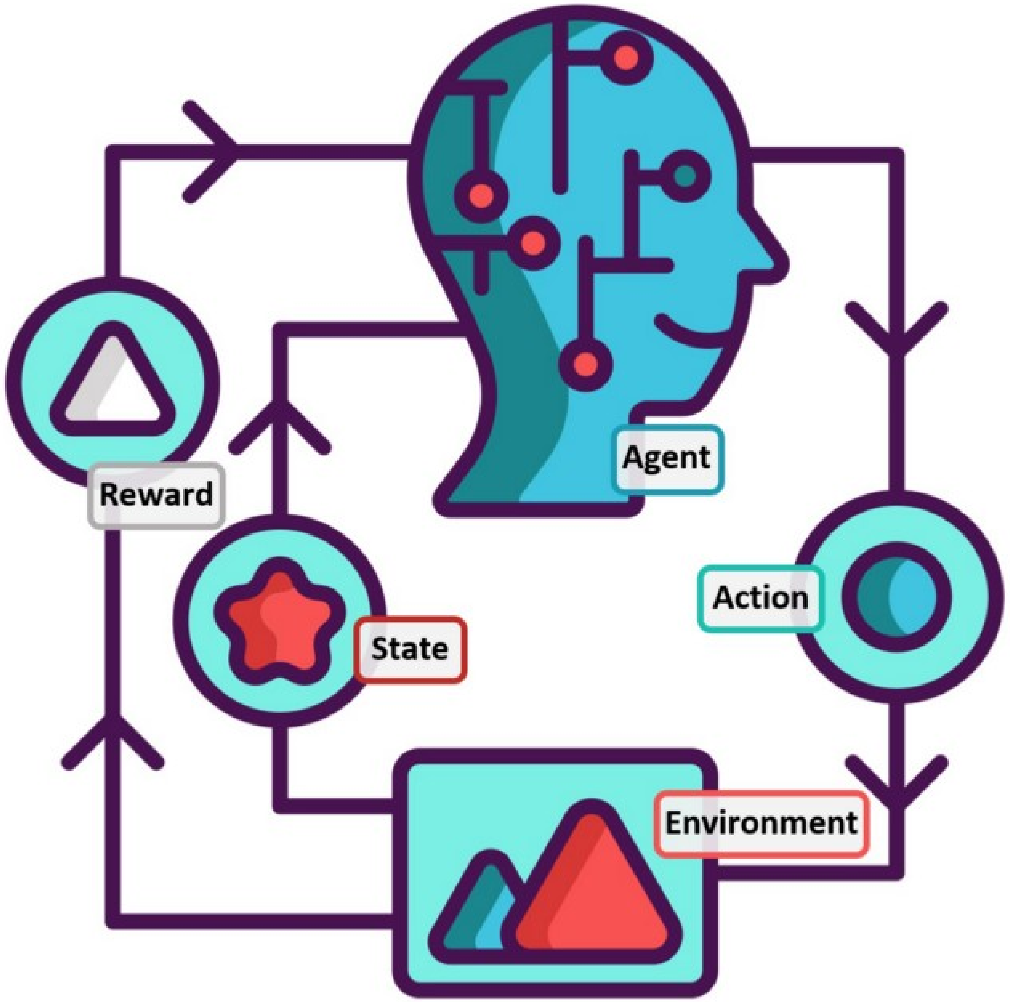
MAFTEC Days
2023

Predicate-based explanation of Reinforcement Learning

Léo Saulières (3rd year PhD student)

Martin C. Cooper – Florence Bannay

Reinforcement Learning



HXP

Goal: Explain past agent’s interactions with the environment (history) through the prism of a predicate d

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate d

Question: Which actions were important to ensure that d was achieved, given the agent's policy π?

HXP

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate d

Question: Which actions were important to ensure that d was achieved, given the agent's policy π?

Idea: Compute the *action importance score* for each state-action (s,a) in the length-k history h

HXP

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate d

Question: Which actions were important to ensure that d was achieved, given the agent's policy π?

Idea: Compute the *action importance score* for each state-action (s,a) in the length-k history h

> *The action importance score of an action a, from a state s in the history is the difference between the utility of a and the average utility of any other action $a' \in A(s) \setminus \{a\}$*

