Reinforcement Learning in Infinite Mario

Shiwali Mohan and John E. Laird
shiwali@umich.edu laird@umich.edu

The Soar Group at University of Michigan

Research Question

- In a constrained cognitive architecture, does
  - describing the world using symbolic, object-oriented representations,
  - hierarchical task decomposition and learning
  - including internal goals and rewards in the design of the agent

result in better reinforcement learning in a complex task?

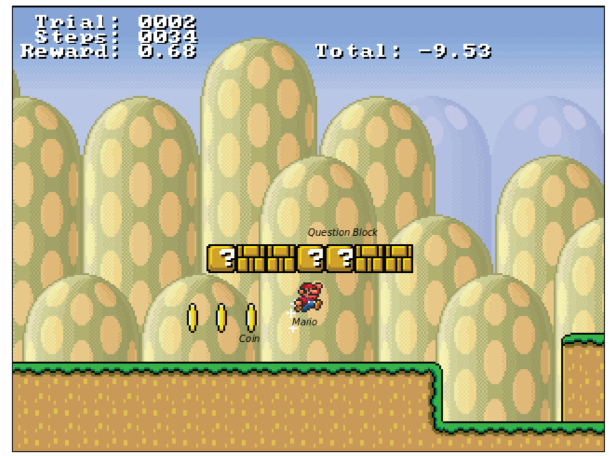
  - higher average reward
  - faster convergence

Outline

- Domain
- Challenges
- Reinforcement Learning
- Propositional Agent
- Symbolic, Object-Oriented Representation
- Hierarchical Reinforcement Learning
- Design
- Nuggets and Coal

Domain

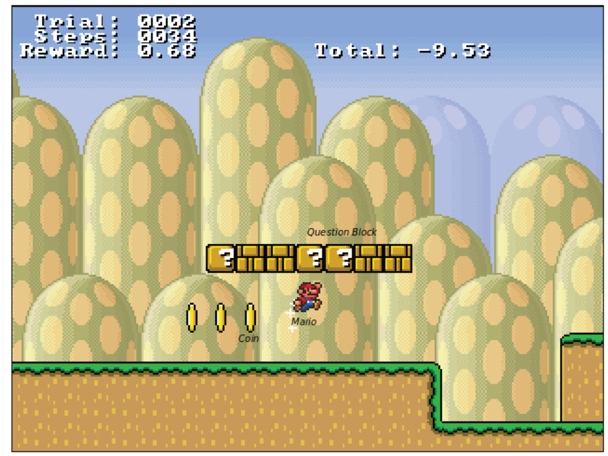
- Infinite Mario
  - Side scrolling game
  - Gain points by collecting coins, killing monsters
- Domain developed in RL-Glue
  - Reinforcement Learning Competition 2009
- State Observations
  - Visual Scene – 16 x 22 tiles, 13 different types
    - Episode is of arbitrary length
  - Monsters – can be of different types, speed etc
- Actions
  - Typical Nintendo Controls
    - Step right, left or stay
    - Jump
    - Speed toggle



- Reward
  - +100 on reaching the finish line
  - +1.00 for collecting coin, killing monster
  - -10.00 for termination of the game
  - -0.01 every step
- Sample Agent
  - Heuristic policy
  - Remembers sequence of actions

Domain

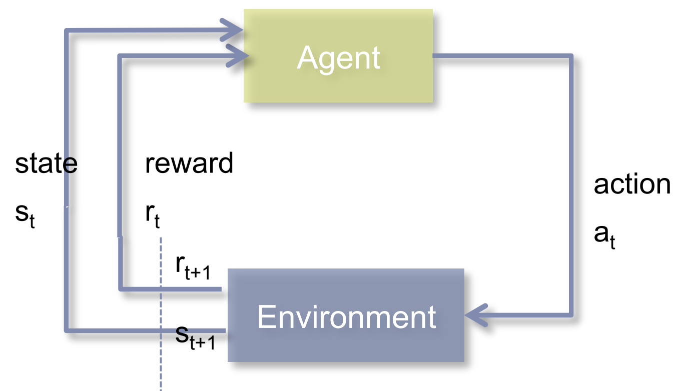
- Learning computationally expensive
  - Episode with 300 steps has ~5000 tiles of 13 different types
  - *Use 'good' state representations*
- Partial Observations
  - Only a part of the game instance is visible at a time.
  - *Assume that only local conditions matter, MDP assumption*
- Large, continuous, growing state space
  - Position, speed of objects (monster) are real numbers
  - Many different objects
  - *Value function augmentation, good state representation*
- Highly dynamic environment
  - High degree of relative movement
  - Despite available input data, predicting behavior is hard
  - *Learn from experience*



