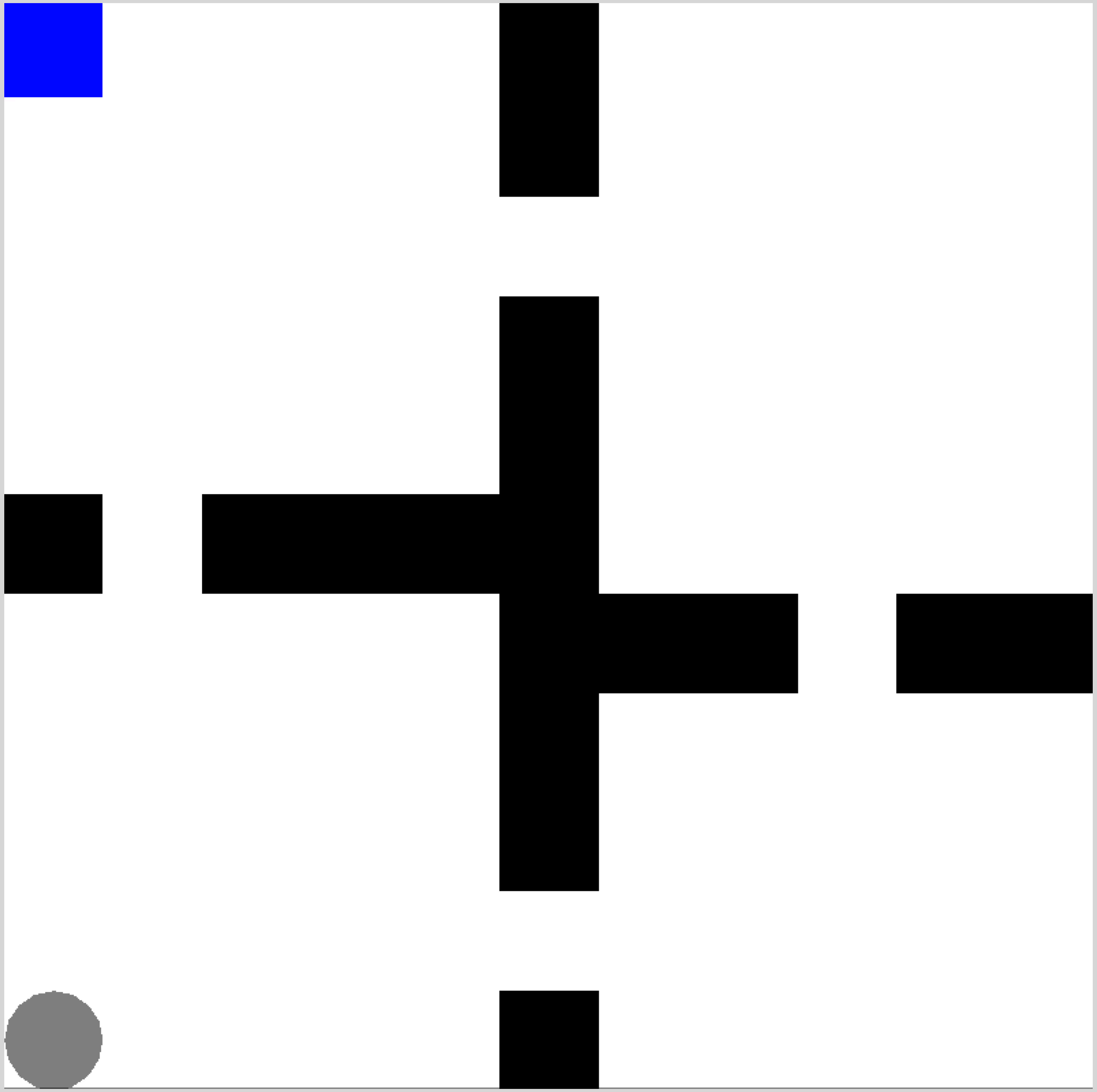
Figure

everything varies

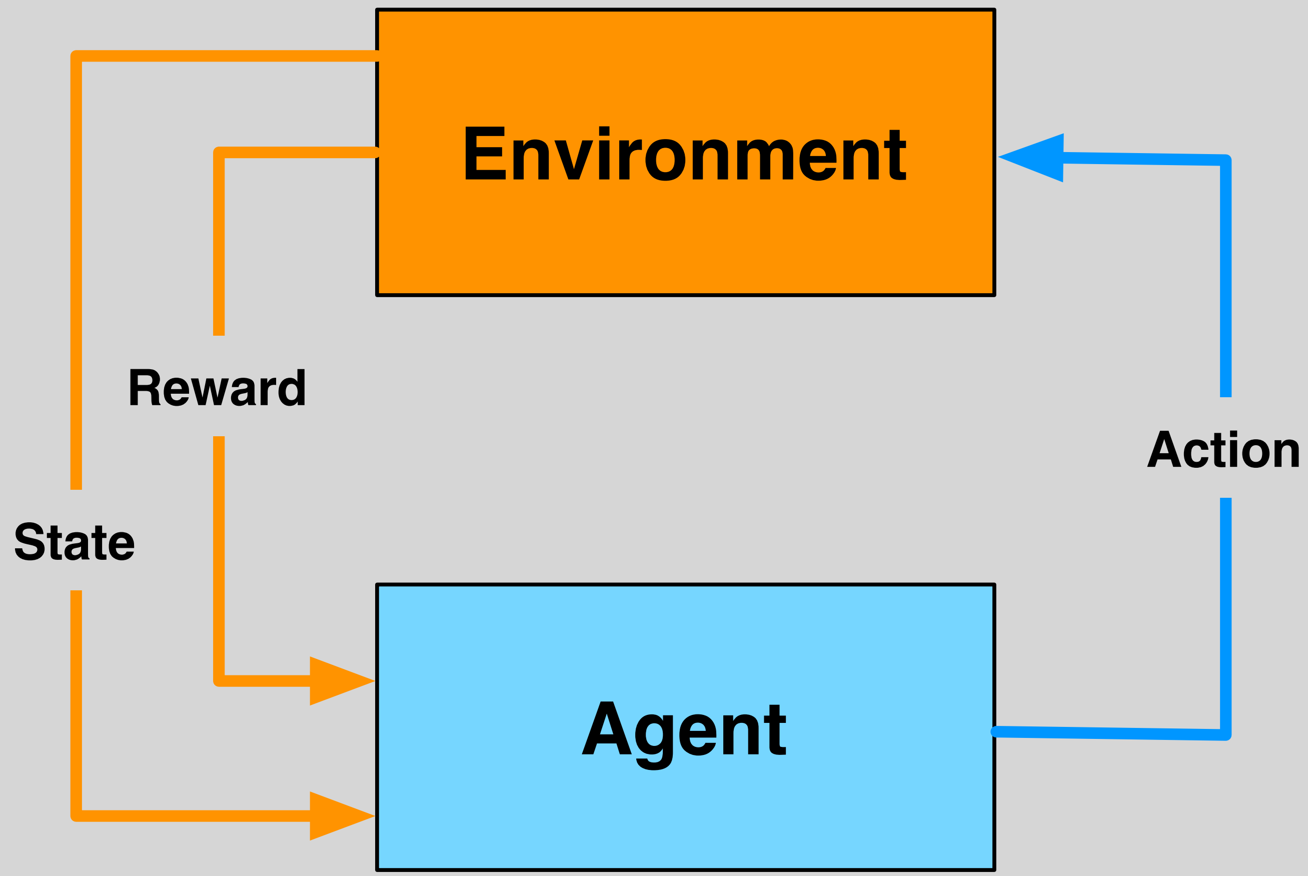
Reinforcement Learning



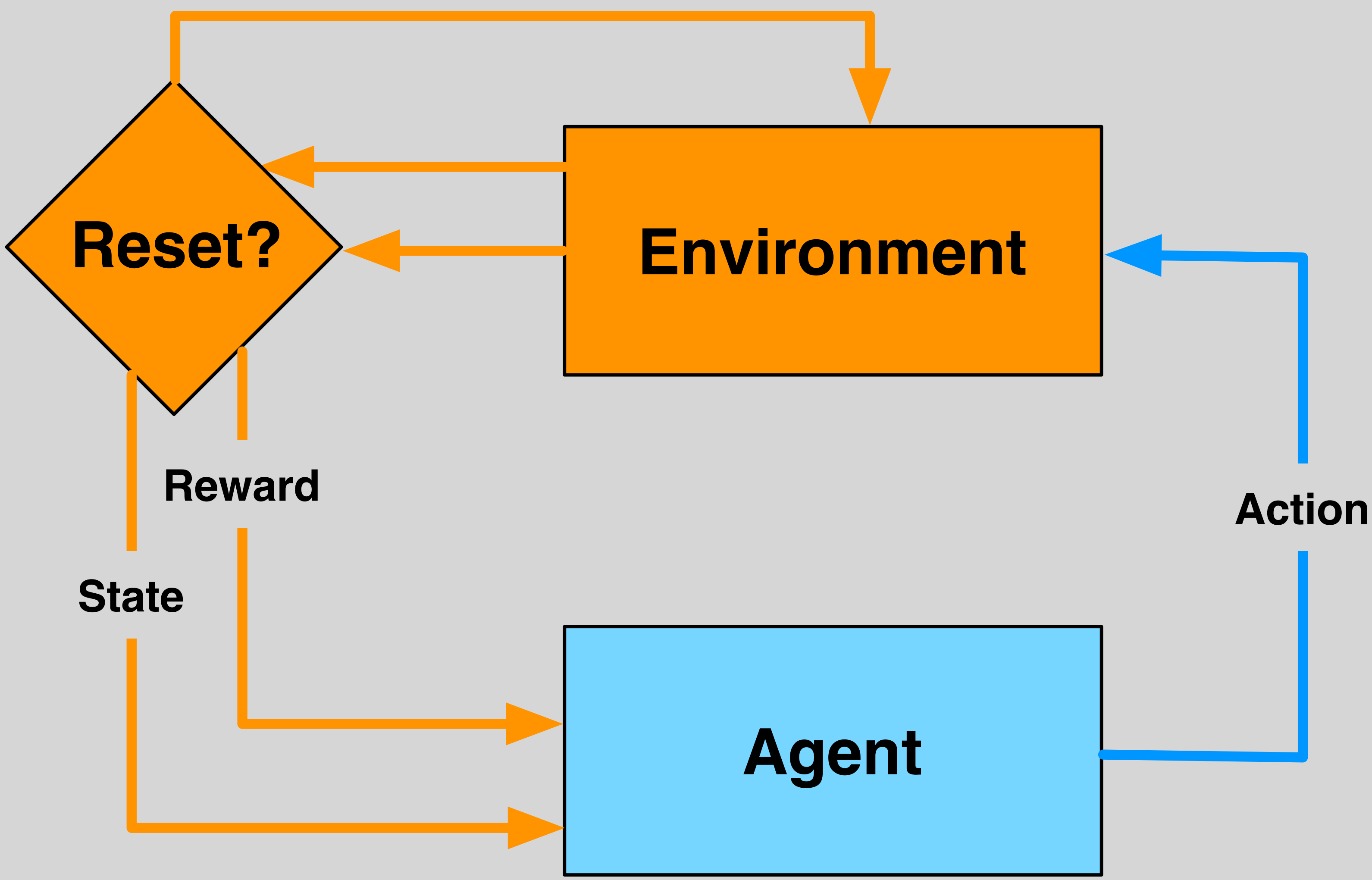
What if something changes?



Groundhog Day Assumptions

