Web-Mining Agents
Cooperating Agents for Information Retrieval

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Tanya Braun (Übungen)

Presentation is based on the following talk:

Relational Macros for Transfer in Reinforcement Learning

Lisa Torrey, Jude Shavlik, Trevor Walker
University of Wisconsin-Madison, USA

Richard Maclin
University of Minnesota-Duluth, USA

Transfer Learning Scenario

Agent learns Task A

↓

Agent encounters related Task B

↓

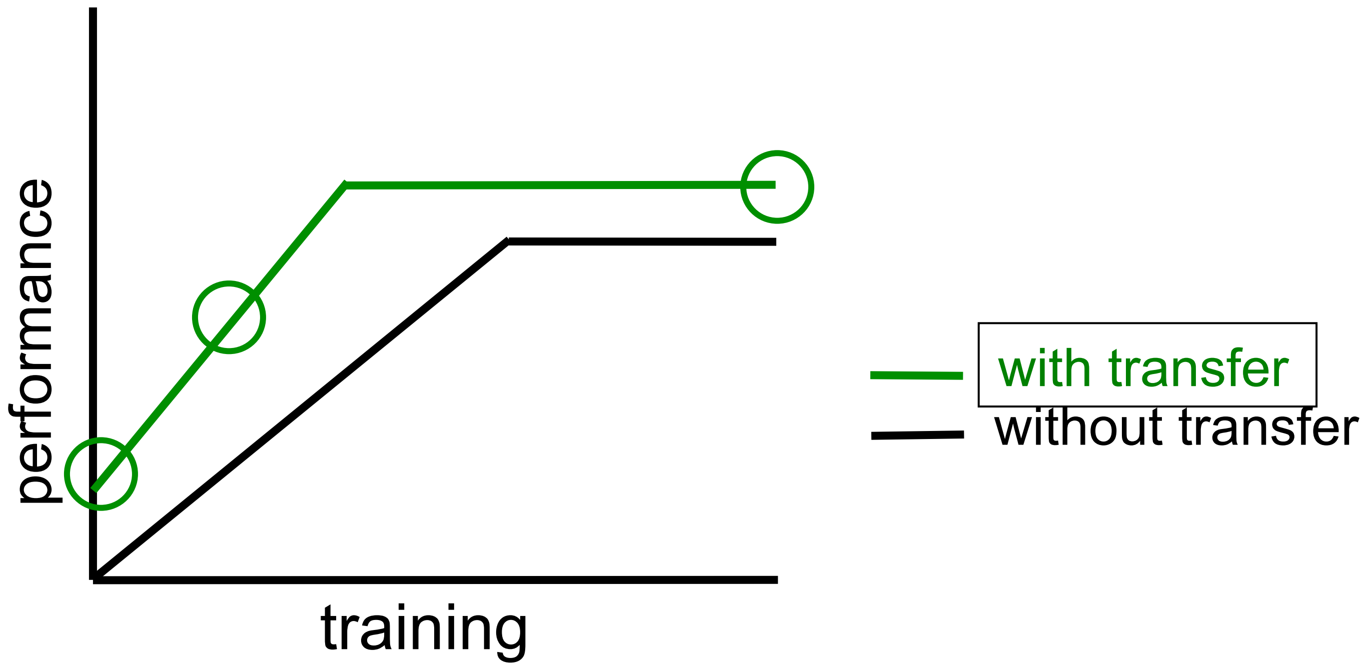
Agent recalls relevant knowledge from Task A