- Learning should generalize across different instances, levels of the game.
  - *Representations that are specific to a particular instance of game cannot be used.*
- Abundant information
  - Lot of extracted, derived data
  - *Learn what is important from instructor*

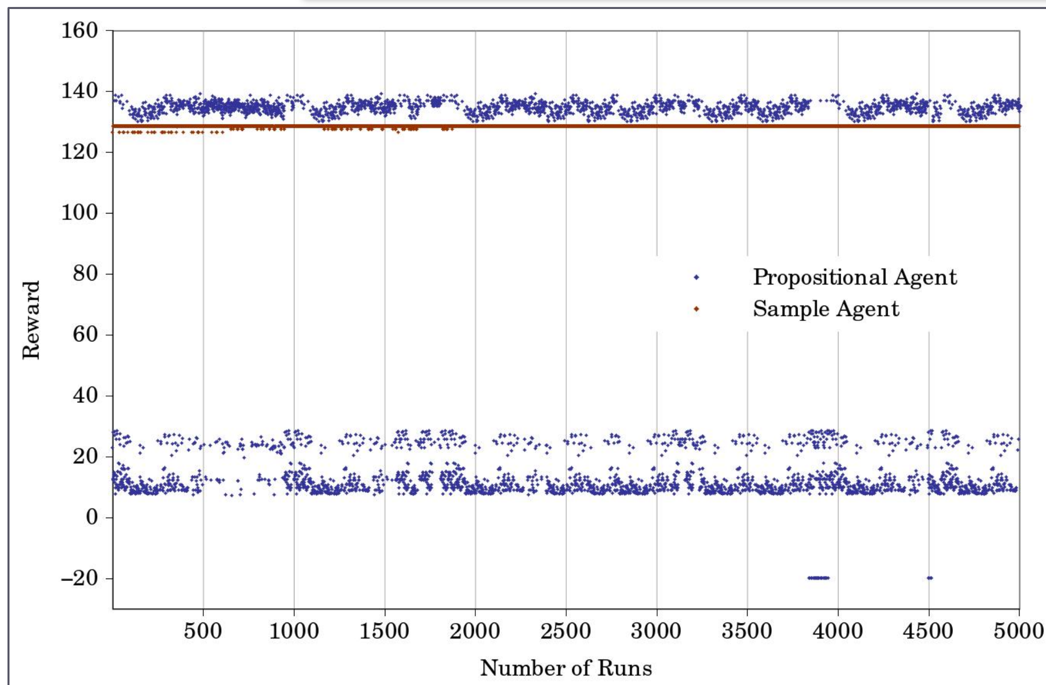
Reinforcement Learning

- Reinforcement Learning
  - Acquire domain knowledge from experience, online learning
- Formally, the basic reinforcement learning model
  - a set of environment states S;
  - a set of actions A; and
  - a set of scalar "rewards" R.
- Based on the interactions the RL agent develops a policy
  - Maximizes the reward earned