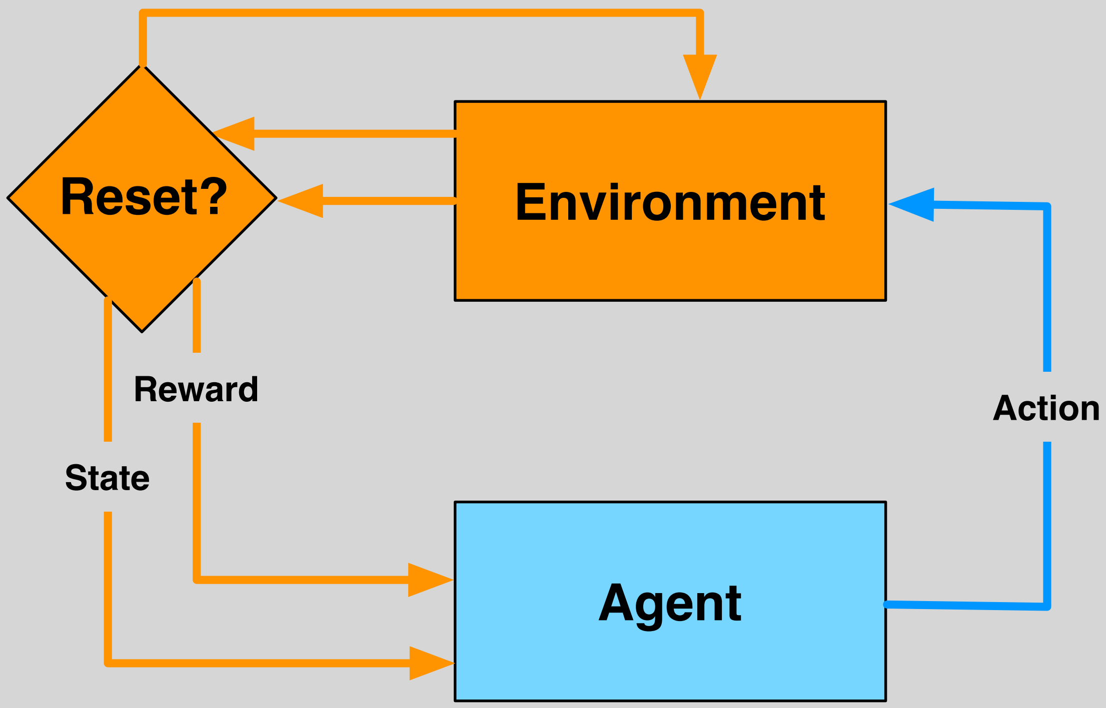


Groundhog Day Assumptions



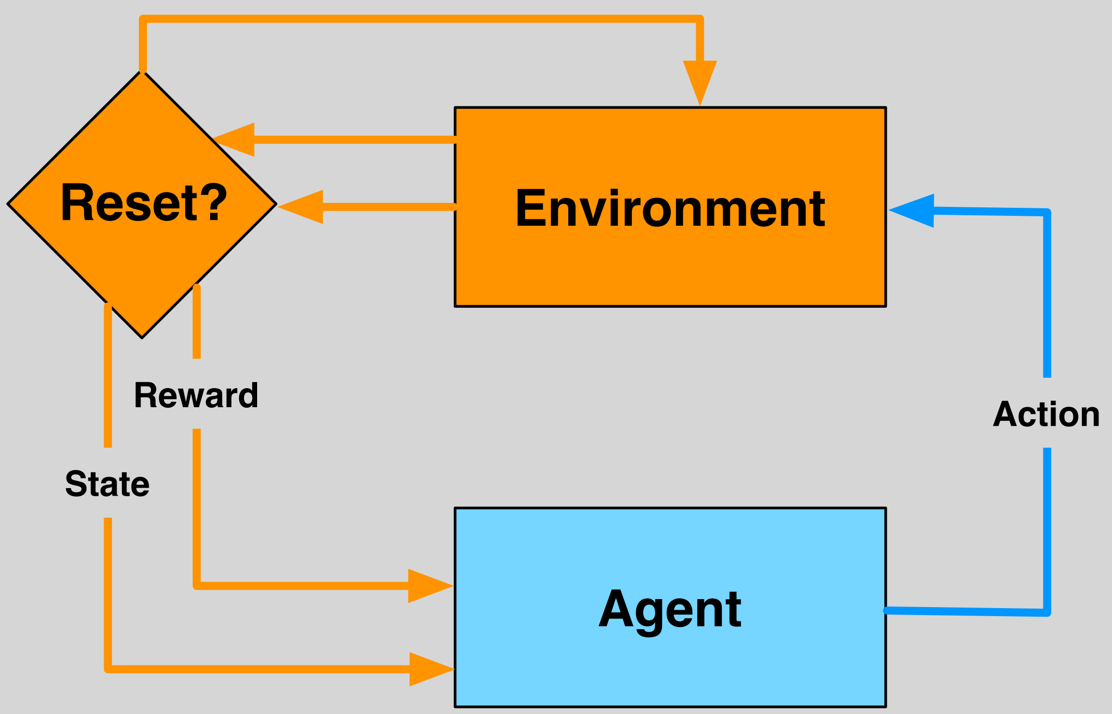
Groundhog Day Assumptions

- Reward function is always the same
- State resets indefinitely
- Resets to states similar to those visited

