

Robotic Software Architecture COMP3431

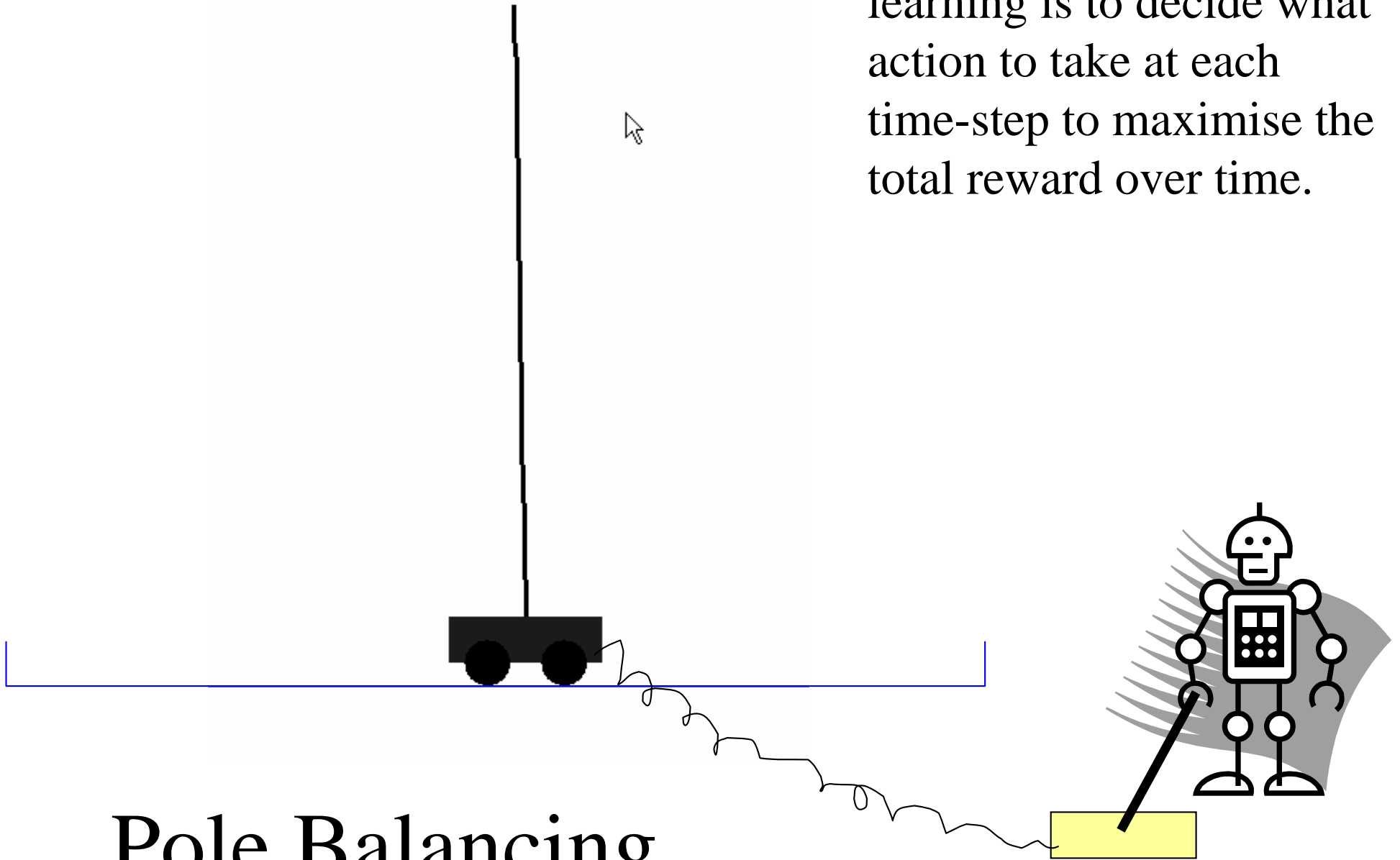
Introduction to Hierarchical Reinforcement Learning

Lecturer: Bernhard Hengst
email: bernhard.hengst@nicta.com.au

19th October 2005



The goal of reinforcement learning is to decide what action to take at each time-step to maximise the total reward over time.

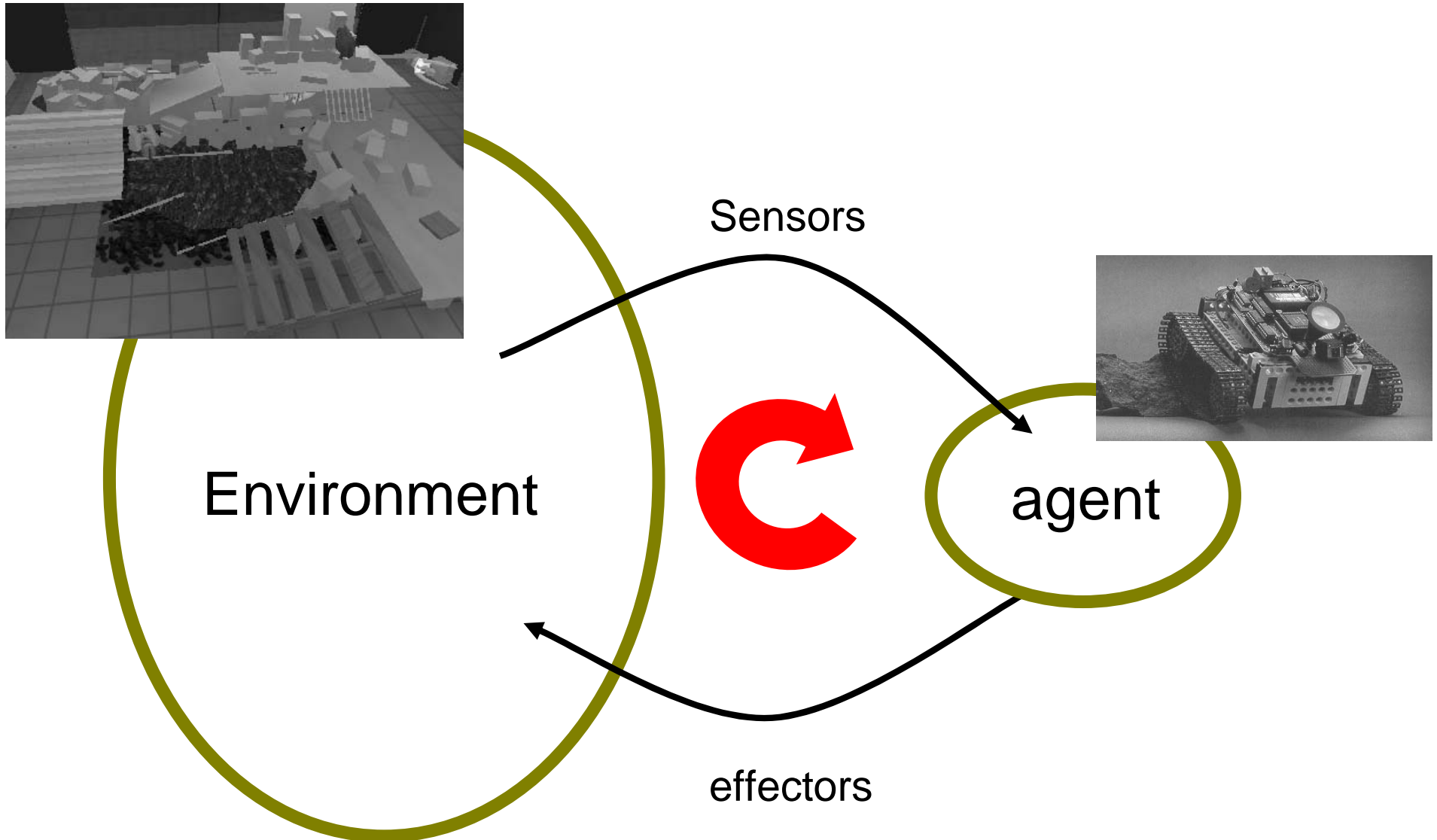


Pole Balancing

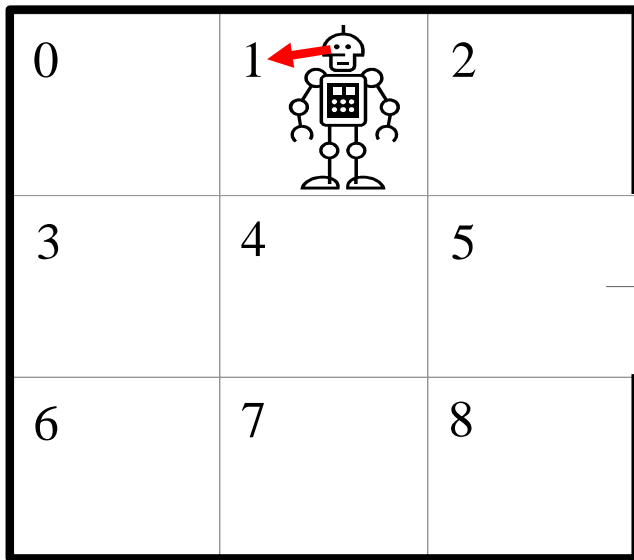
History of Reinforcement Learning

- Psychology/biology
(eg BF Skinner - animal training)
- Operations Research
DP (planning over time)

The Agent View of RL



One Room Problem



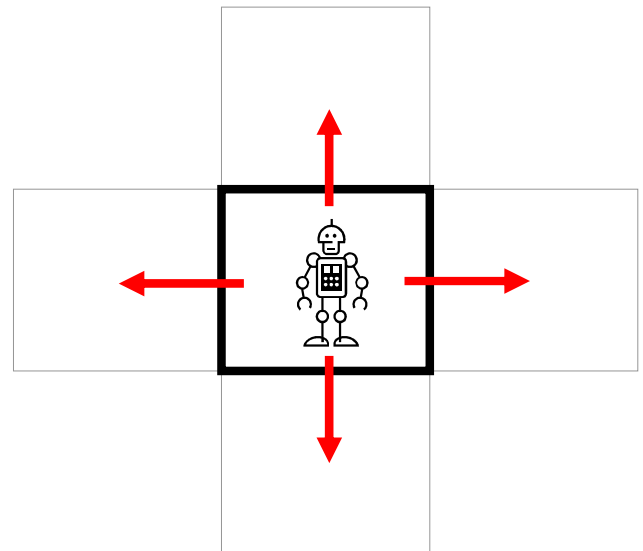
Room States

Exit Reward
\$100



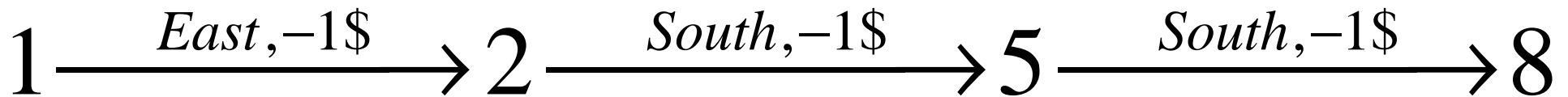
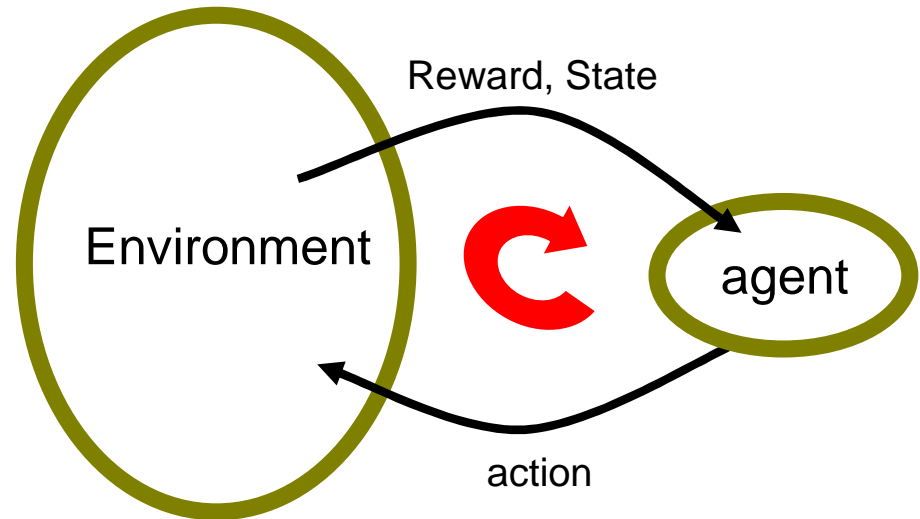
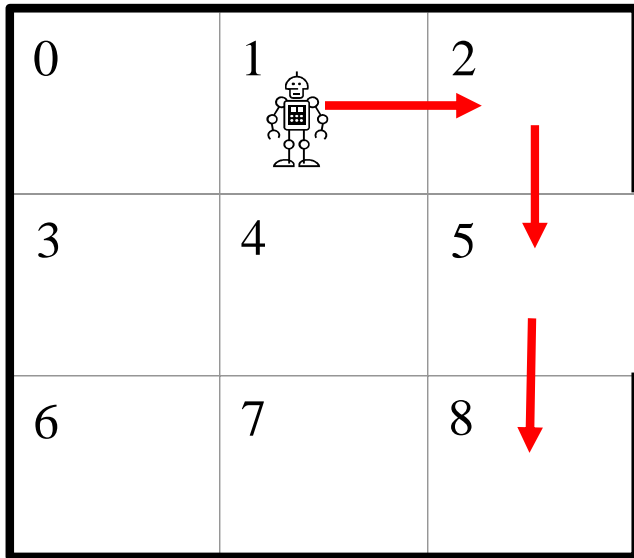
cost \$1 per
time-step
(except \$100)

Actions $\in \{N, S, E, W\}$



State Transitions

Sense – Act Cycle



Policy $\pi : S \rightarrow A$ e.g. $\pi(1)=E, \pi(2)=S, \pi(5)=S, \dots$

Value Function

is the utility of the current state in terms of future rewards given a policy.

policy $\pi = \nearrow$

0 \$93	1 \$98	2 \$99
3 \$94	4 \$97	5 \$100
6 \$95	7 \$96	8 \$95

Transitions: 0 → 1, 1 → 2, 2 → 5, 5 → 8, 8 → 7, 7 → 6, 6 → 3, 3 → 0, 4 → 1, 1 → 4, 5 → 4, 4 → 5, 8 → 5, 5 → 8. Reward: -\$1 from 5 to 4, \$100 from 5 to 8.

$$V^\pi(s_t) \equiv r_t + r_{t+1} + \dots + r_T$$

$$V^\pi(s_t) \equiv r_t + V^\pi(s_{t+1})$$

Optimal Value Function

(* = Optimal)

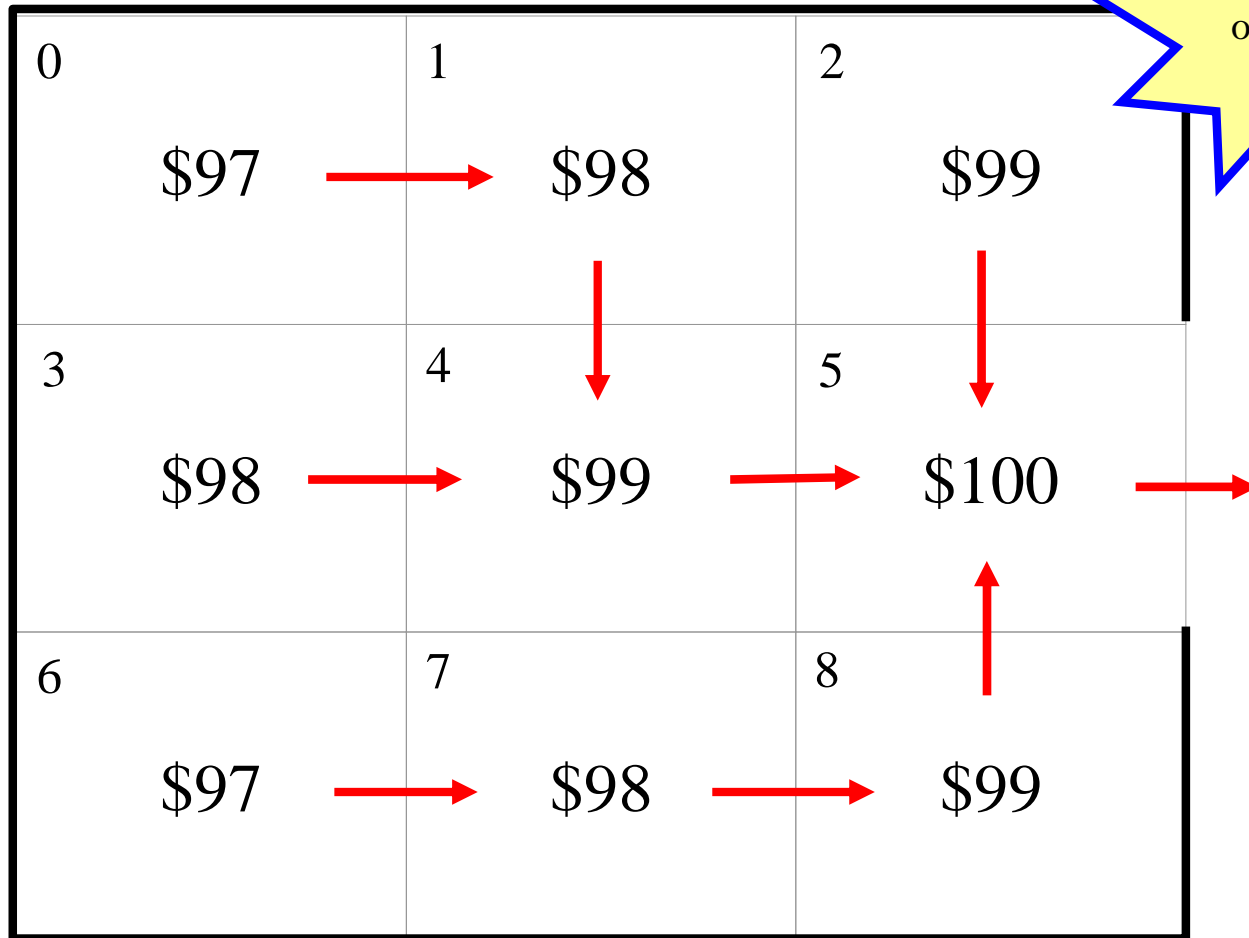
0 \$97	1 \$98	2 \$99
3 \$98	4 \$99	5 \$100
6 \$97	7 \$98	8 \$99

Annotations: A downward arrow labeled "-\$1" points to cell 5. A rightward arrow labeled "\$100" points from cell 5.

$$V^*(s_t) \equiv \max_a [r_t + V^*(s_{t+1})]$$

An Optimal Policy

only one
optimal value function,
but possibly many
optimal policies



$$\pi^*(s_t) \equiv \arg \max_a [r_t + V^*(s_{t+1})]$$

Markov Decision Problem (MDP)

tuple $\langle S, A, \delta, r, s_0 \rangle$

States $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$

$s_0 =$ starting state (eg random)

Actions $A = \{N, S, E, W\}$

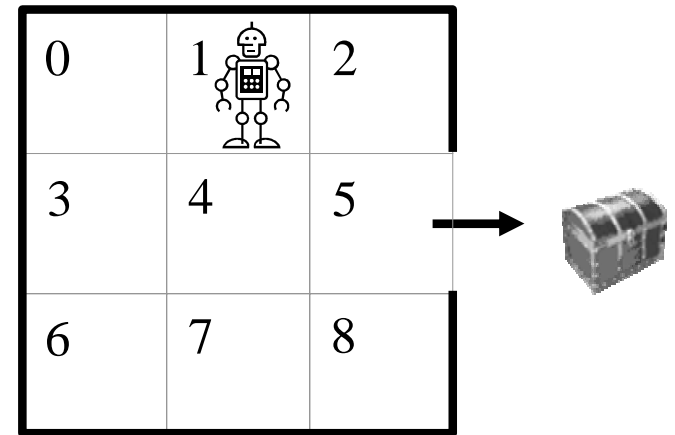
Transition function $\delta(s_t, a_t) = s_{t+1}$

Reward function $r(s_t, a_t) = r_t$

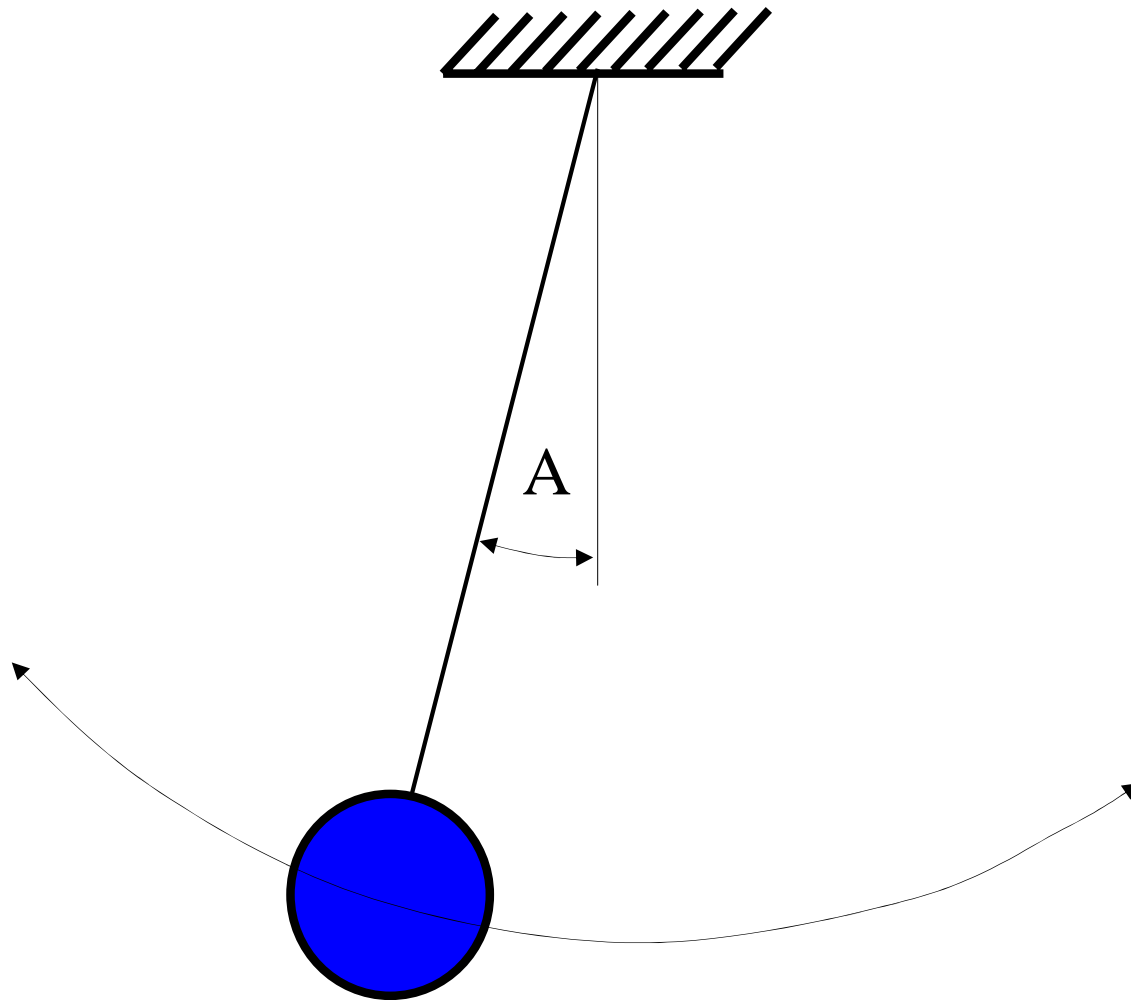
} Model of the environment

Optimisation criterion eg $V^\pi(s_t) \equiv r_t + r_{t+1} + r_{t+2} \dots r_T$

Optimal policy $\pi^* \equiv \arg \max_{\pi} V^\pi(s), \forall s$



The Markov Property



$$\Pr(s_{t+1} | s_t) = \Pr(s_{t+1} | s_t, s_{t-1}, s_{t-2}, s_{t-3}, \dots)$$

Solution to MDP - Value Iteration

initialise $V(s) = 0$ for all s

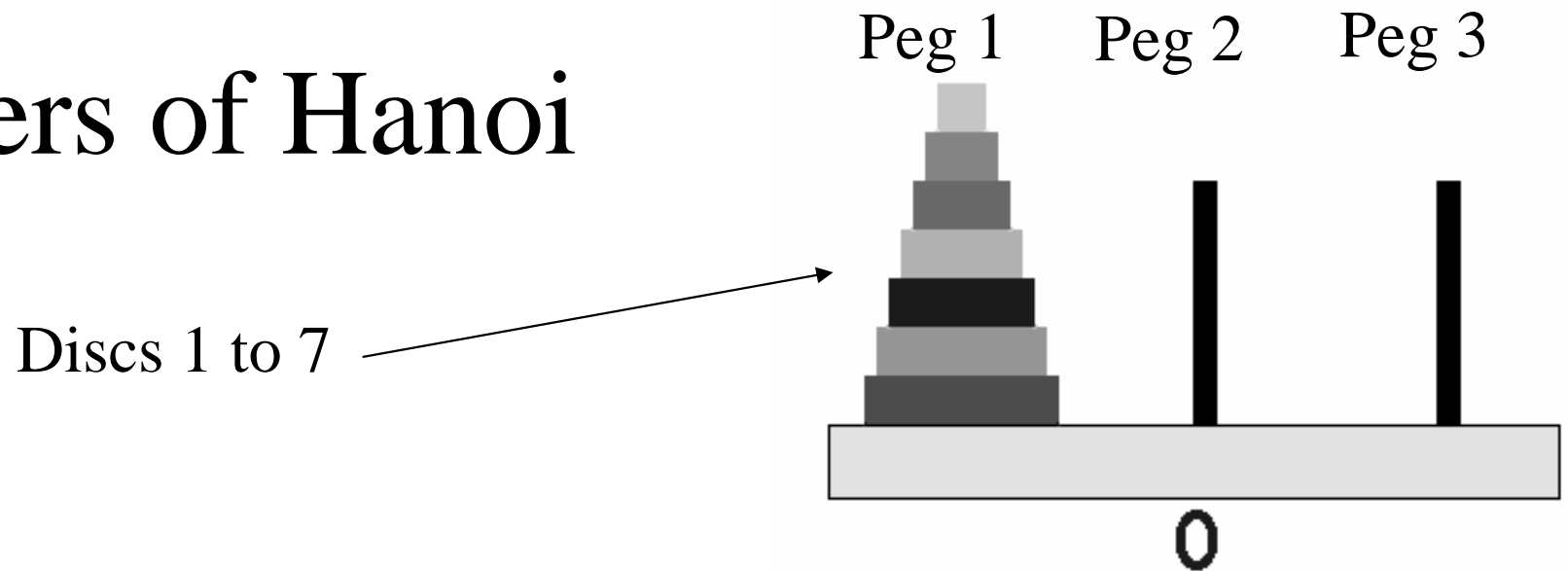
repeat for each $s \in S$

$$V(s) \leftarrow \max_a [r(s, a) + V(\mathcal{D}(s, a))]$$

until $V(s)$ does not change for any s

Output $\pi^*(s) \equiv \arg \max_a [r(s, a) + V(\mathcal{D}(s, a))]$

Towers of Hanoi



States = $\{(1,1,1,1,1,1,1), (2,1,1,1,1,1,1), \dots (3,3,3,3,3,3,3)\}$ $|S|=2187$

$S_0 = (1,1,1,1,1,1,1)$, $s = (3,3,3,3,3,3,3)$ is goal or terminal state

Actions = $\{\text{moveDisc12}, \text{moveDisc13}, \text{moveDisc21},$
 $\text{moveDisc23}, \text{moveDisc31}, \text{moveDisc32}\}$

$\delta: (1,1,1,1,1,1,1), \text{moveDisc12} \rightarrow (2,1,1,1,1,1,1),$

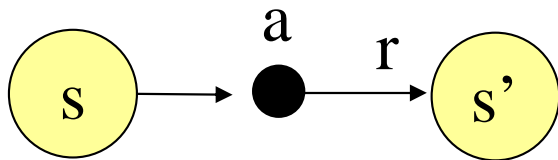
$(1,1,1,1,1,1,1), \text{moveDisc13} \rightarrow (3,1,1,1,1,1,1)$, etc

r: all moves cost -1

$V^*(1,1,1,1,1,1,1) = -127$

$\pi^*(1,1,1,1,1,1,1) = \text{moveDisc13}$, etc (NB there are 10^{1701} policies!!!)

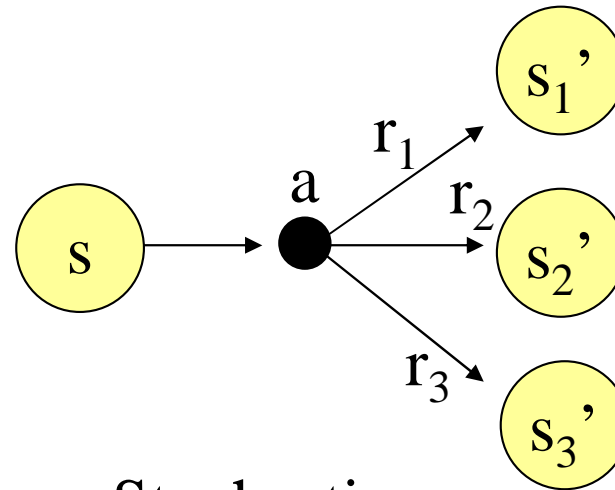
State Transitions and Rewards can be Stochastic



Deterministic

0	1	2
3	4	5
6	7	8

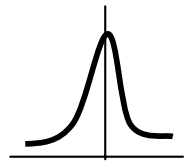
A 3x3 grid with cells containing numbers 0 through 8. A red arrow points from cell 1 to cell 2.



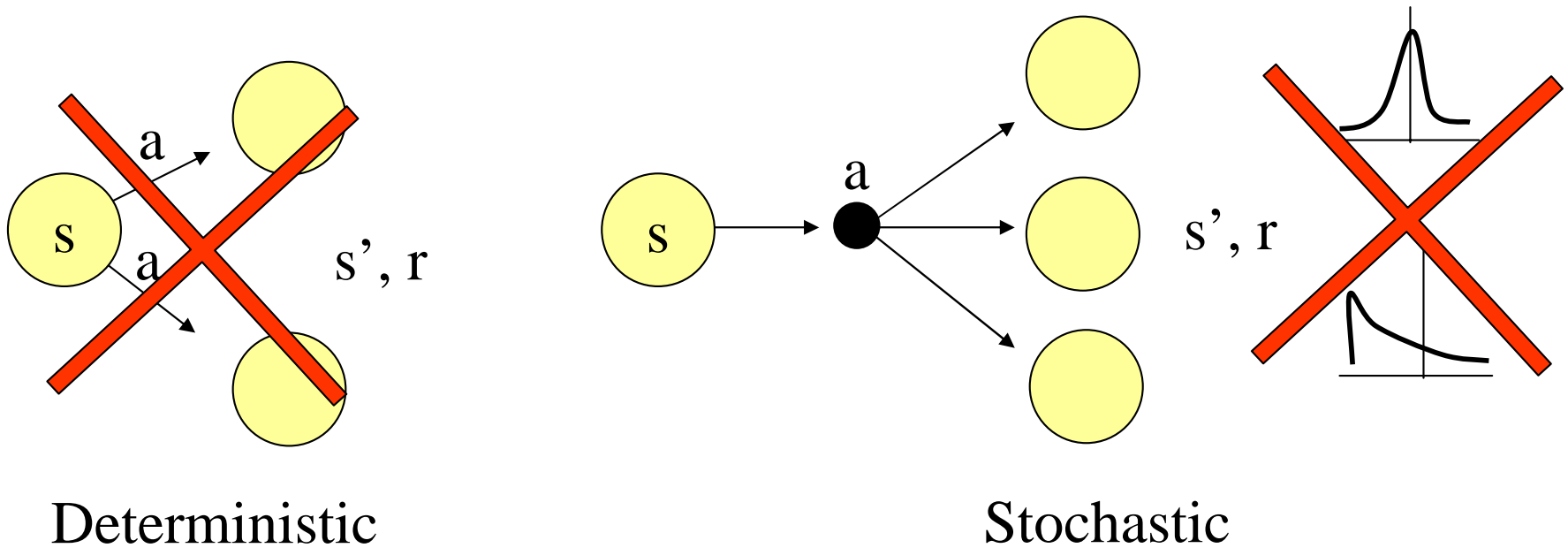
Stochastic

0	1	2
3	4	5
6	7	8

A 3x3 grid with cells containing numbers 0 through 8. Red arrows point from cell 1 to cells 0, 2, and 4.