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate *d*

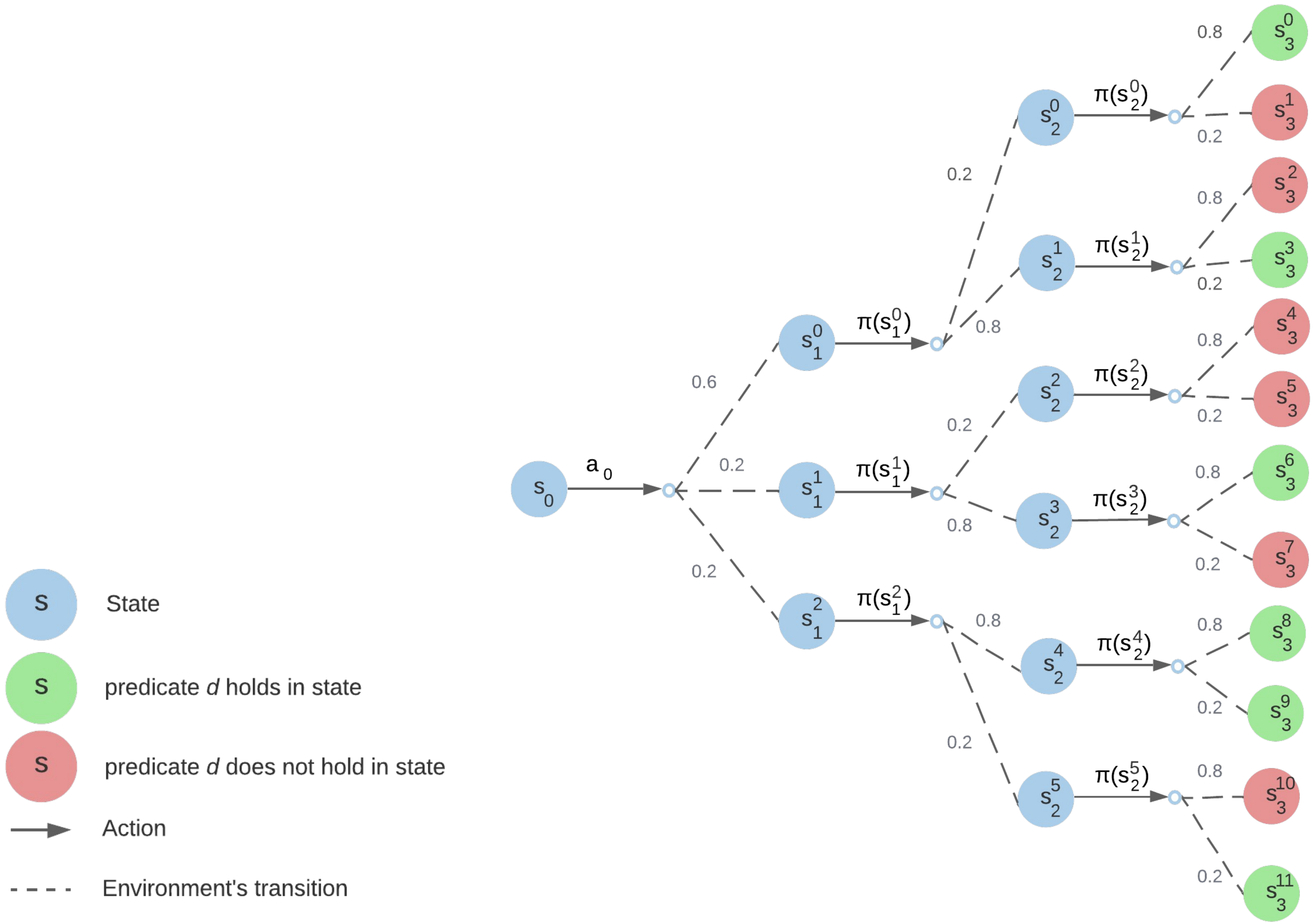
Question: Which actions were important to ensure that *d* was achieved, given the agent's policy π?

Idea: Compute the *action importance score* for each state-action *(s,a)* in the length-*k* history *h*

> *The action importance score of an action a, from a state s in the history is the difference between the utility of a and the average utility of any other action $a' \in A(s) \setminus \{a\}$*

Utility: probability to reach a final state at horizon *k* which respects *d*
Action importance score lies in range [-1;1]

HXP



s_0
a_0
0.6
0.2
0.2
s_1^0
s_1^1
s_1^2
$\pi(s_1^0)$
$\pi(s_1^1)$
$\pi(s_1^2)$
0.2
0.8
0.2
0.8
0.8
0.2
s_2^0
s_2^1
s_2^2
s_2^3
s_2^4
s_2^5
$\pi(s_2^0)$
$\pi(s_2^1)$
$\pi(s_2^2)$
$\pi(s_2^3)$
$\pi(s_2^4)$
$\pi(s_2^5)$
0.8
0.2
0.8
0.2
0.8
0.2
0.8
0.2
0.8
0.2
0.8
0.2
s_3^0
s_3^1
s_3^2
s_3^3
s_3^4
s_3^5
s_3^6
s_3^7
s_3^8
s_3^9
s_3^{10}
s_3^{11}
S
State
S
predicate *d* holds in state
S
predicate *d* does not hold in state
Action
Environment's transition

HXP

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate d

Question: Which actions were important to ensure that d was achieved, given the agent's policy π?

Idea: Compute the *action importance score* for each state-action (s,a) in the length-k history h

> *The action importance score of an action a, from a state s in the history is the difference between the utility of a and the average utility of any other action $a' \in A(s) \setminus \{a\}$*

Utility: probability to reach a final state at horizon k which respects d
Action importance score lies in range [-1;1]

Problem: Computationnaly expensive method (#W[1]-hard)

HXP

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate d

Question: Which actions were important to ensure that d was achieved, given the agent's policy π?