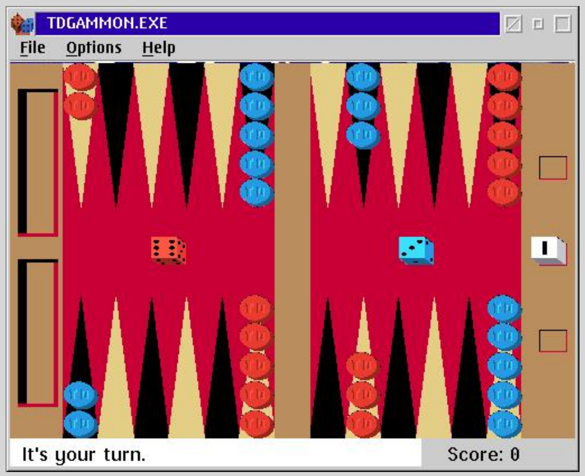


Groundhog Day Assumptions

- Reward function is always the same
- Resets indefinitely
- Resets to states similar to those visited
- Enables **hyper optimization**



Groundhog Day Successes



Tesauro, 1995

Crites and Barto, 1996

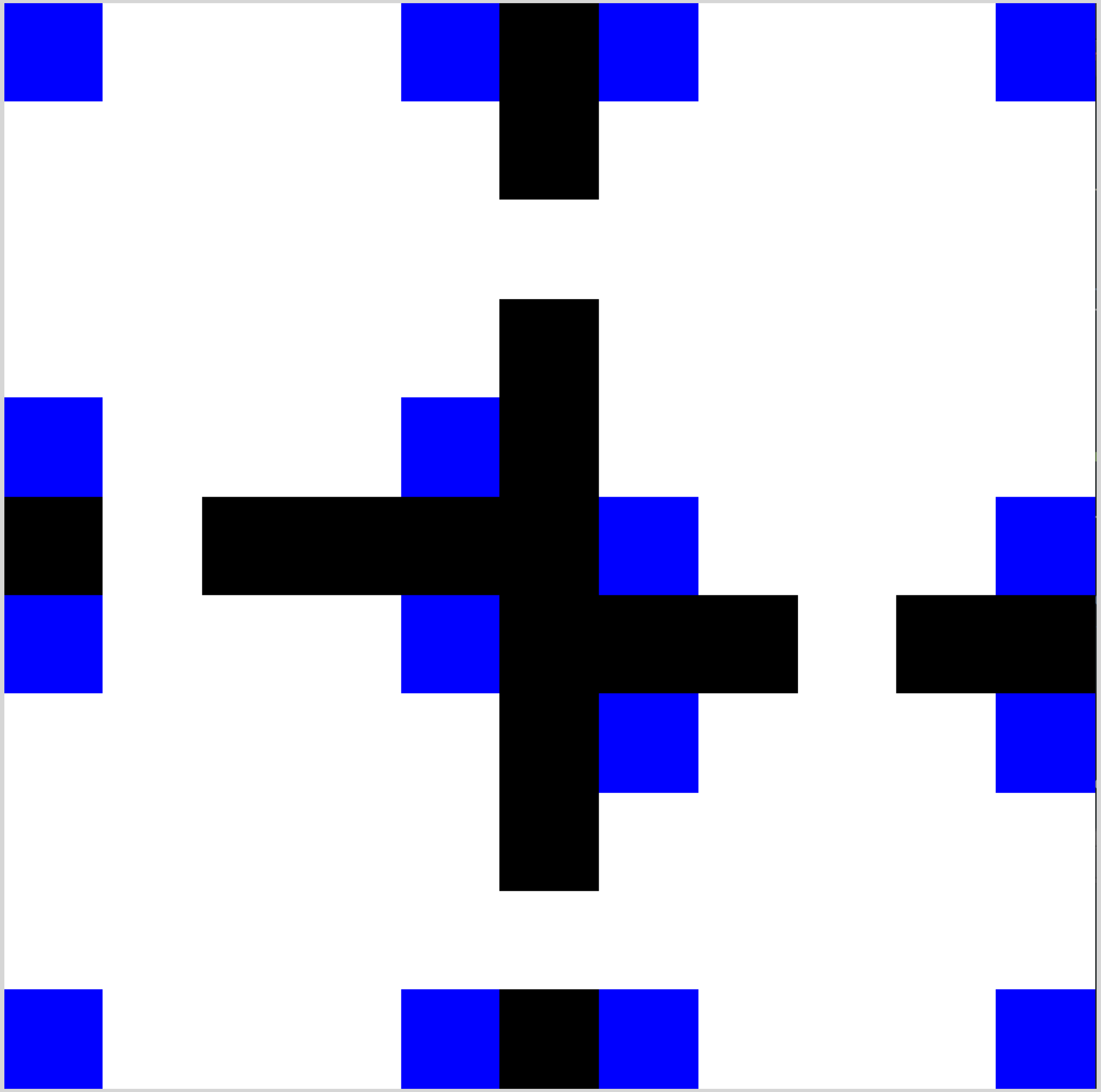
Singh and Bertsekas, 1997

Ng et al., 2004

Peters and Schaal, 2007

Mnih et al. 2015

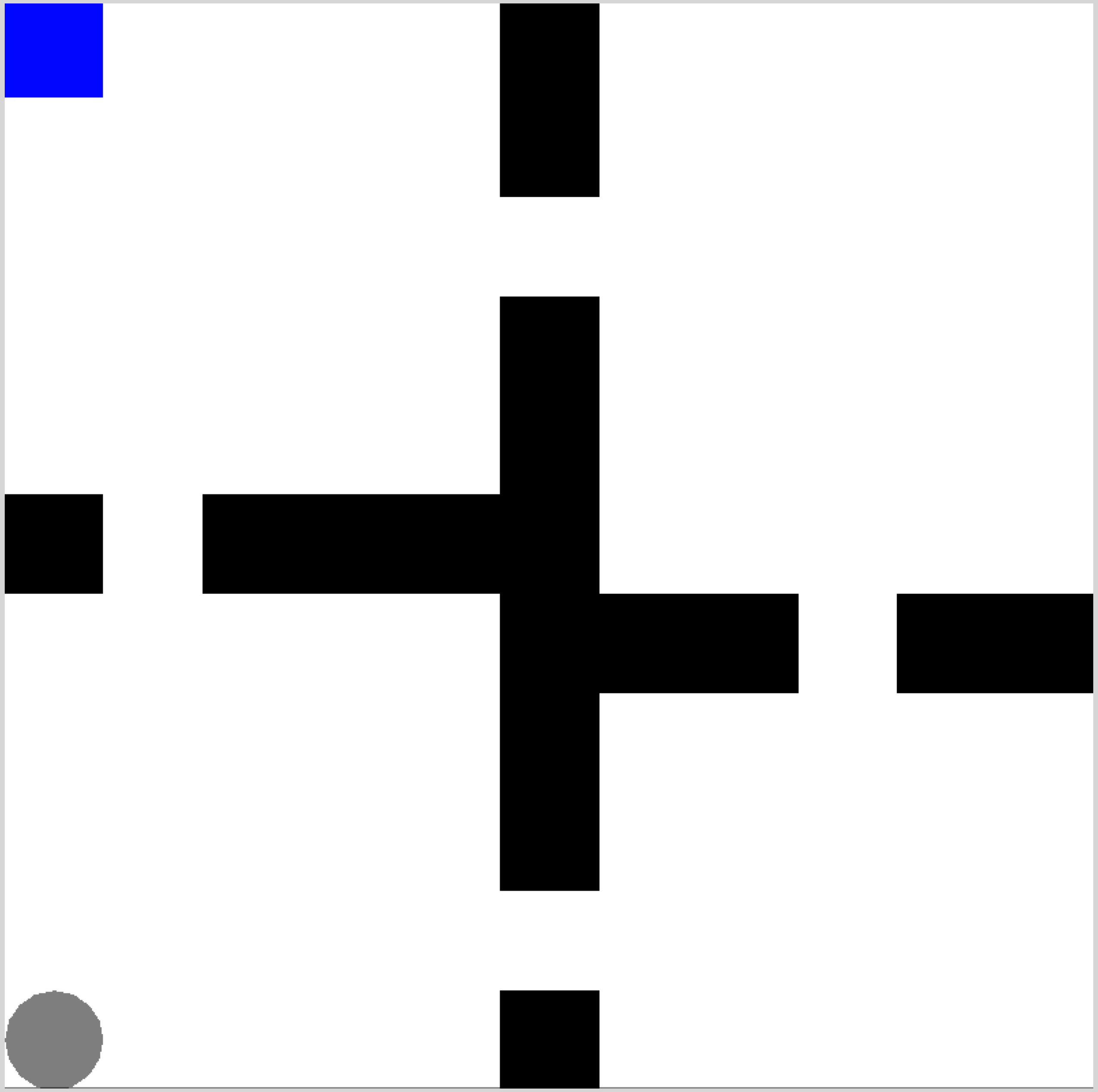
Between Ground and Figure



Learn what changes

- RMax learns the transitions and keeps them

- Subsequently only learns about goal

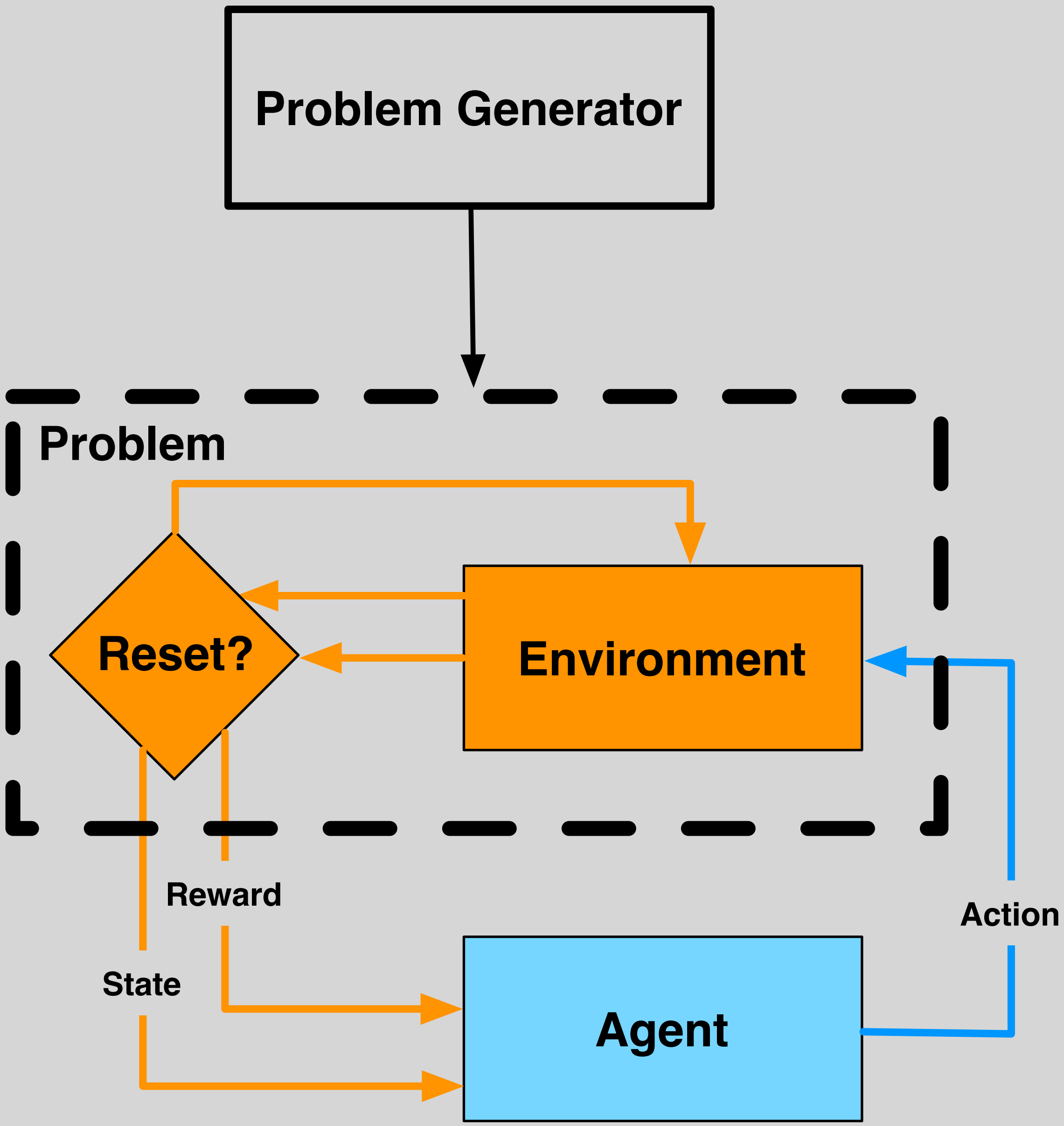


Escaping Groundhog Day