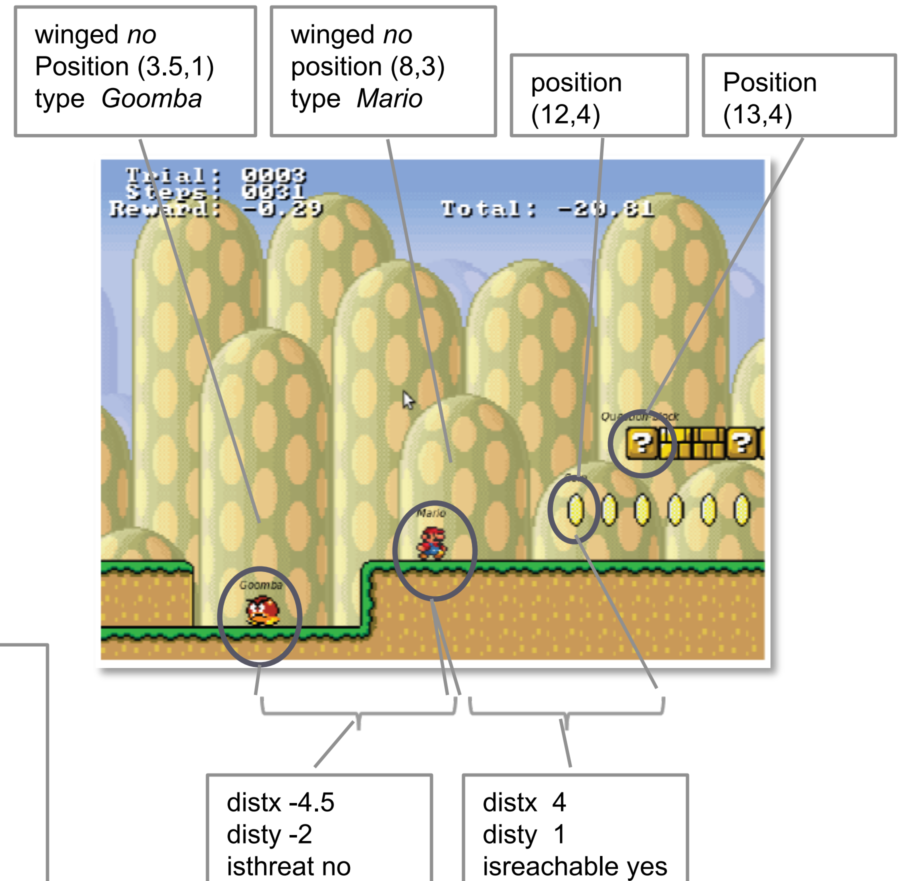
Propositional Agent

- Enormous state space
  - Visual Scene – 16*22 (352) tiles, of 13 different kinds = 13^352 states
    - All states do not really occur in the game
- Use very local information
  - 5*3 tiles around Mario
  - Include monsters that are within this range
- Learning is hard
  - Huge state-space
  - Reward achieved after a long sequence of steps
- Not clear how to provide background knowledge to aid learning
  - Extremely difficult, maybe impossible



Symbolic, Object-Oriented Representation
(agents 2, 3, 4)

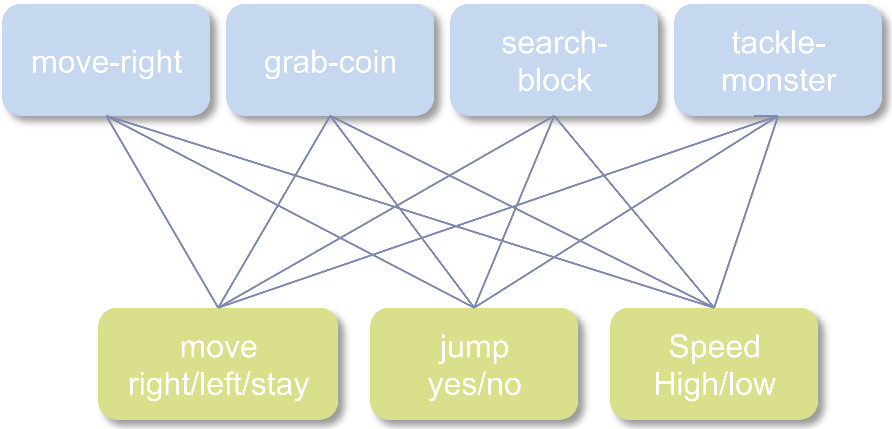
- Extract regular objects from inputs
  - Monsters, coins, question-blocks, platforms, pits
- Associate object with its features
  - *speed, type, position*
- Derive features
  - Relative distances between objects
  - Relative distances of objects from Mario
  - attributes like *'isreachable', 'isthreat'* if a object is close enough and should affect agents behavior
- Describe state
- Provide background knowledge
  - *If attribute 'isreachable' for a platform is set, and there is a coin on if, then set attribute 'isreachable' for the coin.*




```
state <s> ^name mario-soar            <m1> ^type Goomba
          ^monster <m1>                    ^winged no
          ^coin <c1>                        ^distx -4.5
          ^coin <c2>                        ^disty -2
          ^coin <c3>                         ^isthreat no
          ^coin <c4>
          ^coin <c5>                   <c1>   ^distx 4
          ^coin <c6>                          ^disty 1
          ^question-block <q1>                 ^isreachable yes
          ^question-block <q2>
```