Idea: Compute the *action importance score* for each state-action (s,a) in the length-k history h

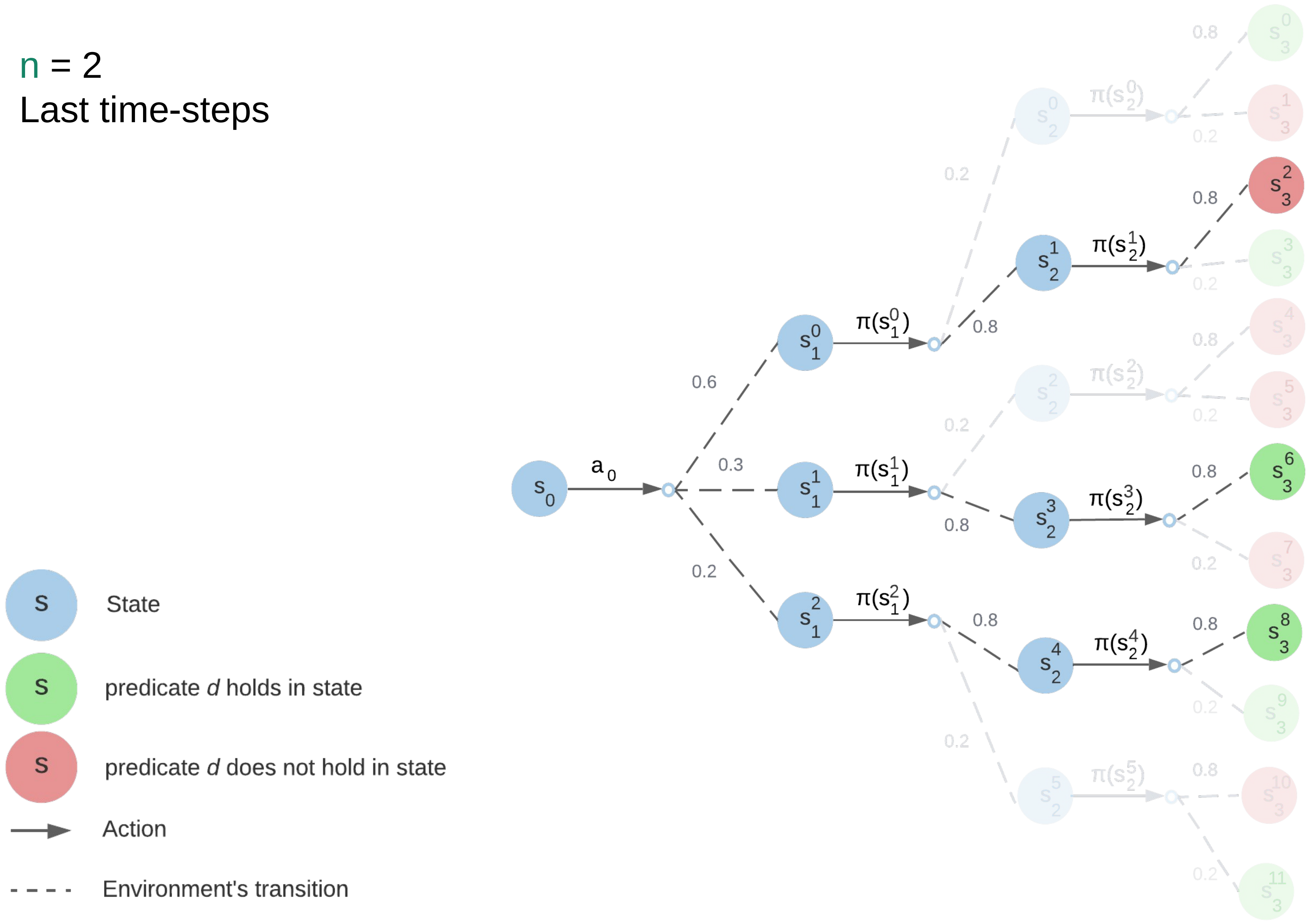
> *The action importance score of an action a, from a state s in the history is the difference between the utility of a and the average utility of any other action $a' \in A(s) \setminus \{a\}$*

Utility: probability to reach a final state at horizon k which respects d
Action importance score lies in range [-1;1]

Problem: Computationnaly expensive method (#W[1]-hard)

Solution: Generate a large range of scenarios, but not the unlikely ones
n-last approximate HXP: most probable transition at the n last time-step(s)

n = 2
Last time-steps



HXP

Goal: Explain past agent's interactions with the environment (history) through the prism of a predicate d

Question: Which actions were important to ensure that d was achieved, given the agent's policy π?

Idea: Compute the *action importance score* for each state-action (s,a) in the length-k history h

The action importance score of an action a, from a state s in the history is the difference between the utility of a and the average utility of any other action $a' \in A(s) \setminus \{a\}$