- Relax groundhog day assumptions
- Investigate benchmark problems generators
- Develop appropriate learning machinery

Escaping Assumptions

- Problem generator that can affect
  - initial state distribution
  - reward function/goal
  - action model
- After learning, a new problem is generated
- Some things remain the same, some vary
- Learn to behave across distribution



Related Areas

- Learning hierarchical actions
- Transfer learning
- Bayesian RL

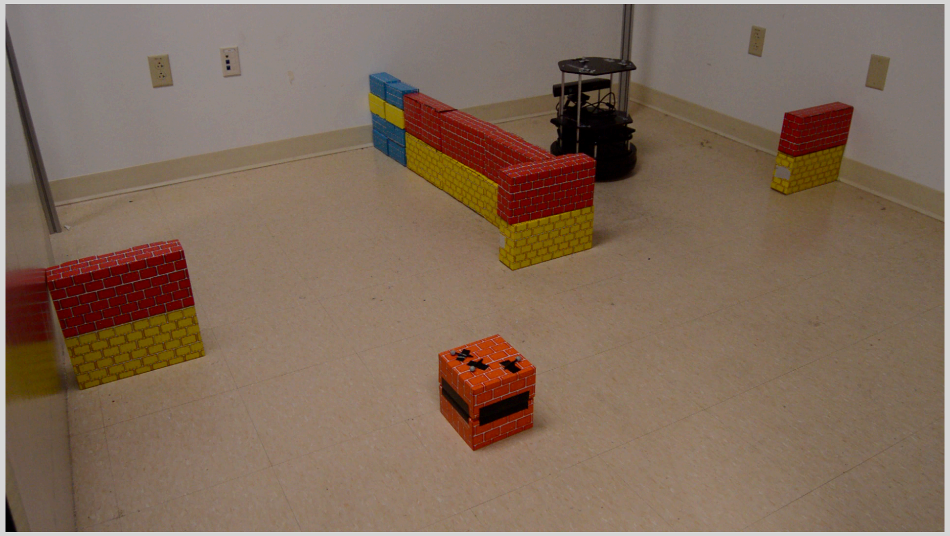
Problem Generators

Robotics

Minecraft

Robotics

- The real world is complex
- Easy to create variation in the environment
- Examine tasks other than motion controllers