State Transitions and Rewards must be Markov



Optimality Criteria

Sum of rewards to termination (Stochastic Shortest Path)

$$V^\pi(s_t) \equiv E\{r_t + r_{t+1} + r_{t+2} \dots r_T\}$$

Sum of rewards for next N time steps

$$V^\pi(s_t) \equiv E\{r_t + r_{t+1} + r_{t+2} \dots r_N\}$$

Discounted sum of rewards

$$V^\pi(s_t) \equiv E\{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots\}, \quad 0 \leq \gamma < 1$$

Average reward per step

$$V^\pi(s_t) \equiv \lim_{n \rightarrow \infty} \frac{1}{n} E\{r_t + r_{t+1} + r_{t+2} \dots r_n\}$$

Be Careful Defining the MDP!

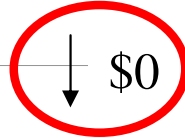
0	1	2
*\$100	*\$100	*\$100
3	4	5
*\$100	*\$100	↓ \$0
6	7	8
*\$100	*\$100	*\$100

\$100
the only reward

$$V^{\pi}(s_t) \equiv r_t + r_{t+1} + r_{t+2} \dots r_T$$

Discounted Value Function

0	1	2
*\$72.9	*\$81	*\$90
	4	5
*\$81	*\$90	*\$100
6	7	8
*\$72.9	*\$81	*\$90



\$100

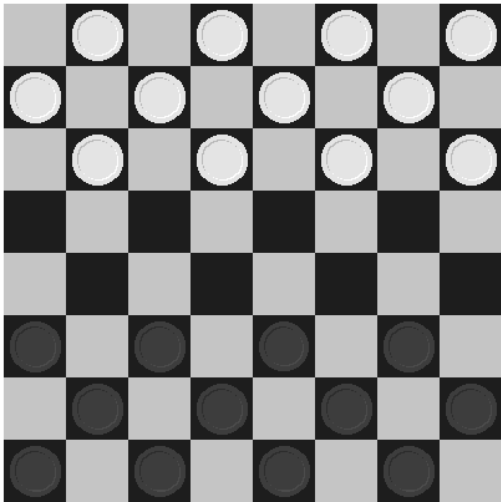


RL learns with
delayed reward

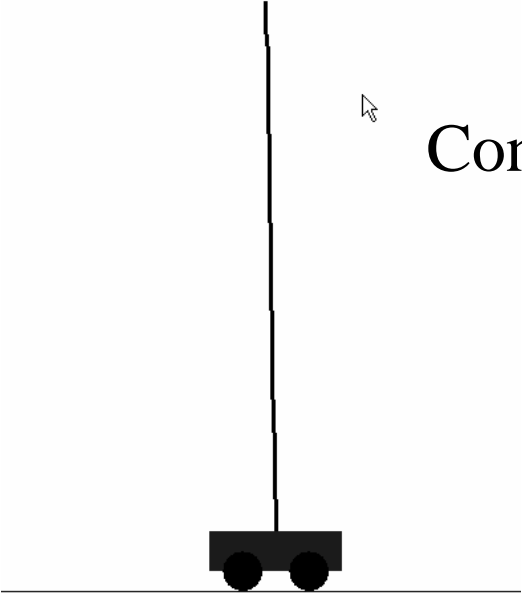
$$V^\pi(s_t) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots \gamma^T r_T, \quad \gamma = 0.9$$

Infinite Horizon (Continuing) Tasks

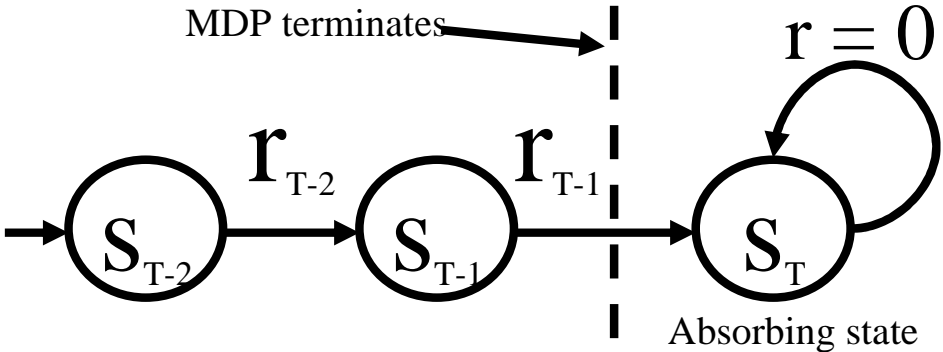
Checkers



Episodic



Continuing



unify episodic and continuing

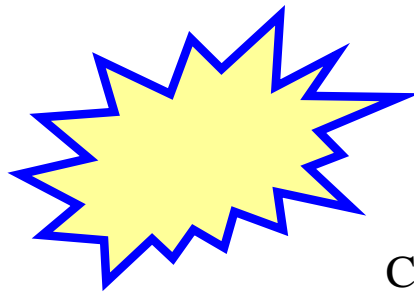
Model Free Learning

Model $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$

The transition and reward function may not be known by the agent.

But, in the Value Iteration algorithm

$$V(s) \leftarrow \max_a [r(s, a) + V(\delta(s, a))]$$



Major breakthrough is to learn the action value function $Q(s,a)$ instead of the value function $V(s)$

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, 1989.

Action Value Function Q

Define $Q^*(s, a) \equiv r(s, a) + V^*(\delta(s, a))$

0	1	2
\$97	\$98	\$99
3	4	5
\$98	\$99	\$100
6	7	8
\$97	\$98	\$99

Red arrows indicate transitions from state 3:

- Up arrow to state 0
- Down arrow to state 6
- Left arrow to state 3
- Right arrow to state 4

 A black arrow points from state 5 to a label "\$100". A black arrow points from state 5 to a label "-\$1".

$$Q(3, N) = -1 + 97 = 96$$

$$Q(3, E) = -1 + 99 = 98$$

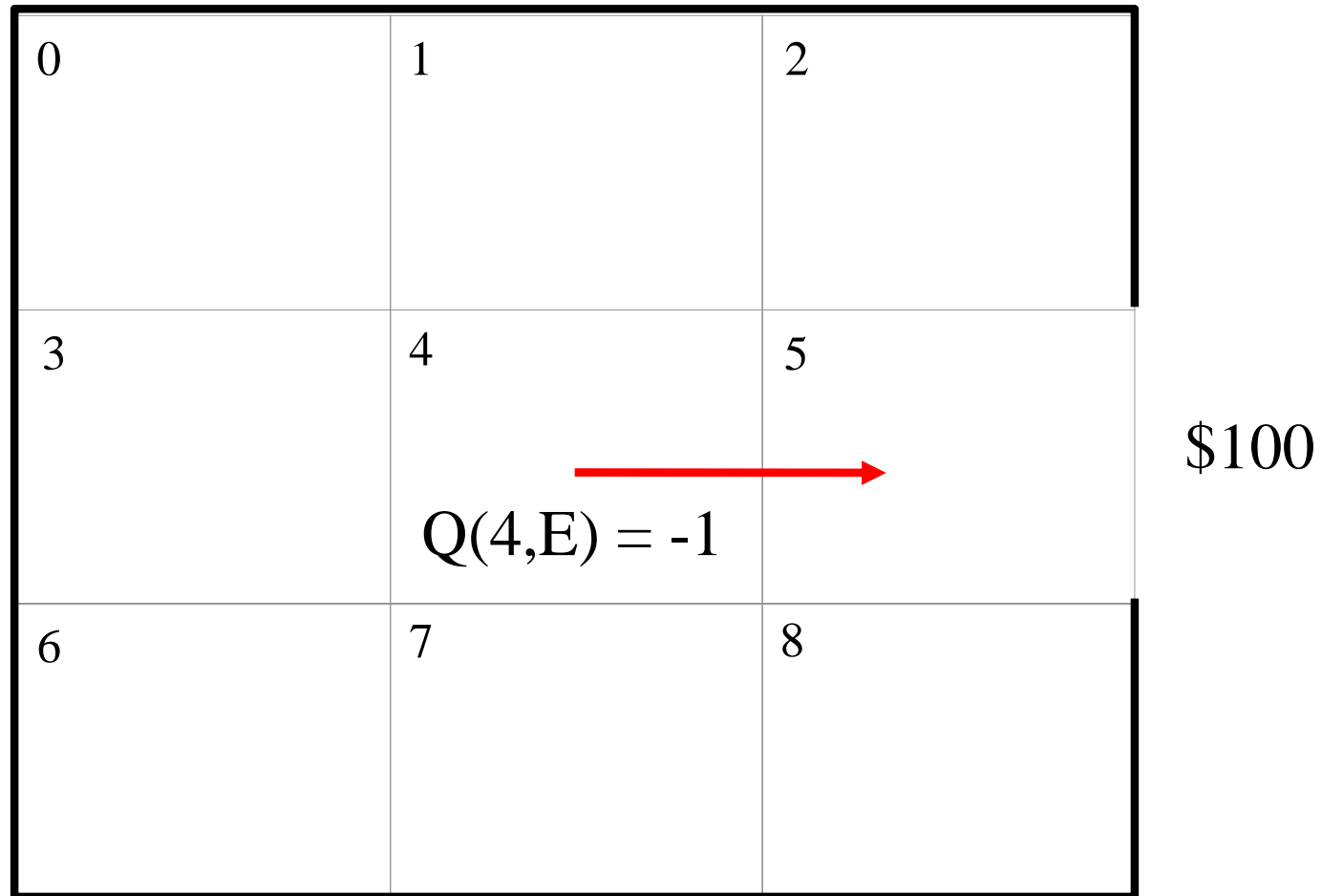
$$Q(3, S) = -1 + 97 = 96$$

$$Q(3, W) = -1 + 98 = 97$$

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

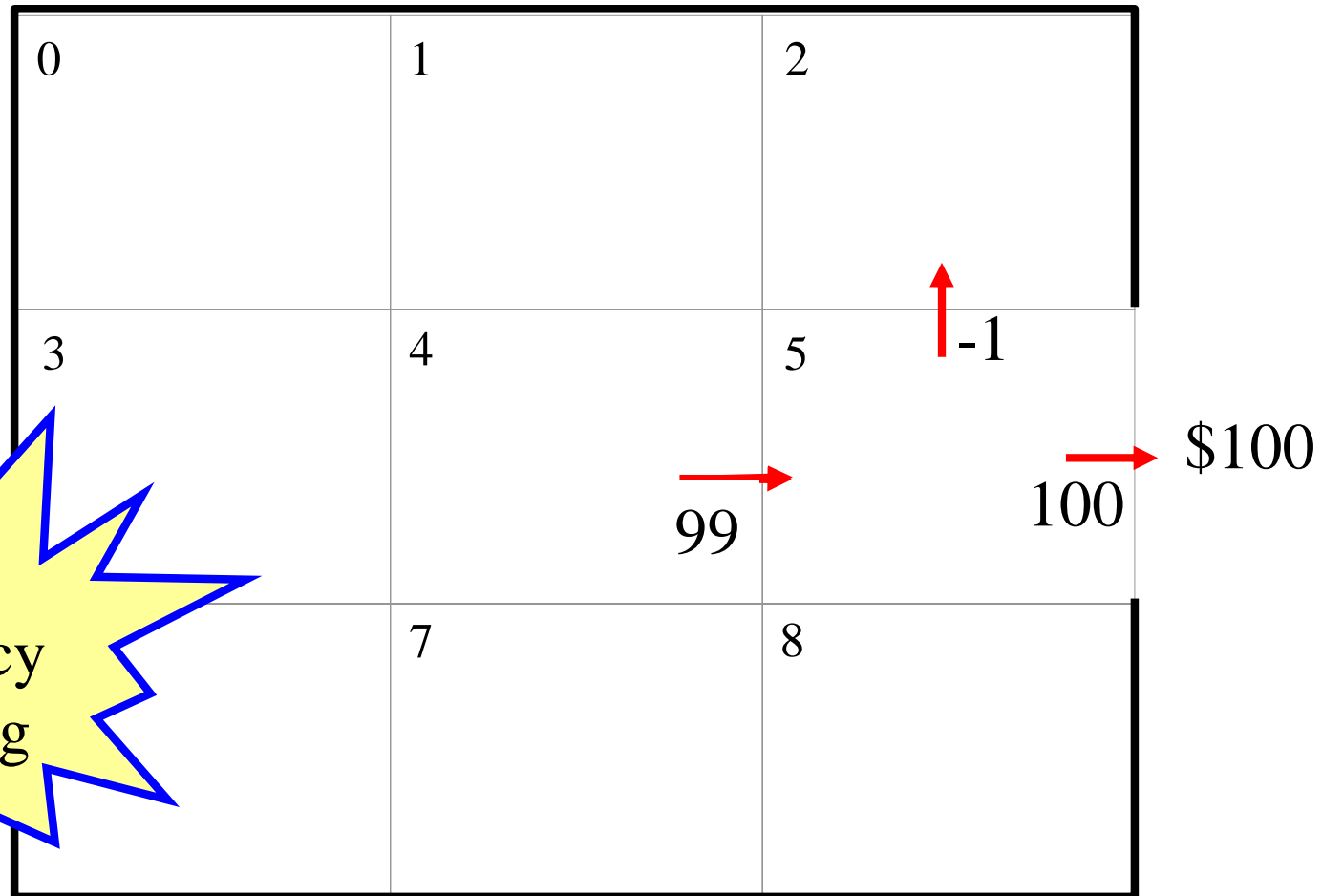
$$\pi^*(3) = E$$

Q-Learning (Temporal Difference)



Training Rule: $Q(s, a) \leftarrow r + \max_{a'} Q(s', a')$

Q-Learning (Temporal Difference)



Training Rule: $Q(s, a) \leftarrow r + \max_{a'} Q(s', a')$

Training Rule to Learn $Q(s,a)$

$$s, a \rightarrow r, s'$$

$$Q(s, a) \leftarrow r + \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow (1 - \alpha_n) Q(s, a) + \alpha_n [r + \gamma \max_{a'} Q(s', a')]$$

Q Learning

initialise $Q(s, a) = 0$ for all s and a

observe current state s

repeat

 select an action a and execute it

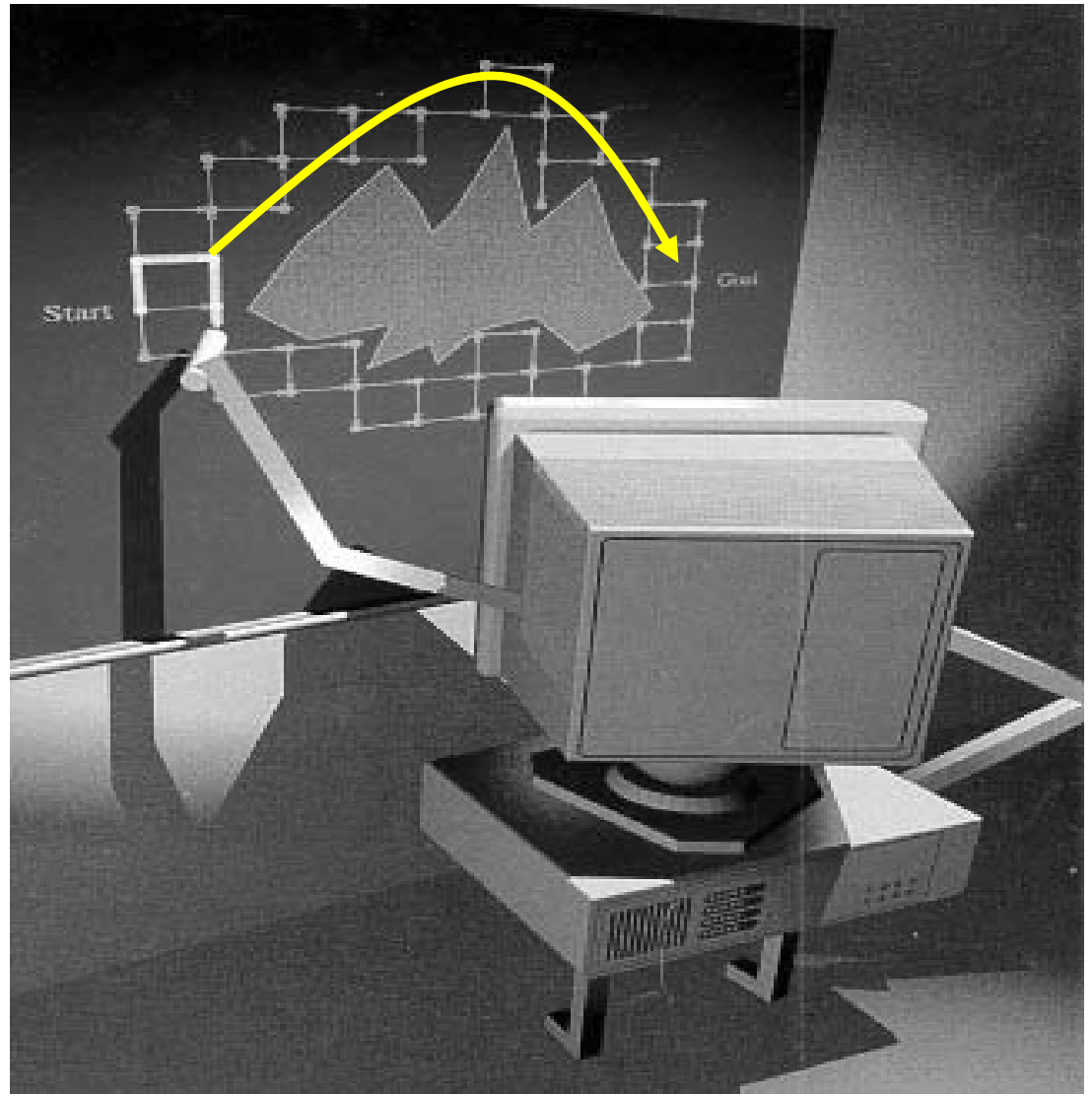
 observe immediate reward r and next state s'

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a')]$$

$$s \leftarrow s'$$

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

Exploration VS Exploitation



Some Exploration Policies





- Greedy: Choose the action with the highest estimated value
- ϵ -greedy: Behave greedily most of the time, but occasionally, with probability ϵ , choose an action at random.
- Softmax: Vary the probability of action as a graded function of estimated value:



$$\Pr(a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}$$

$\tau = \textit{temperature parameter}$

- Random

Eligibility Traces

0	1	2
		
3	4	5  ↓ \$0
		 \$100  \$100
6	7	8

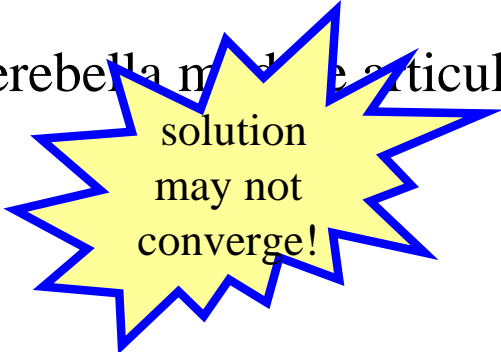
0	1	2
\$72.9	\$81	
3	4	5  ↓ \$0
	\$90	100  \$100
6	7	8

Function Approximation

(Generalising from Examples)

What if we have continuous or large state and/or action values?

- We may have an infinite number of states and/or actions!!
- Use a kind of generalisation called function approximation.
 - State Aggregation: States are grouped together, with one table entry (value estimate) used for the group.
 - Linear Methods, eg as introduced in Ch1 ML (Mitchell), page 9 for checkers
 - Instance based methods: value approximated by nearest neighbour
 - Artificial neural networks
 - Others, eg tile coding or CMAC (cerebellar model articulator controller)



solution
may not
converge!

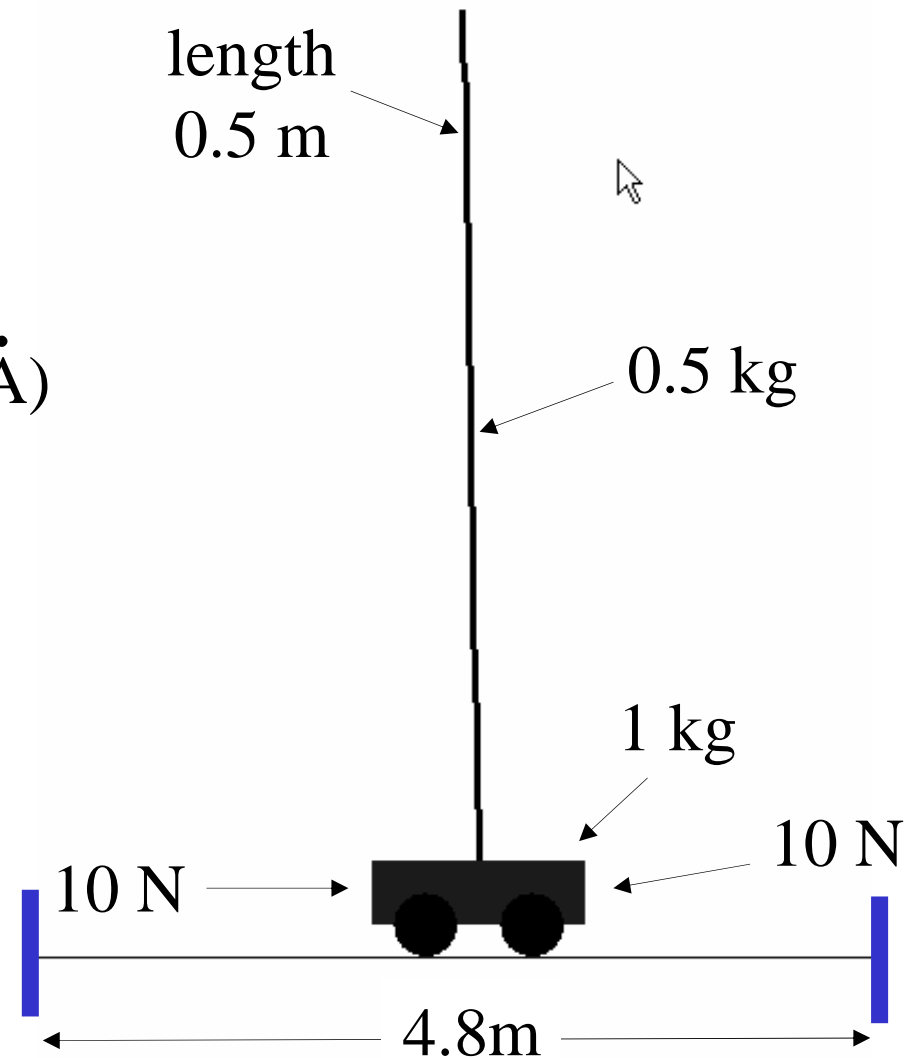
Example: Pole Balancing

Markov State

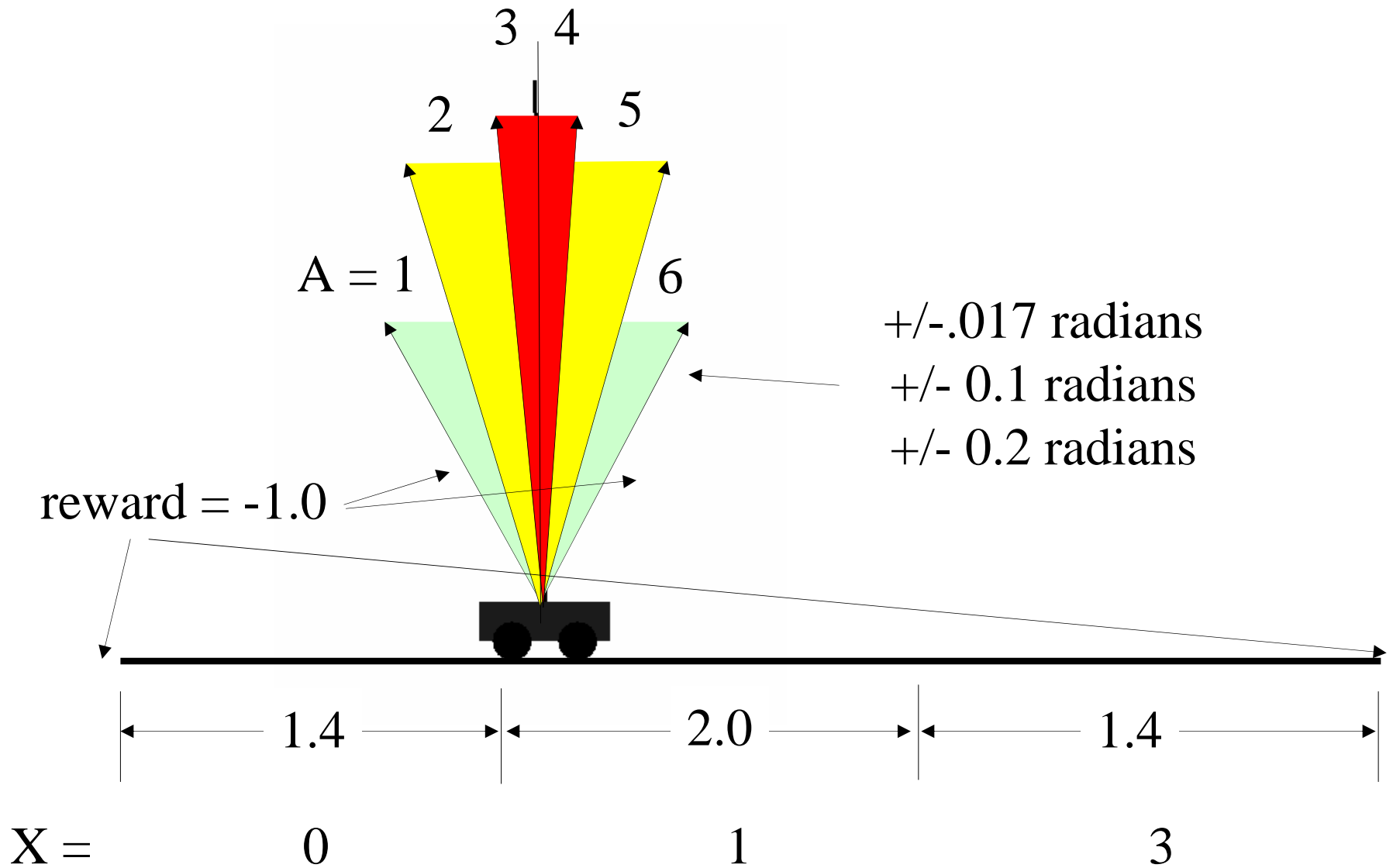
- pole angle (θ)
- pole angular velocity ($\dot{\theta}$)
- cart position (X)
- cart velocity (\dot{X})

Actions

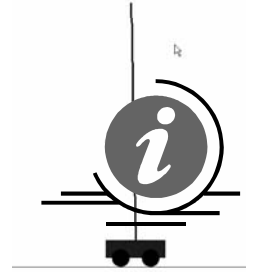
- push left
- push right



Pole Balancing: Function Approximation



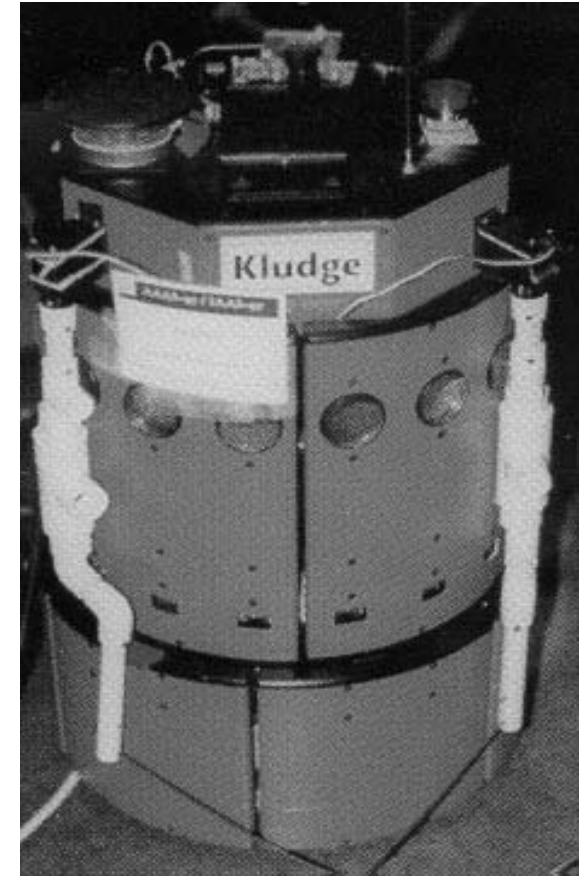
Pole Balancing



- State: continuous, discretised to $|A|=6$, $|\dot{A}|=4$, $|X|=3$, $|\dot{X}|=4$.
Total $|S|=288$
- Actions: push left, push right
- Reward: -1.0 if leaning too much or run off track, 0.0 other-wise
- Infinite horizon
- Model free
- Q learning (one step backups)
- Optimality criteria: maximise sum of discounted future rewards
- Parameters:
 - learning rate (alpha) fixed at .2
 - discount rate (gamma) = 0.99