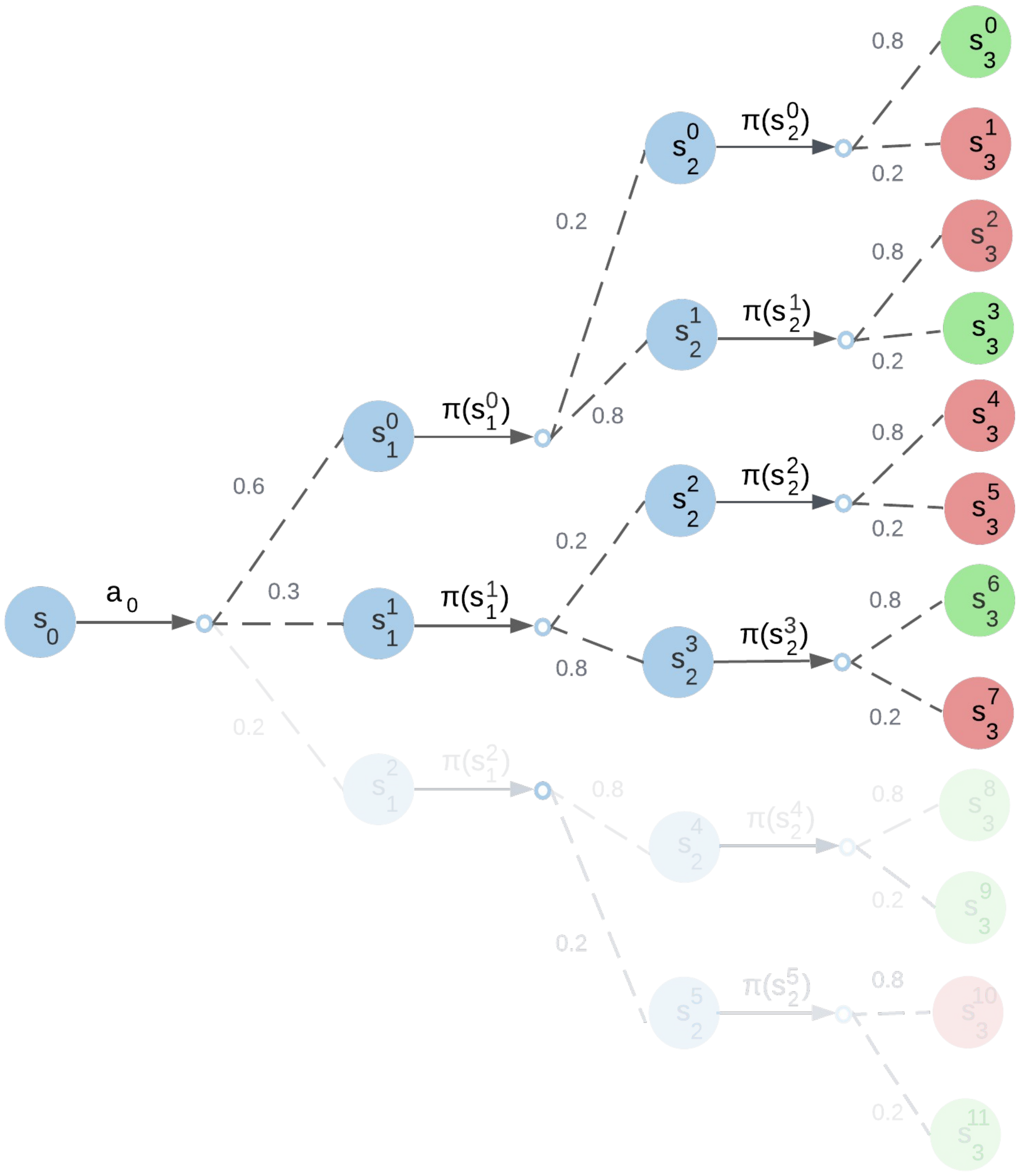
Utility: probability to reach a final state at horizon k which respects d
Action importance score lies in range [-1;1]

Problem: Computationnaly expensive method (#W[1]-hard)

Solution: Generate a large range of scenarios, but not the unlikely ones
m-transition approximate HXP: m most probable transition at each time-step

$m = 2$

Each time-step



s State

s predicate *d* holds in state

s predicate *d* does not hold in state

Action

Environment's transition

HXP

Problem: Large history requires an important number of deterministic transitions to provide in reasonable time action importance scores.

This leads to a decrease of the importance scores quality.

HXP

Problem: Large history requires an important number of deterministic transitions to provide in reasonable time action importance scores.
This leads to a decrease of the importance scores quality.

Solution: Backward HXP

Backward HXP

Data:

- History i.e. state-action sequence $H = (s_0, a_0, s_1, ..., a_{k-1}, s_k)$
- Predicate *d*
- Length of studied sub-sequence *l*

Backward HXP

Data:

- History i.e. state-action sequence $H = (s_0, a_0, s_1, ..., a_{k-1}, s_k)$
- Predicate *d*
- Length of studied sub-sequence *l*

Idea: Search for the most important action a_i among the *l* last actions of $H_{(k-m,k)}$

Get the associated state s_i

Redefine the predicate to study based on s_i

Search for the most important action of $H_{(i-m,i)}$

Iterate this process through the entire history

Backward HXP

Data:

- History i.e. state-action sequence $H = (s_0, a_0, s_1, ..., a_{k-1}, s_k)$
- Predicate *d*
- Length of studied sub-sequence *l*

Idea: Search for the most important action a_i among the *l* last actions of $H_{(k-m,k)}$

Get the associated state s_i

Redefine the predicate to study based on s_i

Search for the most important action of $H_{(i-m,i)}$

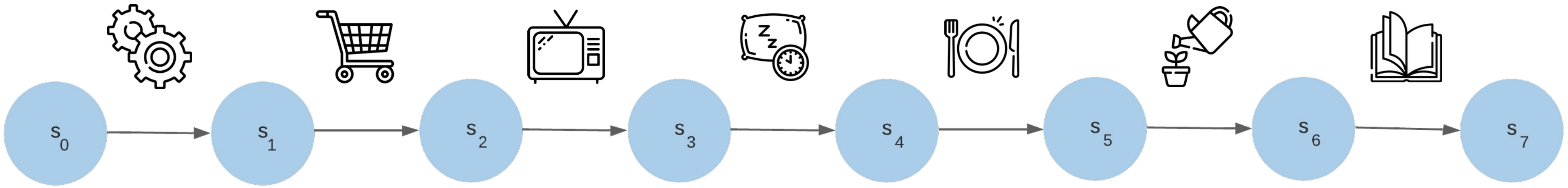
Iterate this process through the entire history