↓

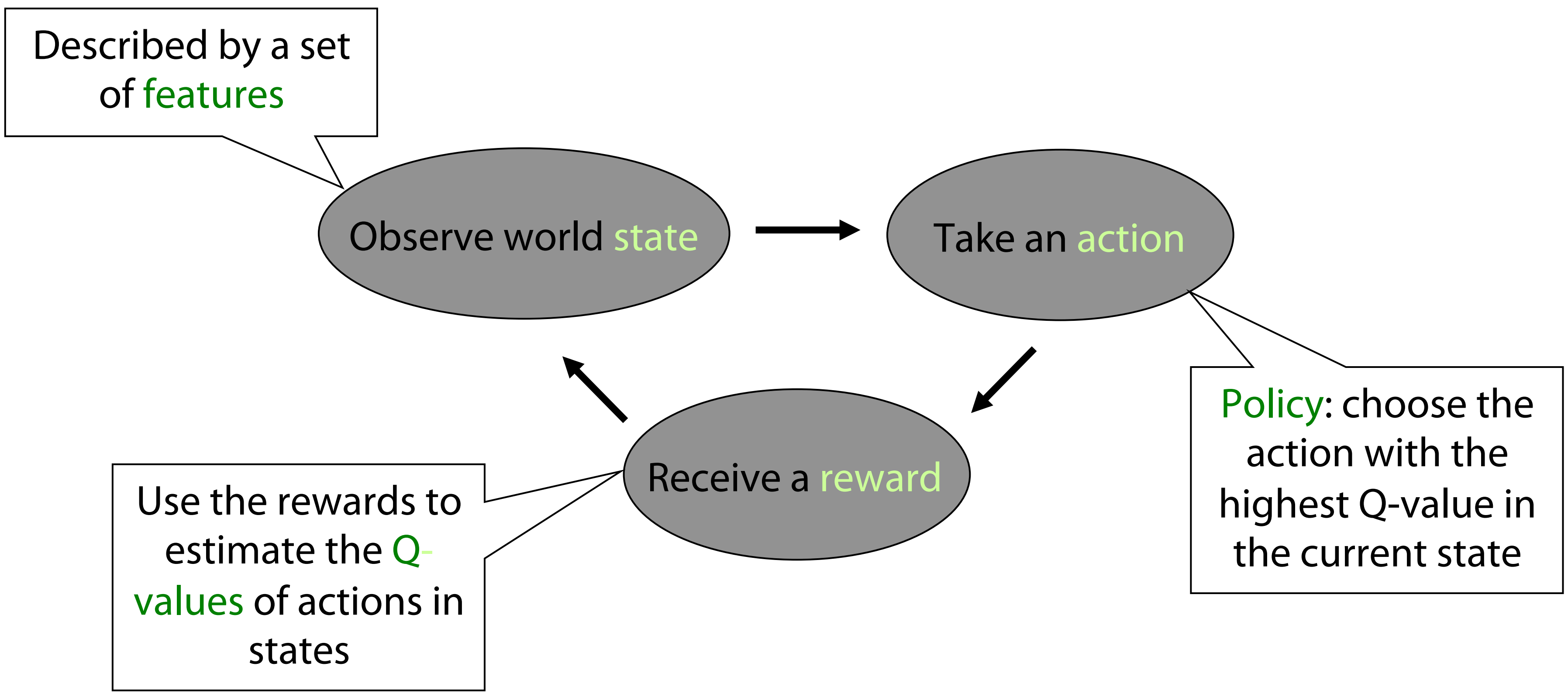
Agent uses this knowledge to learn Task B quickly