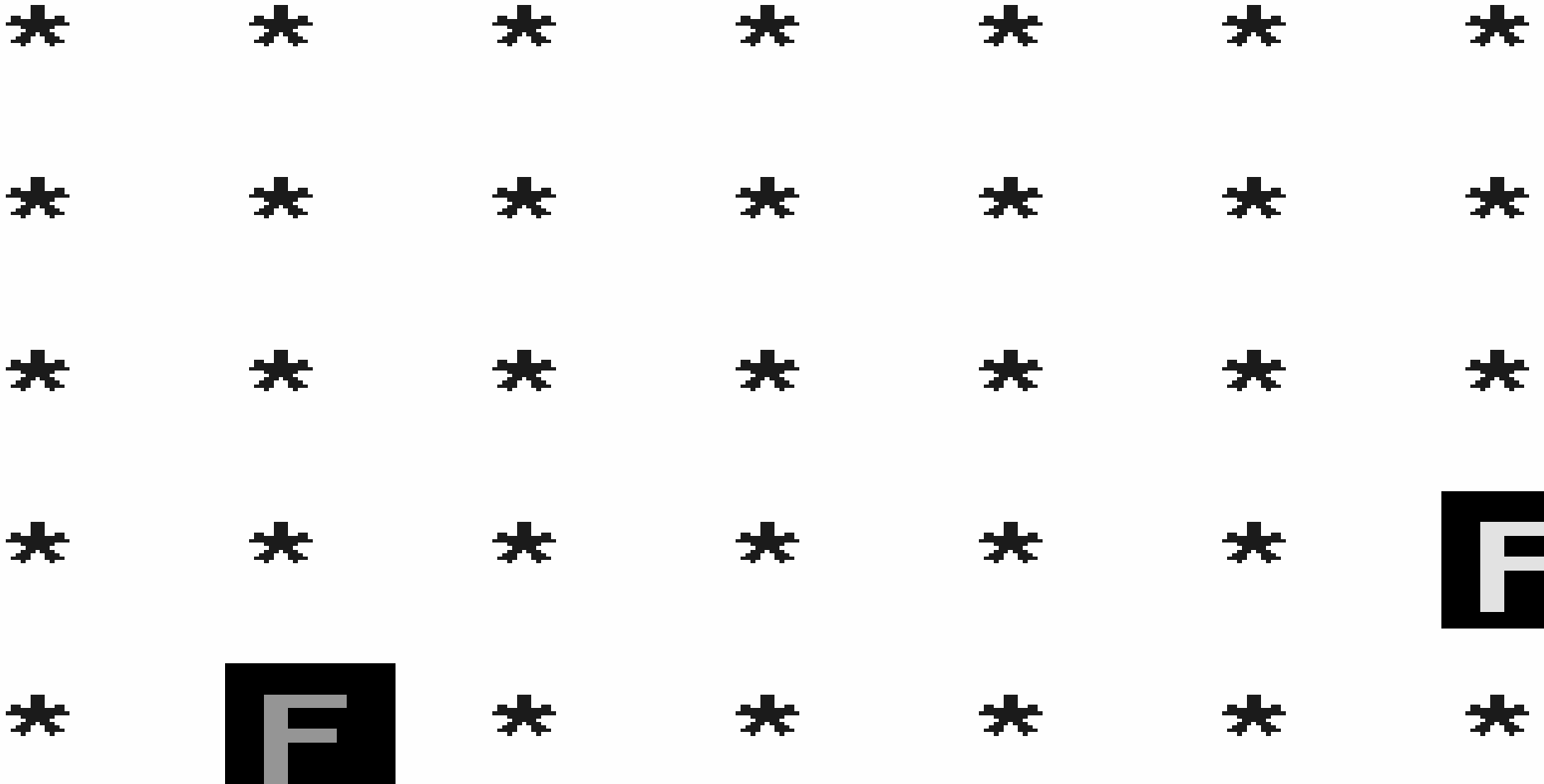
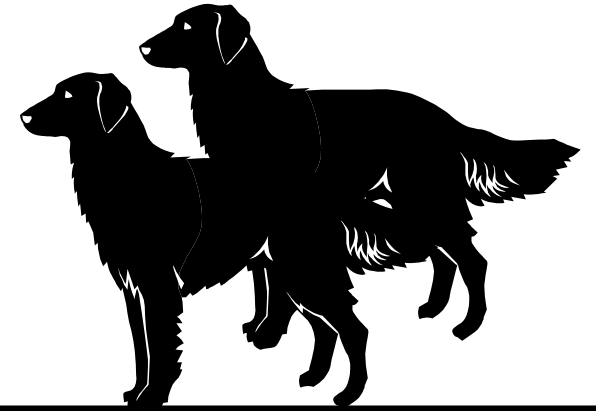
Hidden state

9	10	10	8	10	10	12
5			5			5
5			<i>G</i>			5
5			5			5
3	10	10	2	10	10	6

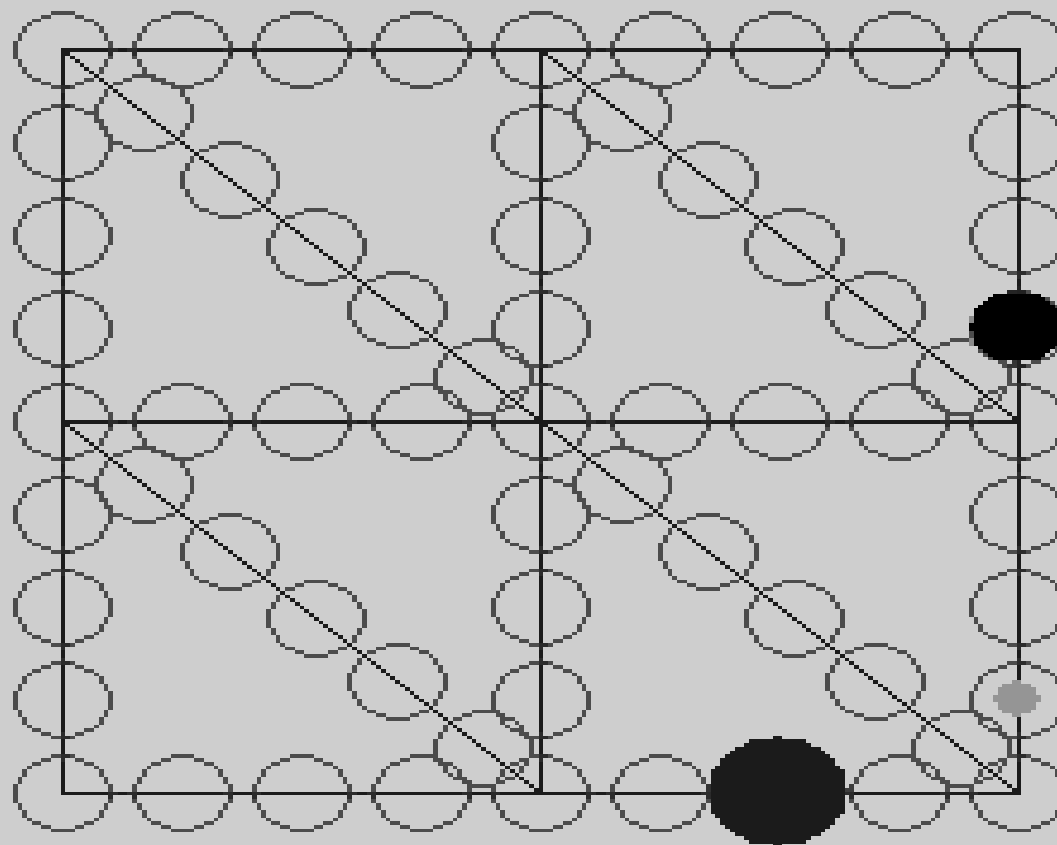




Co-Evolution

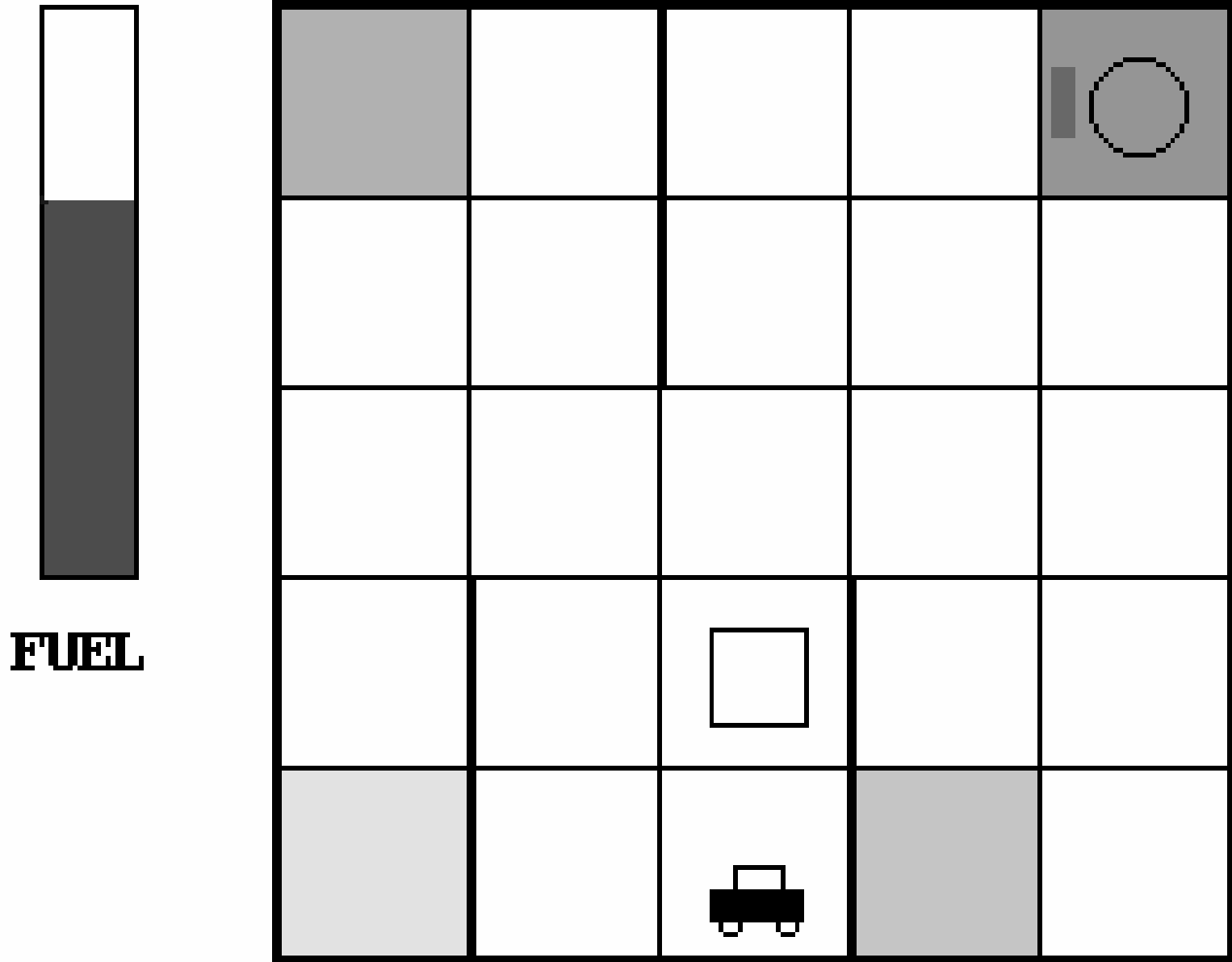


NICTA Project LEAR



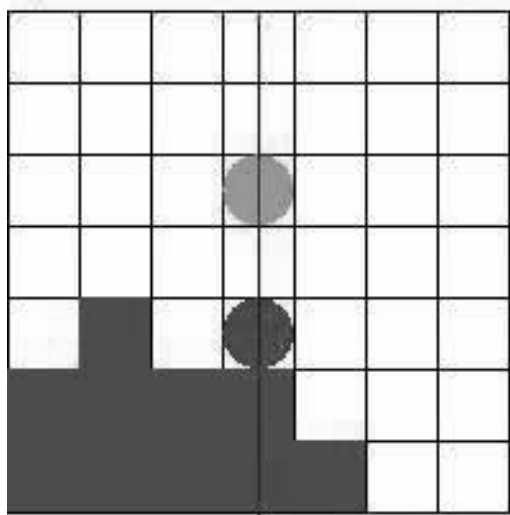
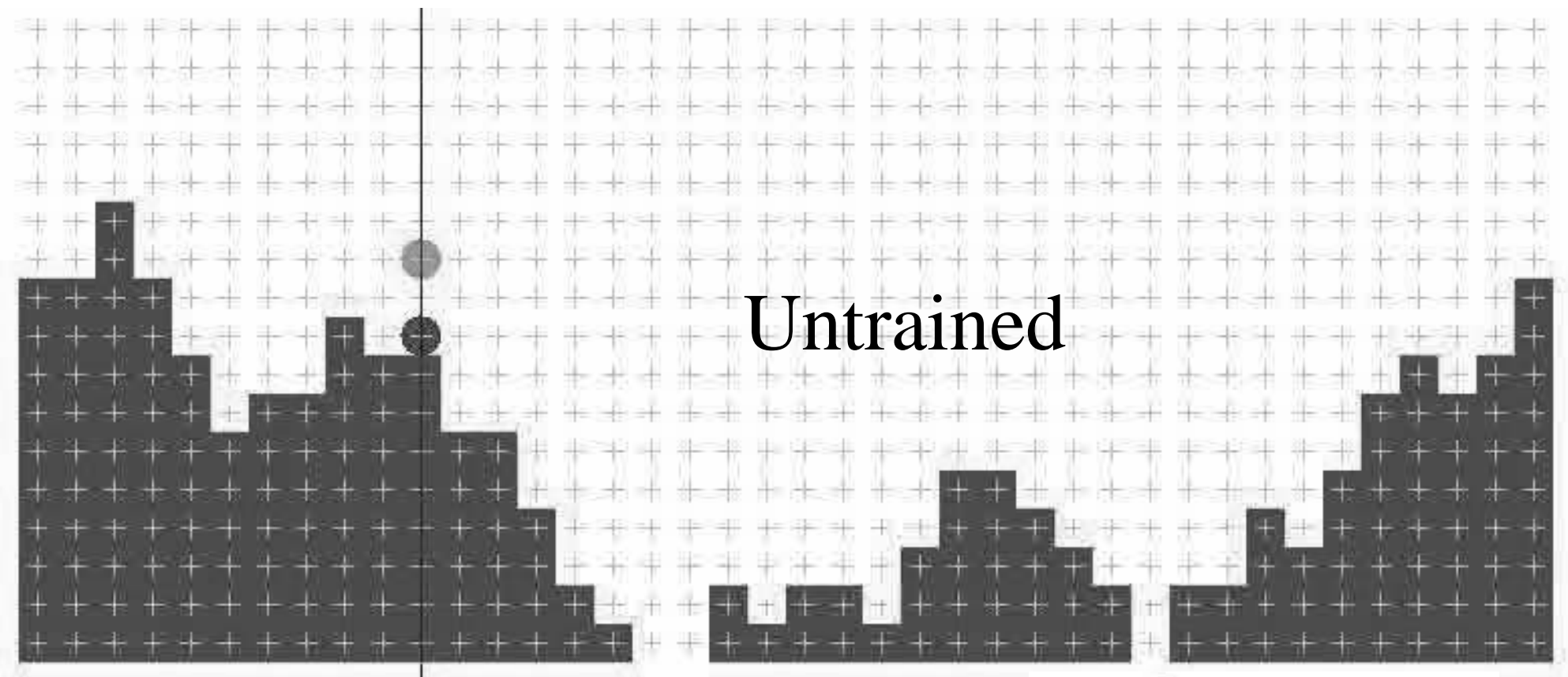
Adonis Antoniades 6 May 2004

The Taxi Domain with Fuel

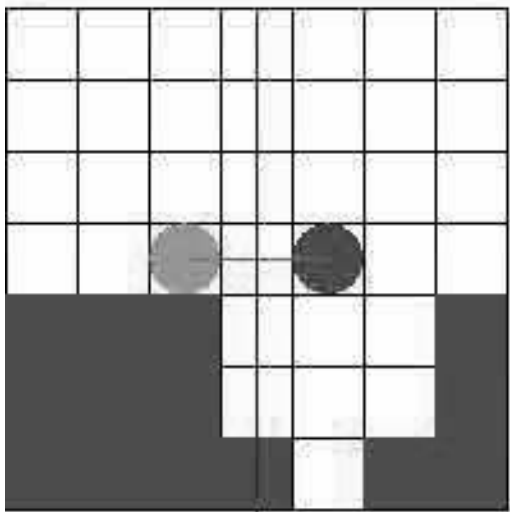
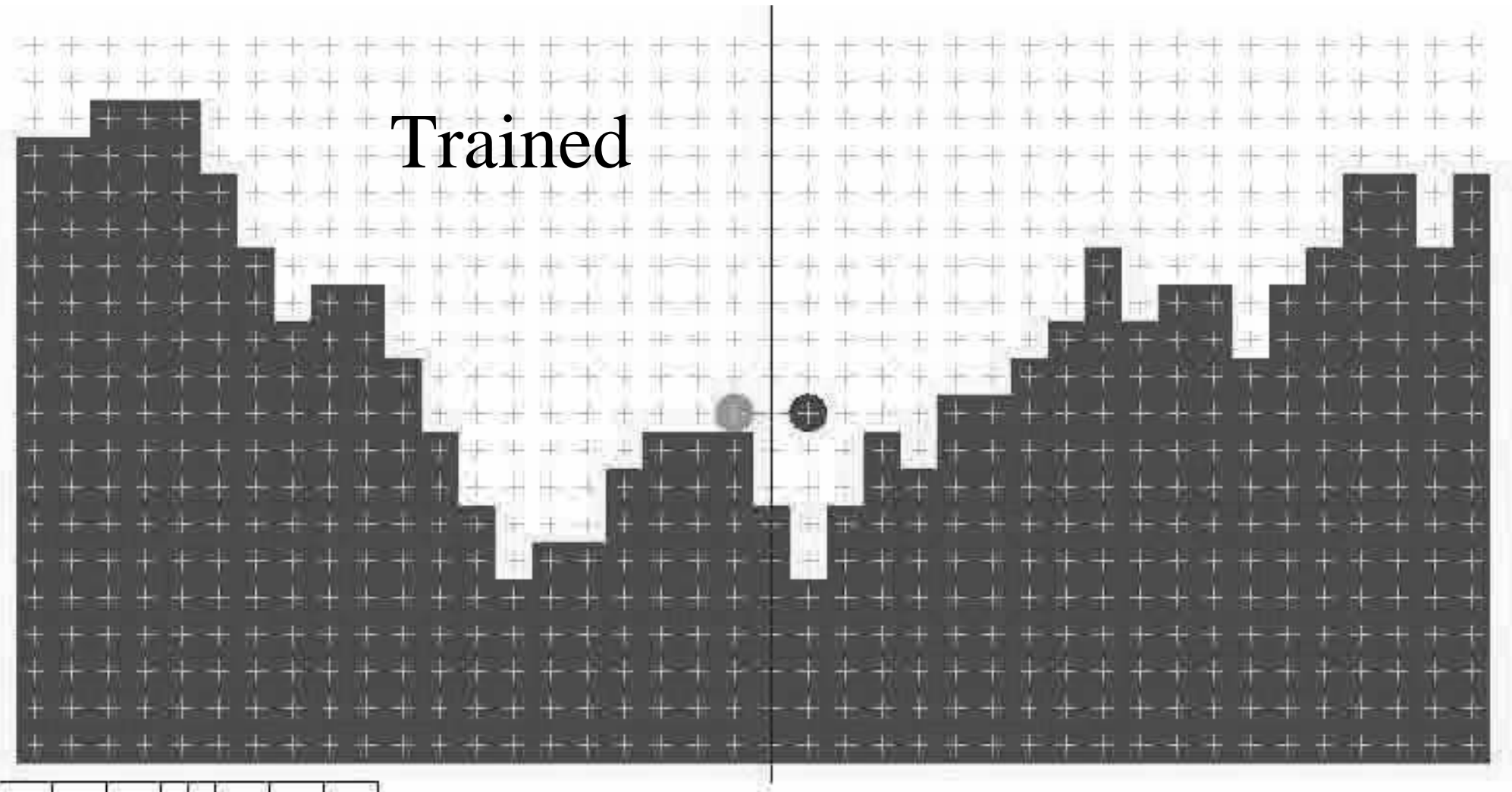


6500 states

Raymond Sheh's Flipper 2005

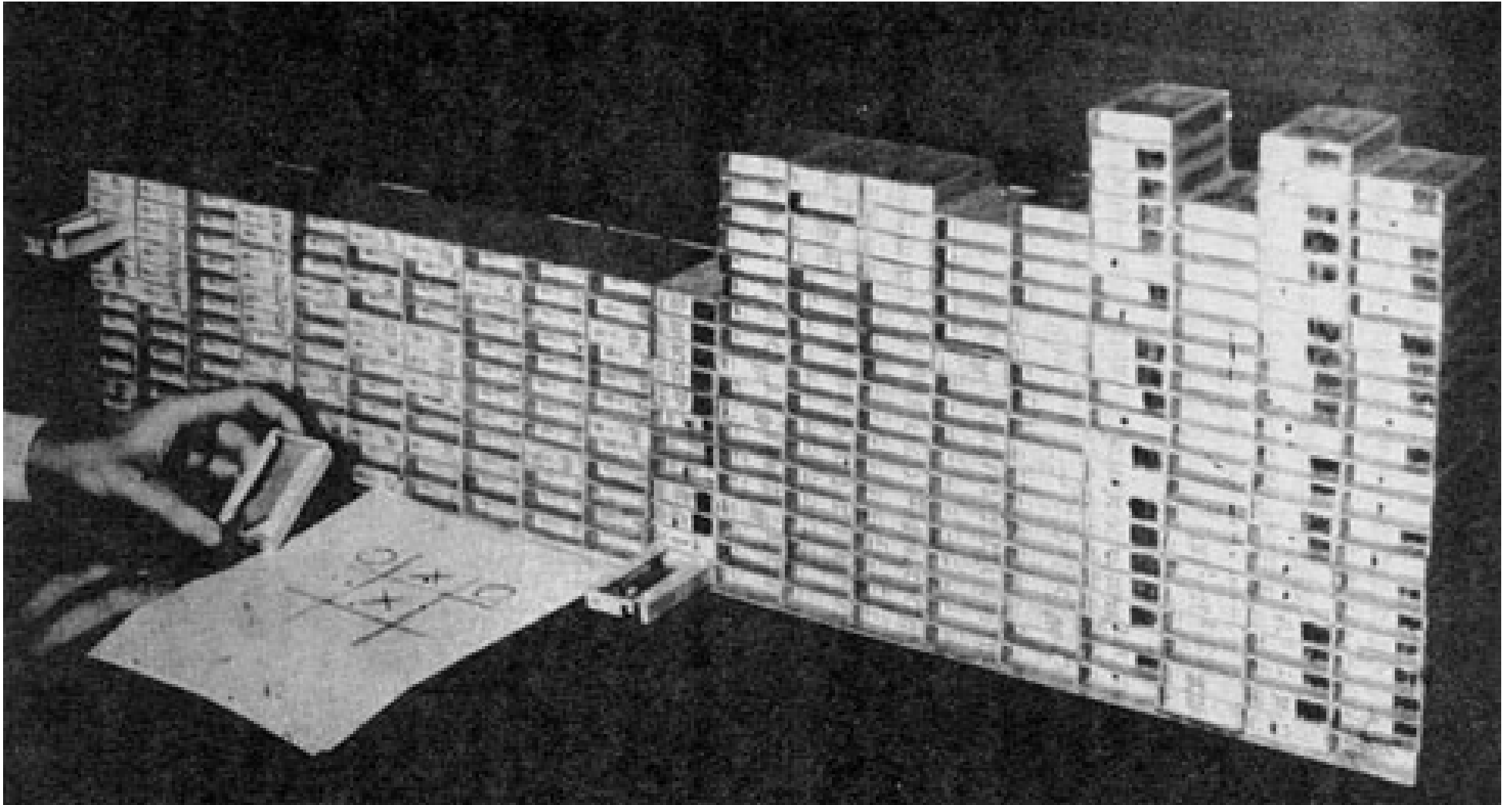


Trained



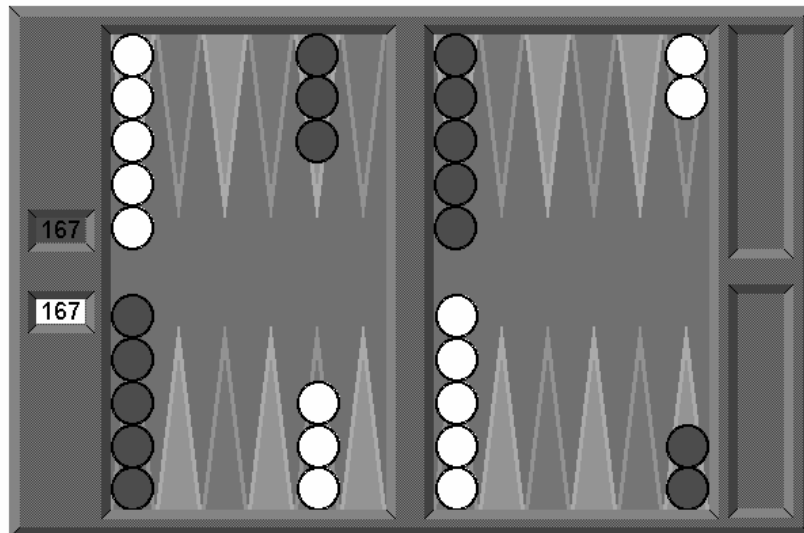
MENACE

(Machine Educable Noughts and Crosses Engine – D.Michie, 1961)



RL Examples

- Checker Player [Arthur Samuel, 1959,1967]
- Trial and Error [Michie, 1961]
- TD-Gammon [Tesauro, 1995]
- Job-shop Scheduling [Zang and Dietterich, 1995, 1996]
- Elevator Dispatching [Crites & Barto, 1996]
- Dynamic Channel Allocation [Singh and Bertsekas, 1997]
- KeepAway Soccer [Stone and Sutton, 2001]



References

Chapter 13, *Machine Learning* T. Michell, 1997

Reinforcement Learning, An Introduction

Richard R Sutton and Andrew G. Barto,

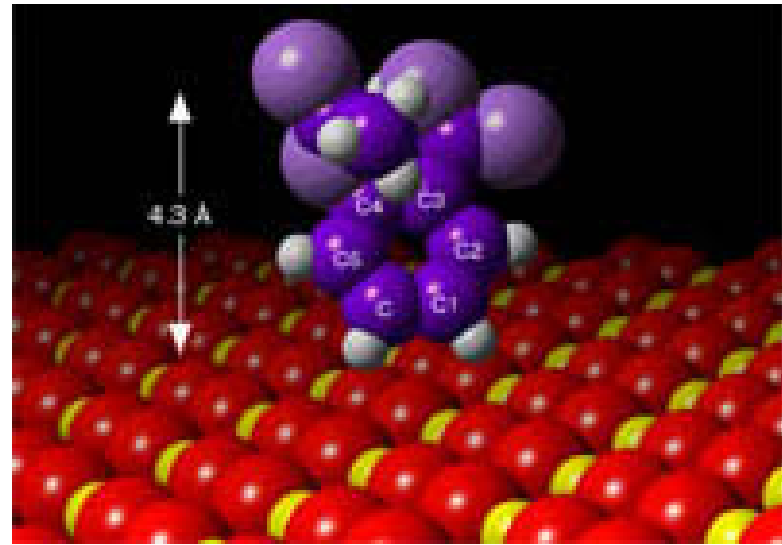
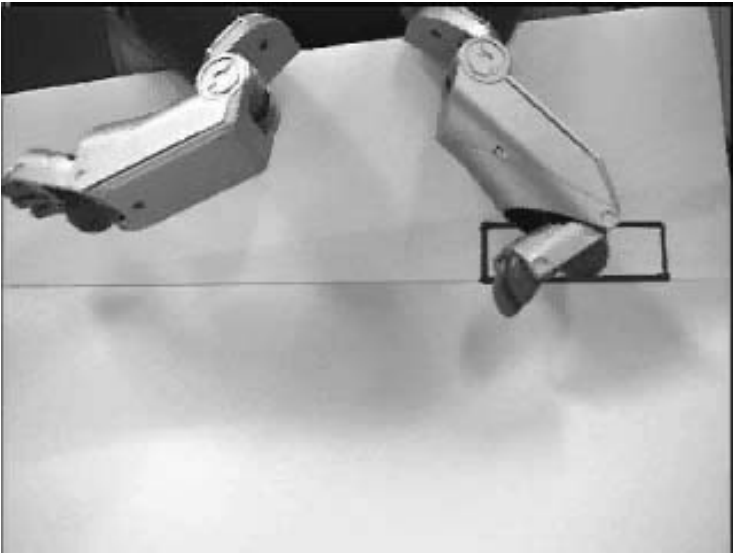
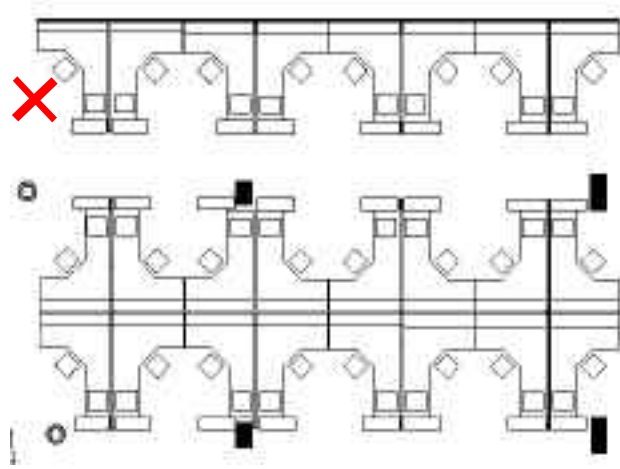
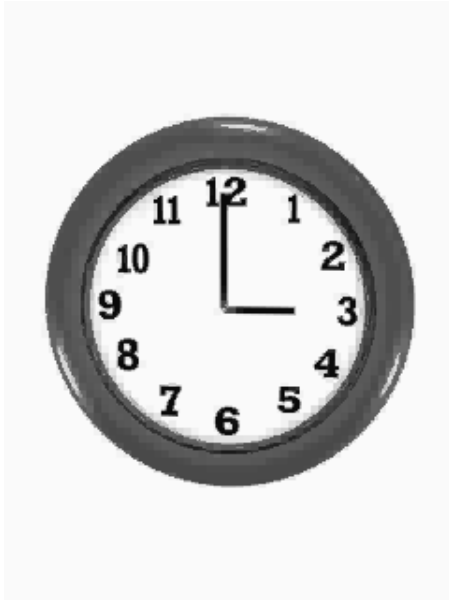
MIT Press 1998

Issues with 'Flat' RL

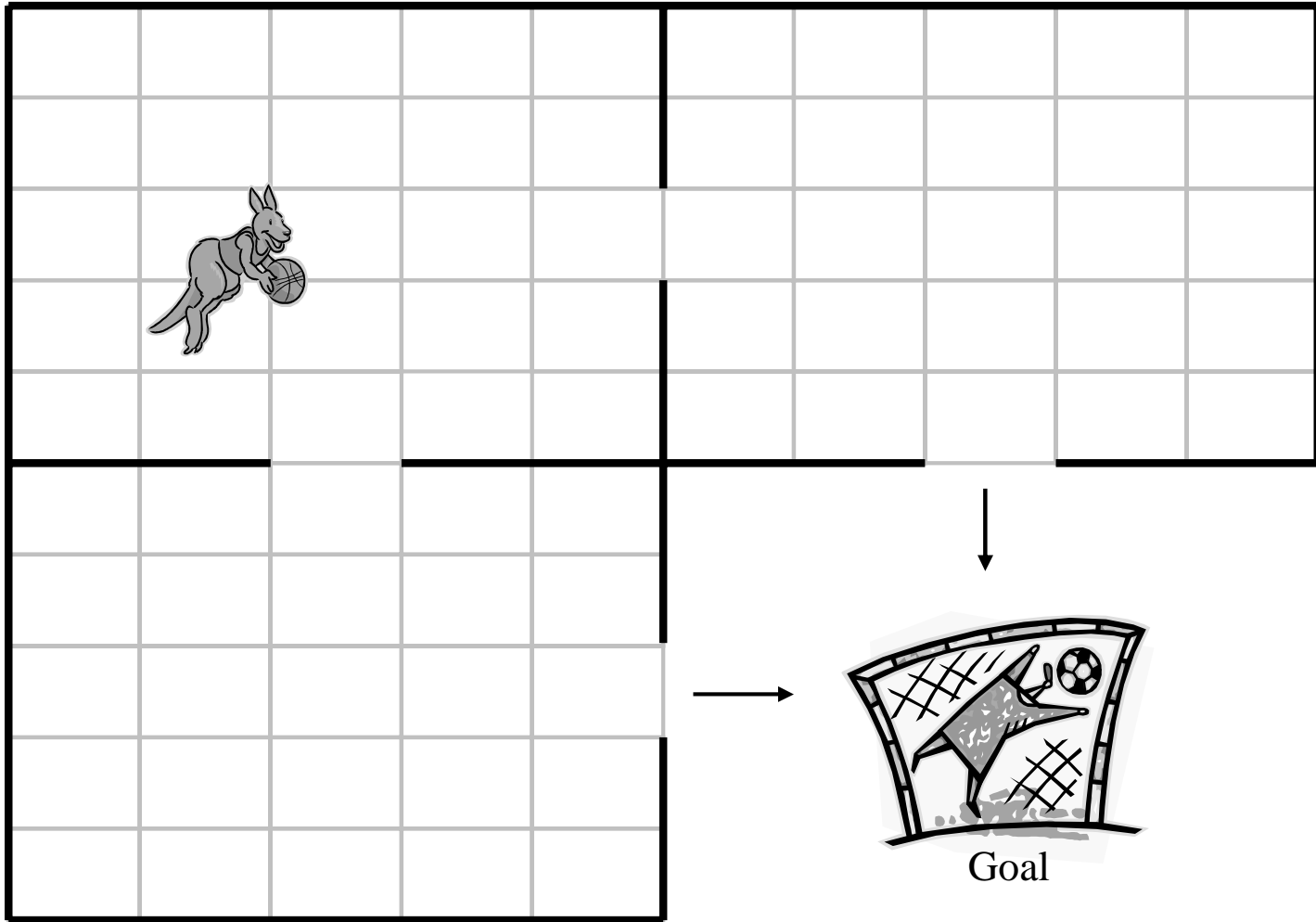
- Long time to converge.
- Huge memory requirements
- Poor generalisation over states and actions
- Hidden State

Remainder of this lecture will look at how hierarchical RL can address some of these issues.