The re-defined predicate is a general description of a set of states

Result: Set of studied predicates and important actions

Backward HXP

Example: the end of Bob's day



Bob's state: *(hunger, happy, tired, fridge, fuel)*

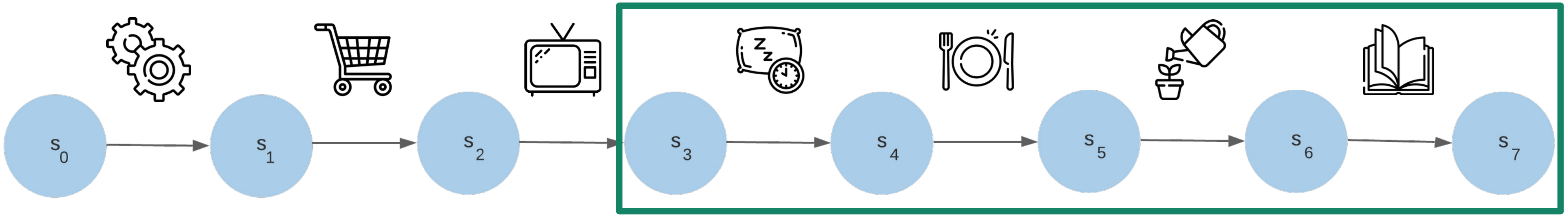
Last history state: *(¬hunger, happy, tired, ¬fridge, ¬fuel)*

Data:

- H: history corresponding to the end of Bob's day
- *d*: 'Bob is not hungry'
- *l*: 4

Which actions were important to ensure that *d* was achieved, given the agent's policy π?

Example: the end of Bob's day

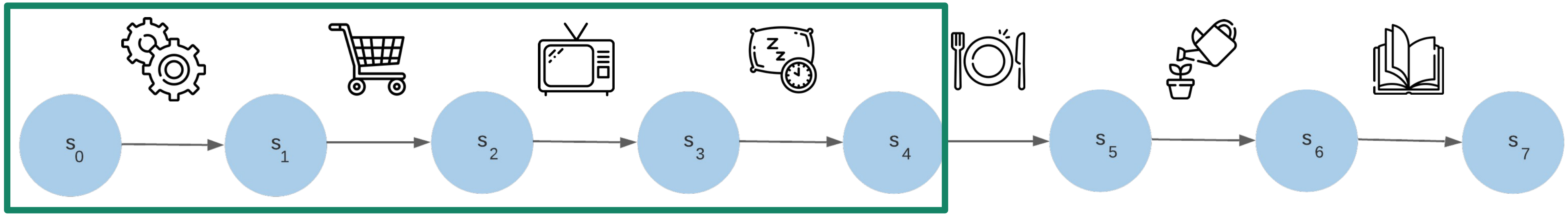


Most important action: *'eat'*

New predicate based on s_4, *d*: *'Bob is hungry and has a full fridge'*

Backward HXP

Example: the end of Bob's day



Most important action: *'shop'*

Result:

- Actions: *'shop', 'eat'*
- Predicates : *'Bob is hungry and has a full fridge', 'Bob is not hungry'*

  Bob isn't hungry because he went shopping (to fill his fridge) and then ate

Backward HXP

Algorithm 2: Backward HXP algorithm

Input : H: history, l: maximal sub-sequence length, π: agent's policy, d: predicate, p: transition function, δ: probability threshold

Output: A: action list, D: predicate list

$A \leftarrow []$;
$D \leftarrow []$;
$i_{max} \leftarrow len(H)$;
while $i_{min} \neq 0$ **do**
  $i_{min} \leftarrow max(0, i_{max} - l)$;
  $a, s, z, idx \leftarrow \text{select}(H_{(i_{min}, i_{max})}, \pi, d, p)$; `// select a state-action couple`
  $d \leftarrow \text{all\_PAXp}(\mathbb{F}, \kappa, s, \delta, \pi, p, d, i_{max} - i_{min})$; `// define a new predicate`
  $A.append(a)$;
  $D.append(d)$;
  $i_{max} \leftarrow idx$;
end
return A, D

Backward HXP

State-action couple selection: Most important action a and associated state s

Backward HXP

State-action couple selection: Most important action a and associated state s

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate d and s

Backward HXP

State-action couple selection: Most important action a and associated state s

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate d and s

PAXp is a formal method to explain classifiers in terms of feature selection

Backward HXP

State-action couple selection: Most important action *a* and associated state *s*

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate *d* and *s*

PAXp is a formal method to explain classifiers in terms of feature selection

Classifier: *Is x at least as useful as s?* $\kappa_s(\mathbf{x}) = u_d(\mathbf{x}) \geq u_d(s)$

Backward HXP

State-action couple selection: Most important action *a* and associated state *s*