Goals of Transfer Learning

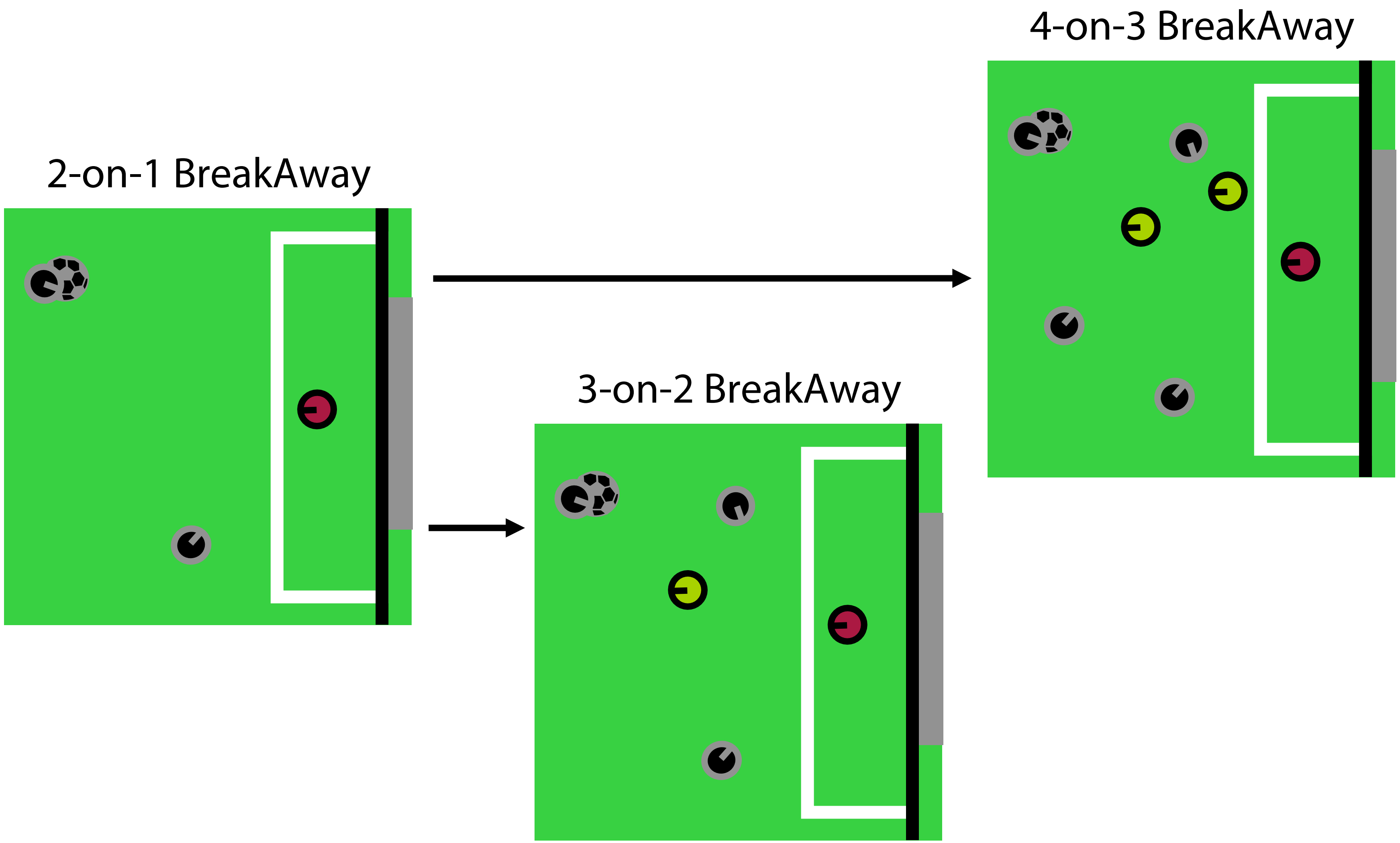
Learning curves in the target task:



Reinforcement Learning



The RoboCup Domain



Transfer in Reinforcement Learning

- Related work
  - Model reuse (Taylor & Stone 2005)
  - Policy reuse (Fernandez & Veloso 2006)
  - Option transfer (Perkins & Precup 1999)
  - Relational RL (Driessens et al. 2006)