Action (Operator) Hierarchy

- GOMS analysis of Mario[1]
  - Predictive of the behavior of human expert
  - Introduced functional-level operators and Keystroke-level
  - Divides the task into smaller tasks



- Two kinds of actions
  - FLOs
    - Functional-level Operators (actions)
    - Abstract macro-actions
    - Sequence of atomic actions
    - With a specific functional goal
  - KLOs
    - Keystroke –level Operators (actions)
    - Atomic actions
    - Move, jump, speed toggle

- Application of Actions
  - Object-Oriented
    - FLOs described for specific objects
    - *tackle-monster* for monsters
  - Control
    - Derived attributes used to control the progression
    - *'isthreat'* , *'isreachable'*

[1] B.E. John and A.H. Vera, "A GOMS analysis of a graphic machine-paced, highly interactive task," Proceedings of the SIGCHI conference on Human factors in computing systems, 1992, pp. 251–258.

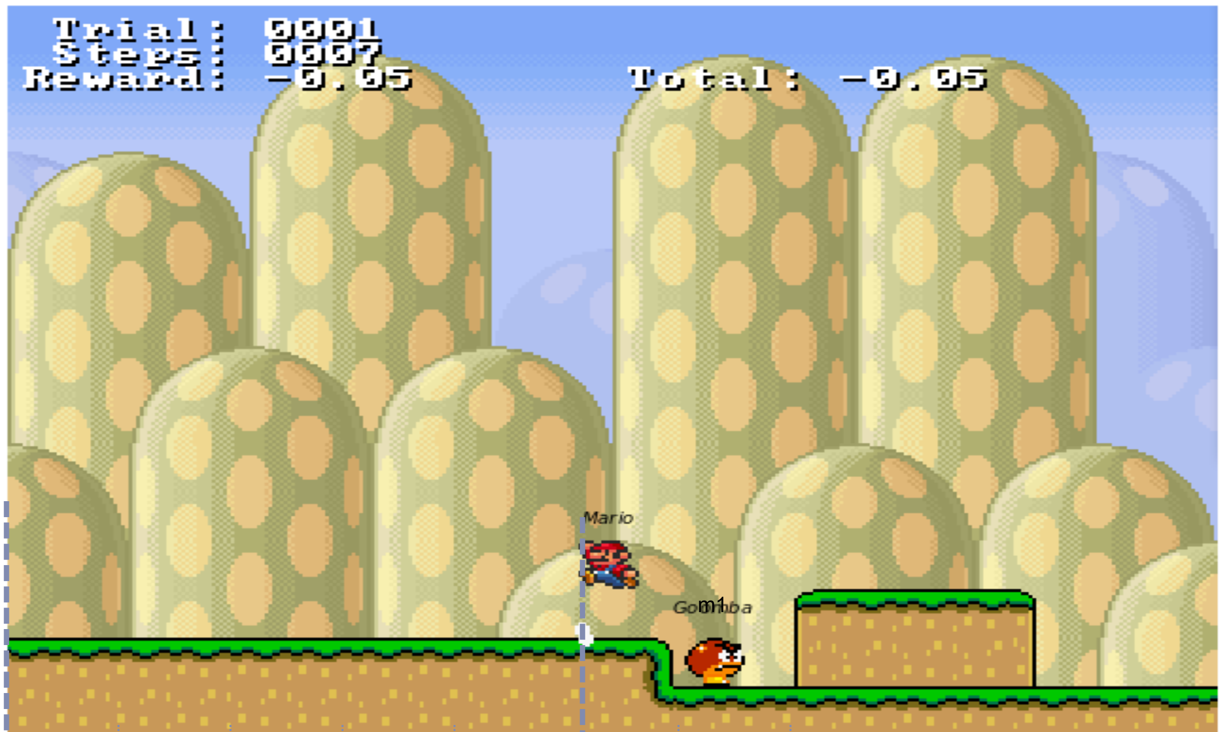
Progression

move-right

grab-coin

search-block

tackle-monster



state: mario-soar
available behavior:
move-right

state: mario-soar
available behavior:
move-right,
tackle-monster(m1)

m1 isthreat yes

state: move-right
available actions:
atomic-actions

state: tackle-monster
available behavior:
Atomic-actions

Agent 2: Learning at KLO level

- Can the agent learn behaviors like grabbing a coin, killing a monster?
- State
  - FLO-level: presence, absence of flag attributes like 'isthreat' or 'isreachable'

    monster <m1> ^isthreat true

    coin <c1> ^isreachable true

  - KLO-level: features extracted/derived from input

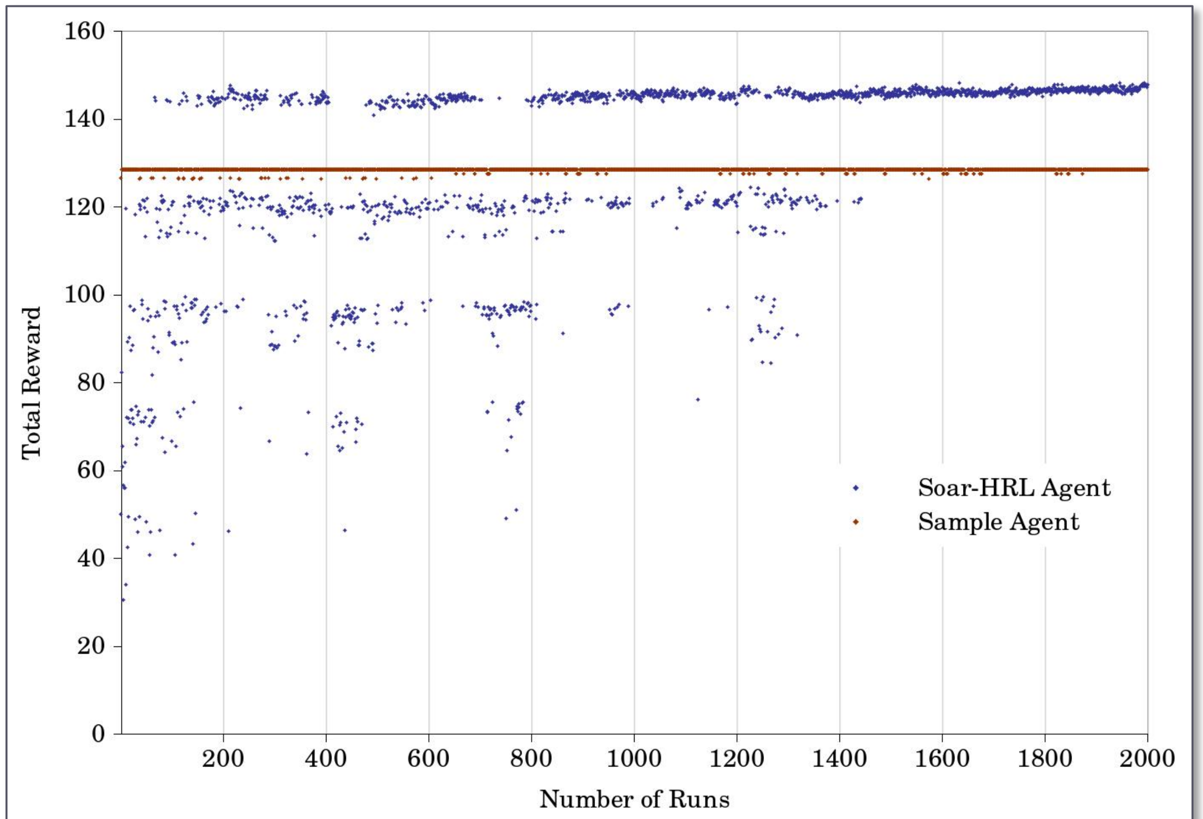
    monster <m1> ^type Goomba
^winged no
^distx <x>
^disty <y>
…