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate *d* and *s*

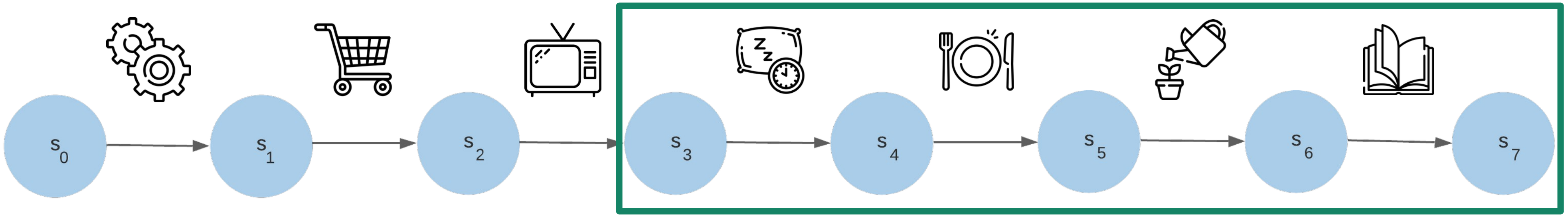
PAXp is a formal method to explain classifiers in terms of feature selection

Classifier: *Is x at least as useful as s?* $\kappa_s(\mathbf{x}) = u_d(\mathbf{x}) \geq u_d(s)$

Weak PAXp: *A subset of fixed features for which the probability of predicting a class c is at least δ (with δ ∈ [0,1])*

Backward HXP

Example: Bob's end day



<u>Most important action</u>: *'eat'*

<u>Associated state s_4</u>: (hunger, ¬happy, ¬tired, fridge, ¬fuel)

$u(s_4) = 1$, $\delta = 0.8$

Weak PAXp: *(hunger, fridge, ¬tired)* → new predicate

At least 80% of states described by *(hunger, fridge, ¬tired)* have a utility of 1

Backward HXP

State-action couple selection: Most important action *a* and associated state *s*

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate *d* and *s*

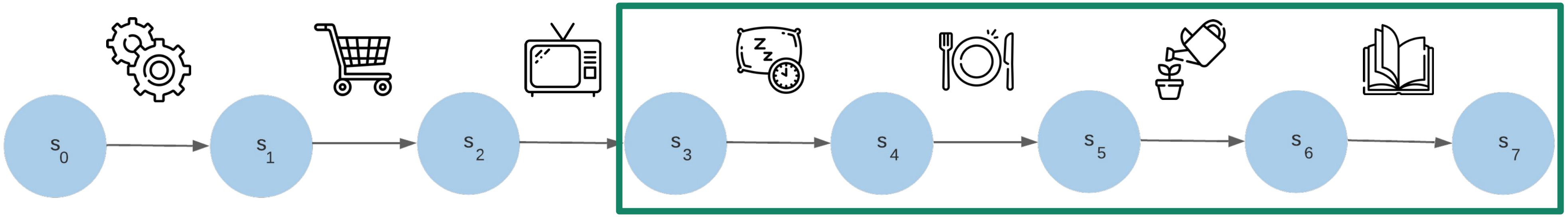
PAXp is a formal method to explain classifiers in terms of feature selection

Classifier: *Is x at least as useful as s?* $\kappa_s(\mathbf{x}) = u_d(\mathbf{x}) \geq u_d(s)$

Weak PAXp: *A subset of fixed features for which the probability of predicting a class c is at least δ (with δ ∈ [0,1])*

PAXp: *A Weak PAXp which is subset minimal*

Example: Bob's end day



Most important action: *'eat'*

Associated state s_4: (hunger, ¬happy, ¬tired, fridge, ¬fuel)

$u(s_4) = 1, \ \delta = 0.8$

PAXp: *(hunger, fridge)* → new predicate

At least 80% of states described by *(hunger, fridge)* have a utility of 1

Backward HXP

State-action couple selection: Most important action *a* and associated state *s*

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate *d* and *s*

PAXp is a formal method to explain classifiers in terms of feature selection

Classifier: *Is x at least as useful as s?* $\kappa_s(\mathbf{x}) = u_d(\mathbf{x}) \geq u_d(s)$

Weak PAXp: *A subset of fixed features for which the probability of predicting a class c is at least δ (with δ ∈ [0,1])*

PAXp: *A Weak PAXp which is subset minimal*

Problem: Finding one *PAXp* is computationnaly expensive

Backward HXP

State-action couple selection: Most important action *a* and associated state *s*

Predicate definition: Disjunction of all probabilistic Abductive eXplanations (PAXp) based on predicate *d* and *s*

PAXp is a formal method to explain classifiers in terms of feature selection

Classifier: *Is x at least as useful as s?* $\kappa_s(\mathbf{x}) = u_d(\mathbf{x}) \geq u_d(s)$

Weak PAXp: *A subset of fixed features for which the probability of predicting a class c is at least δ (with δ ∈ [0,1])*

PAXp: *A Weak PAXp which is subset minimal*

Problem: Finding one *PAXp* is computationnaly expensive

Solution: Generate the new predicate with only one *Locally-minimal PAXp*, a class of *Weak PAXp* which is easier to compute

Backward HXP

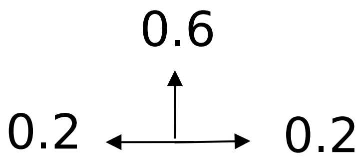
Algorithm 2: findLmPAXp.