Minecraft

- Can expand to very large worlds
- Turing complete complexity
- Safe; no hardware failures
- Many possible goals
- Very easy to manipulate

Reasoning with a Problem Generator

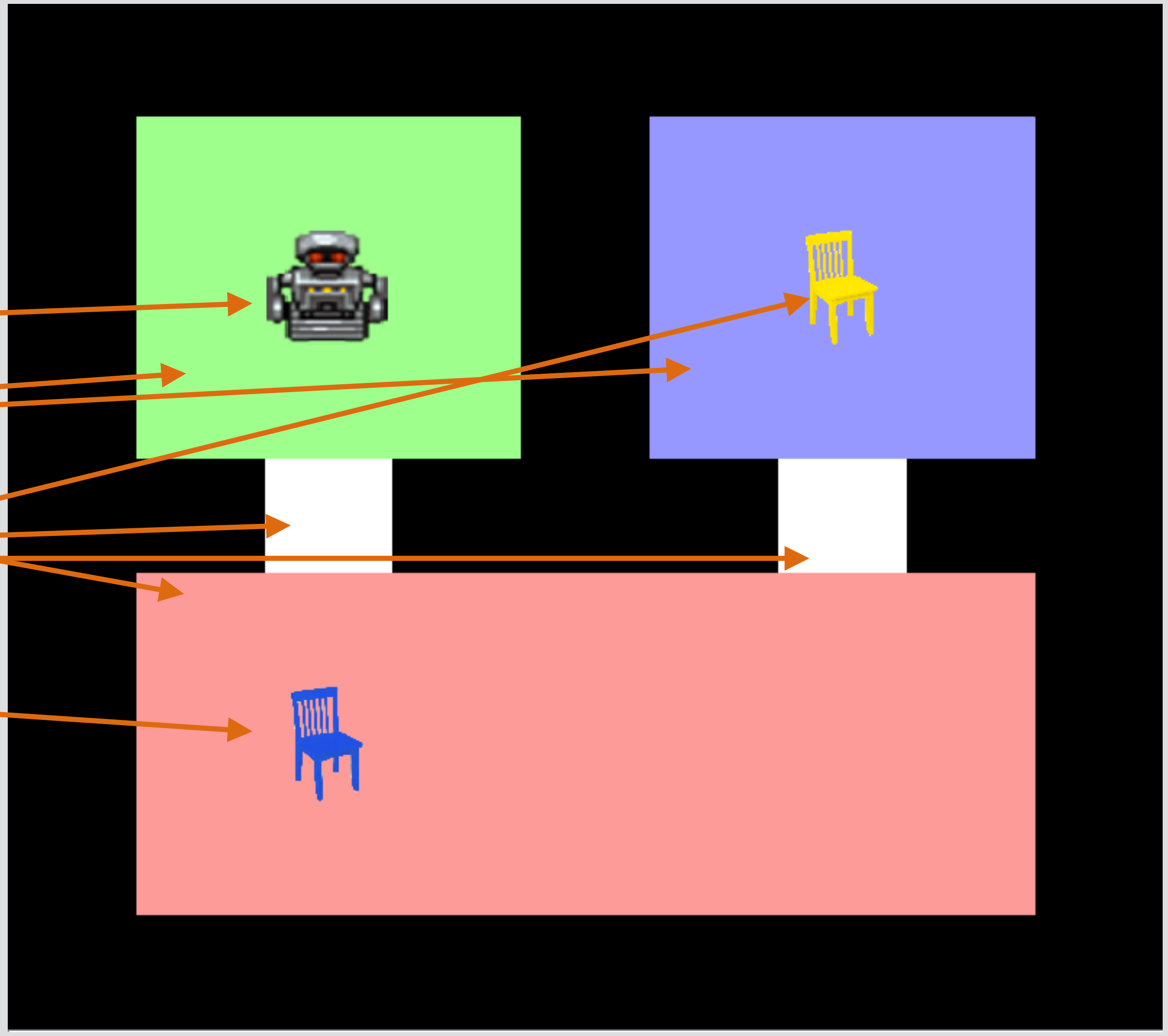
- Need mechanisms to generalize knowledge across problems
  - Requires reasoning about the state
- Some existing approaches
  - Agent space features (Konidaris and Barto, 2007)
  - Intertask mappings (Taylor, Stone, and Liu, 2007)
  - Horde (Sutton, Modayil, Delp, Degris, Pilarski, White, and Precup, 2011)
- We will highlight **Object-oriented MDPs** (Diuk, Cohen, and Littman, 2008)
  - Works well for robotics environments and Minecraft

OO-MDPs

(Diuk, Cohen, Littman, 2008)