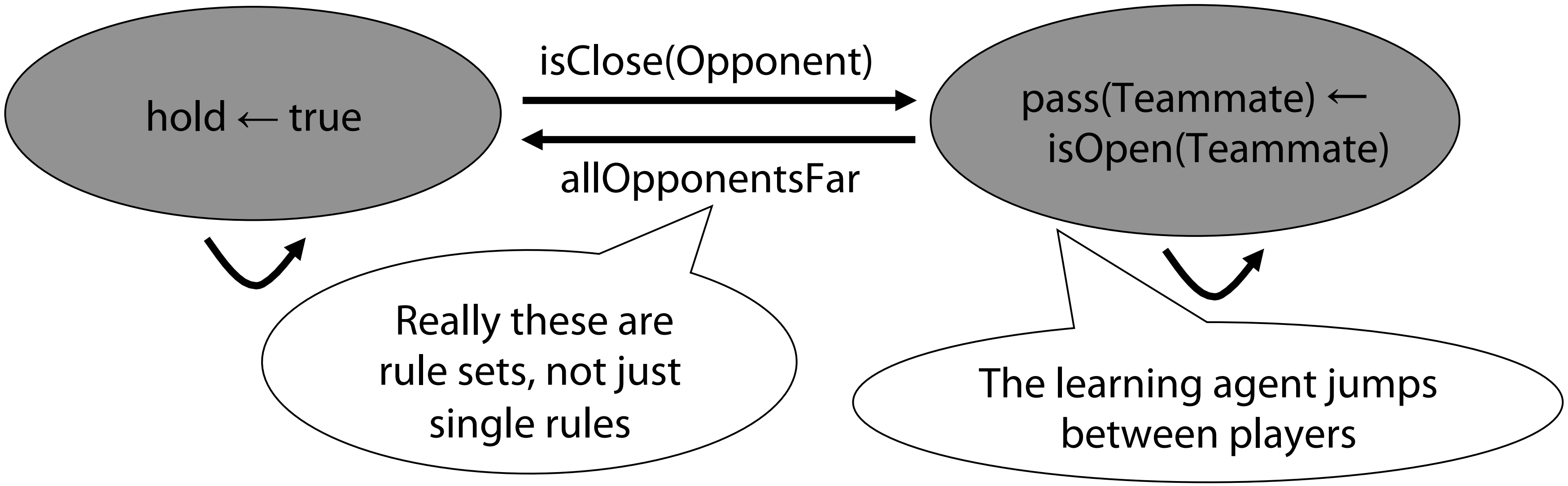
- Our previous work
  - Policy transfer (Torrey et al. 2005)
  - Skill transfer (Torrey et al. 2006)

Copy the Q-function

Learn rules that describe when to take individual actions

Now we learn a strategy instead of individual skills

Representing a Multi-step Strategy



- A relational macro is a finite-state machine
- Nodes represent internal states of agents in which limited independent policies apply
- Conditions for transitions and actions are in first-order logic

Our Proposed Method

- Learn a relational macro that describes a successful strategy in the source task
- Execute the macro in the target task to demonstrate the successful strategy
- Continue learning the target task with standard RL after the demonstration

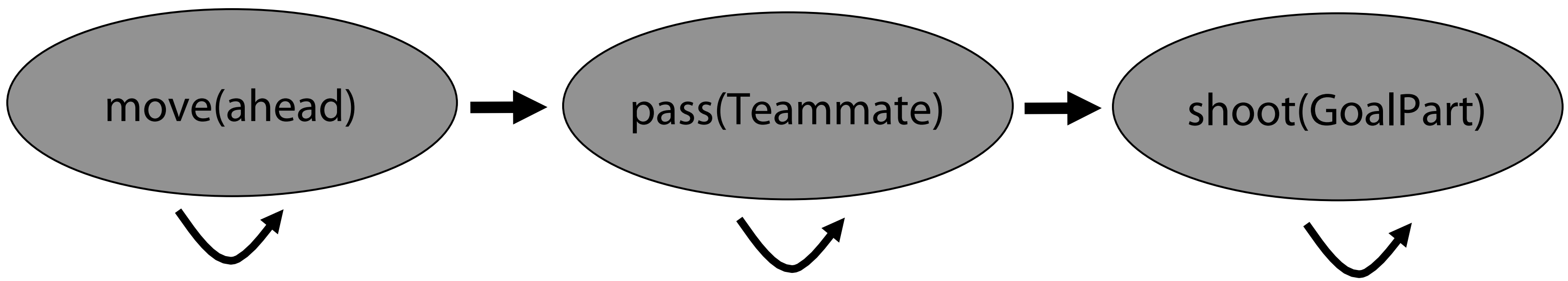
Learning a Relational Macro

- We use ILP to learn macros
- We learn a macro in two phases
  - The action sequence (node structure)
  - The rule sets for actions and transitions

Learning Macro Structure

- Objective: find an action pattern that separates good and bad games

macroSequence(Game) ←
  actionTaken(Game, StateA, move, ahead, StateB),
  actionTaken(Game, StateB, pass, _, StateC),
  actionTaken(Game, StateC, shoot, _, gameEnd).