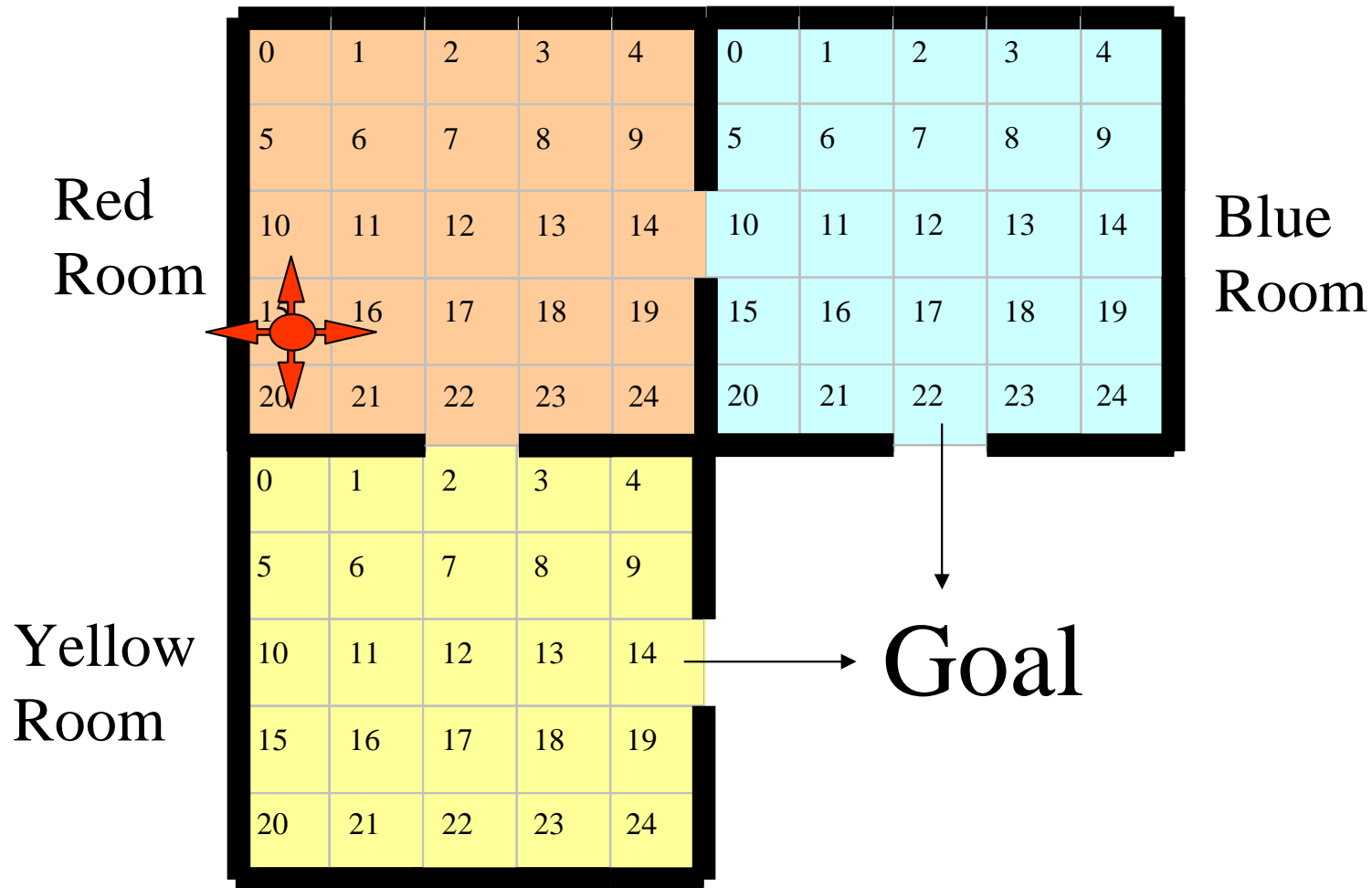
Repetitive Models



An Illustrative Example



Objective: Minimise distance to goal



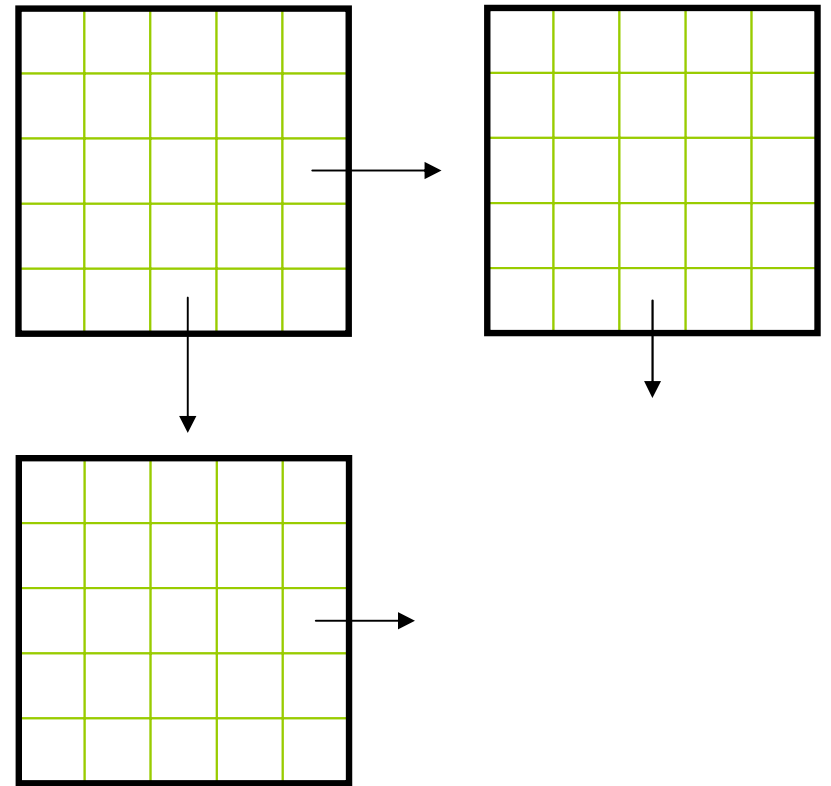
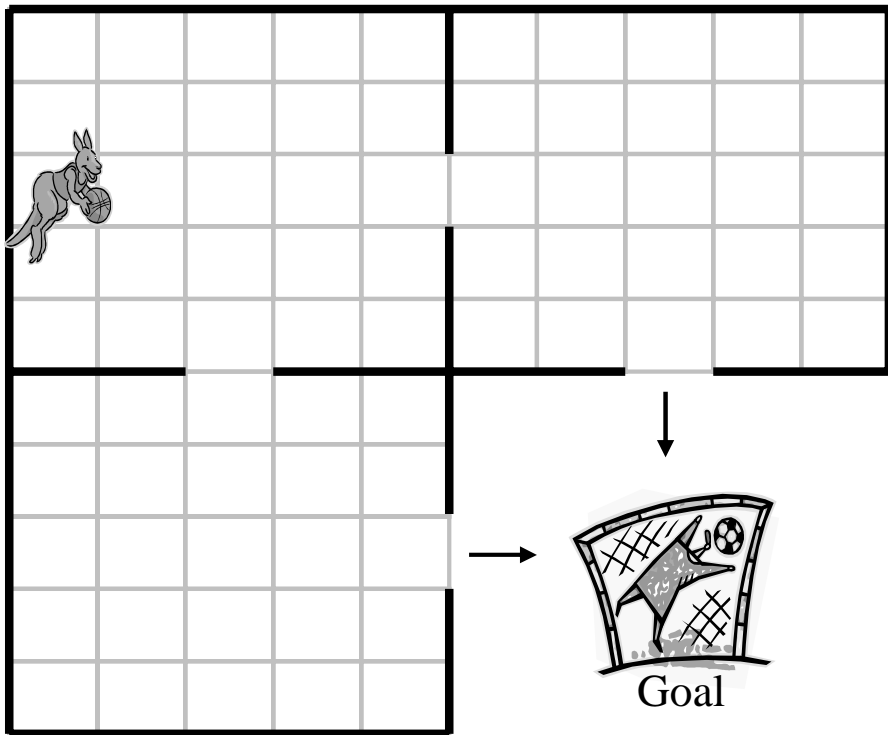
Value Function

11	10	9	8	7	6	5	4	5	6
10	9	8	7	6	5	4	3	4	5
9	8	7	6	5	4	3	2	3	4
8	7	6	7	6	3	2	1	2	3
7	6	5	6	7	2	1	0	1	2
6	5	4	3	2			↓		
5	4	3	2	1					
4	3	2	1	0	→				
5	4	3	2	1					
6	5	4	3	2					

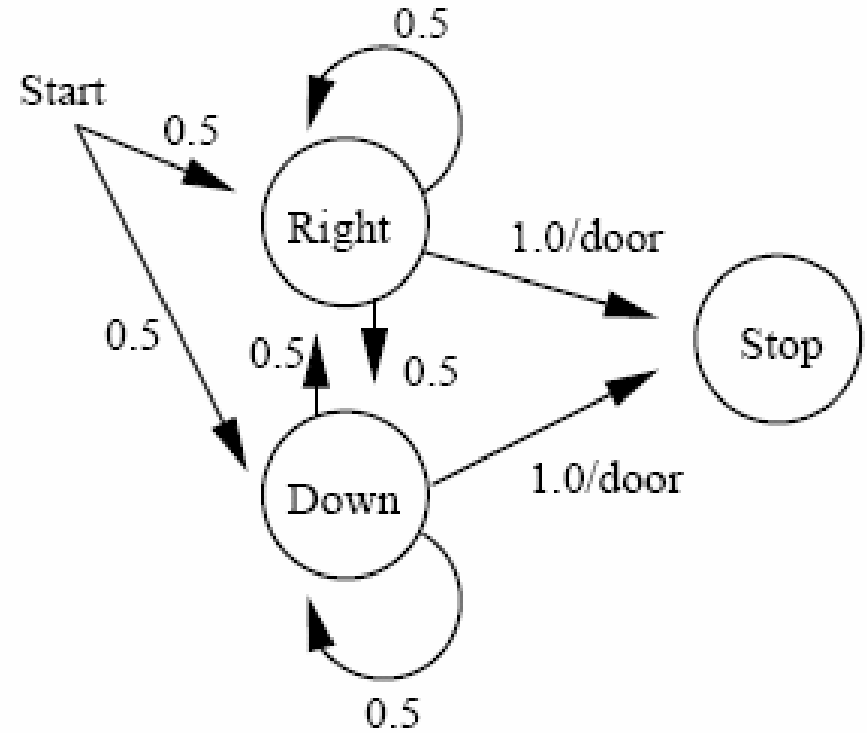
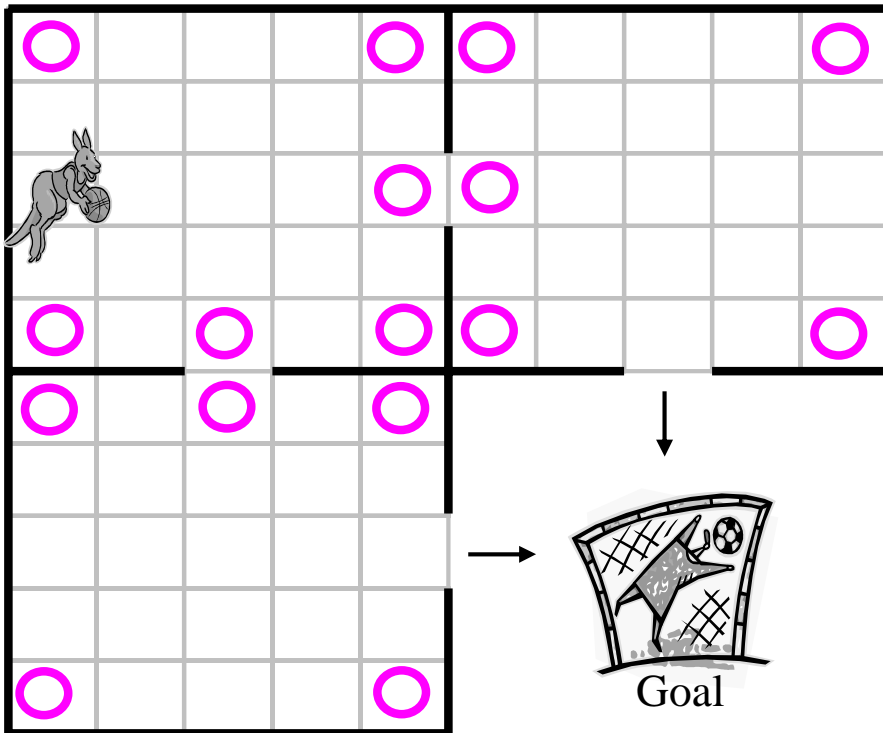
Basis of HRL

- Abstract Actions (or temporally extended actions)
 - Macros
 - Options (Sutton, R.S., Precup, D.)
 - Machines (R. Parr - HAMQ)
 - Sub-tasks (T. Dietterich - MAXQ)
- Semi-MDPs: MDPs where the actions are generalised to be temporally extended
- State abstraction

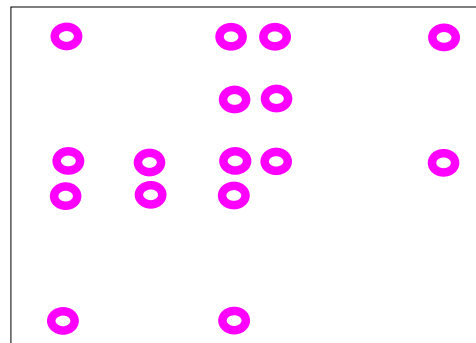
Options $\langle I, \pi, \beta \rangle$



HAMQ

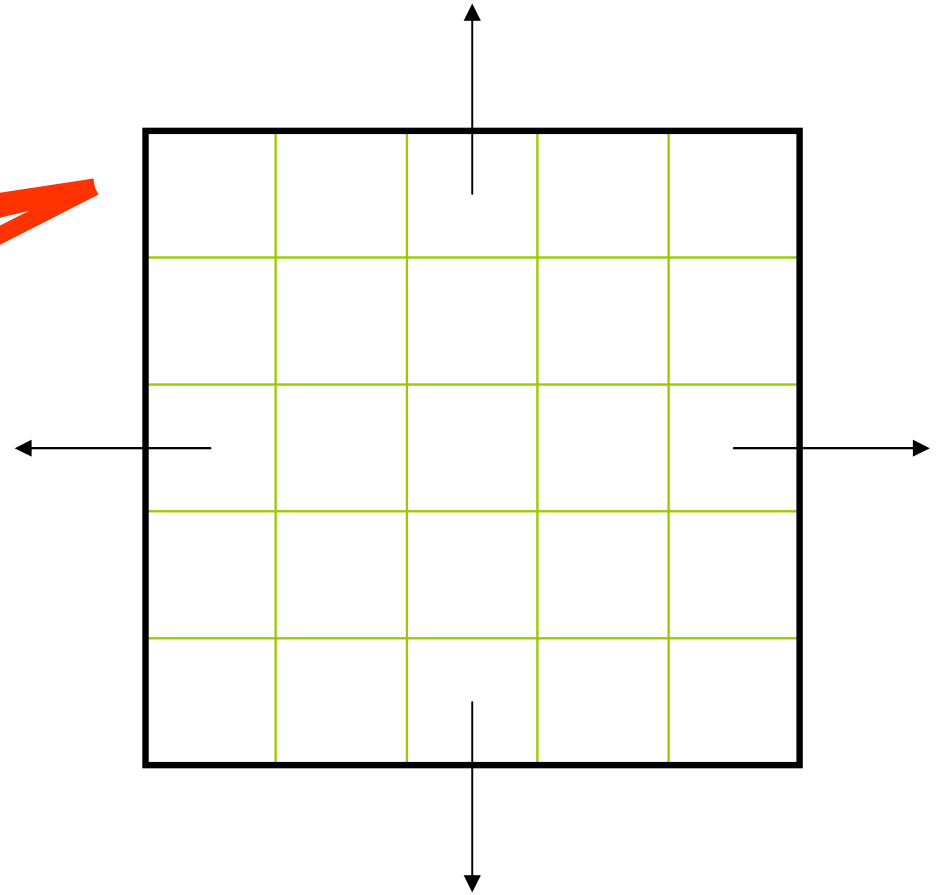
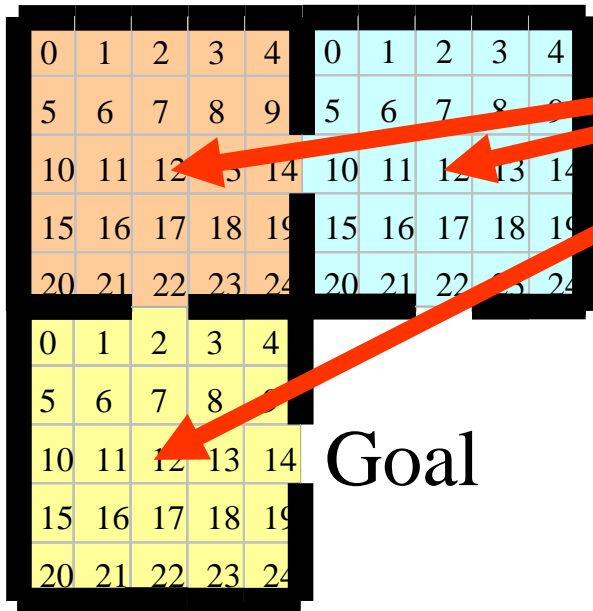


Machine (eg Stochastic Finite State Automata)



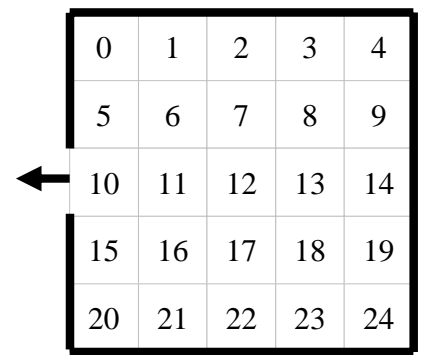
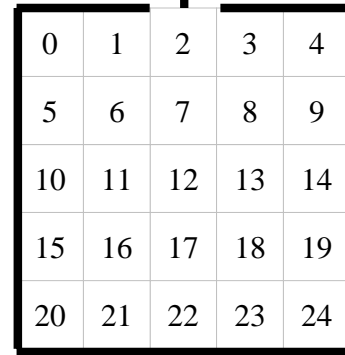
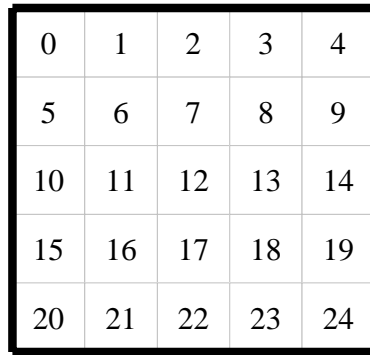
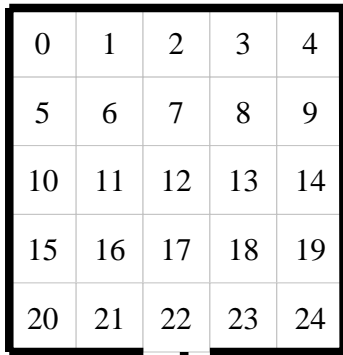
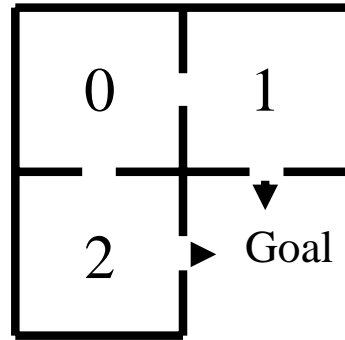
Reduced Abstract Machine

MAXQ

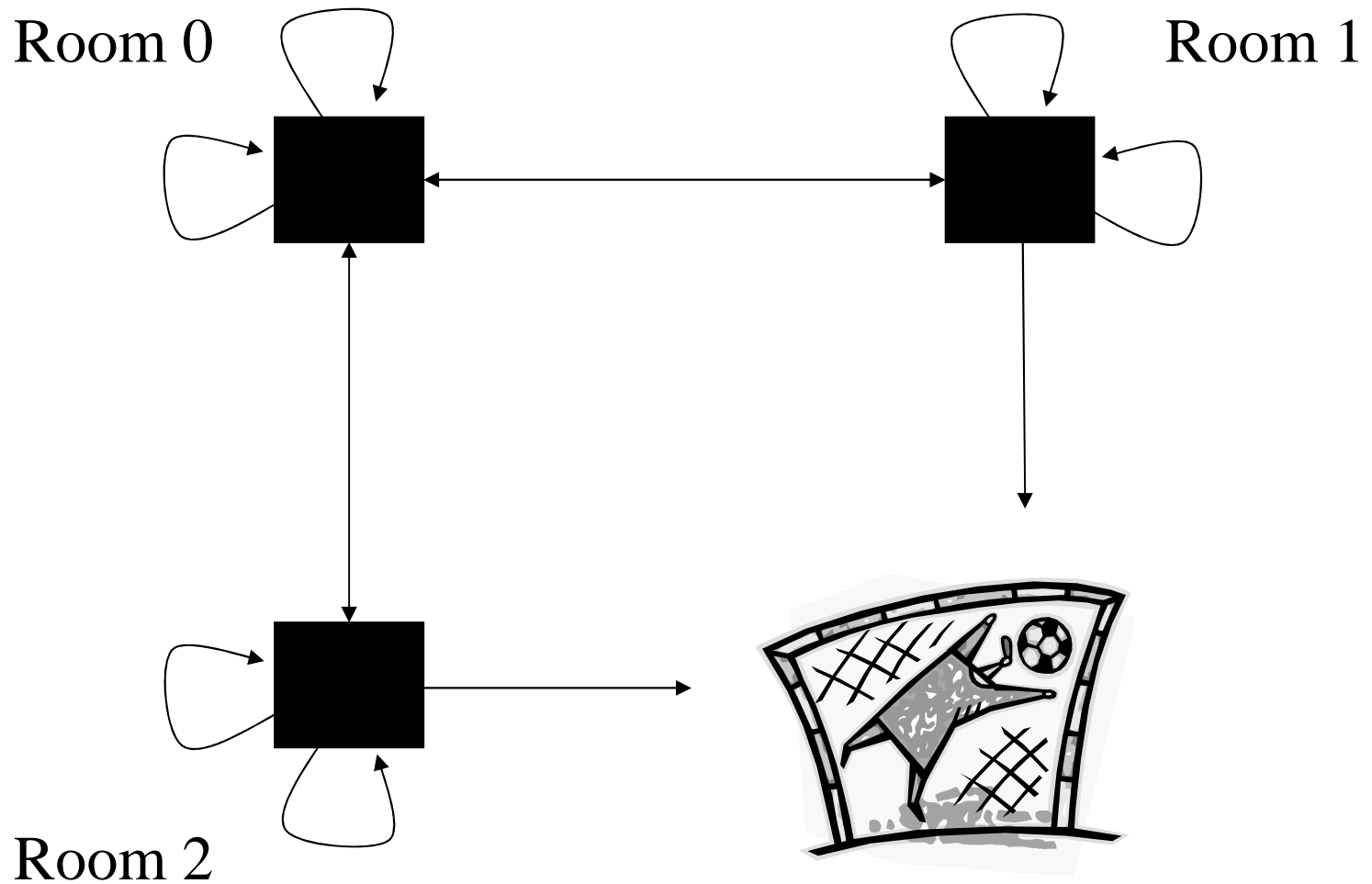


a typical room
Showing possible terminations

MAXQ task hierarchy



The Abstract Problem



Create Top Level Abstract MDP

Level 2
(Colour)

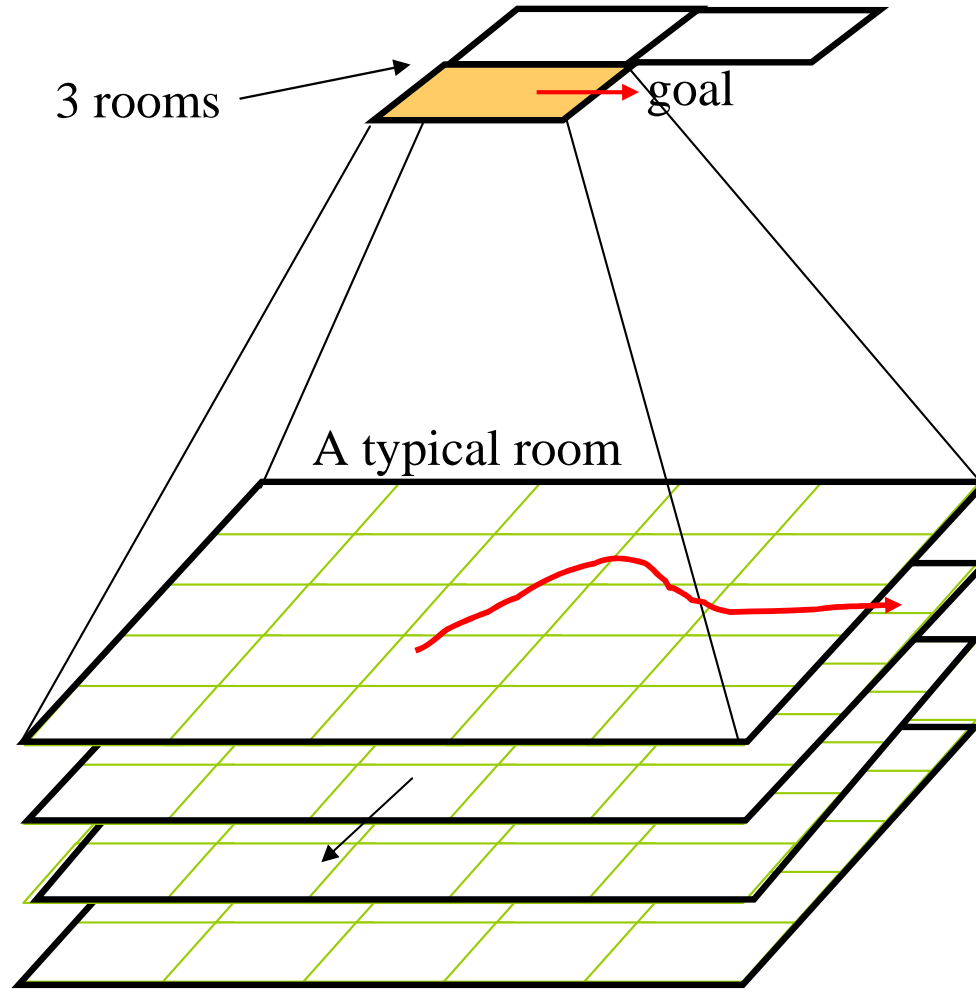
3 rooms

goal

1. Subtask reuse
2. Value Function decomposition
3. State abstraction

A typical room

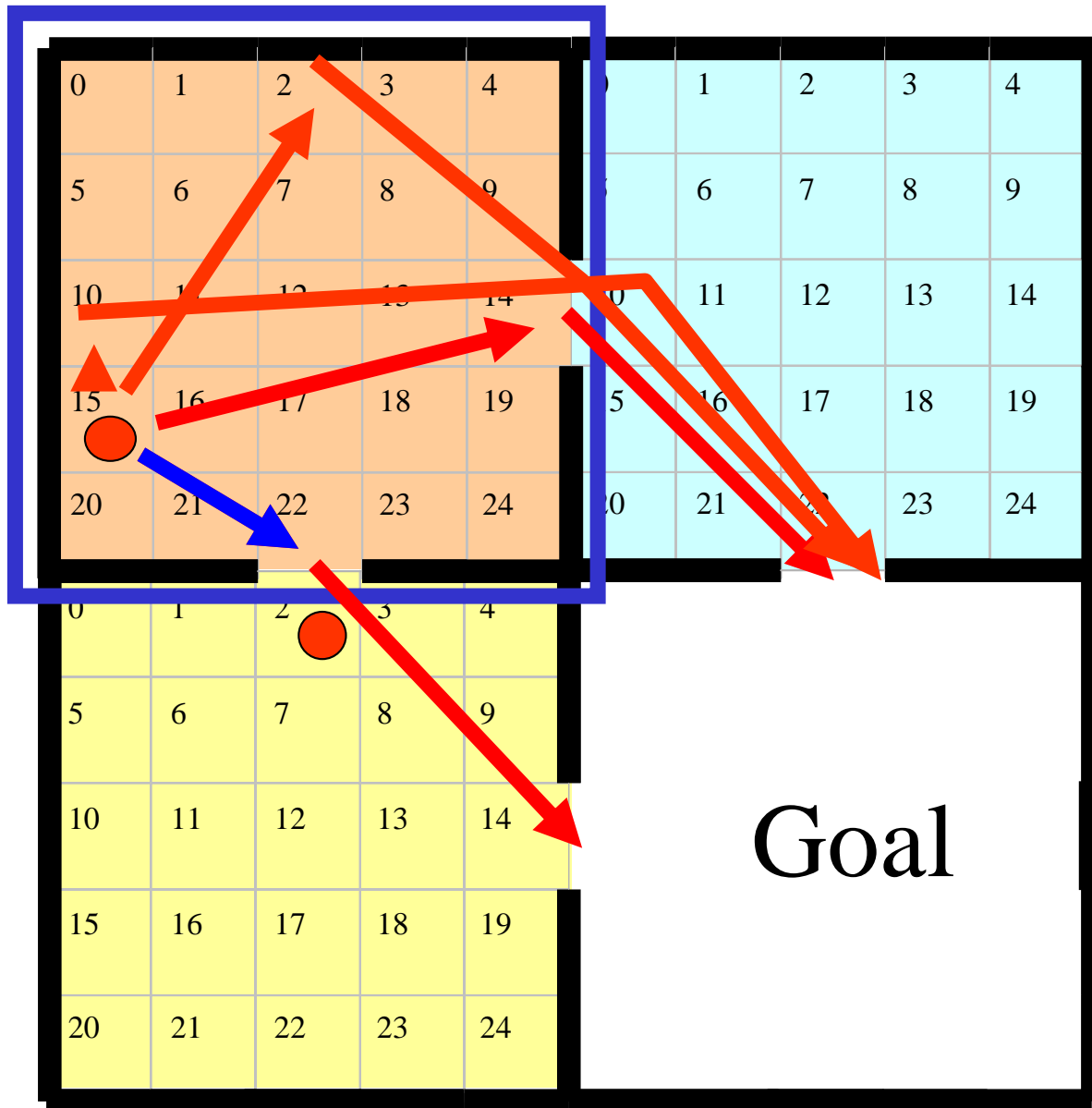
Level 1
(Position)



Rooms become abstract states

Room leaving policies become abstract actions

Hand Simulation



Initial state =
(red room, position 15)

Room level action =

$\operatorname{argmin} \{Q^2(\text{top, red, exit})$
plus $\min Q^1(\text{exit, 15, a})\}$

