

4	R				G
3	O				
2					
1					
0	Y			B	
	0	1	2	3	4

# Thoughts on Abstraction in RL

Tom Dietterich

Oregon State University

# What is Abstraction?

- State abstraction:
  - dimensionality reduction  $\chi(s)$  in the state space that ignores some information
    - interesting case: information losing in the sense that the full MDP value function or policy cannot be represented in this space. However,  $V$  or  $\pi$  can be represented for some subtask/subroutine/sub-agent
- Action abstraction:
  - temporally extended action (options, MAXQ subprocedures)
- Agent abstraction:
  - factoring the action space into separate “agents” that can operate using a state abstraction (possibly with limited communication or coordination with the other agents).

# Open Problem 1

- Automatic discovery of MAXQ hierarchies
  - Given: experimental access to an MDP
  - Find: MAXQ hierarchy with state and action abstractions
  - Exact and approximate cases
  - Additional challenges
    - Can we do it fast enough to benefit on this MDP
    - Or do we only do this in a transfer learning setting?

# Open Problem 2

- Automatic factoring of complex MDP into multiple agents (including state abstractions) while minimizing coordination costs
  - Is there something analogous to the MAXQ value function decomposition that works for multi-agent MDPs?
  - Under what conditions?

# Open Problem 3

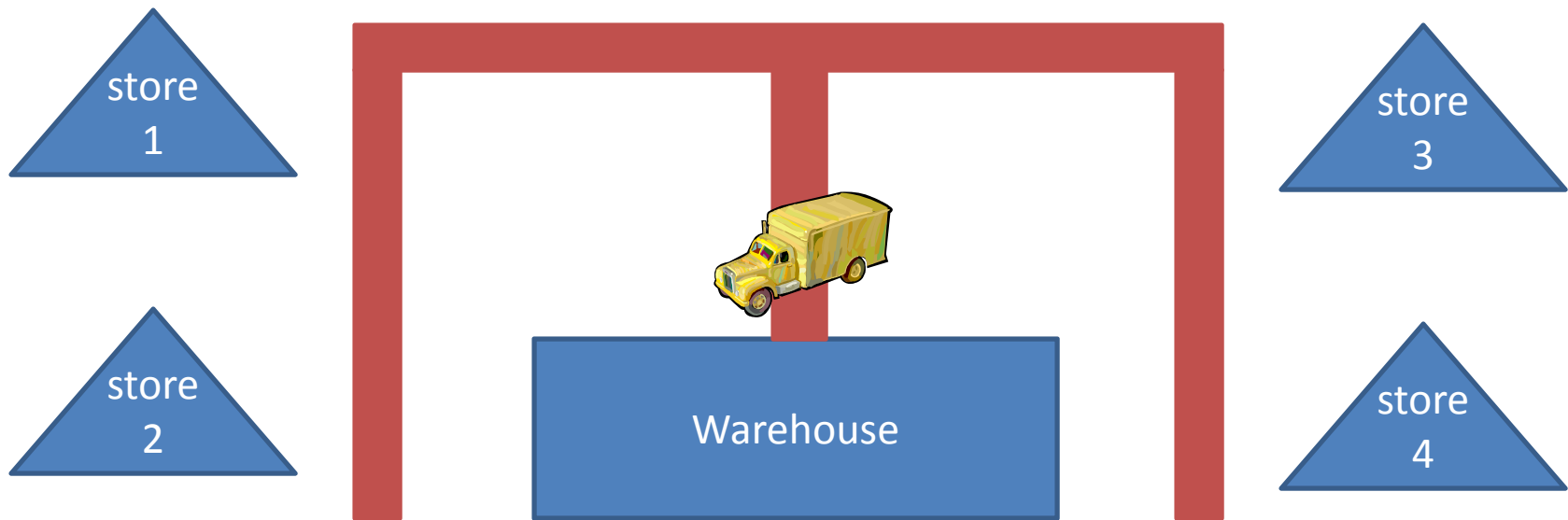
- Learning Complex State Abstractions
  - Existing abstractions consist of ignoring a subset of the state variables
  - What about transforming the state variables?
    - “destination is reachable without refilling gas tank”

# Open Problem 4

- Learning policy-dependent state abstractions
  - Consider a consumable resource (e.g., gasoline, battery power).
  - If the policy invokes a subroutine only when there is enough resource available for the subroutine to terminate, then the subroutine can ignore the resource
  - Example: Fuel Tax
- This abstraction is unsafe in general, but by positing an invariant (enforced by the parent policy), it becomes safe.
- Another view: Reward Decomposition (see below)

# Open Problem 5

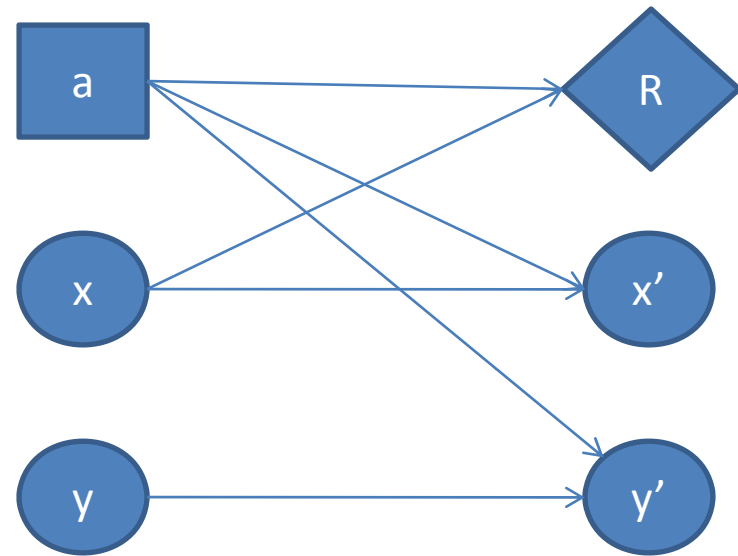
- Automatically introducing “intentions” to gain state abstraction
  - K kinds of products sold in stores
  - Each store has demand  $D[1\dots K]$
  - Truck has capacity  $Q$
  - Cost for driving around (should follow shortest path)
  - Reward for satisfying the demand of each store
  - Wholesale cost for each unit delivered



intention: Set of stores to handle on next trip

# Rewards Matter in Abstraction

$y$  is irrelevant for MDP  $M$  if  
the state variables  $s$  can be partitioned as  
 $s = (x, y)$  such that for all actions  $a$   
 $P(s' | s, a) = P(x' | x, a) P(y' | x, y, a)$  and  
 $R(s' | s, a) = R(x' | x, a)$



- Goal: Abstract as much as possible while still being able to represent  $V(s)$
- If  $R$  is constant, then all state variables are irrelevant
- Do intrinsic rewards produce good abstractions?
- Reward decomposition enables better abstractions