Learning Macro Conditions

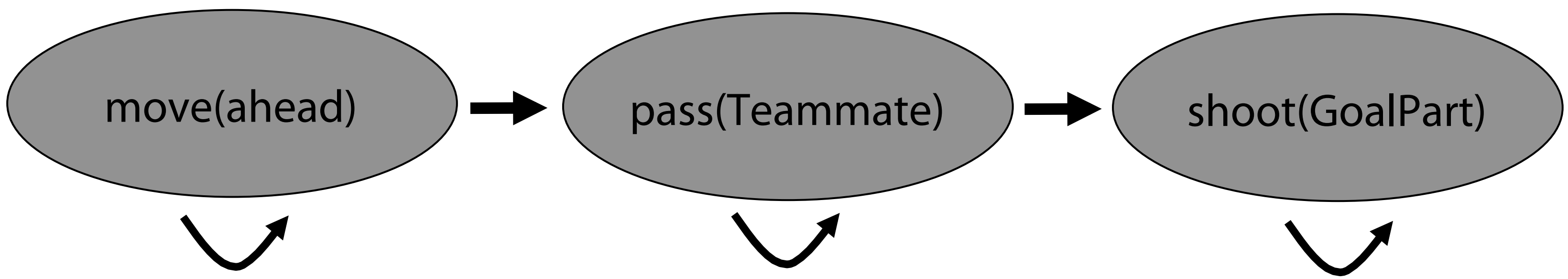
- Objective: describe when transitions and actions should be taken

For the transition from *move* to *pass*

transition(State) ←
    feature(State, distance(Teammate, goal)) < 15.

For the policy in the *pass* node

action(State, pass(Teammate)) ←
    feature(State, angle(Teammate, me, Opponent)) > 30.



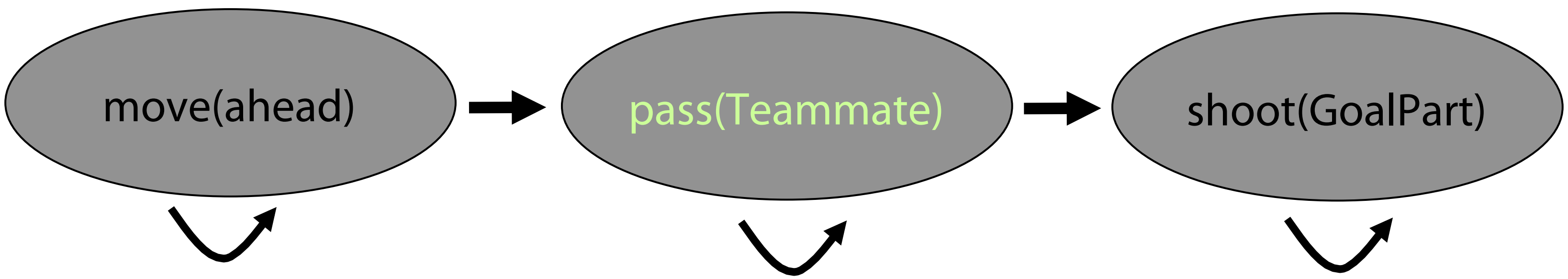
Examples for Actions

positive

scoring	Game 1:	move(ahead)	pass(a1)	shoot(goalRight)
	Game 2:	move(ahead)	pass(a2)	shoot(goalLeft)

negative

non-scoring	Game 3:	move(right)	pass(a1)	
	Game 4:	move(ahead)	pass(a1)	shoot(goalRight)