World consists of objects that belong to classes

- robot
- room
- door
- block

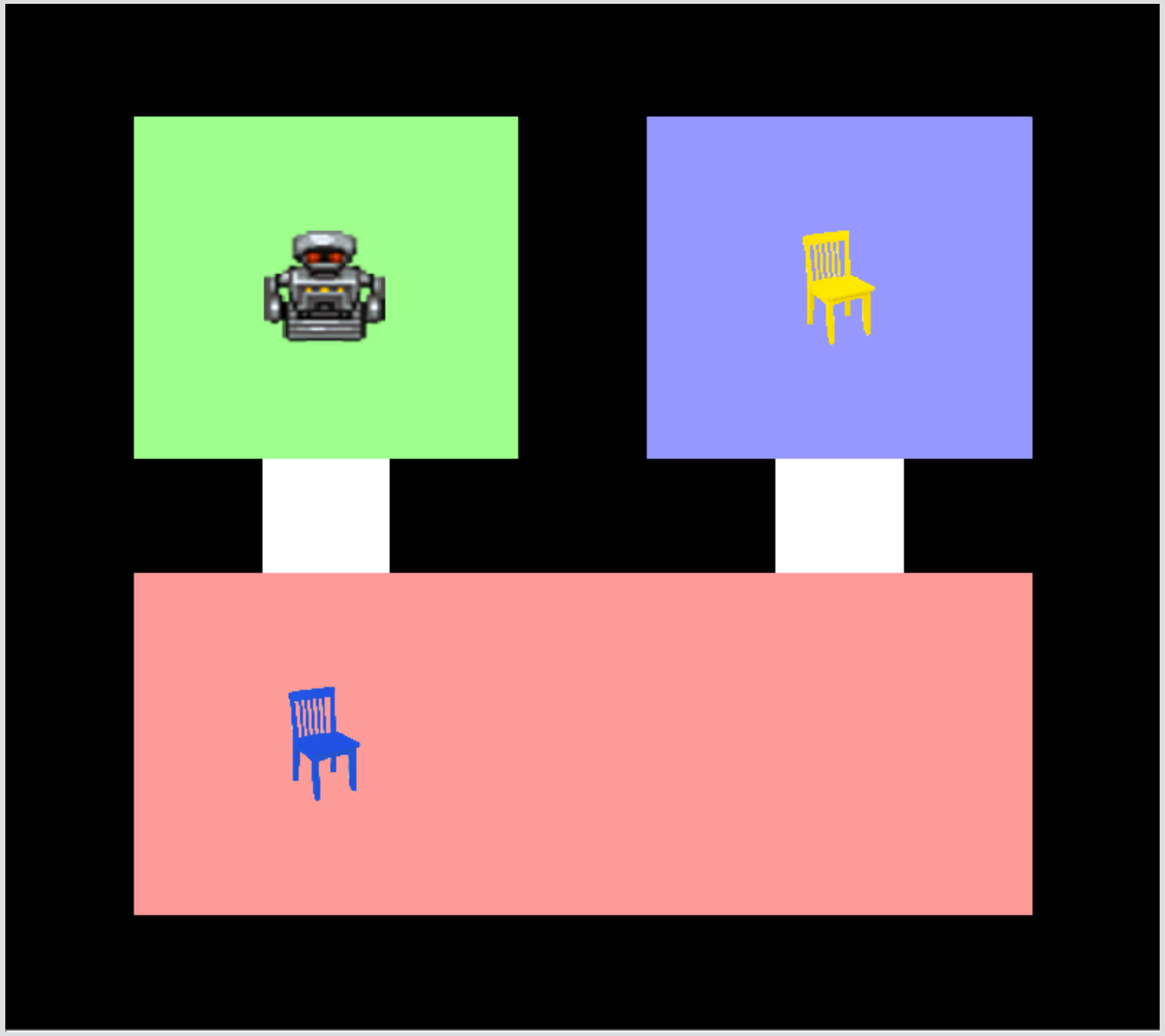


OO-MDPs

(Diuk, Cohen, Littman, 2008)

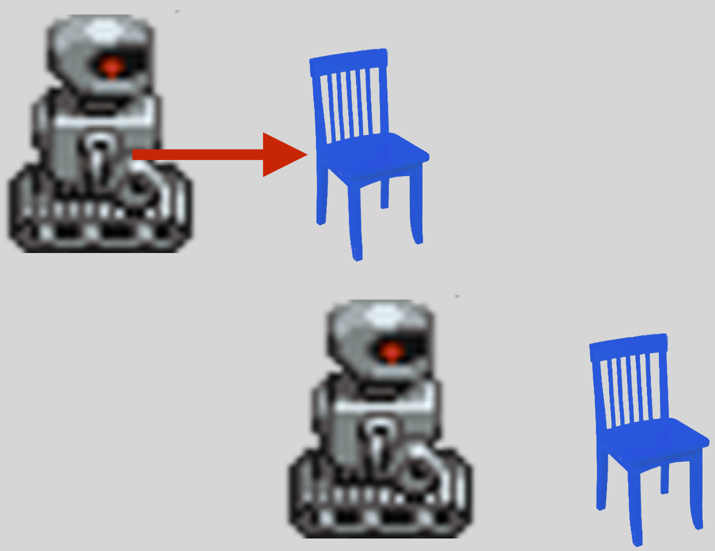
Each object has a value assignment to its attributes

- robot0
  (x,y) := (2,6)
- block1
  (x,y,color,shape) := (2,3,blue,chair)
- etc.



OO-MDP Generalization

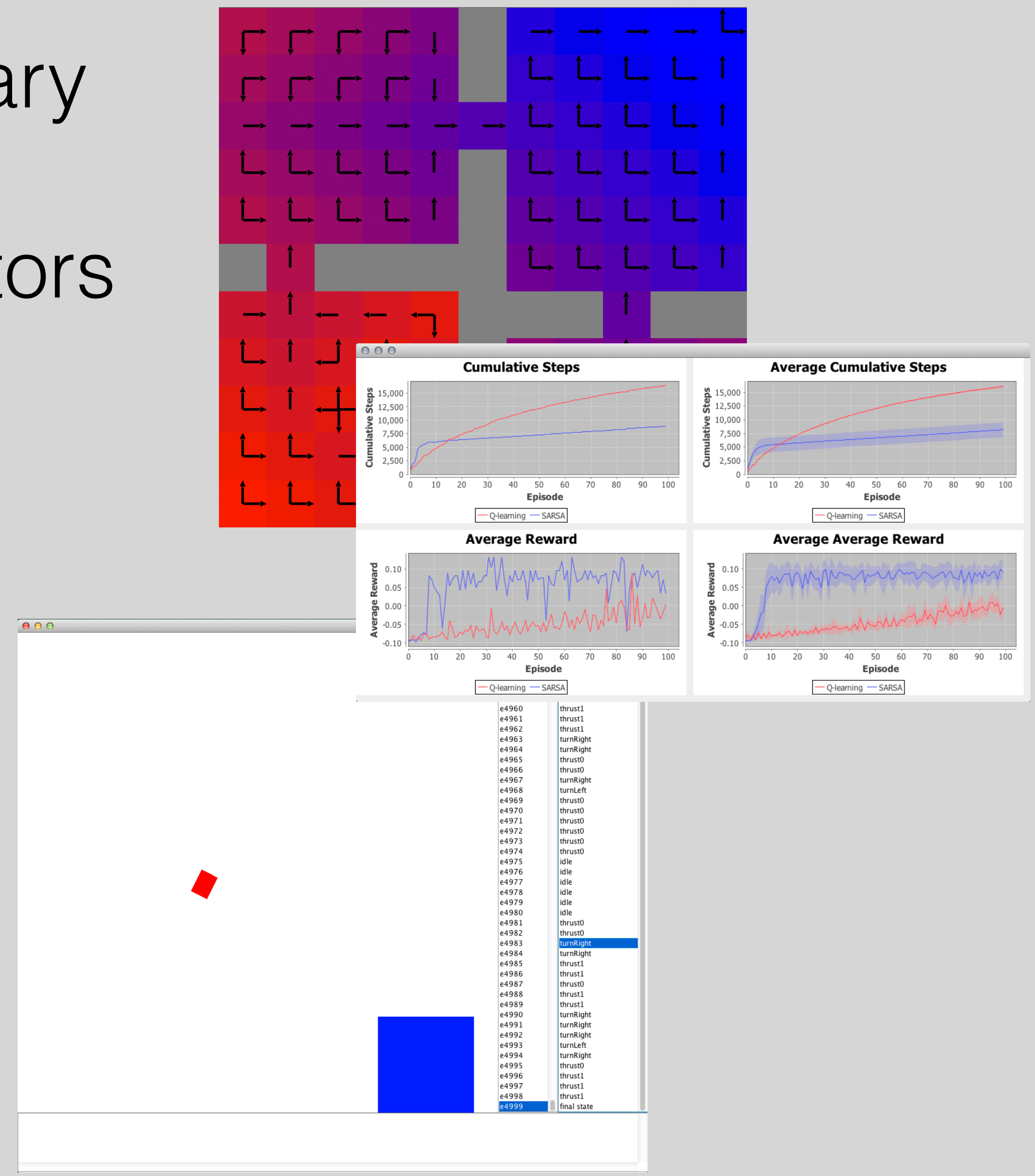
- Transition dynamics factored by objects
- DOORMax (Diuk, Cohen, and Littman, 2008)
- Physics based Prior (Scholz, Levihn, Isbell, and Wingate, 2014)



BURLAP

http://burlap.cs.brown.edu

- Java RL and Planning Library
- Problem and State Generators
- OO-MDP Representation
- ROS interface
- Minecraft interface
  github.com/h2r/burlapcraft