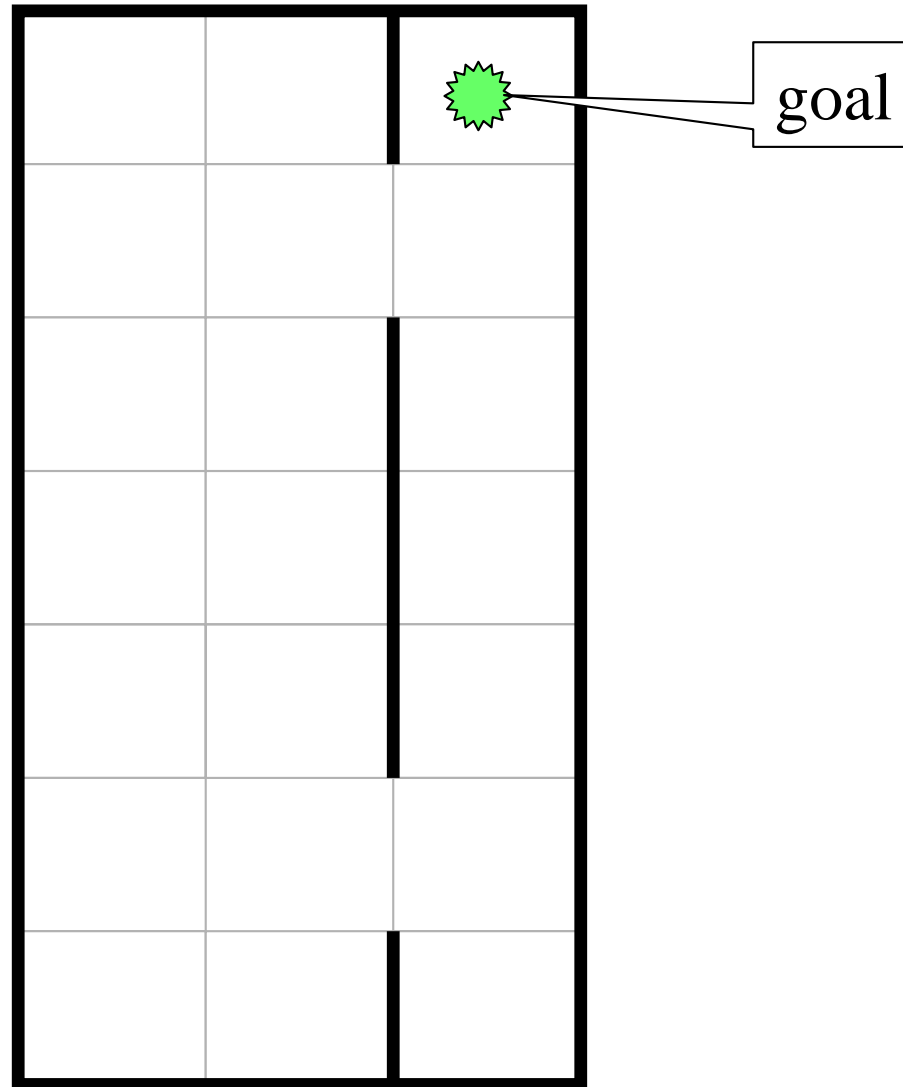
Choose abstract action:
leave-room-at-bottom

Invoke leave-room-at-bottom sub-MDP

Robot executes sub-MDP
policy until it exits

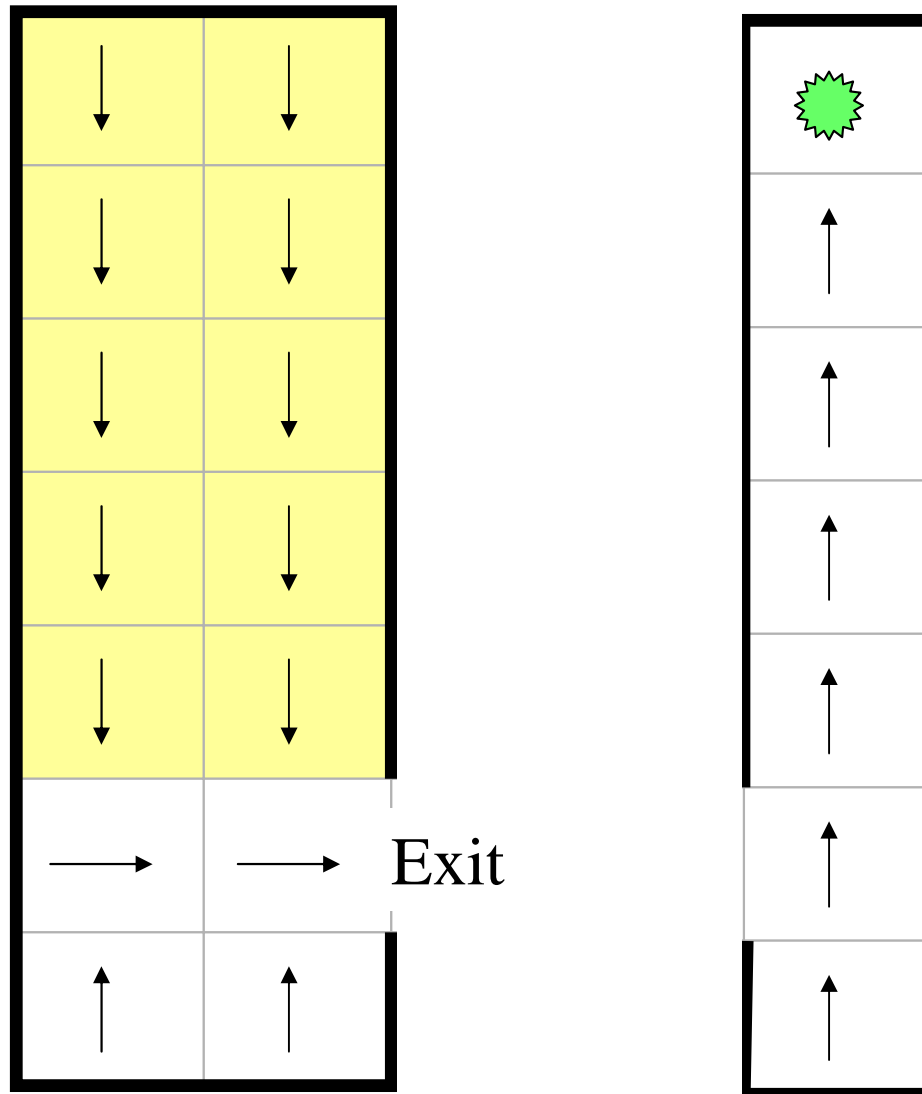
Goal

HRL and Optimality



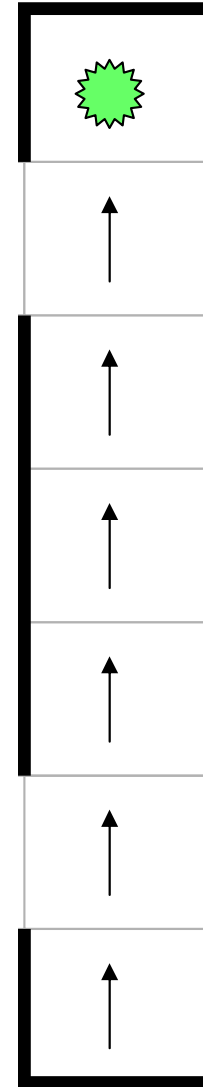
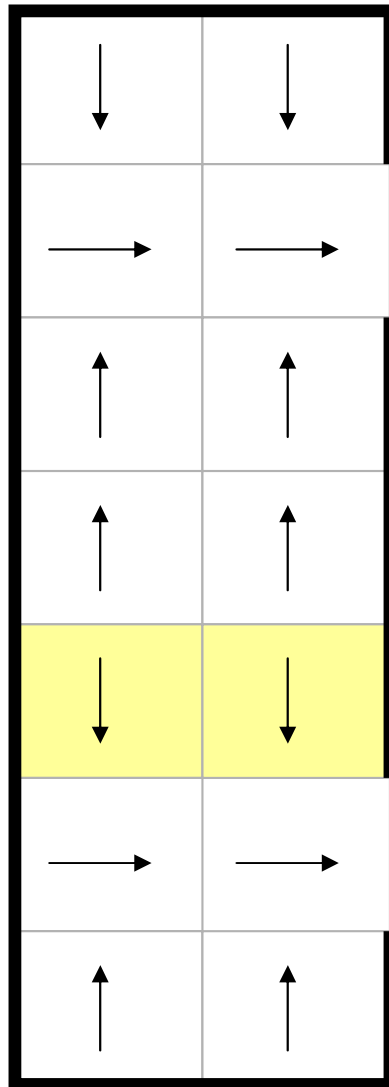
Hierarchically Optimal

(not globally optimal)

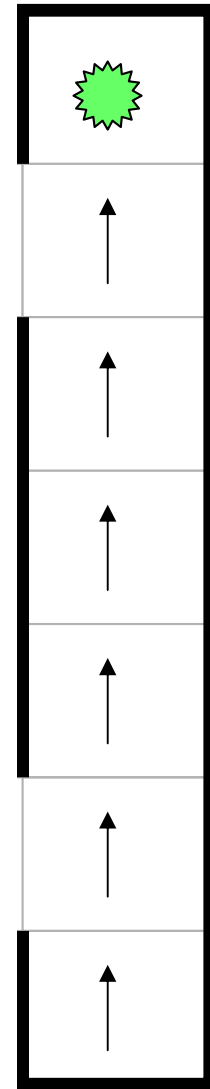
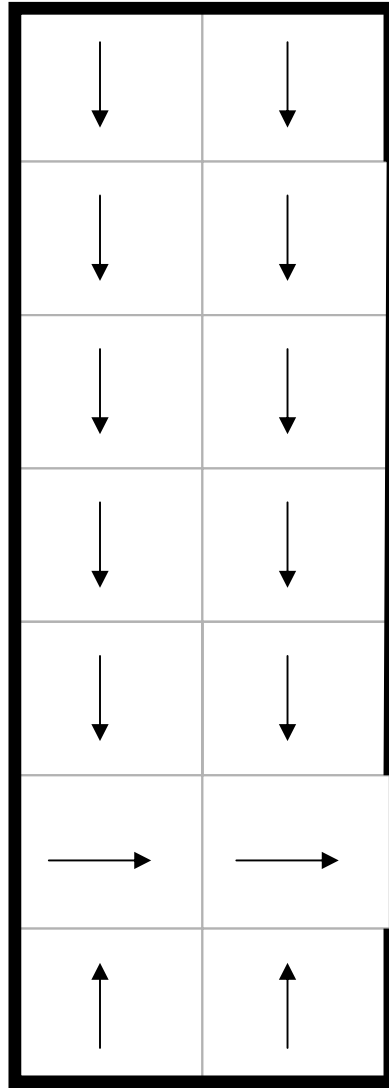
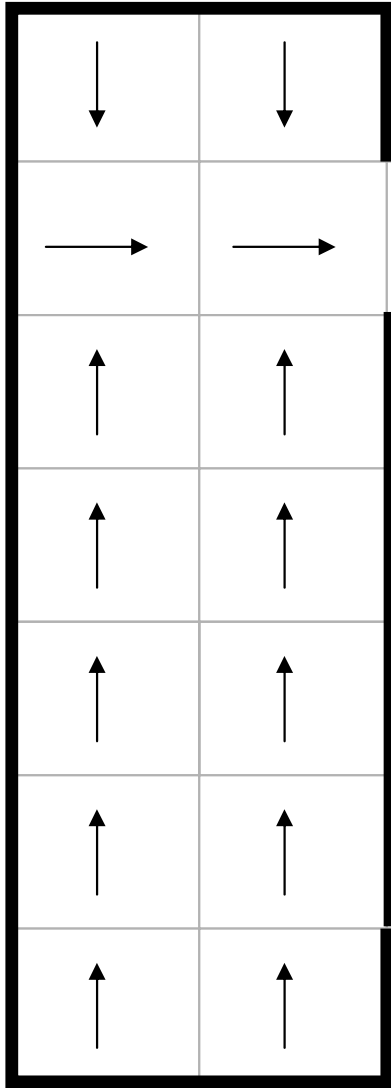


Recursively Optimal

(not globally optimal or hierarchically optimal)



Optimal?



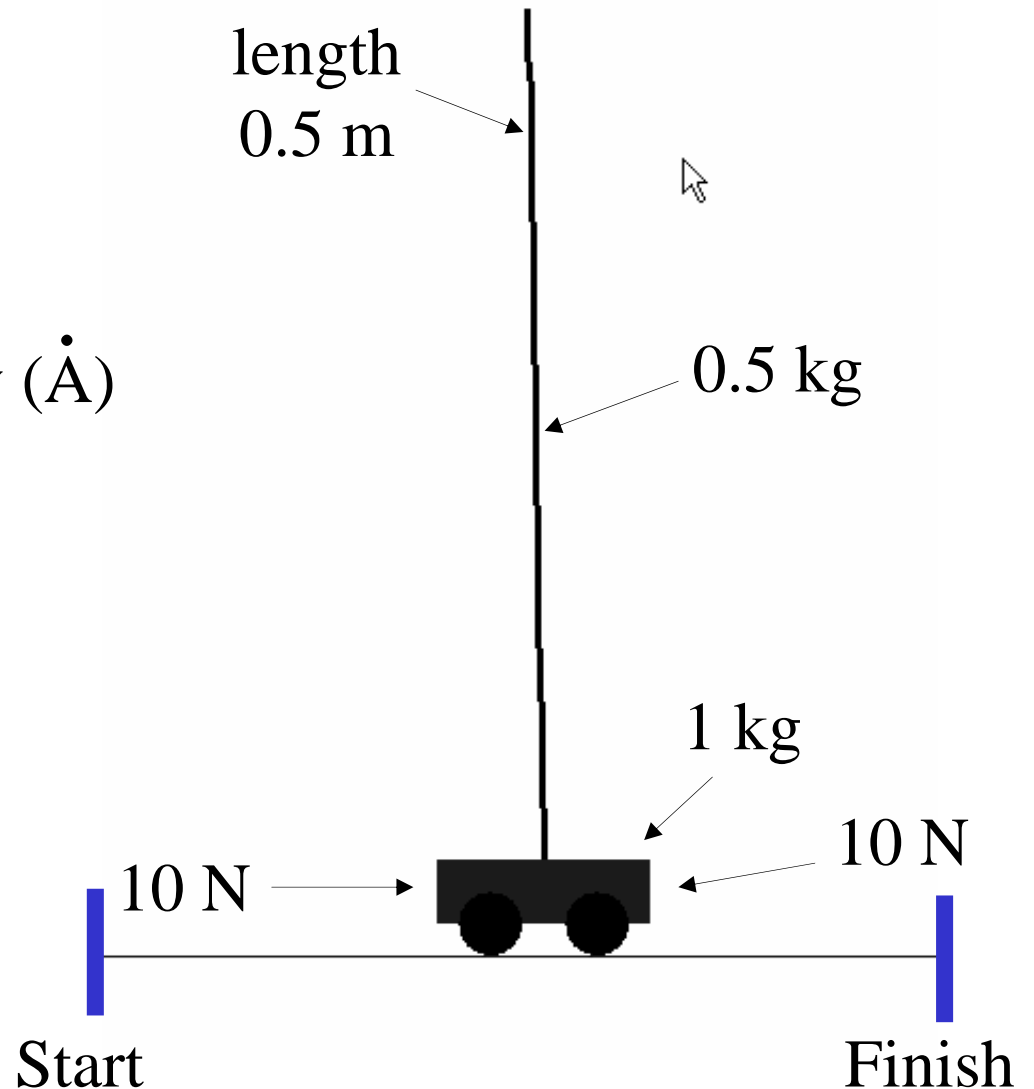
Hierarchical Pole Balancing & Translation

Markov State

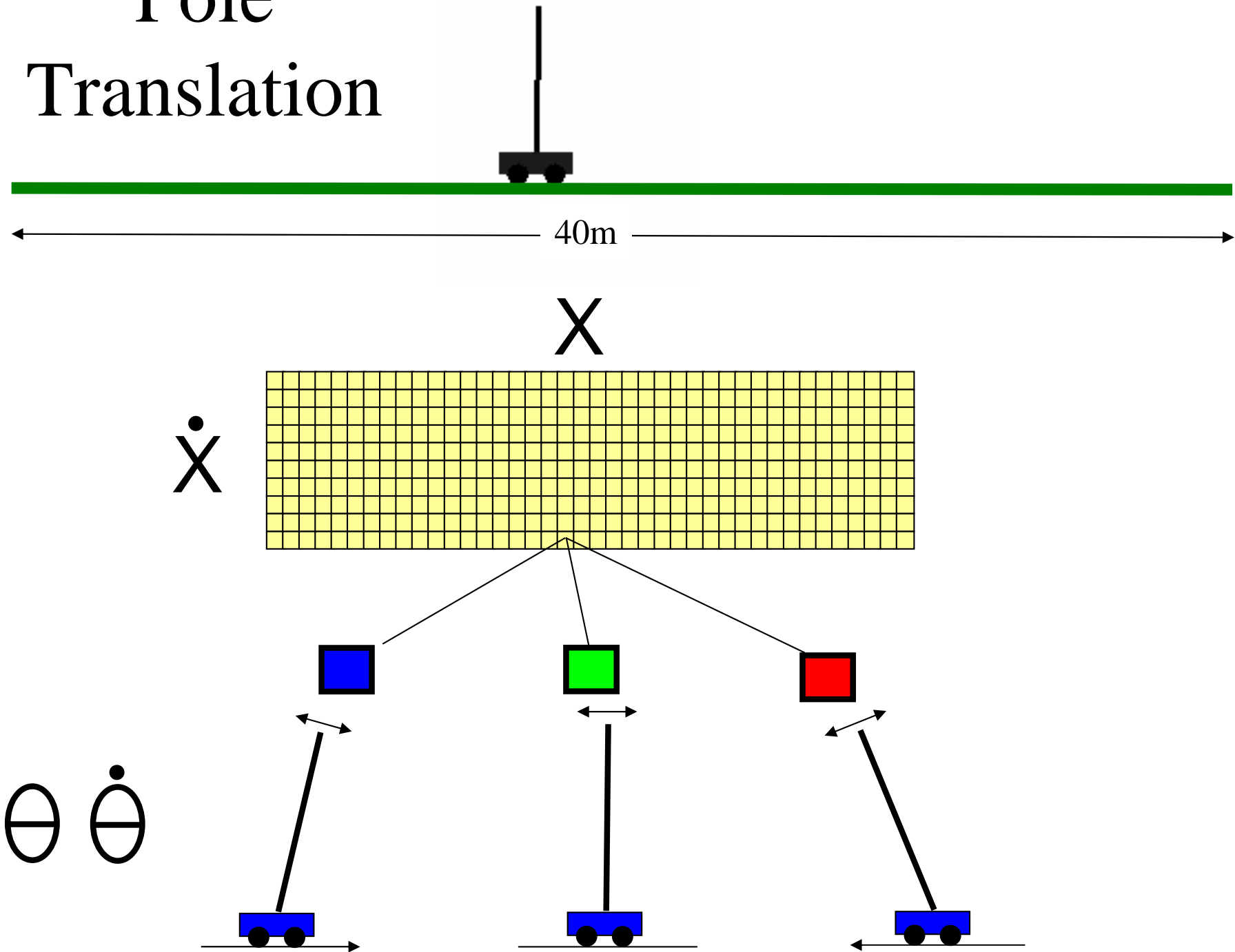
- pole angle (θ)
- pole angular velocity ($\dot{\theta}$)
- cart position (X)
- cart velocity (\dot{X})

Actions

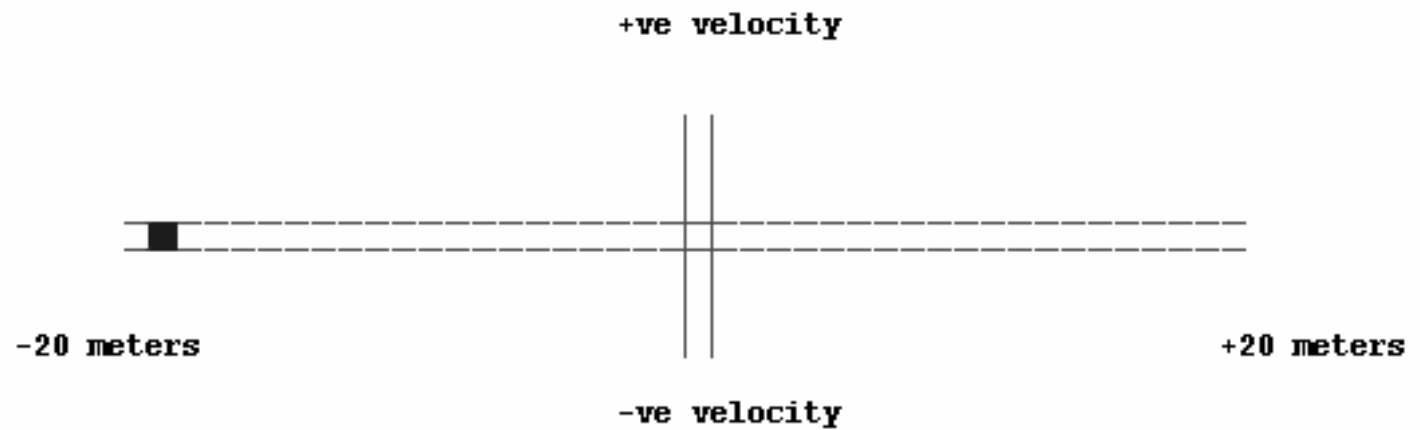
- push left
- push right



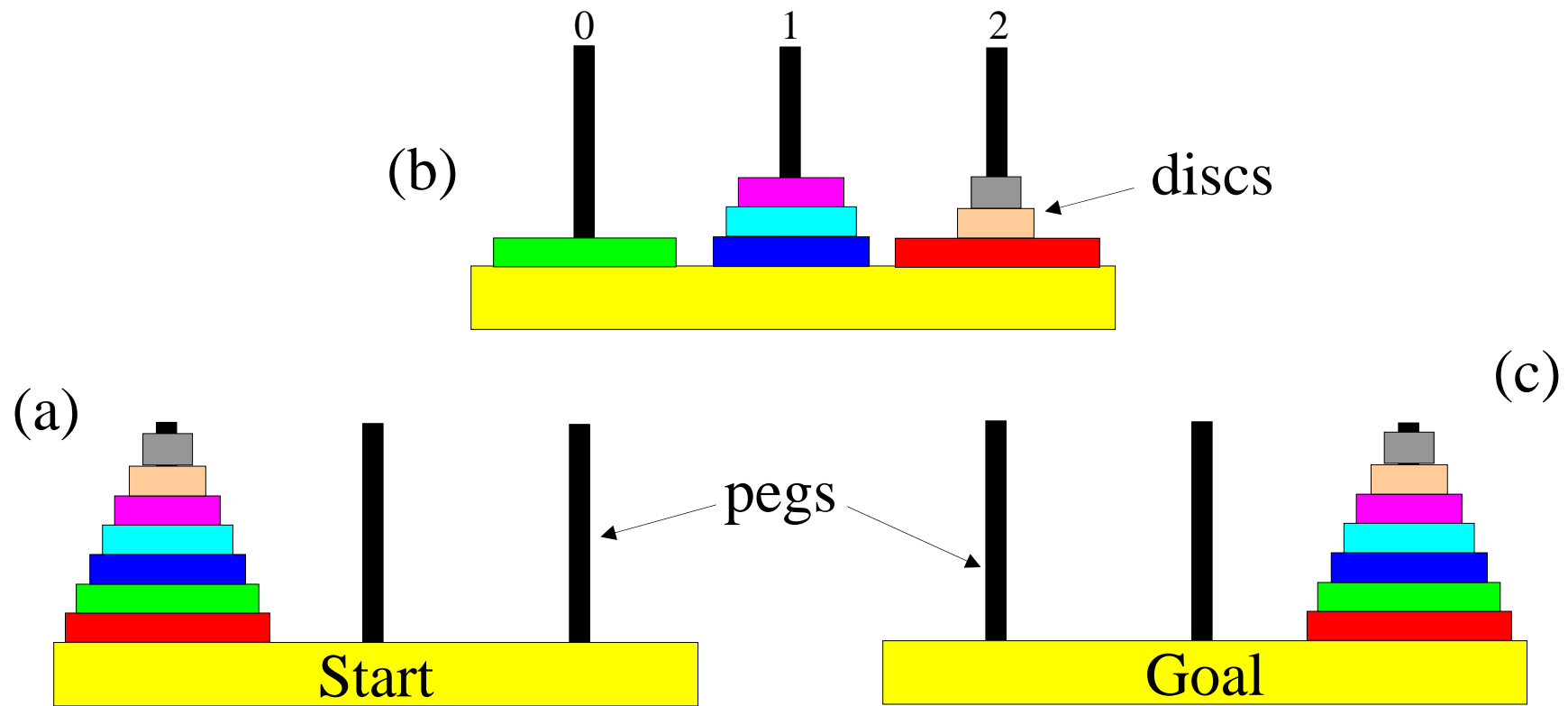
Pole Translation

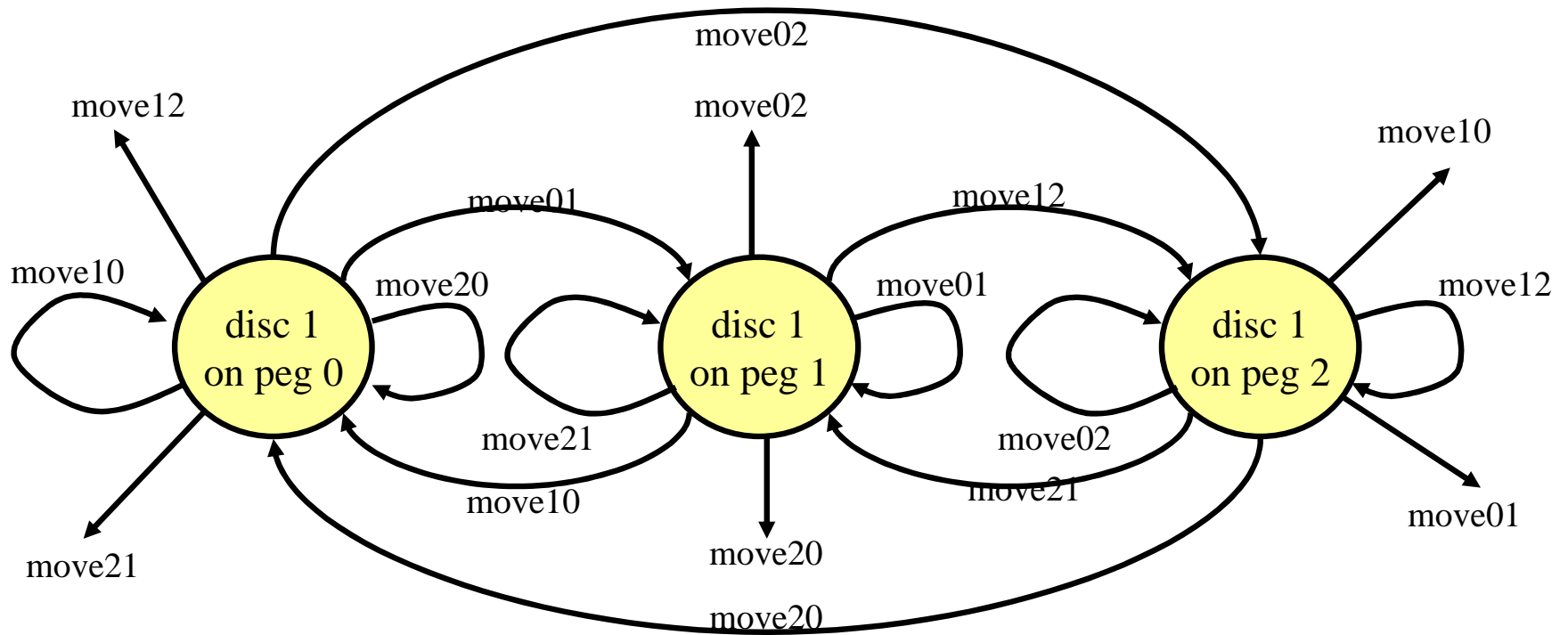
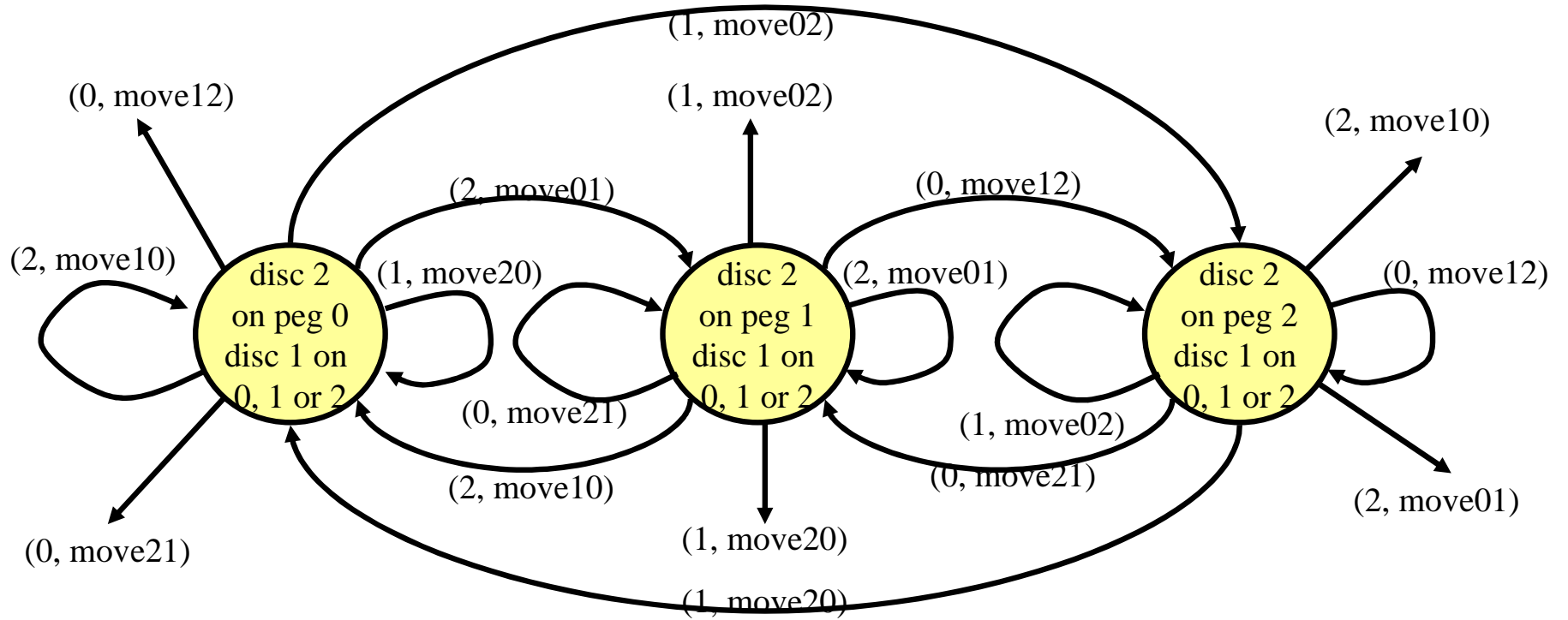