    coin <c1> ^distx <x>
^disty <y>

- Actions
  - FLOs : tackle-monster, grab-coin, search-question etc
    - Symbolic preferences used to break a tie
  - KLOs: move{right, left, stay} x jump{yes,no} x speed toggle {on,off}
- Learning
  - Given symbolic preferences at the FLO level
    - tackle-monster > move-right
  - Learn the most rewarding sequence of KLOs
- Reward
  - As provided by the environment , at KLO level

Results - Agent 2

- Averaged over 10 trials of 2000 runs each
- Learning algorithm - SARSA
- Learning rate – 0.3
- Discount rate – 0.9
- Exploration policy – Epsilon-greedy
- Epsilon – 0.01
- Reduction-rate – 0.99
- Policy converges at ~1400 episodes
- Average reward earned by converged policy (last 100 runs)=145.97
- Agent performs better than the sample agent.



Agent 3: Hierarchical Learning

- Can the agent learn preferences between objects as it learns behaviors?
  - Similar to MAXQ-0[1]
- State
  - FLO-level: presence, absence of flag attributes like ‘isthreat’ or ‘isreachable’

```
monster <m1> ^isthreat true
             ^distx <x>
             ^disty <y>
```

```
coin <c1> ^isreachable true
          ^distx <x>
          ^disty <y>
```

  - KLO-level: features extracted/derived from input

```
monster <m1> ^type Goomba
             ^winged no
             ^distx <x>
             ^disty <y>
             …
```

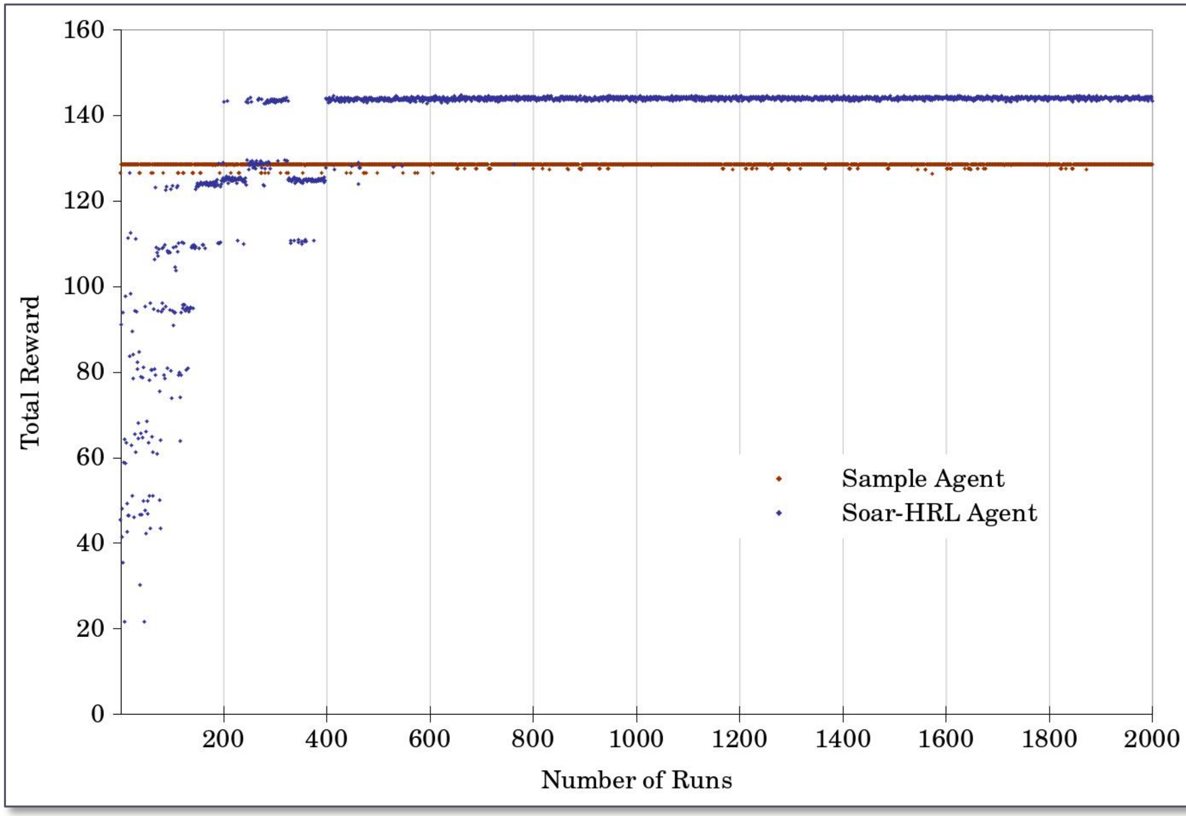
```
coin <c1> ^distx <x>
          ^disty <y>
```