Input : Feature $\{1, \ldots, m\}$; feature space $\mathbb{F}$, classifier κ, instance $(\mathbf{v}, c)$, threshold δ
Output: Locally-minimal PAXp $\mathcal{S}$

$\mathcal{S} \leftarrow \{1, \ldots, m\}$;
for $i \in \{1, \ldots, m\}$ **do**
  if *WeakPAXp*$(\mathcal{S} \backslash \{i\}; \mathbb{F}, \kappa, \mathbf{v}, c, \delta)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \backslash \{i\}$;
  end
end
return $\mathcal{S}$

Frozen Lake

Transition function (⬆)



Actions ⬆⬇⬅➡

State

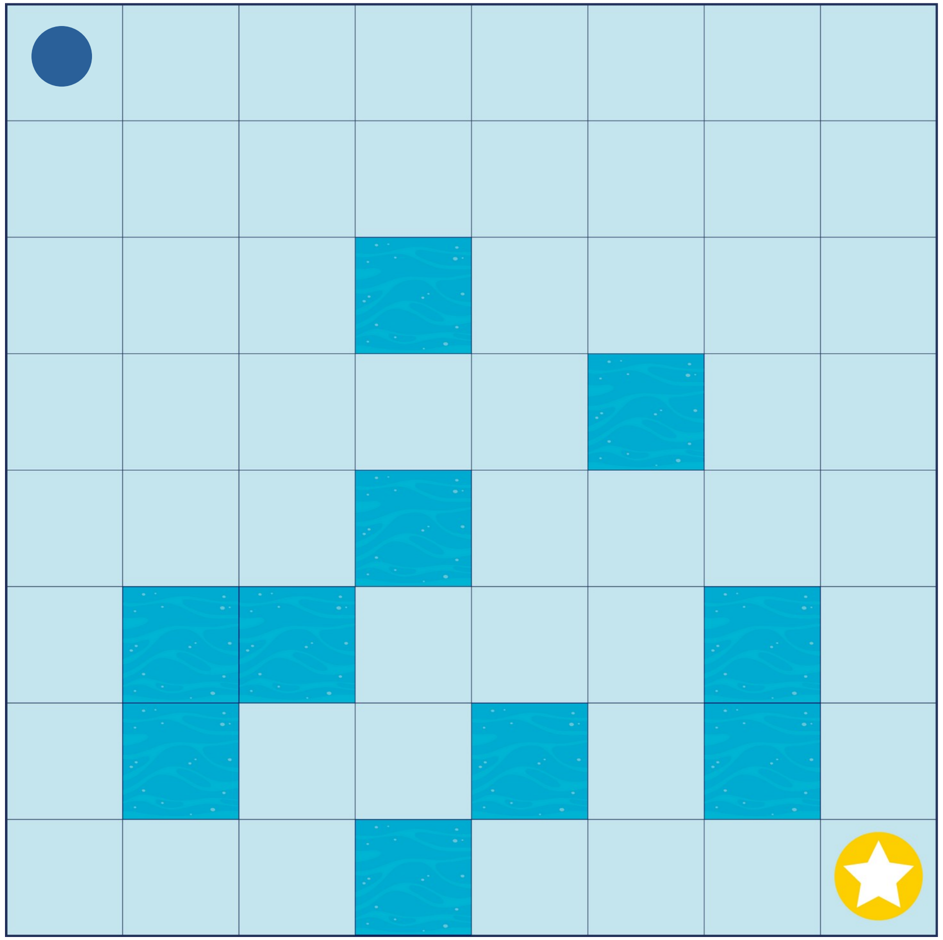
- Position
- Previous position
- Position of a closest hole
- Distance starting/current position
- Number of holes