Pole Translation

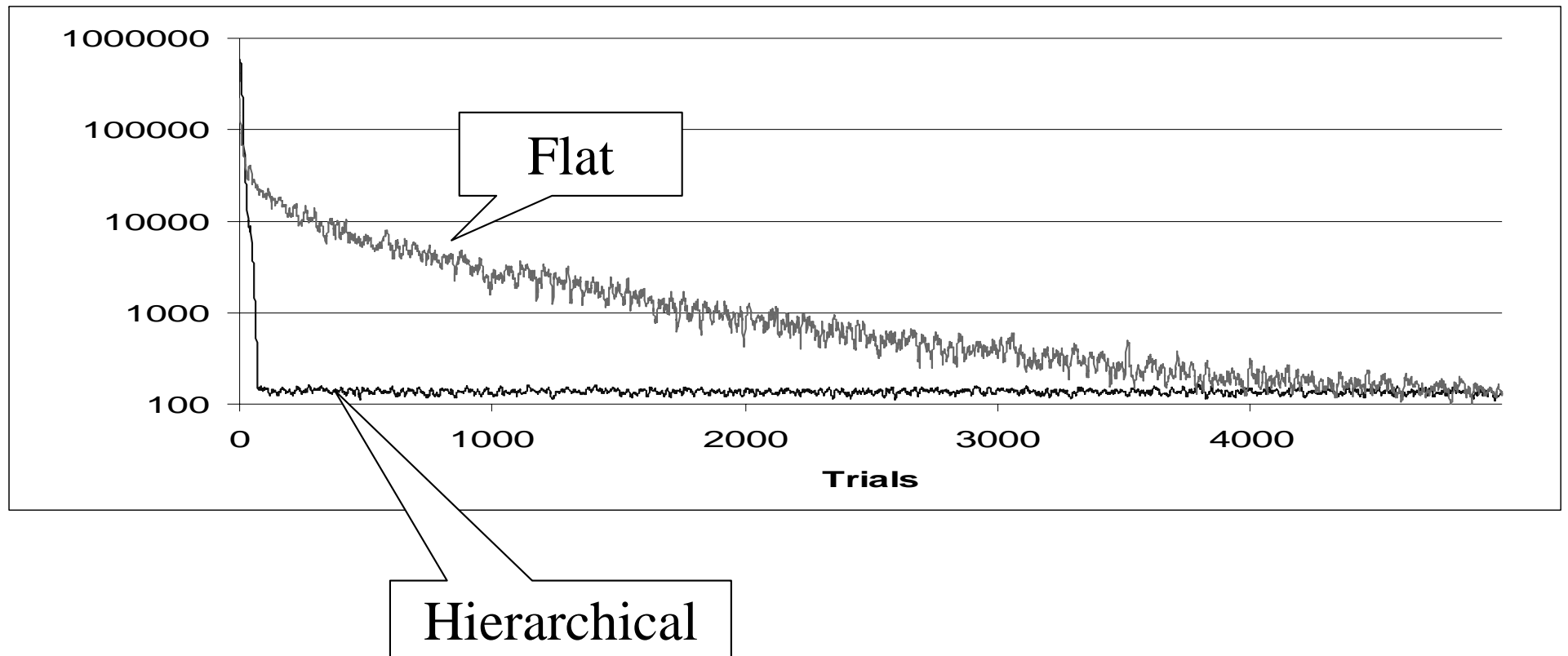


Towers of Hanoi





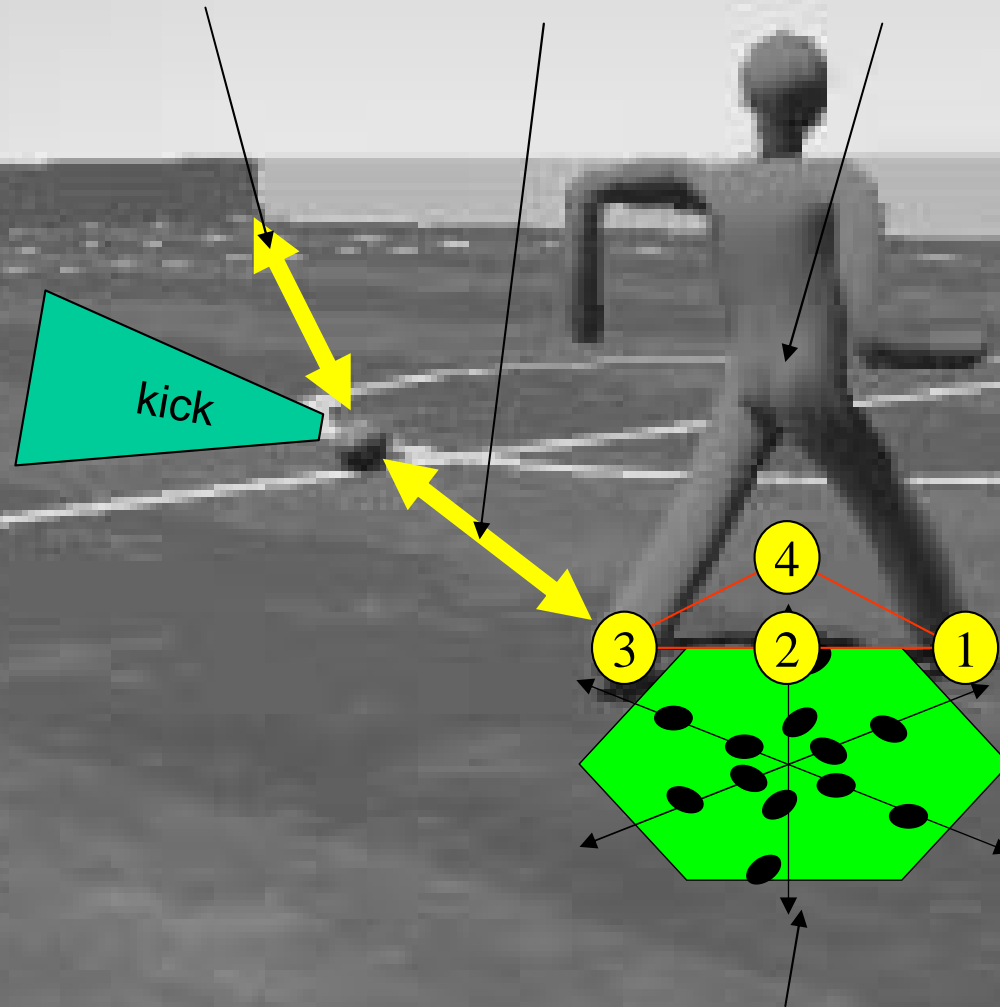
ToH Hierarchical Solution



Simulated Soccer Agent

Sensor = (ball position, ball distance, robot stance)

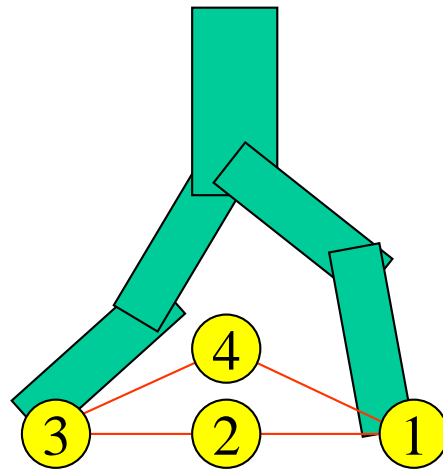
Goal =
Only Reward



Actions = {move legs, turn}

Robot-Stance Variable

State = (ball-position-on-field, robot-relative-to-ball, robot-stance)



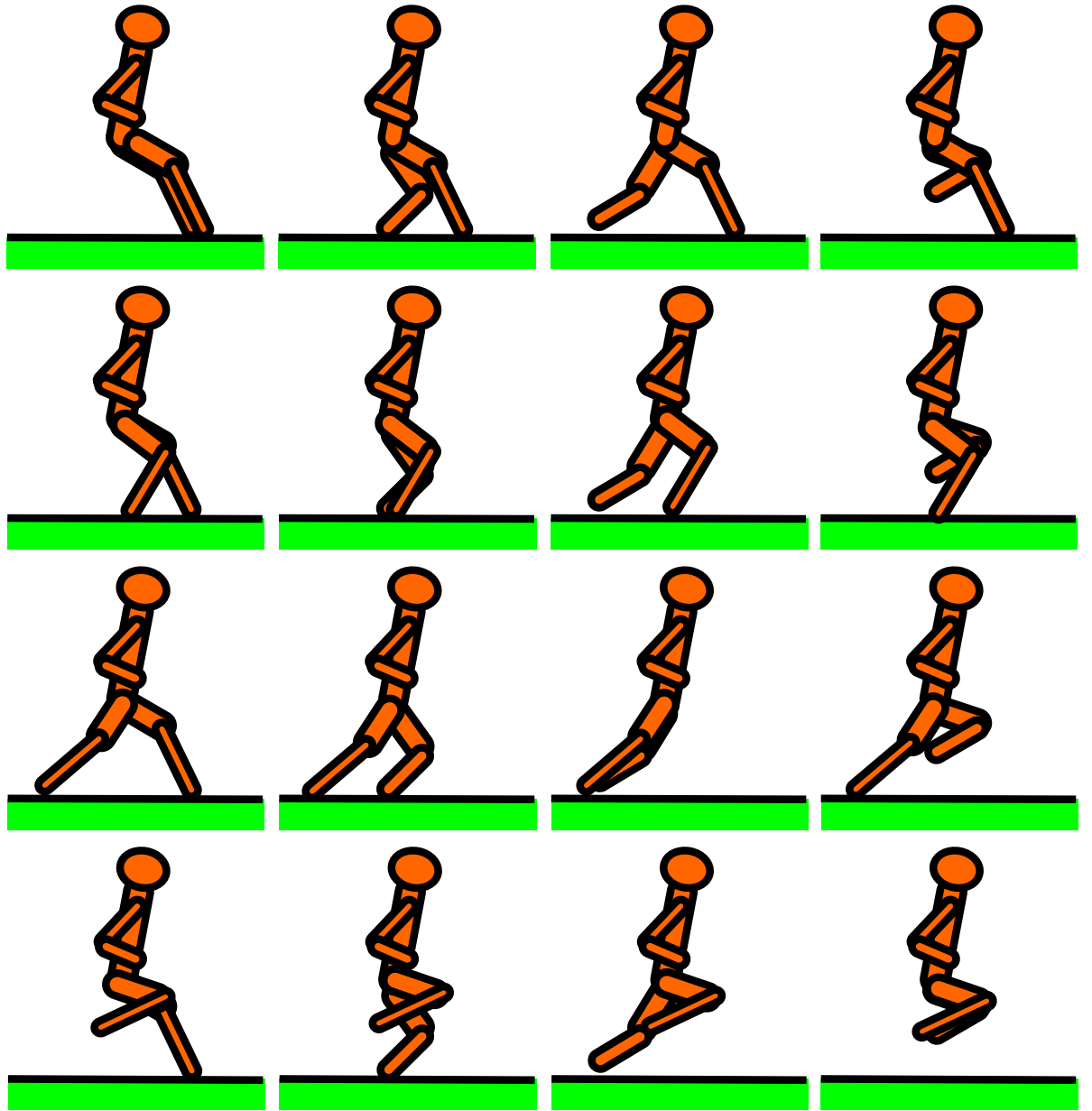
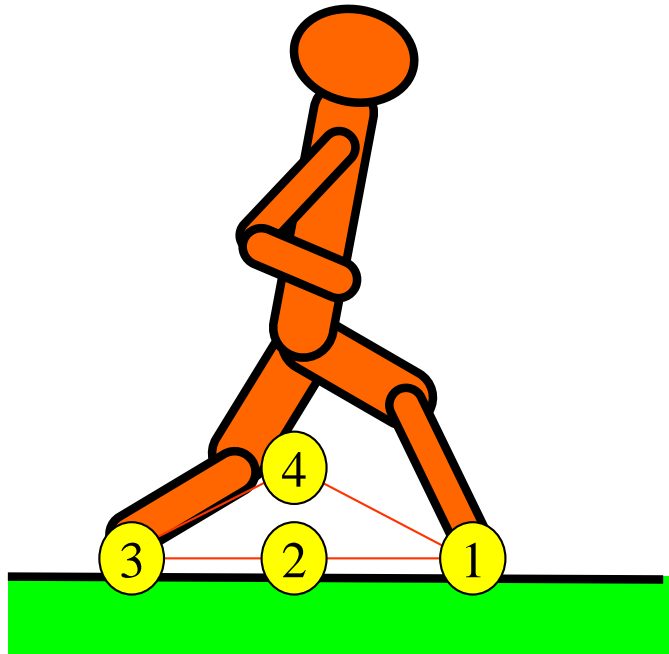
2 legs

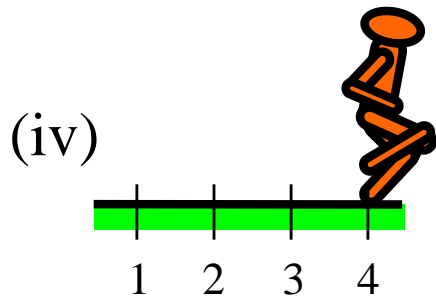
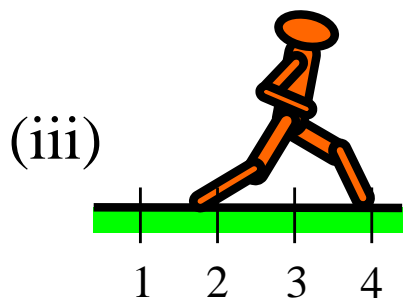
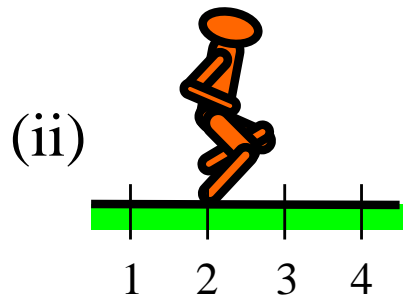
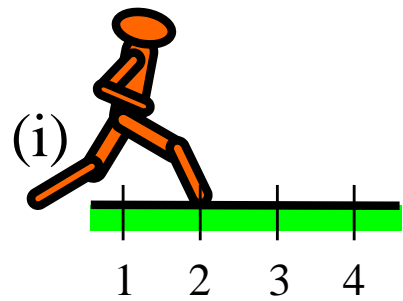
4 settings per leg

6 directions on field

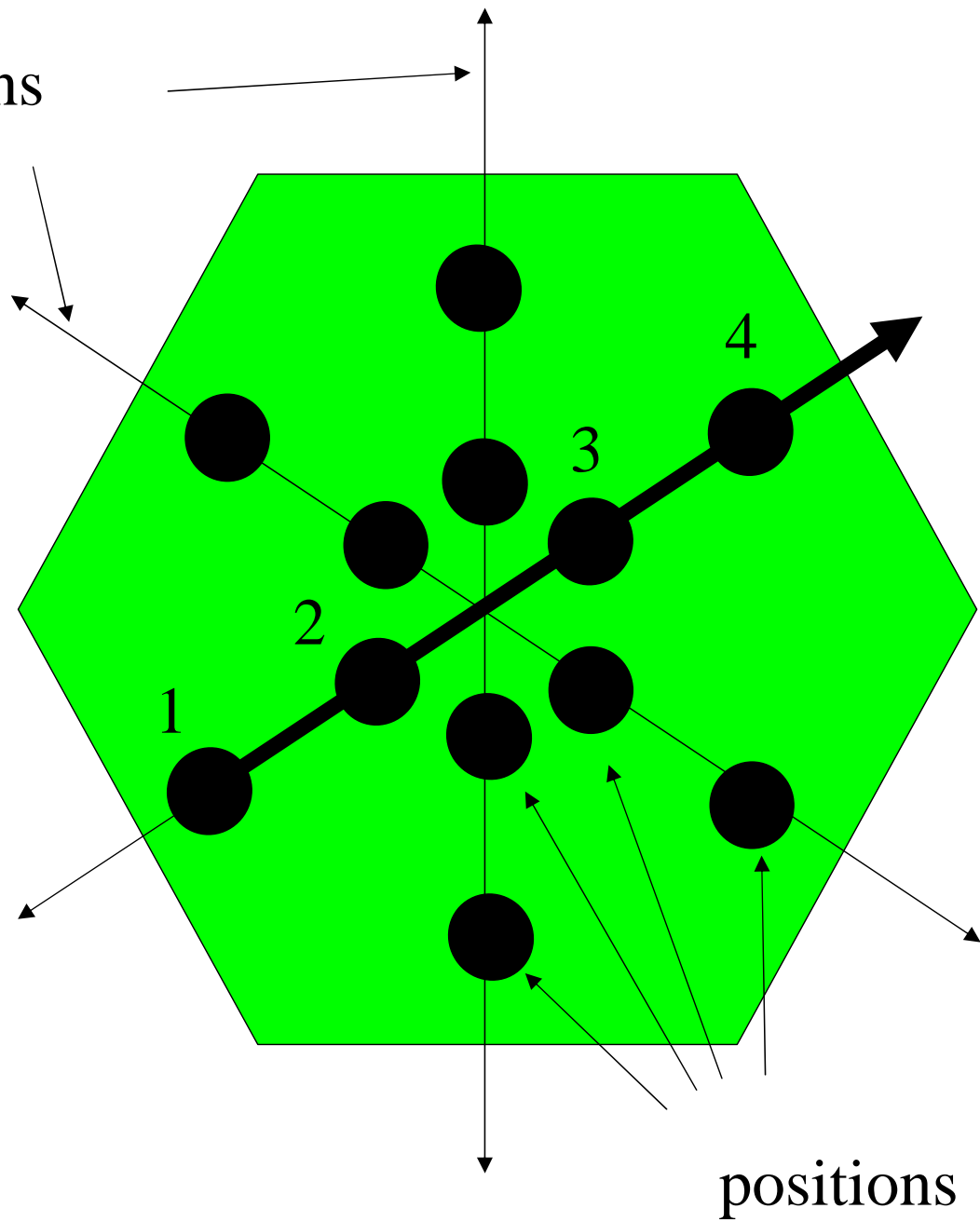
4 positions per direction

384 state values



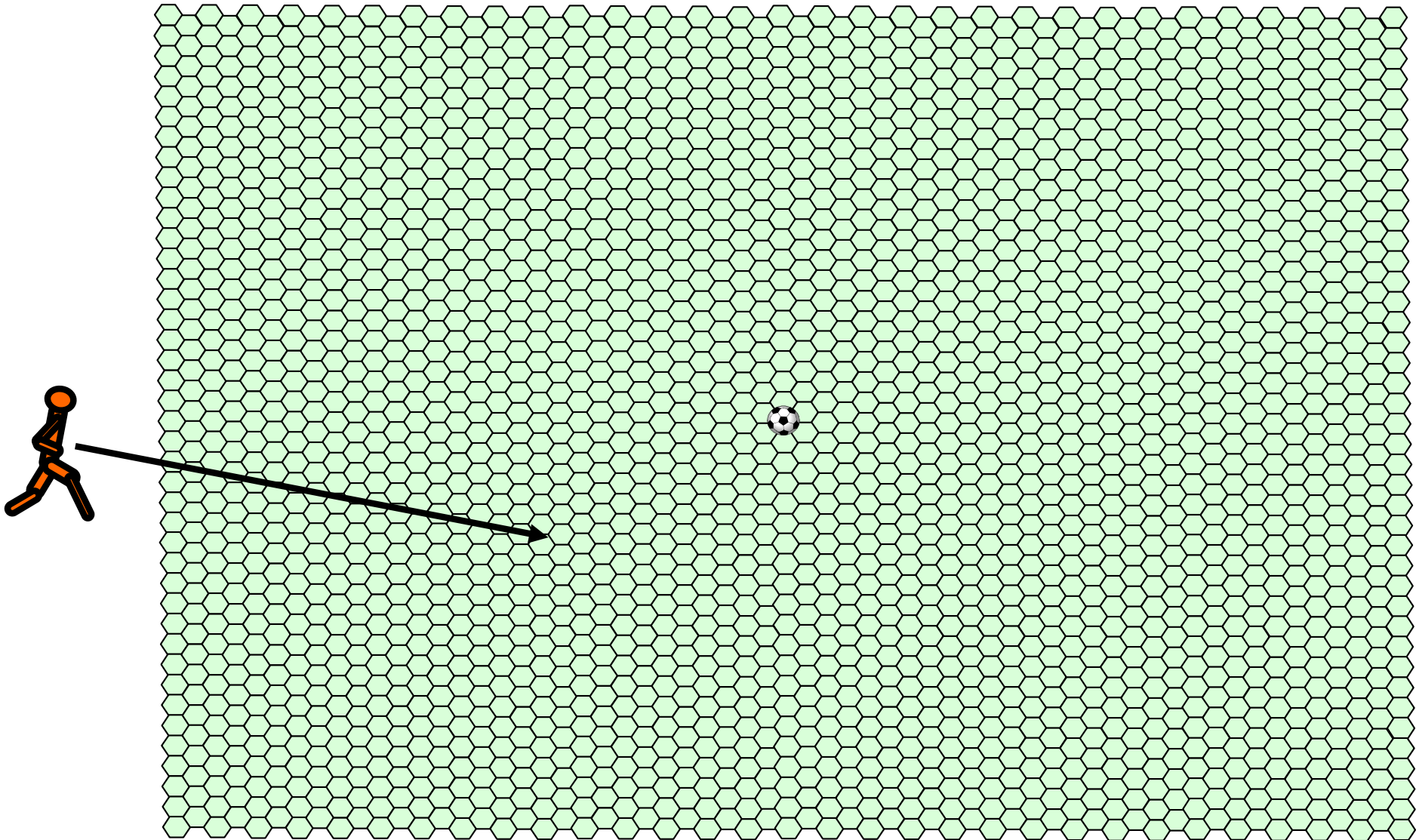


directions

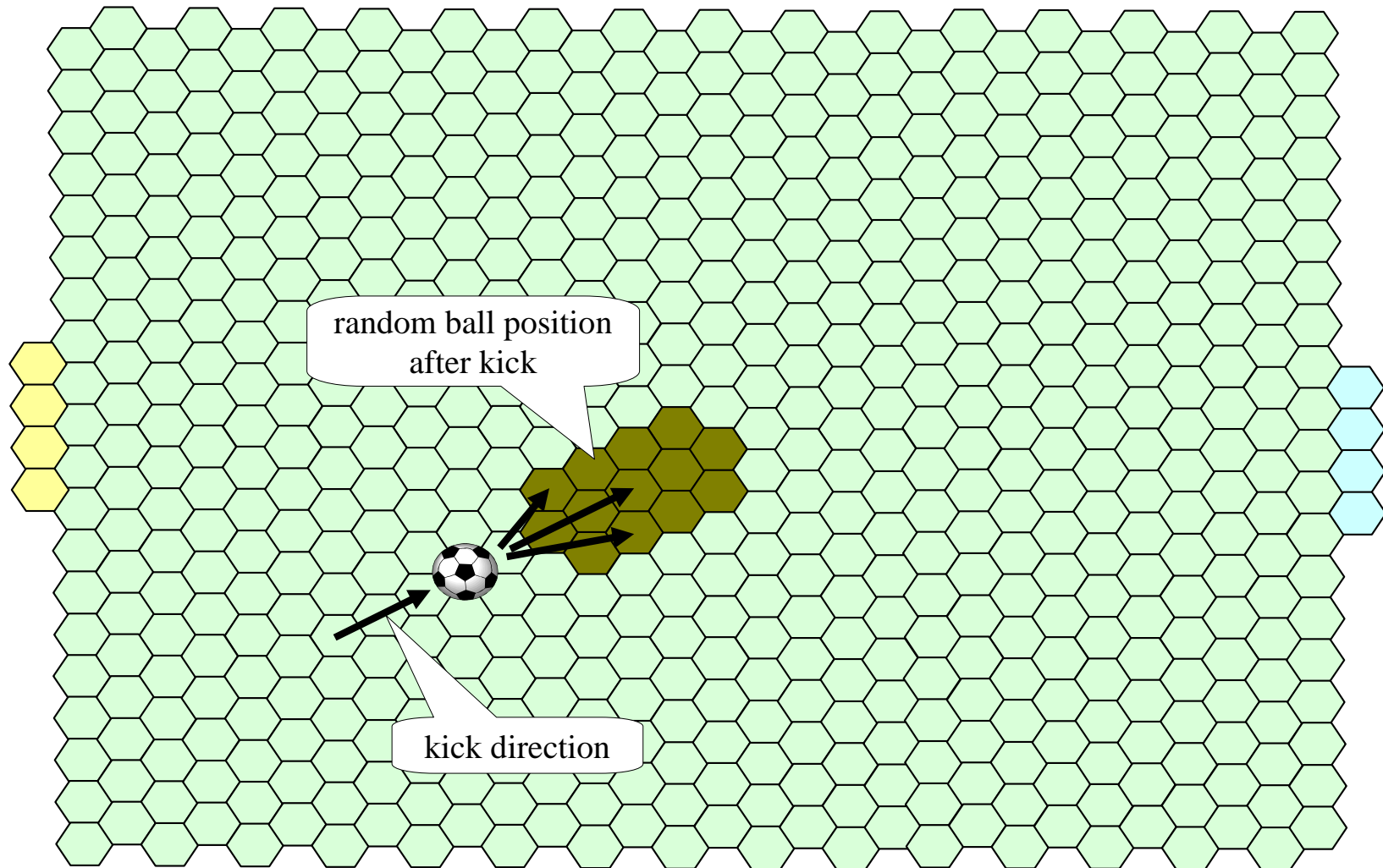


positions

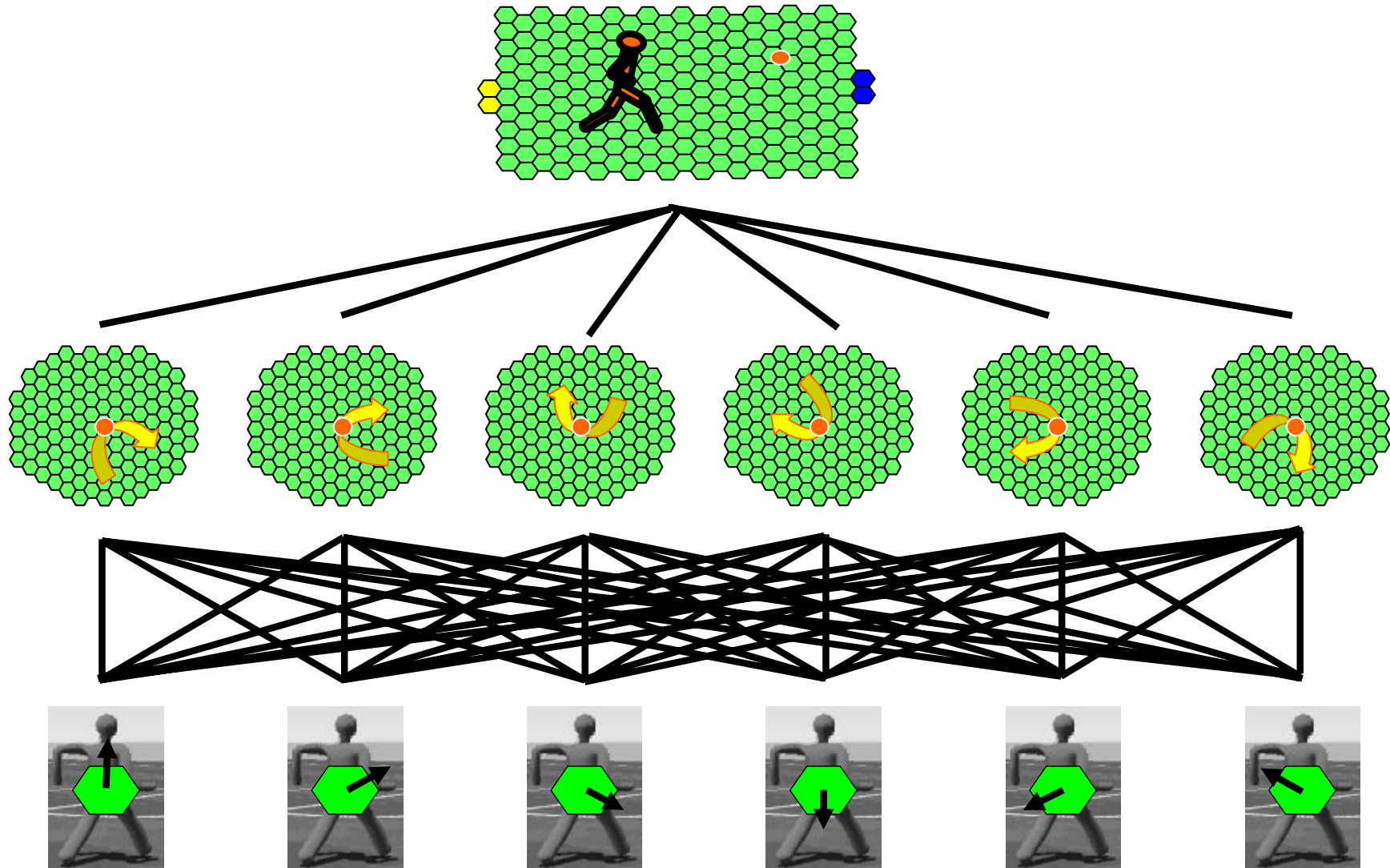
Sensor =
(robot relative to ball, ball position on field, robot stance)



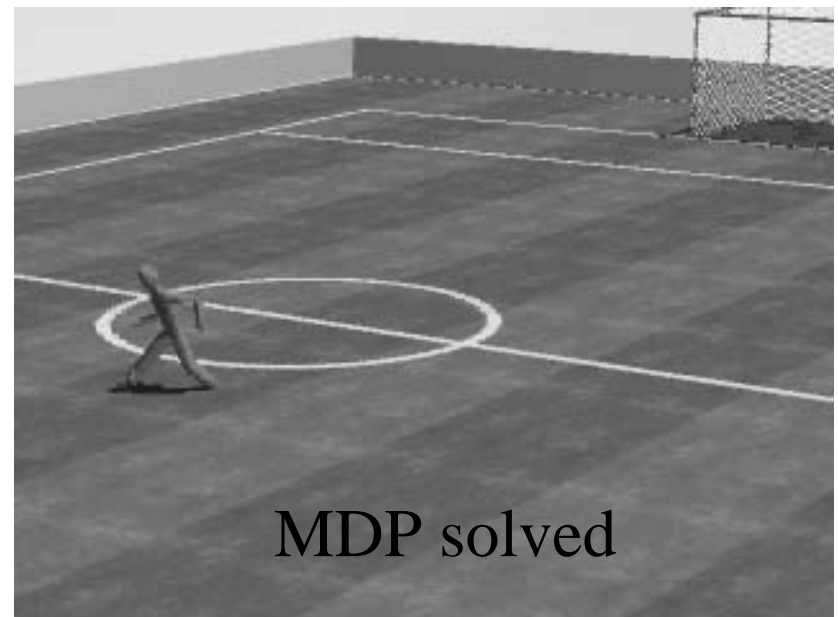
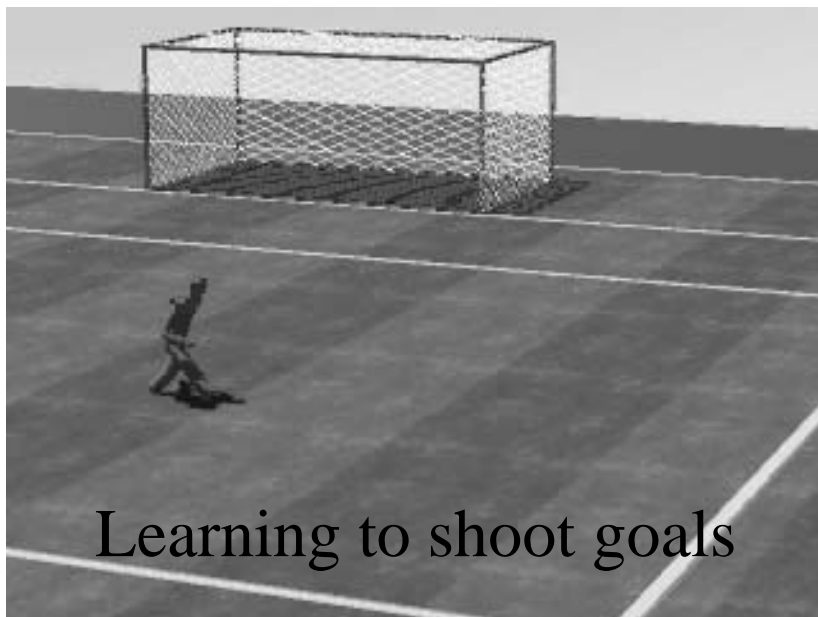
Sensor =
(robot relative to ball, ball position on field, robot stance)