- Actions
  - FLOs : tackle-monster, grab-coin, search-question etc
  - KLOs: move{right, left, stay} x jump{yes,no} x speed toggle {on,off}
- Learning
  - Learn numeric preferences at the FLO level
  - Learn the most rewarding sequence of KLOs
- Reward
  - As provided by the environment , at both KLO and FLO level

[1] T.G. Dietterich, “Hierarchical reinforcement learning with the MAXQ value function decomposition,” Journal of Artificial Intelligence Research, vol. 13, 2000, pp. 227–303.

Results – Agent 3

- Averaged over 10 trials of 2000 runs each
- Learning algorithm - SARSA
- Learning rate – 0.3
- Discount rate – 0.9
- Exploration policy – Epsilon-greedy
- Epsilon – 0.01
- Reduction-rate – 0.99
- Agent converges to a policy at 400 episodes
- Average reward earned by converged policy (last 100 runs)=144.68
- Agent converges faster than Agent 2 (1400 runs)
- Learns a more specific policy which might be better



Agent 4: Reward Shaping

- Can the performance of the agent be improved by introducing internal rewards and goals in the agent design?
  - MAXQ-Q[1]
  - Building in intrinsic goals and rewards[2]

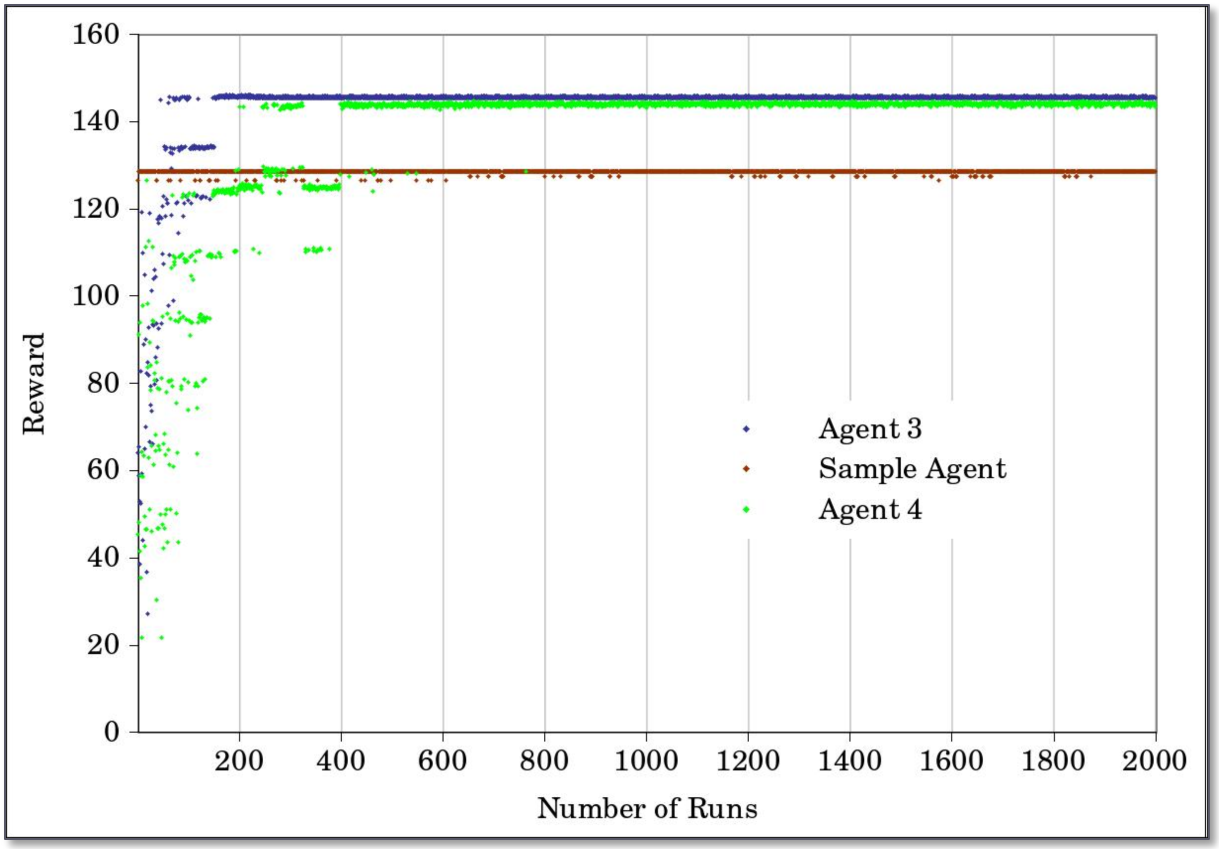
- State, Action, Learning
  - As in Agent 3

- Reward
  - Agent 3 uses the reward as provided by the environment
    - Agent may get rewarded even if it does not execute the selected FLO correctly,
      - grabbing a coin while tackle-monster is selected
  - Reward the agent at the KLO level only when the goal is achieved
    - +1.00 for correctly executing the selected FIO,