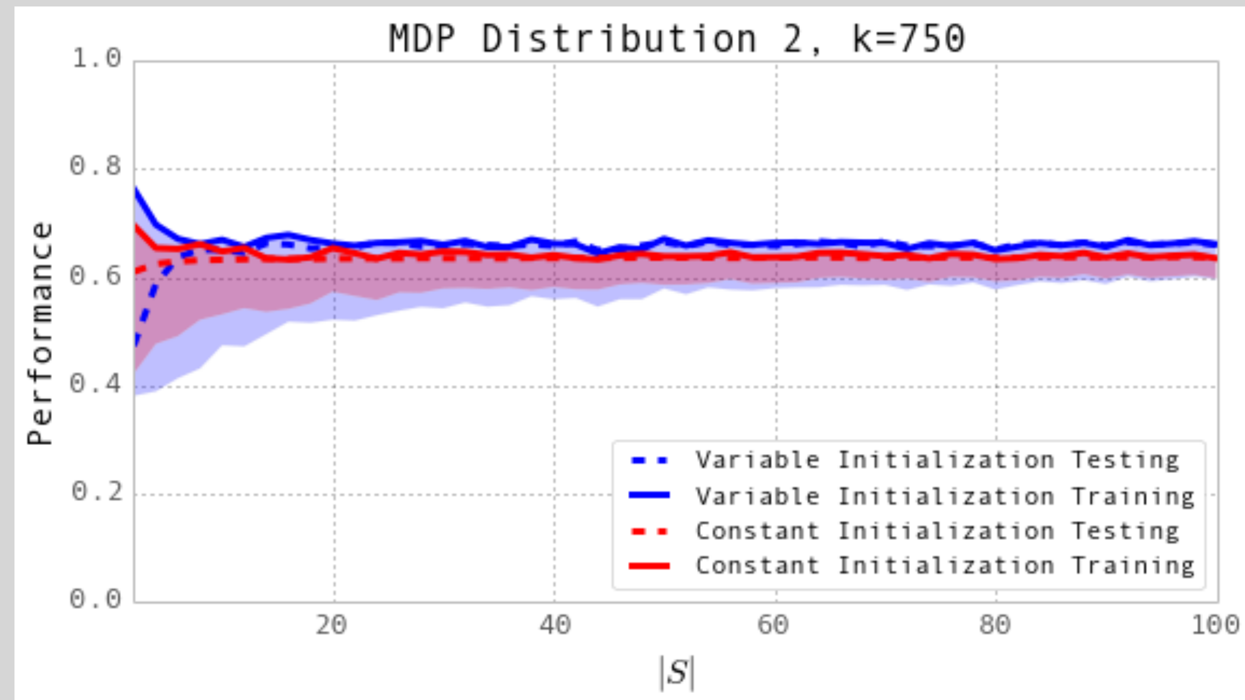


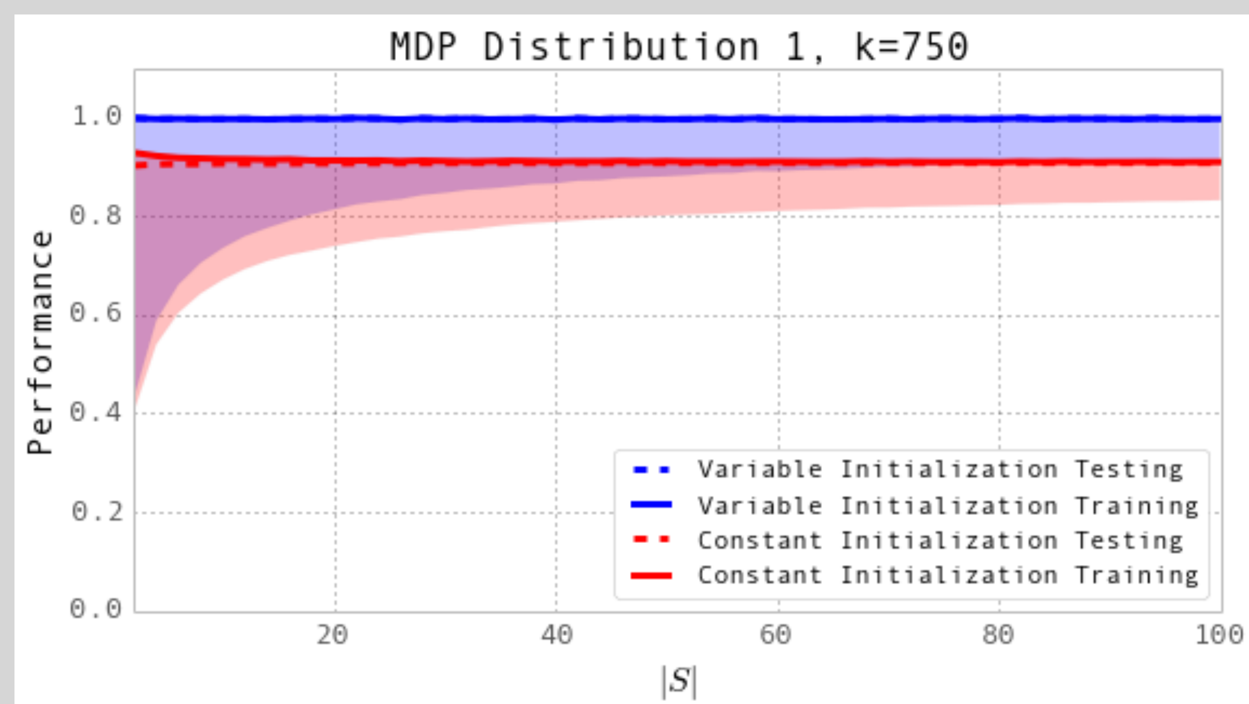
What we can learn

- World physics
- Learning to learn
- Learning to plan
- Task decomposition and representation
- Learning about natural language

Learning to Learn

- Given two parameterized algorithms, which do we use?
- For single problem tune each and extract policy
- For problem distribution, need to worry about over and under fitting
- Compute theoretical generalization bounds
- Works with weak parameter optimization and samples