Task Hierarchy using HEXQ



Hierarchical RL (HEXQ)



Robot Soccer MDP Q Values

Flat MDP

|States| = $600 \times 2501 \times 384 = 5.76 \times 10^8$

|Actions| = 21

|Q values| = 1.21×10^{10}

HEXQ

Level 1: 384 states x 21 actions x 6 sub-MDPs

Level 2: 2501 states x 6 actions x 6 sub-MDPs

Level 3: 600 states x 6 actions

|Q values| = 1.42×10^5

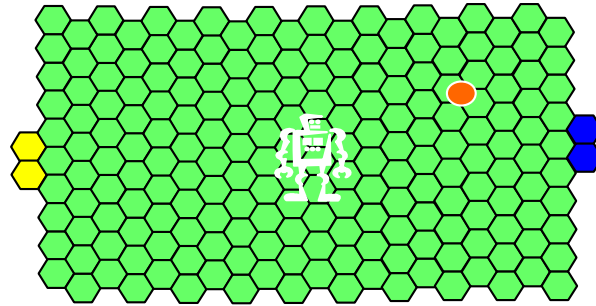
Saving: 5 orders of magnitude

Abstraction Hierarchy

Concept

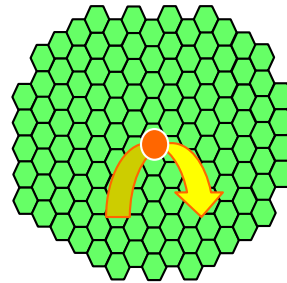
Subgoal

Dribble ball
on field



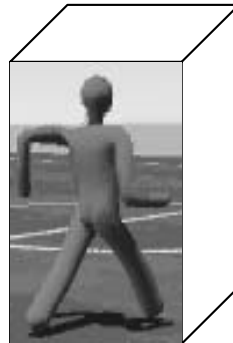
Kick goal

Ball approach



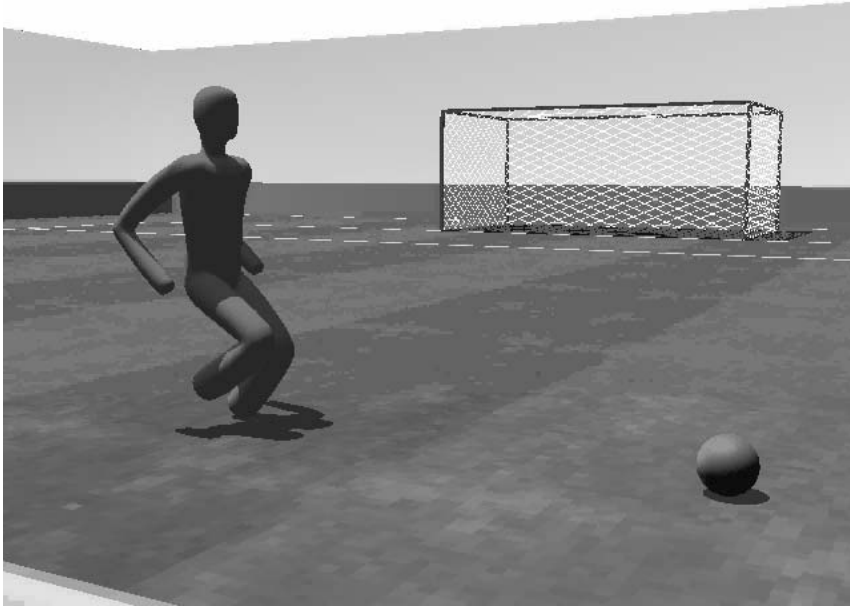
kick in direction

Gait and turn



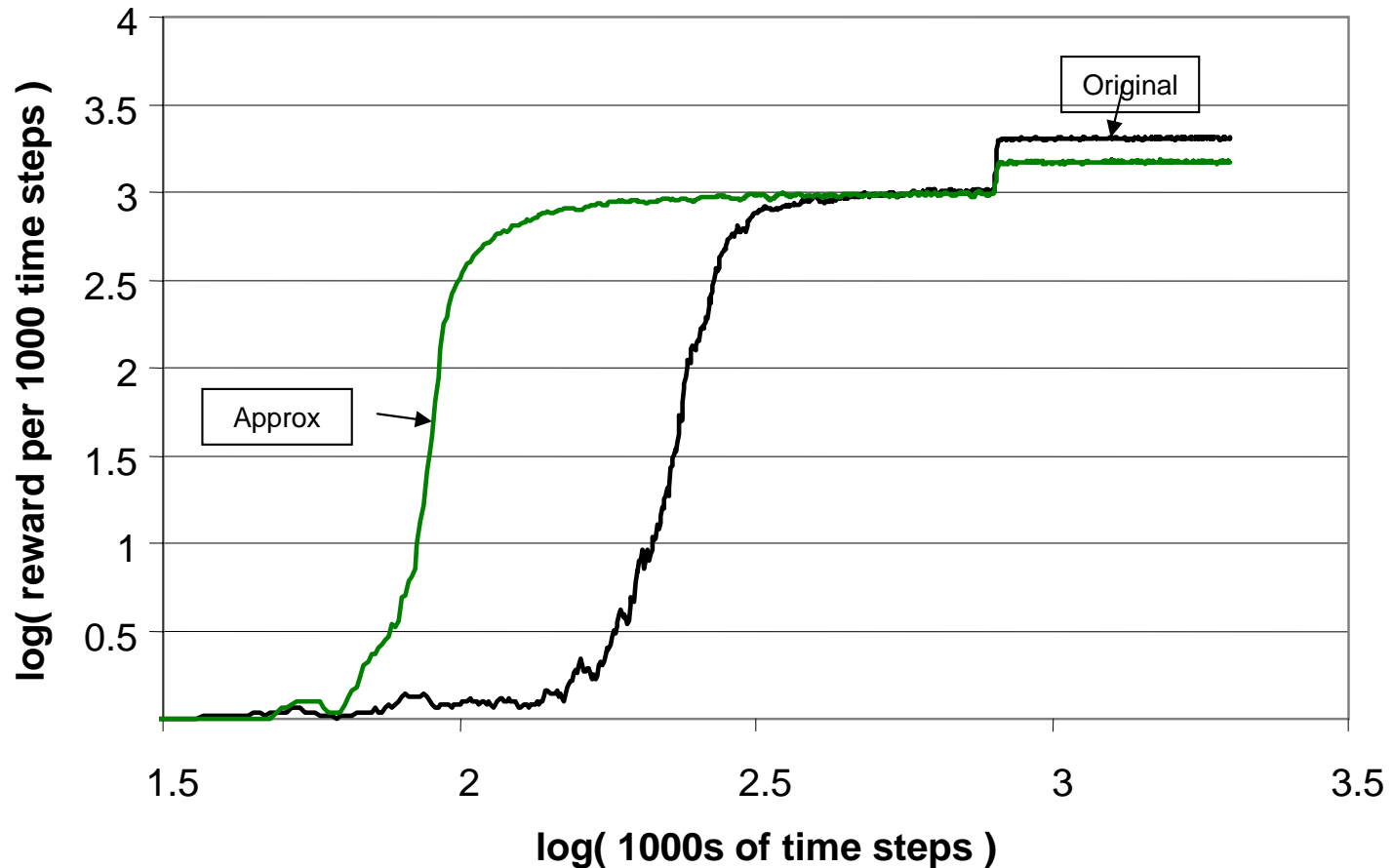
Step in direction

Infinite Horizon or Continuing HRL



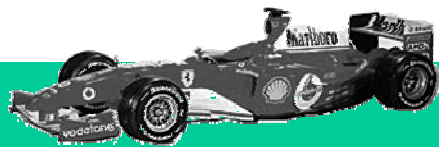
HRL Approximations

- Hierarchical depth of value function
- Coarseness of subtask termination conditions



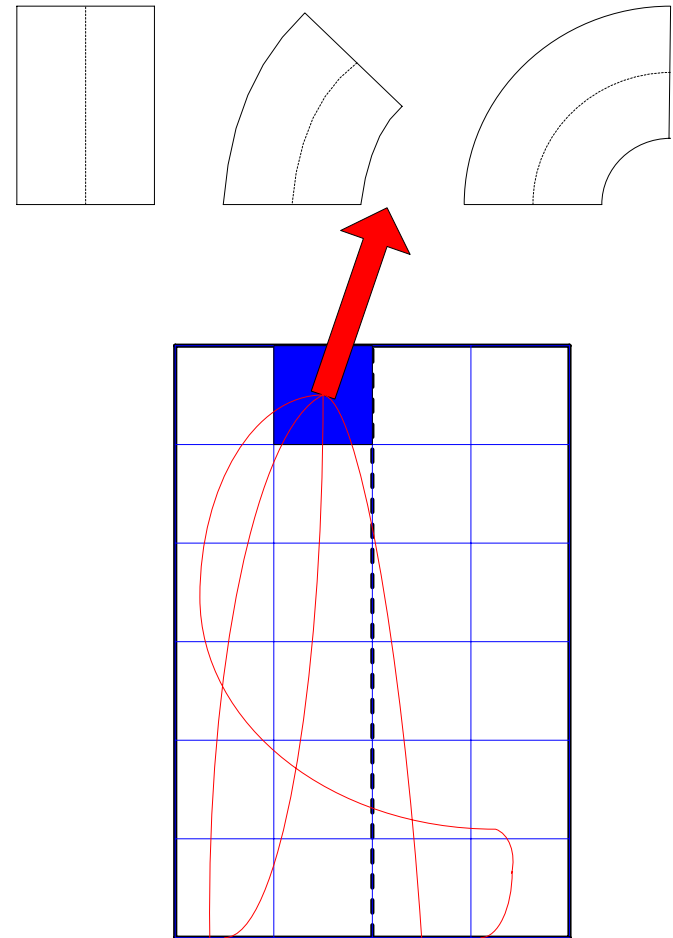
Racing Cars

Adam Schuck



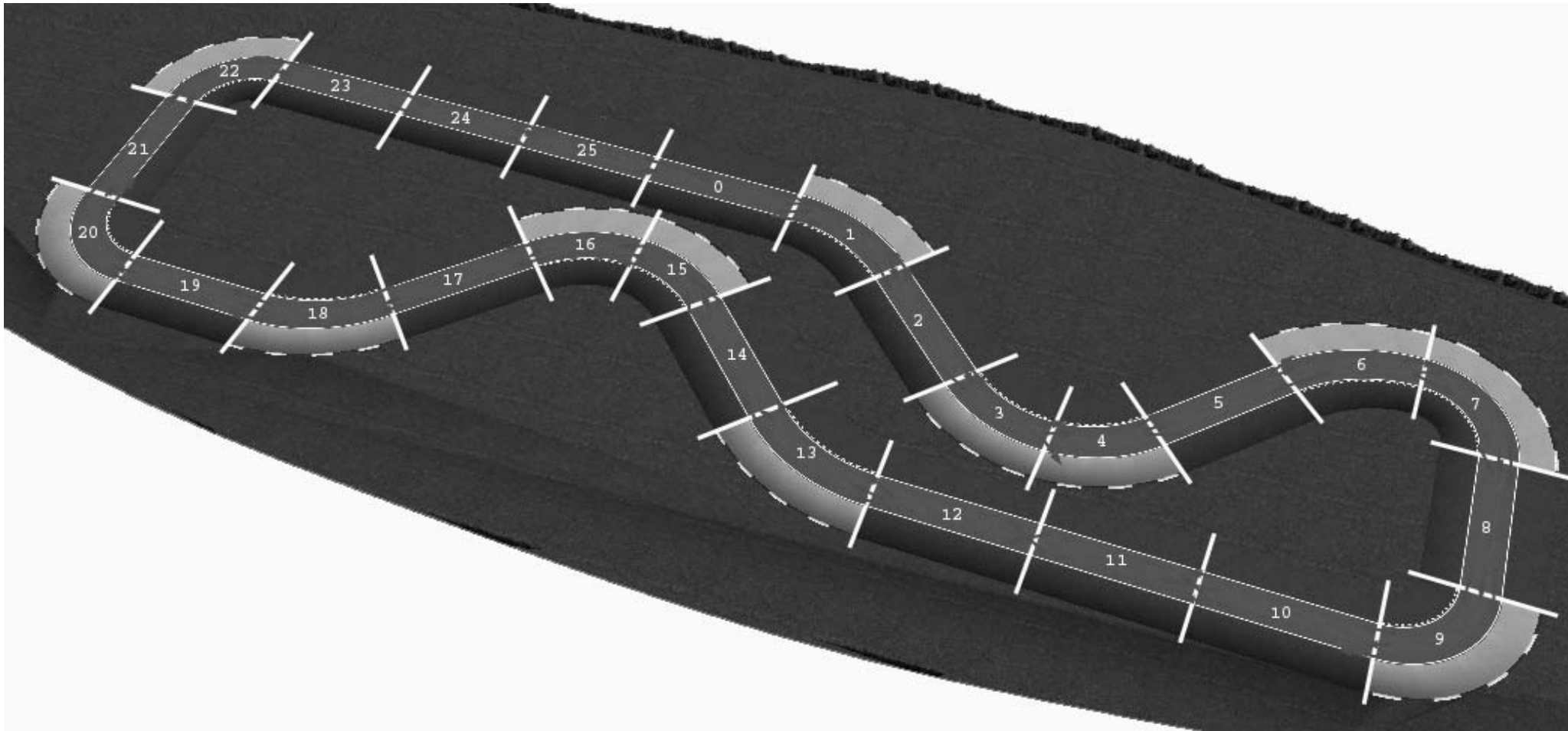
Low-level Learner

- Racetracks are composed from simple pieces
- Lower level treats track segment as MDP
- Determines optimal policy for different exit strategies (position, angle, speed)



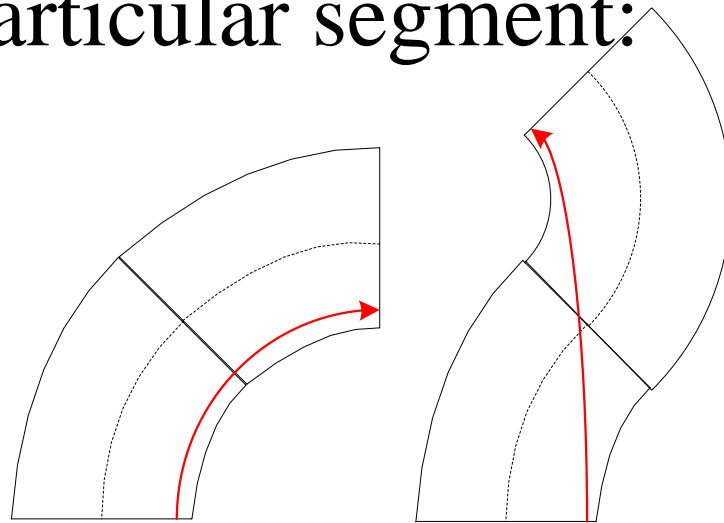
High-level Learner

- Determine optimal policy for abstracted SMDP



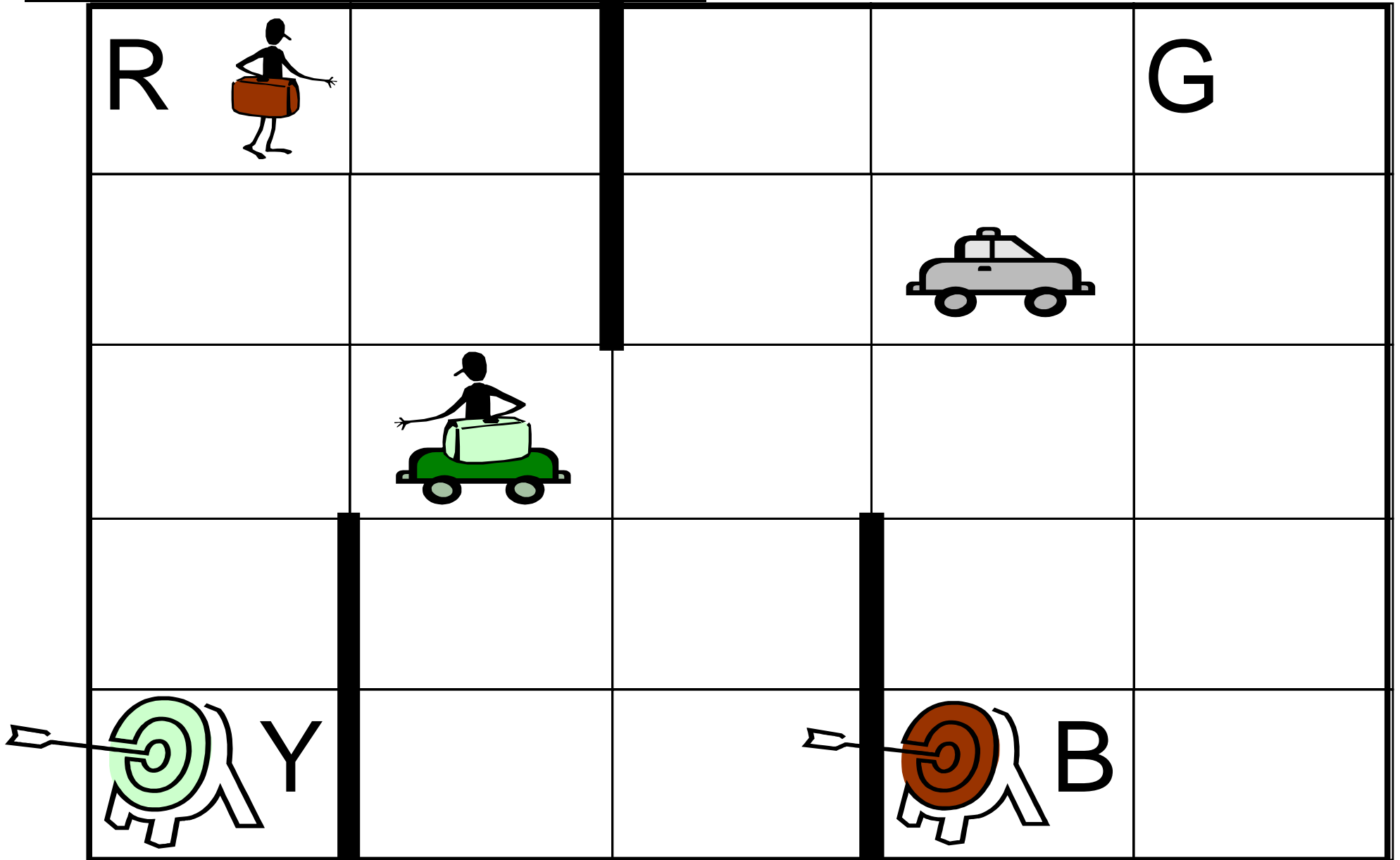
High-level Learner

- Learns how to use low level skills appropriately
- Context has large effect on which policy to use in a particular segment:

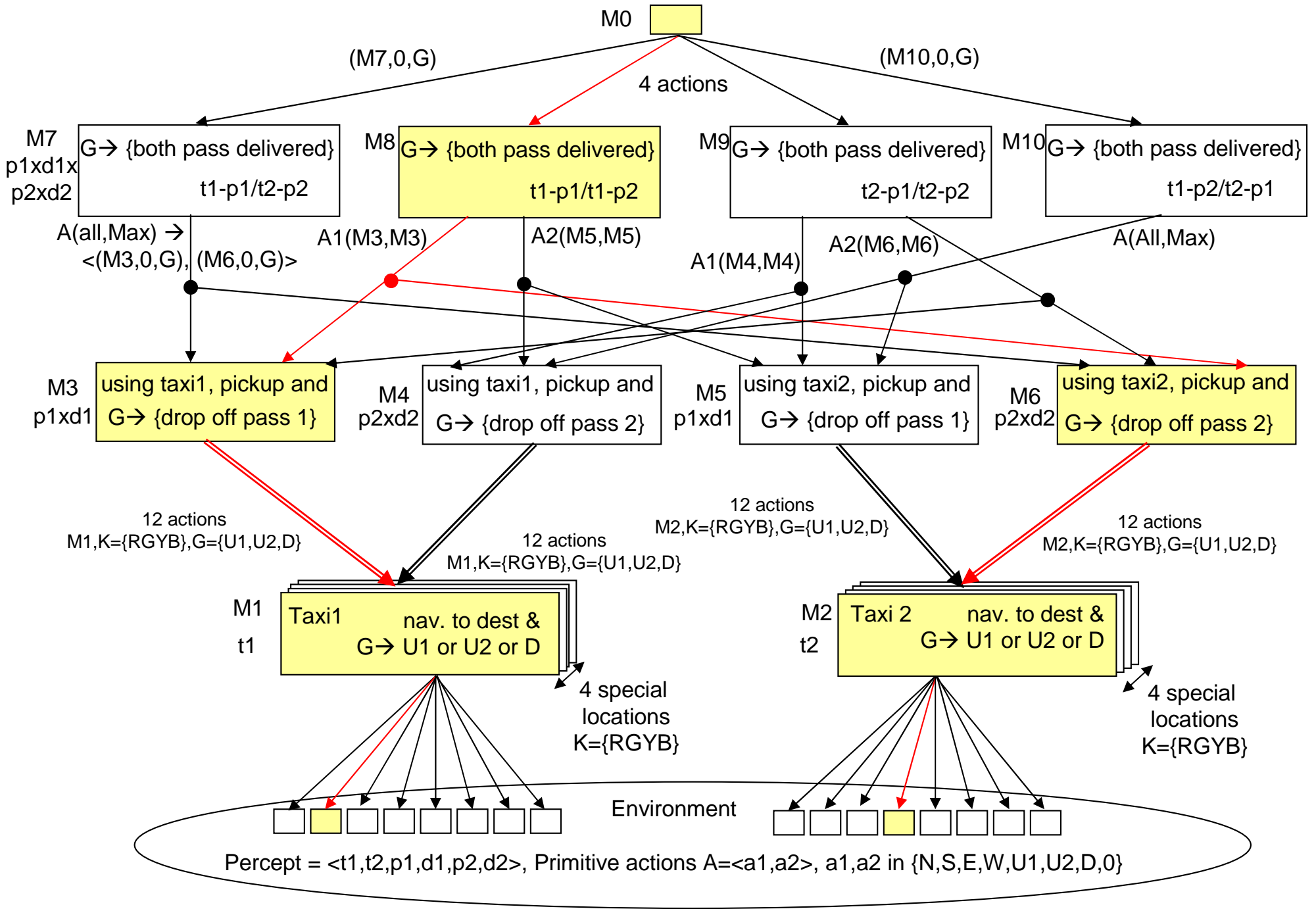


Two Taxi Task

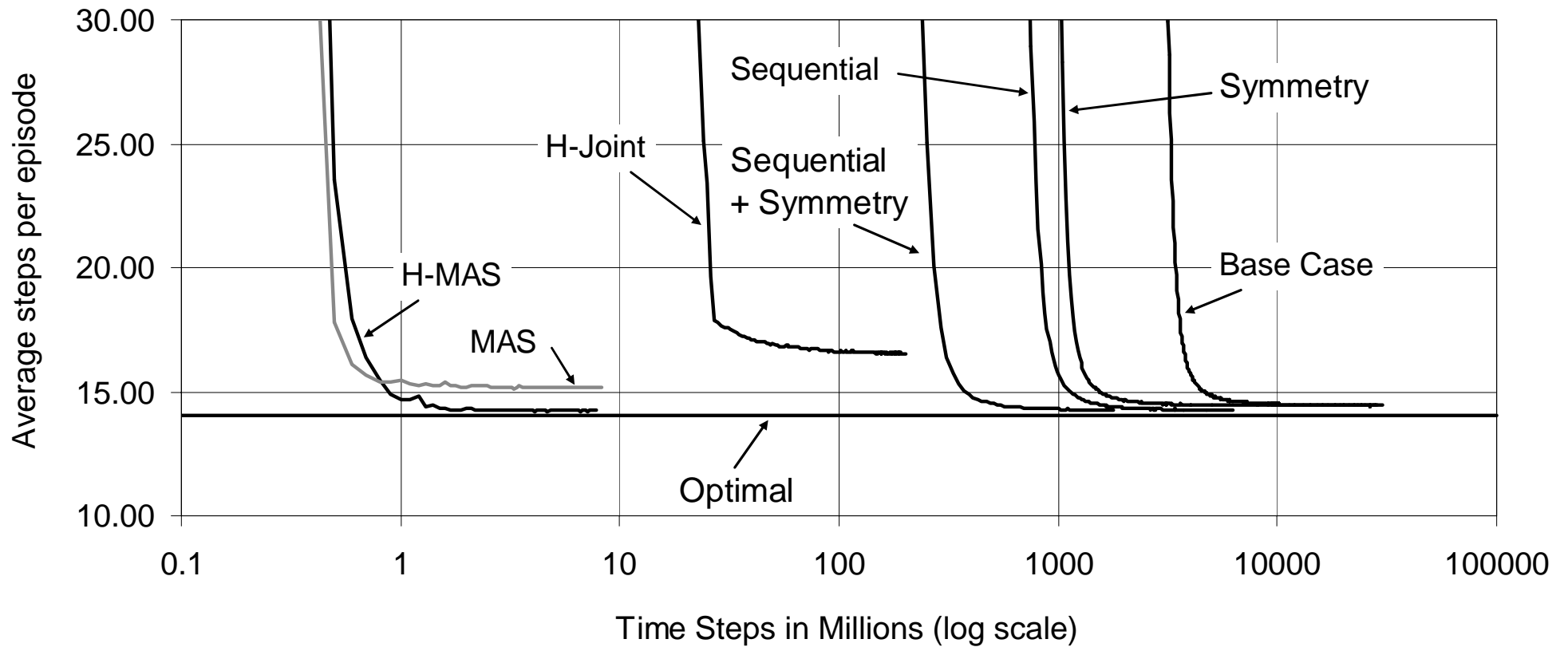
2T2C HEXQ 108 BH 13 Feb 2009



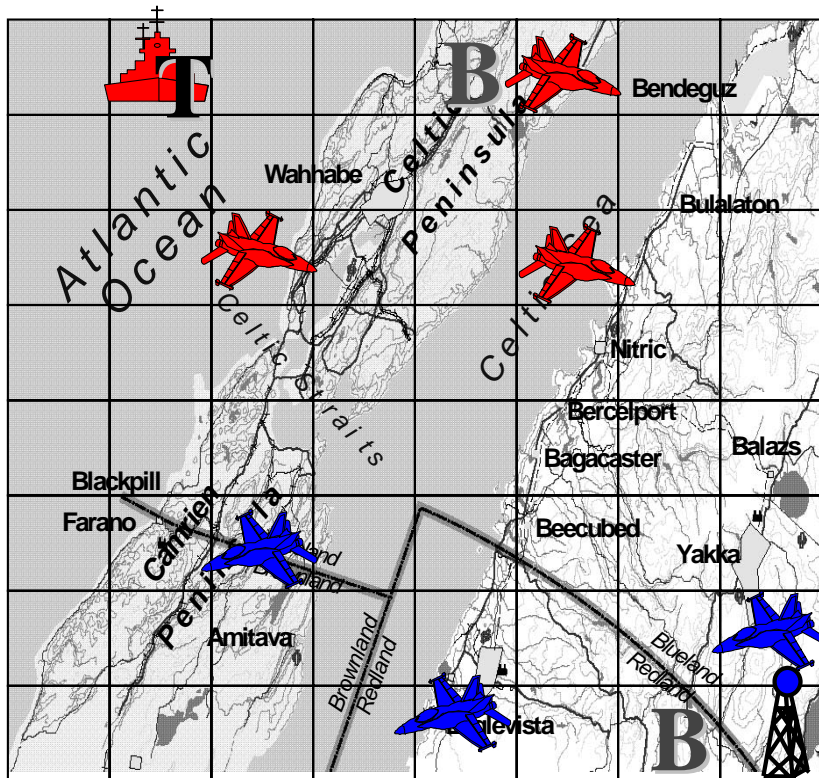
Two Taxi Task Example



Two Taxi Task



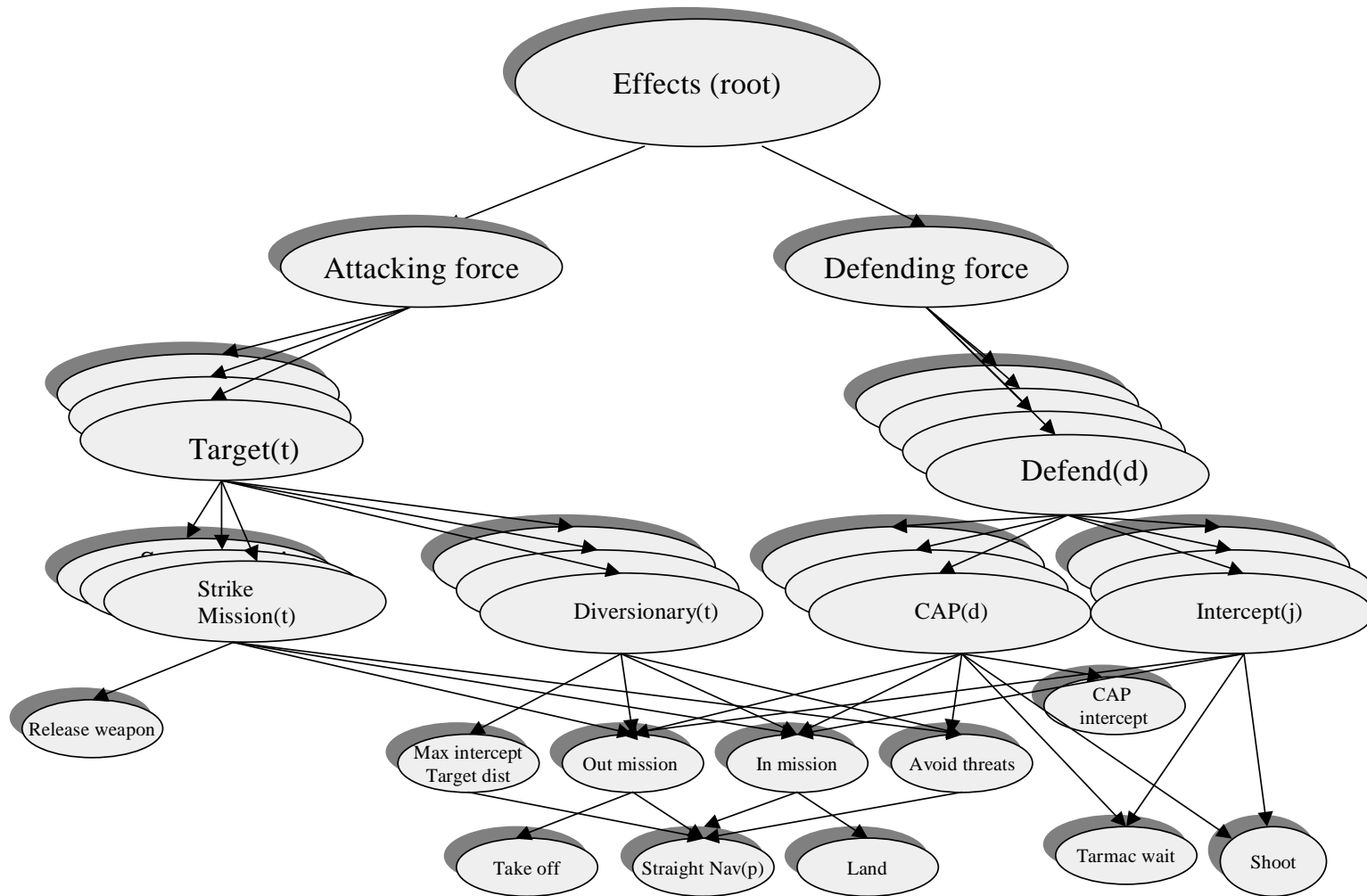
LEAR: Machine learn strategies

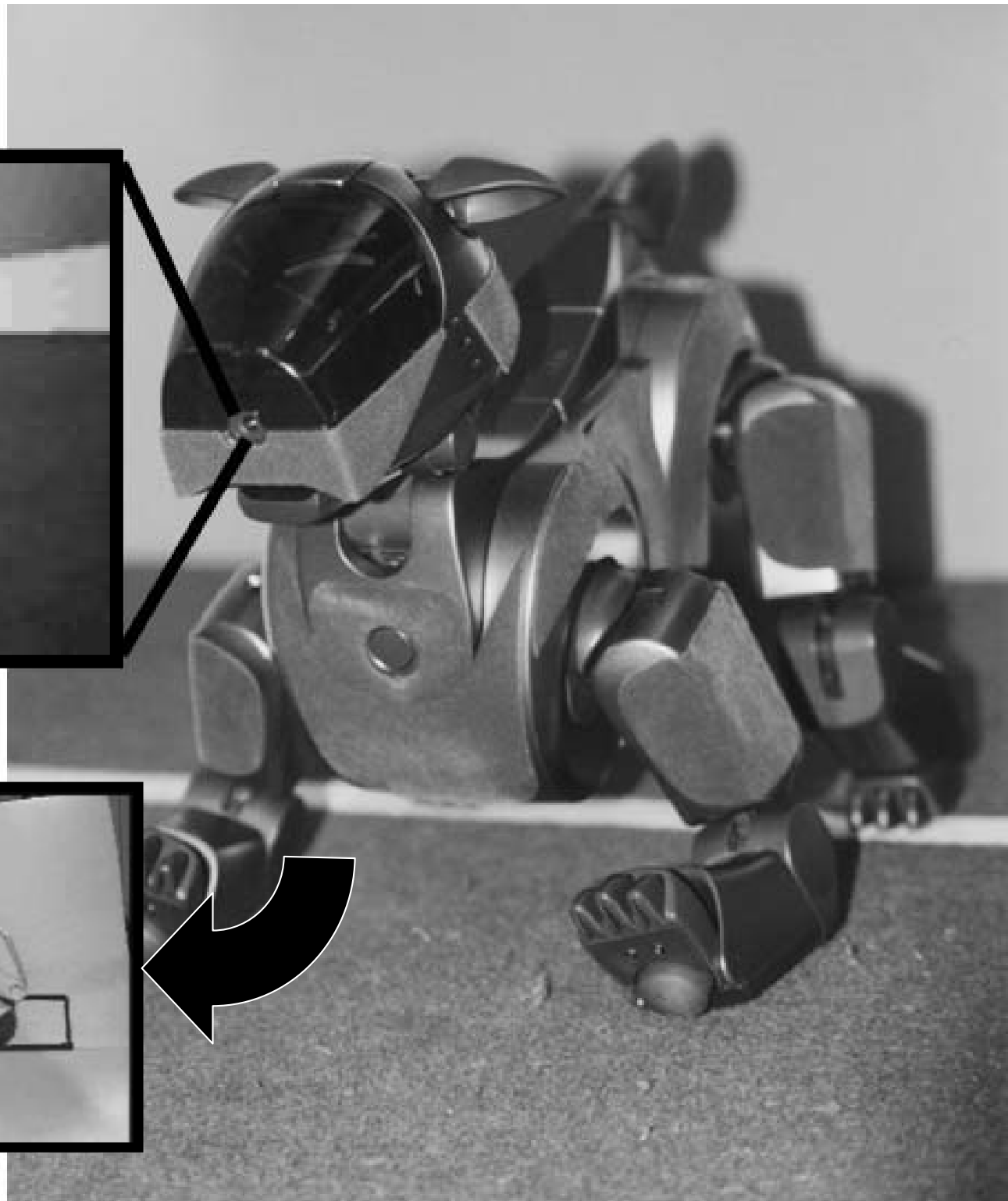
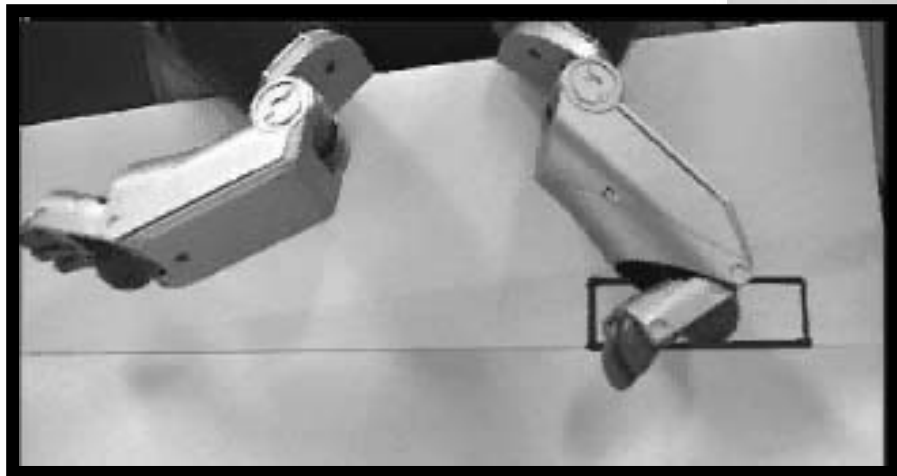
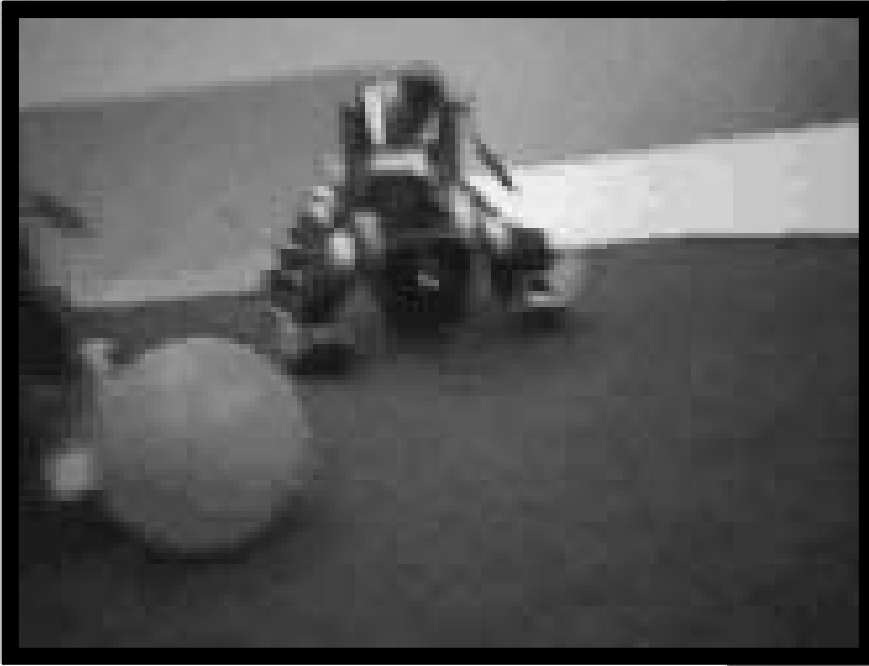


The distinguishing research features of this project are

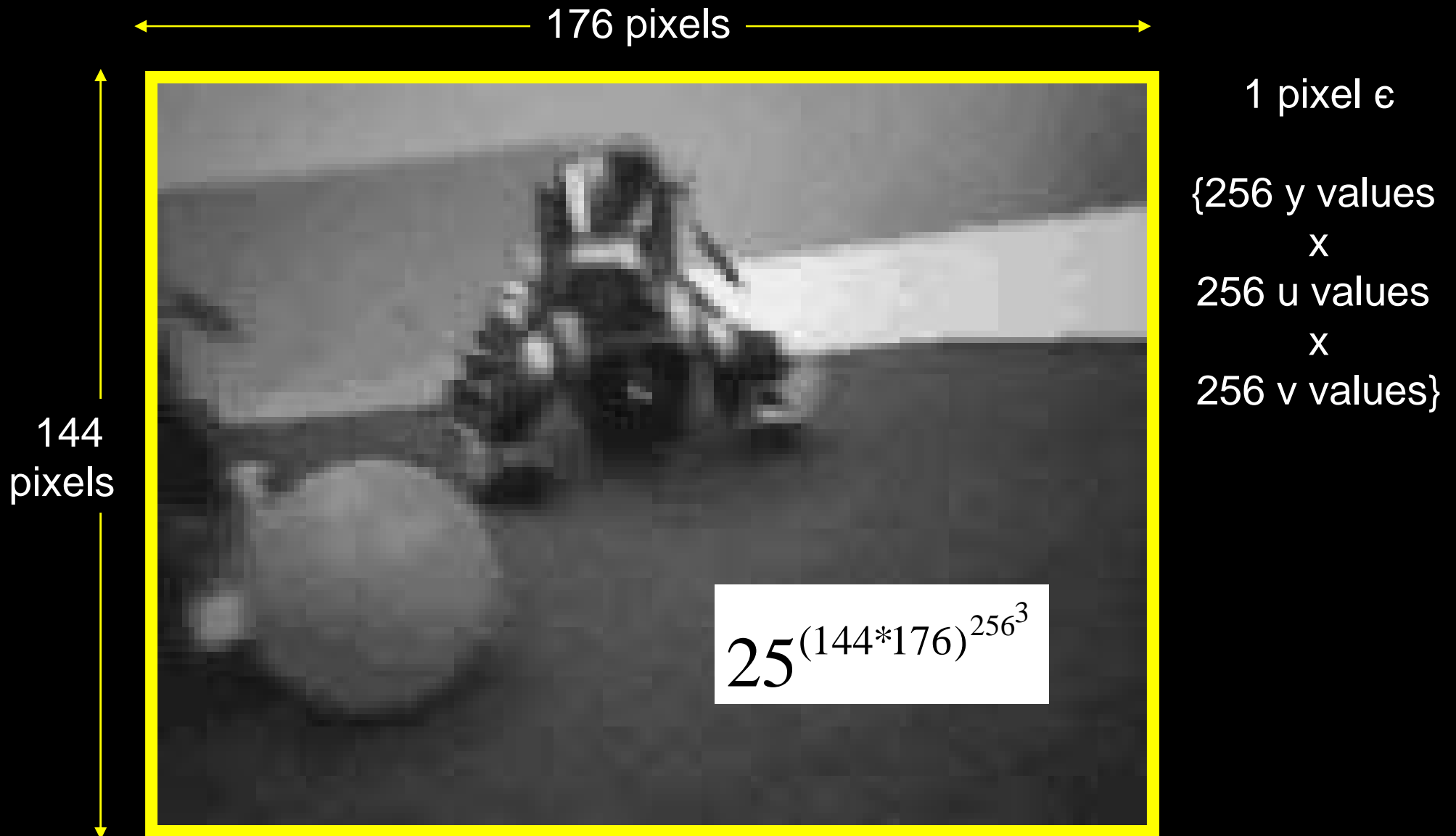
- (1) the study of operational art and
- (2) machine learning and representation.

A LEAR Task Hierarchy





AIBO Visual Sensor



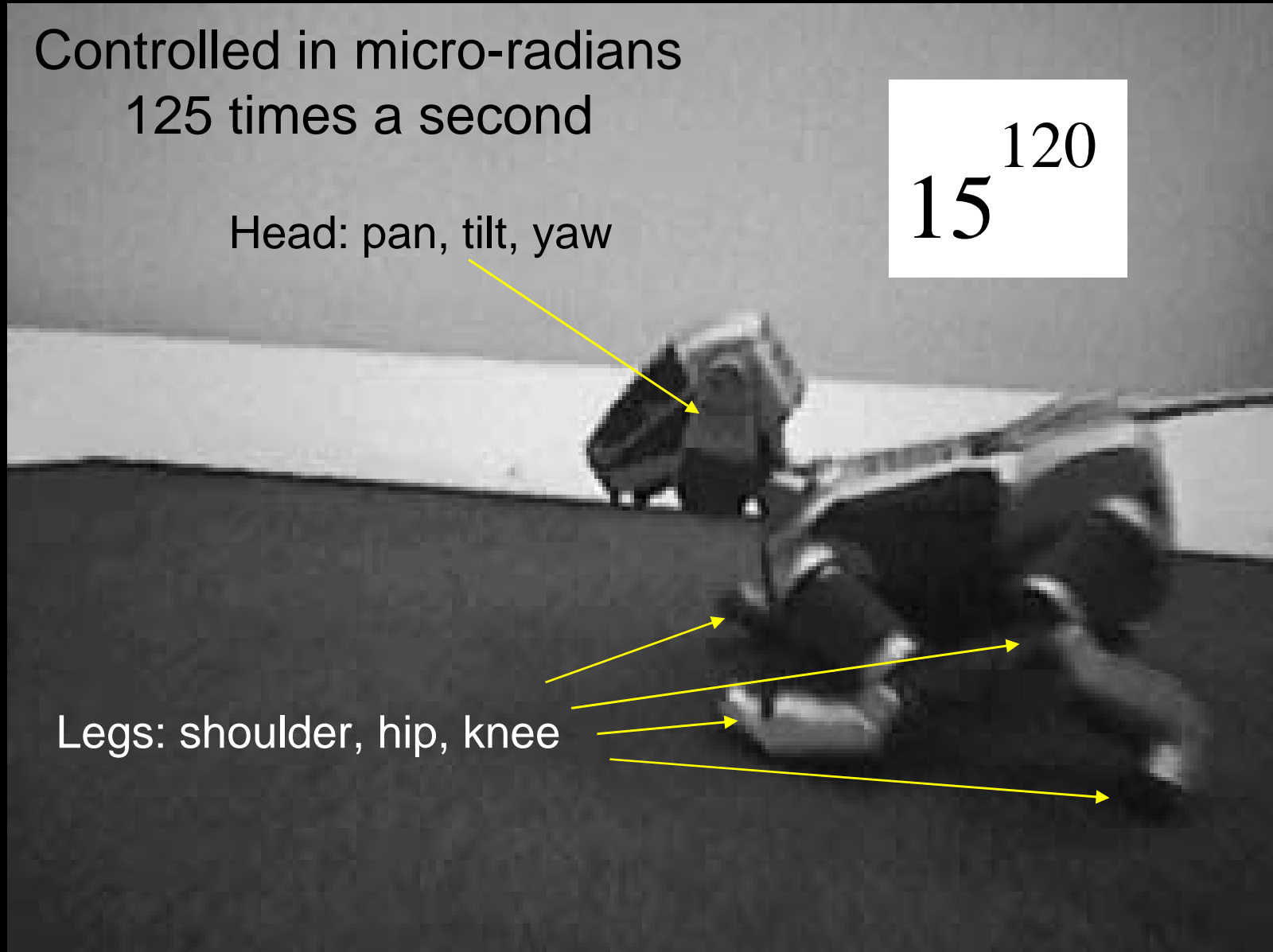
Effector

Controlled in micro-radians
125 times a second

Head: pan, tilt, yaw

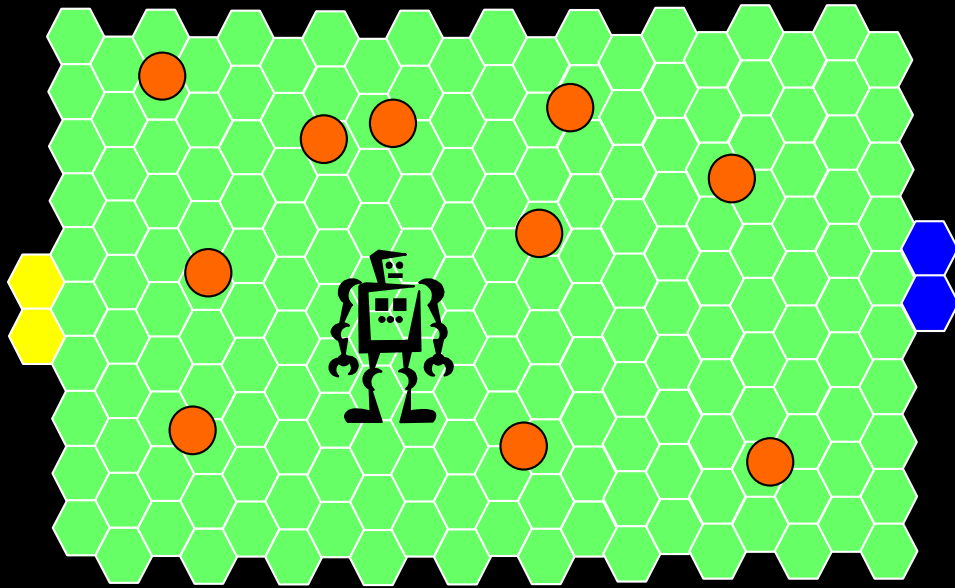
15¹²⁰

Legs: shoulder, hip, knee

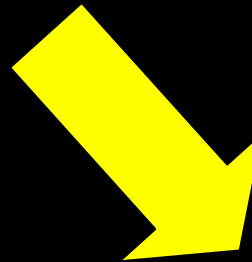


Fukuoka, Japan 2002

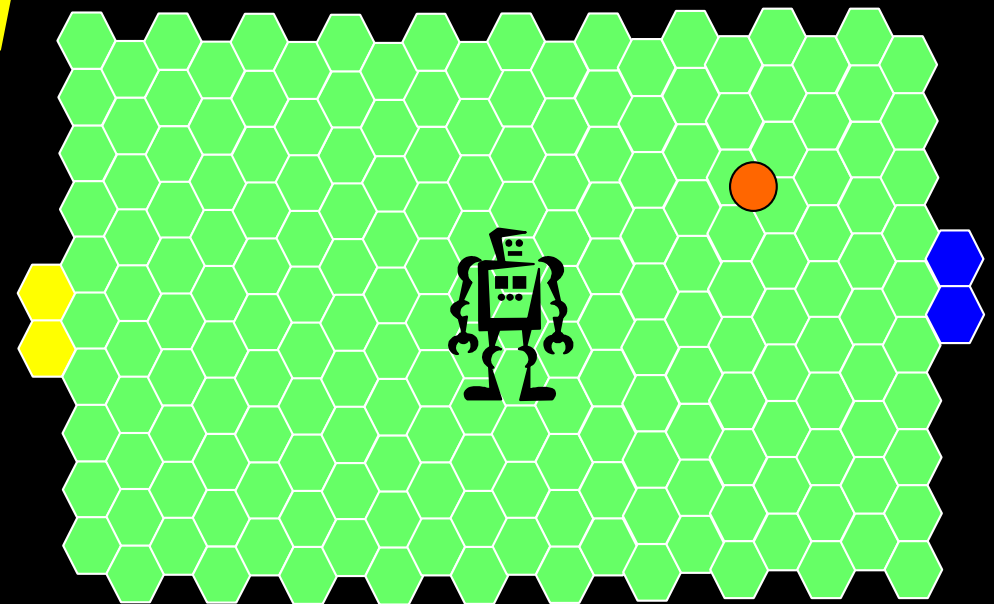




Selective
Perception
&
Serialise
Subgoals

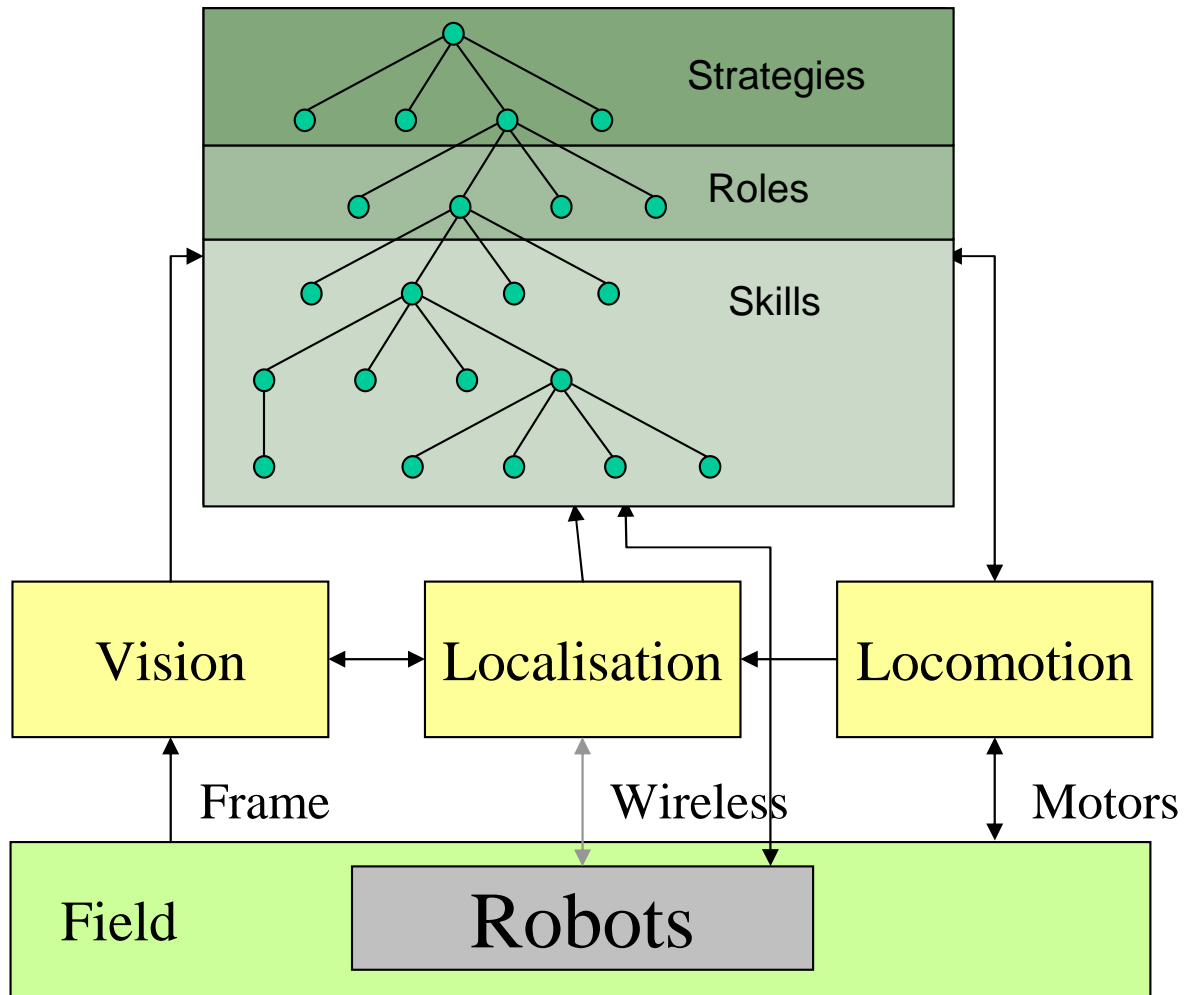


Weakly coupled
environment

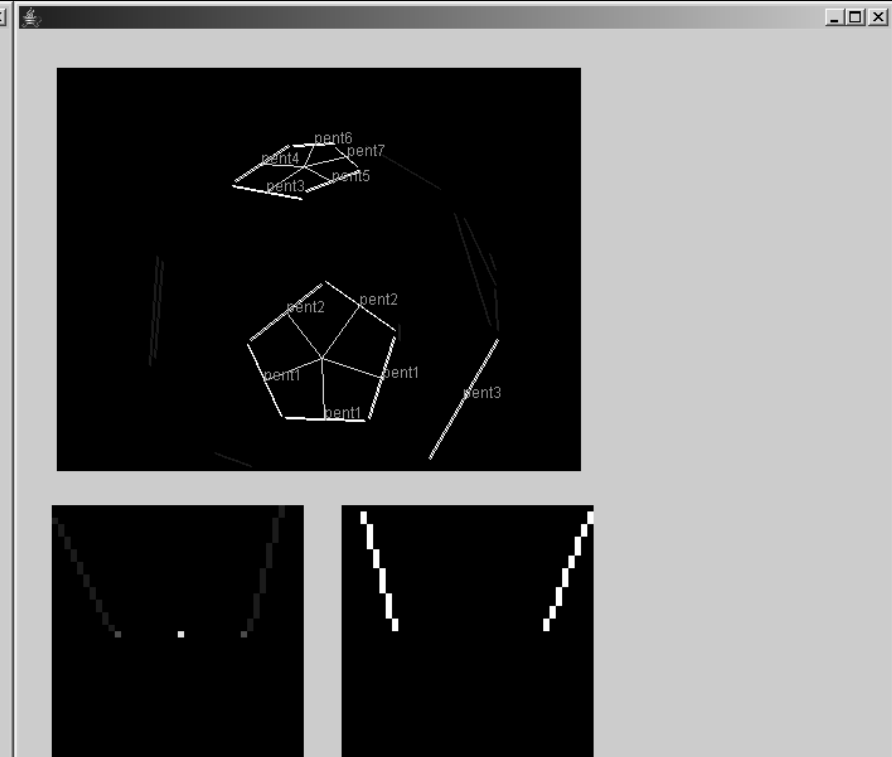
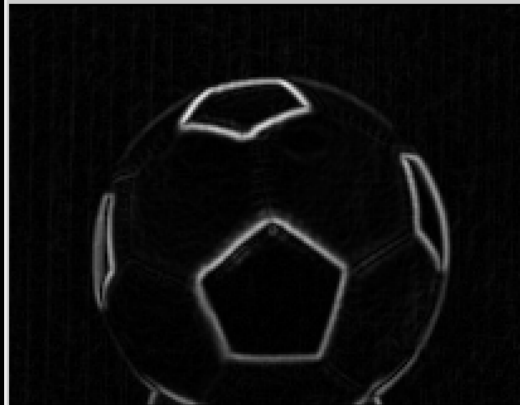
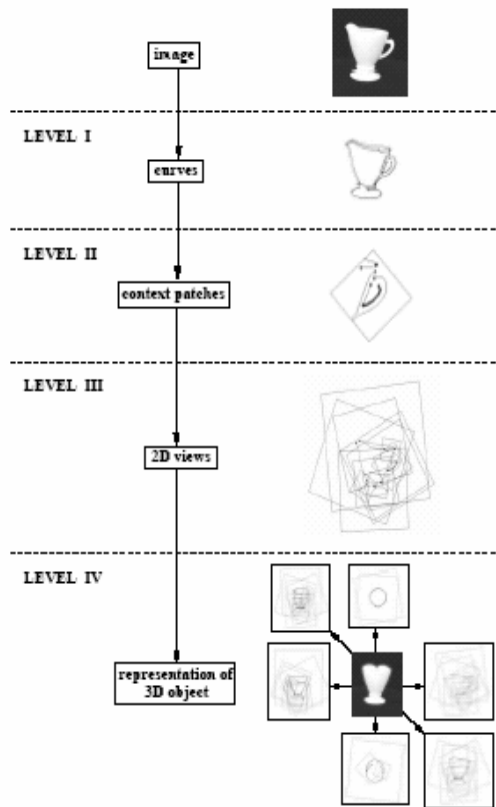


60 orders of magnitude saving.

RoboCup



Hierarchical Appearance Based Object Recognition



Tom Vogelgesang 2004
Wen Tao Lu 2005

load DB

add patch

0

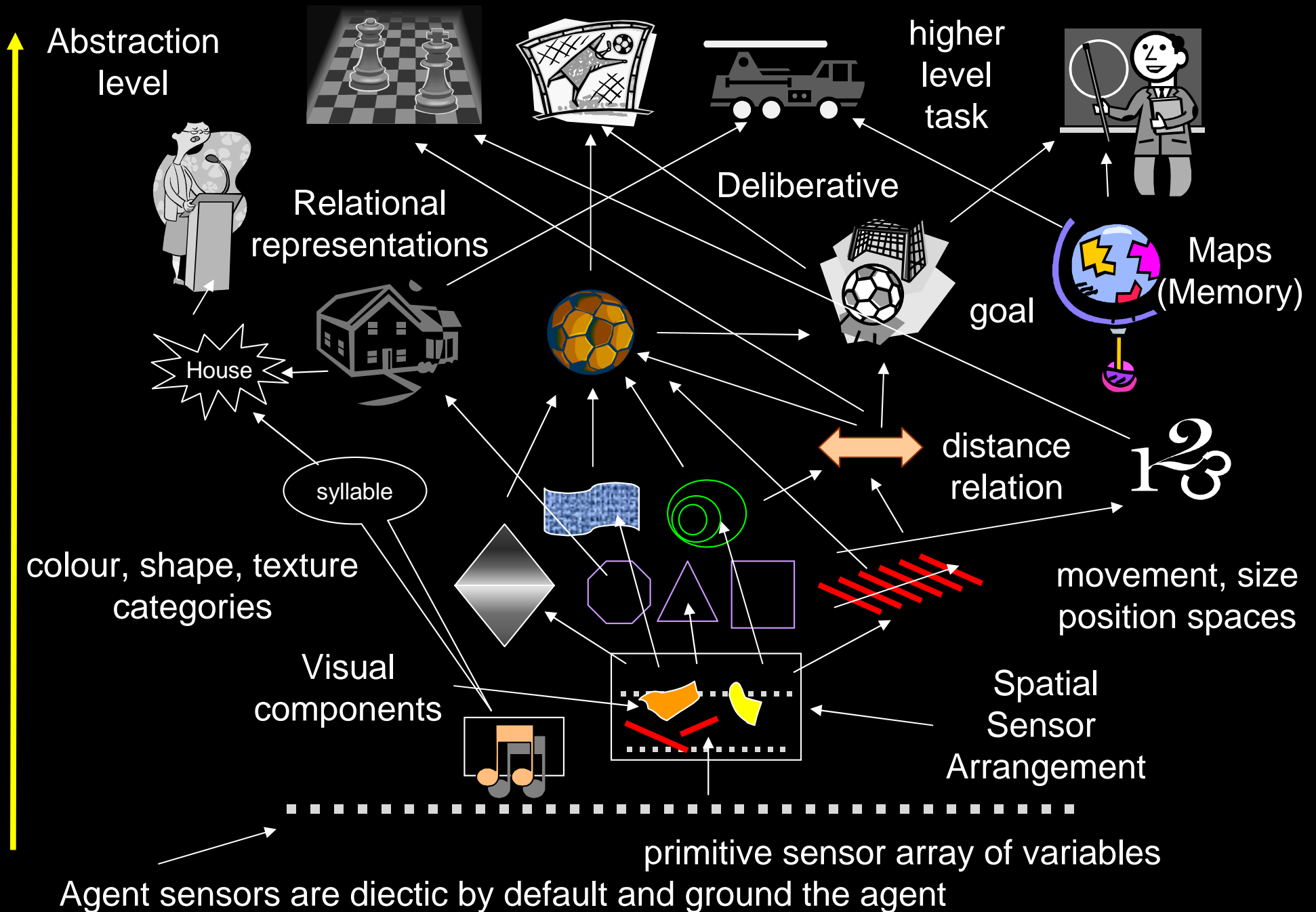
dPatch

0.8425

save DB

Ref: Randal Nelson

Learning Grounded Dynamic Conceptual Hierarchies



Hierarchical Reinforcement Learning

References:

Sridhar Mahadevan, Mohammad Ghavamzadeh, Khashayar Rohanimanesh, Georgios Theocharous, "Hierarchical Approaches to Concurrency, Multiagency, and Partial Observability", Learning and Approximate Dynamic Programming: Scaling up to the Real World, Edited by Jennie Si, Andy Barto, Warren Powell, and Donald Wunsch, John Wiley & Sons, New York.

Andrew Barto and Sridhar Mahadevan, "Recent Advances in Hierarchical Reinforcement Learning", volume 13, Special Issue on Reinforcement Learning, Discrete Event Systems journal, pp. 41-77, 2003

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research, 13, 227-303. Compressed postscript. Also available from my HTTP directory as Gzipped postscript

Sutton, R.S., Precup, D., Singh, S. (1999). Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. Artificial Intelligence 112:181-211. An earlier version appeared as Technical Report 98-74, Department of Computer Science, University of Massachusetts, Amherst, MA 01003. April, 1998.

Reinforcement Learning with Hierarchies of Machines, Ronald Parr and Stuart Russell. NIPS 97.



The Next Step