      - killing/avoiding a monster when tackle-monster is selected
    - -0.01 for every step
  - Reward at the FLO level is computed from the reward provided by the environment

[1] T.G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," Journal of Artificial Intelligence Research, vol. 13, 2000, pp. 227–303.
[2] S. Singh, R.L. Lewis, A.G. Barto, J. Sorg, and A. Helou, "On Separating Agent Designer Goals from Agent Goals: Breaking the Preferences–Parameters Confound," submitted, 2010.

Results – Agent 4

- Averaged over 10 trials of 2000 runs each
- Learning algorithm - SARSA
- Learning rate – 0.3
- Discount rate – 0.9
- Exploration policy – Epsilon-greedy
- Epsilon – 0.01
- Reduction-rate – 0.99
- Agent converges to a policy at 200 episodes
- Average reward earned by converged policy (last 100 runs)=145.98
- Average standard deviation (last 100 runs) = 2.0083
- Converges faster than Agent 3 (400 runs), correct intrinsic reward



Nuggets and Coal

- Hierarchical division of the task makes operating in the environment easier
- Hierarchical learning allows the agent to learn policies faster
- Demonstrated that object-oriented representations provide structure to state description
- Symbolic representations allow for easy encoding of the background knowledge
- Encoding intrinsic rewards in an agent helps it learn faster.
- Detailed analytical study of the domain
  - Optimality
  - Good comparison metric
  - Parameter sweeps
  - Transfer Learning
- Learning at one level should help in playing a higher level