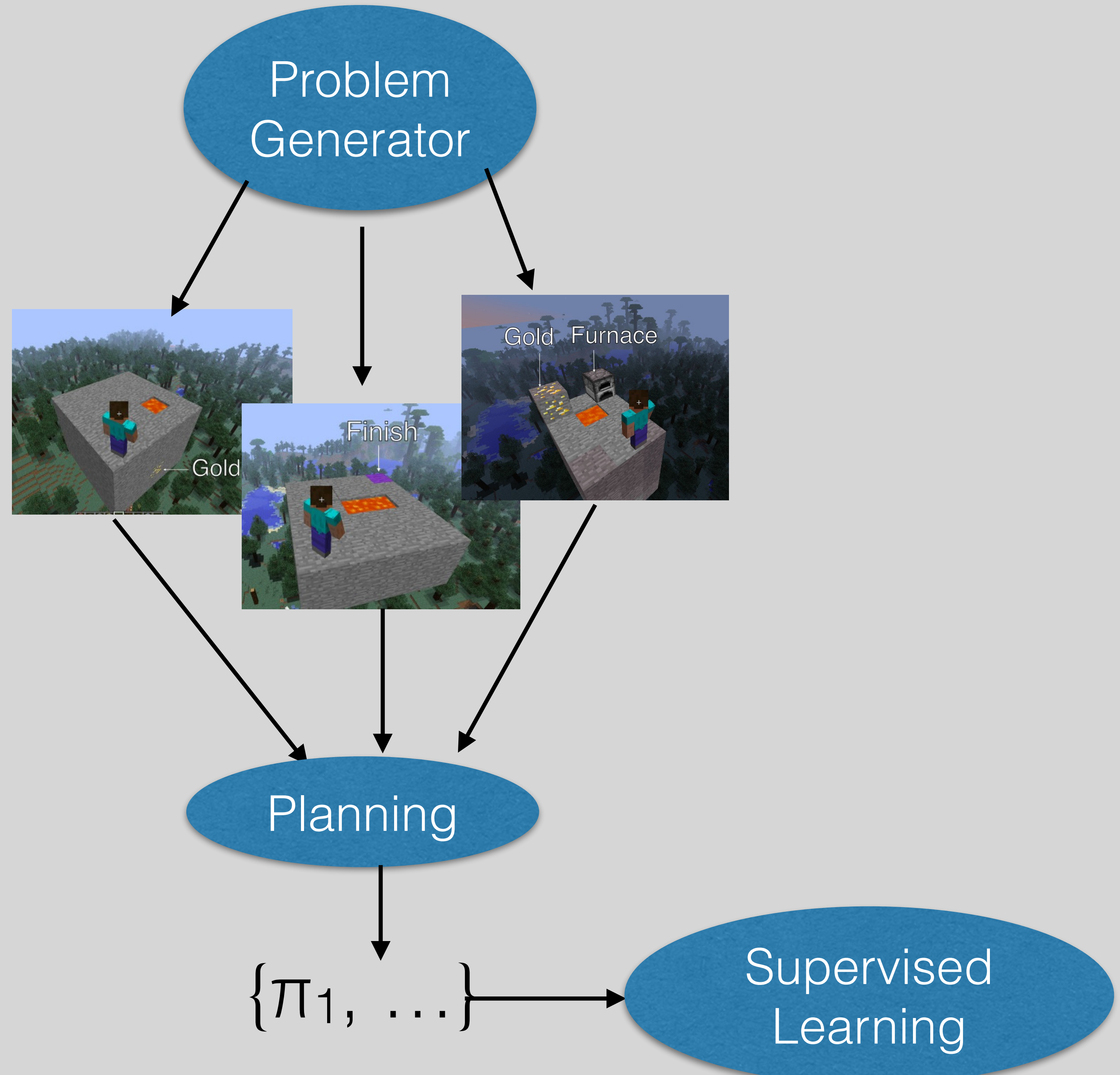




Learning to Plan

(Abel, Hershkowitz, Barth-Maron, Brawner, O'Farrell, MacGlashan, and Tellex, 2015)

- Goal-directed action priors
  - Not all actions are relevant for a given goal-type in every state
  - Learn possibly relevant actions and prune the rest
  - Prune irrelevant actions

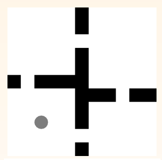
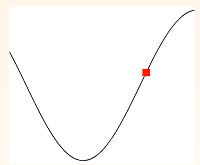
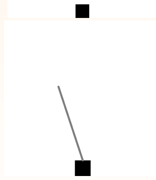


Escaping Groundhog Day

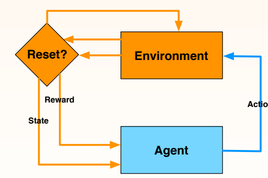
BROWN

James MacGlashan, Stefanie Tellex, and Michael L. Littman

1. Classic RL Is Like the Movie Groundhog Day

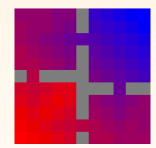




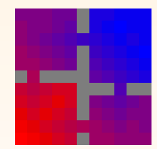

- Wake up
- Act in the world until completion
- Reset back to the beginning
- **Hyper optimize with retries**

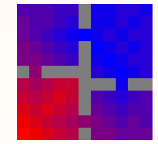


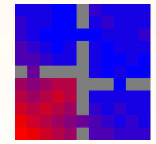
2. Brittle to Changes

Q-learning value function very slowly shifts to the new goal location


Before goal change


100 steps


500 steps

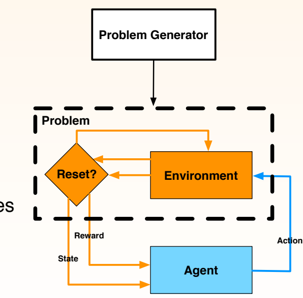

800 steps

3. Escaping Groundhog Day

- Relax assumptions
- Investigate benchmark problem generators
- Develop appropriate learning machinery

4. Relax Problem Assumptions

- Problem can affect
  - Reset states
  - Transition function
  - Reward function
- Learn what is ground and figure



Related Areas
- Learning Action Hierarchies
- Transfer Learning
- Bayesian RL

Ground
everything is constant

Problem Generators
some constants, some variables

Figure
everything varies

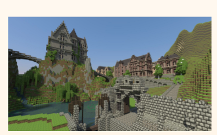
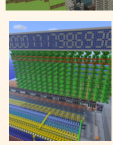
5. Domains for Problem Generators

Robotics


- The real world is complex
- Easy to have variation in the environment
- Range of learning tasks beyond motion controllers

Minecraft

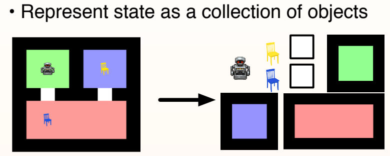
- Enormous worlds
- Turing complete complexity
- Safe
- No hardware failures
- Many possible goals
- Easy to manipulate


6. Object-oriented MDPs

(Diuk, Cohen, and Littman, 2008)

- Represent state as a collection of objects

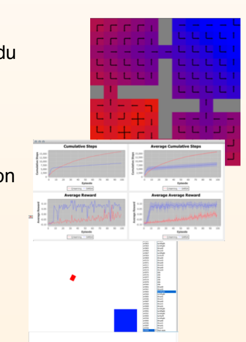


- Each object receives a value assignment, e.g., robot := <2,6>; block0 := <2,2,chair,blue>; ...
- Permits learning object-wise transition functions

7. BURLAP

http://burlap.cs.brown.edu

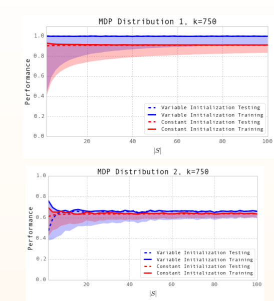
- Java RL and planning library
- Problem and state generators
- OO-MDP Representation
- ROS Interface
- Minecraft interface
  github.com/h2r/burlapcraft
- Function approximation
- Options
- Inverse RL
- Multi-agent
- and more!



8. Learning to Learn

- Tune and select an algorithm for a distribution of problems.
- Introduce Sample Optimized Rademacher Complexity to generate generalization bounds
- Formal bounds from training problems and weak parameter optimization
- Grounds as many parameters as possible

Example
- Two classes of Q-learning parameters to tune
  1) epsilon, learning rate
  2) epsilon, learning rate, all initial Q-values
- On a narrow distribution with little data, choose (2); on wide distribution choose (1)

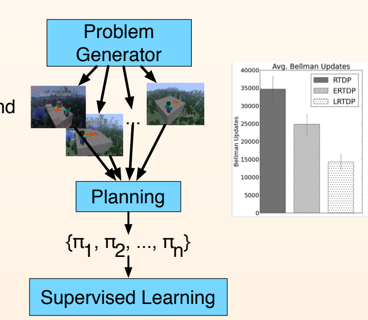


9. Learning to Plan

(Abel, Hershkowitz, Barth-Maron, Brawner, O'Farrell, MacGlashan, and Tellex, 2015)

- Not all actions are relevant for all states and goals
- Prune irrelevant actions
- Learn optimality probability from solved training problems
- Grounds bad action decisions

- We test on Minecraft
- Training data consists of small problems
- Testing is on larger harder problems



Conclusion

- Recent work gearing towards a problem generator paradigm
  - Novel states, reward function, and transition dynamics
  - Some things stay the some others vary
- Robotics and Minecraft offer interesting problems
- New Machinery
  - OO-MDPs, goal-based action priors, algorithm selection,
- BURLAP - problem generators, ROS, and Minecraft
  - http://burlap.cs.brown.edu
  - Minecraft interface: https://github.com/h2r/burlapcraft

Collaborators

- David Abel
- Krishna Aluru
- Gabriel Barth-Maron
- Stephen Brawner
- Ellis Hershkowitz
- Vukosi Marivate
- Kevin O’Farrell
- Matthew Taylor
- Carl Trimbach
- Eli Upfal

Conclusion

- Recent work gearing towards a problem generator paradigm
  - Novel states, reward function, and transition dynamics
  - Some things stay the some others vary
- Robotics and Minecraft offer interesting problems
- New Machinery
  - OO-MDPs, goal-based action priors, algorithm selection,
- BURLAP - problem generators, ROS, and Minecraft
  - http://burlap.cs.brown.edu
  - Minecraft interface: https://github.com/h2r/burlapcraft