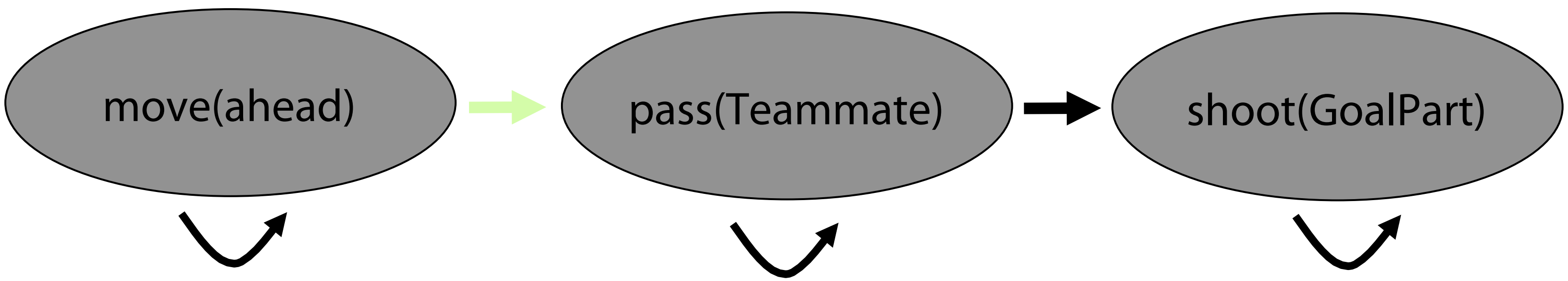
Examples for Transitions

positive

scoring
- Game 1: move(ahead) pass(a1) shoot(goalRight)
- Game 2: move(ahead) move(ahead) shoot(goalLeft)

negative

non-scoring
- Game 3: move(ahead) pass(a1) shoot(goalRight)



Transferring a Macro

- Demonstration
  - Execute the macro strategy to get Q-value estimates
  - Infer low Q-values for actions not taken by macro
  - Compute an initial Q-function with these examples
  - Continue learning with standard RL

- Advantage: potential for large immediate jump in performance
- Disadvantage: risk that agent will blindly follow an inappropriate strategy

Advice in RL

- **Advice** provides constraints on Q values under specified conditions

IF an opponent is near me
AND a teammate is open
THEN *Q(pass(teammate)) > Q(move(ahead))*

- Apply as *soft* constraints in optimization

Model size + C × Data misfit **+ μ × Advice misfit**

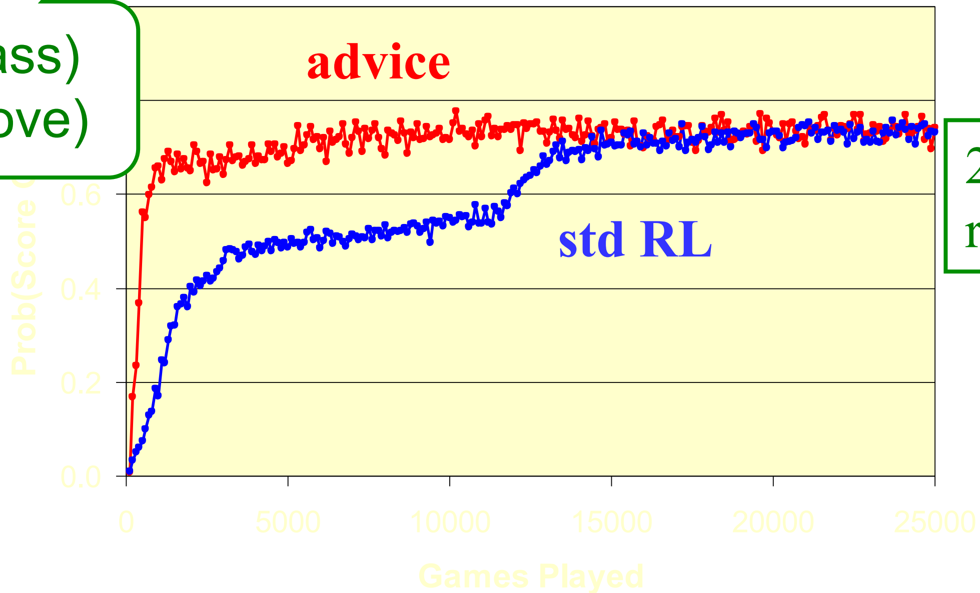
Sample Advice-Taking Results

if *distanceToGoal* [?] *10*

and *shotAngle* ≥ *30*

then prefer *shoot* over all other actions

Q(shoot) > Q(pass)
Q(shoot) > Q(move)



2 vs 1 BreakAway, rewards +1, -1

Conclusions

- This transfer method can significantly improve initial target-task performance
- It can handle new elements being added to the target task, but not new objectives
- It is an aggressive approach that is a good choice for tasks with similar strategies
- Advice taking as a specific kind of transfer learning