Reward function

- +1 in Goal position
- +0 otherwise

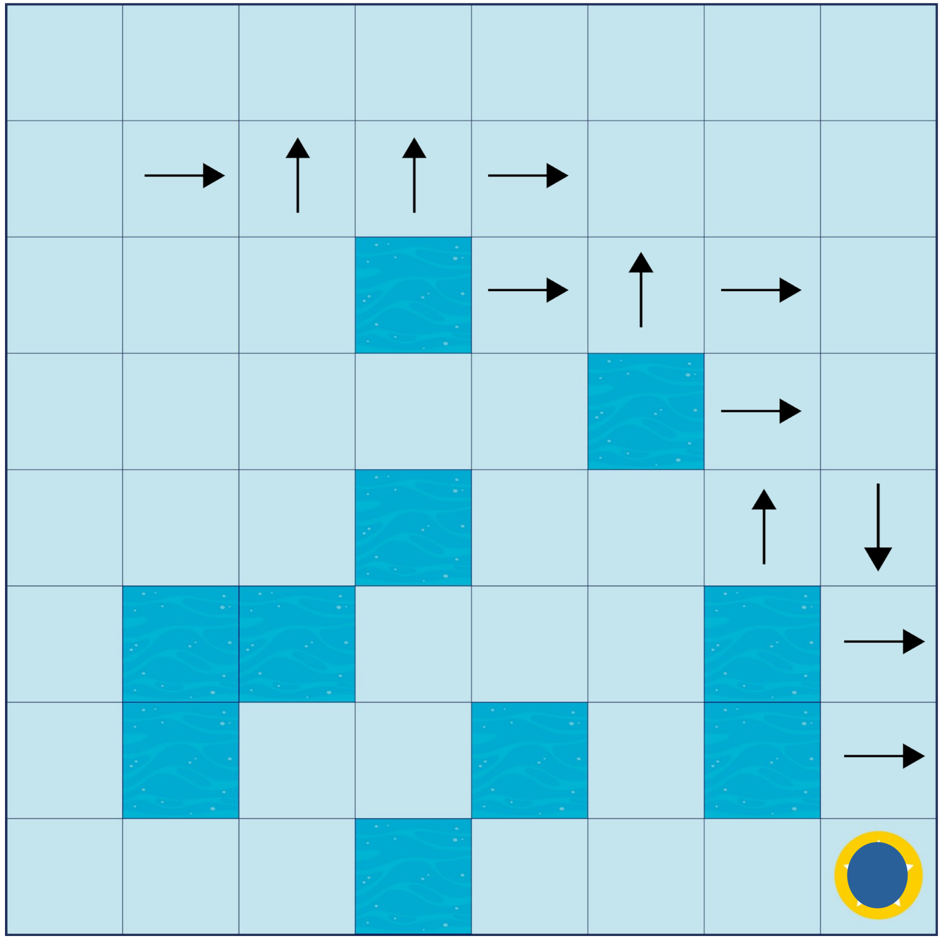
Algorithm Tabular Q-learning

Predicates *goal, holes, region*



Frozen Lake

History



Predicate: *goal*

Frozen Lake

B-HXP ($l = 4$, $\delta = 0.8$)

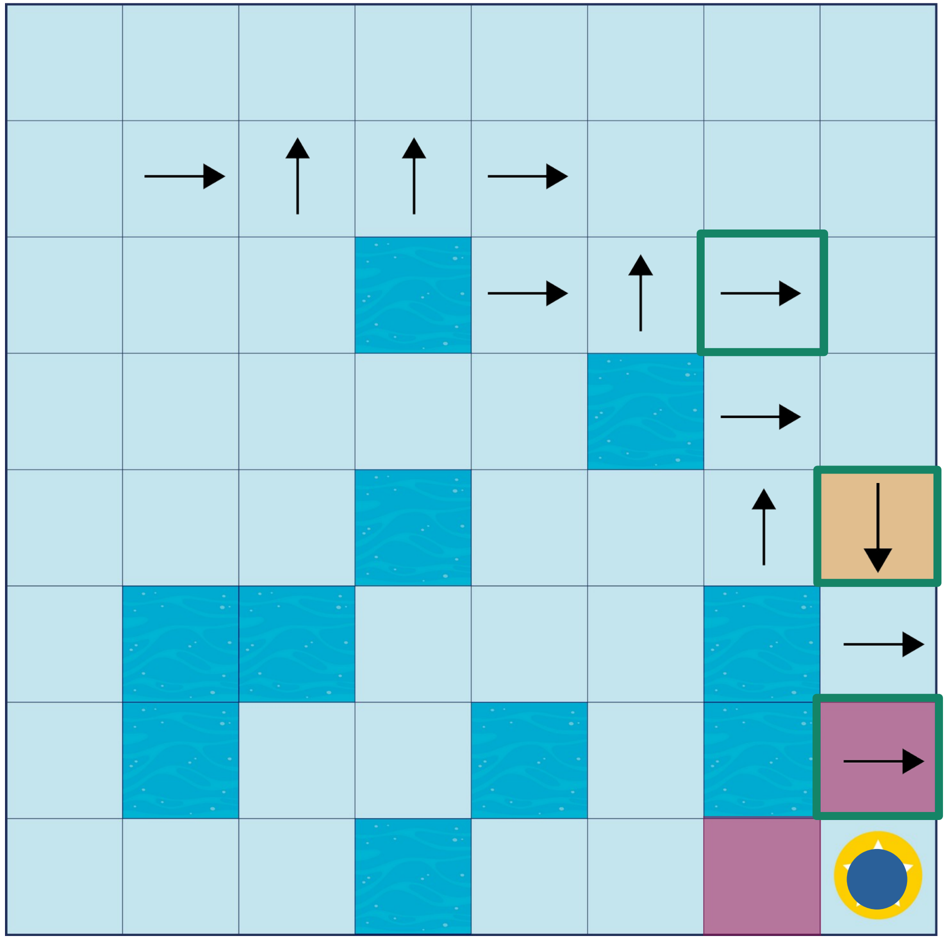
Scores: [-0.0, **0.0**, -0.001, -0.0 |
0.006, -0.009, **0.102**, 0.087 |
-0.001, 0.04, 0.012, **0.114**]

Predicates:

- Position, Previous position, Close hole position
- Distance starting/current position
- Goal

Runtime: 2.45s

History



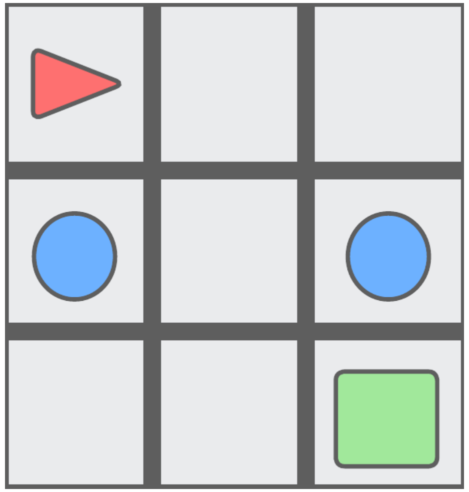
Dynamic Obstacles

Transition function Obstacles moves

Actions

- Move forward
- Rotate 90° left
- Rotate 90° right

State 7x7 view



Reward function

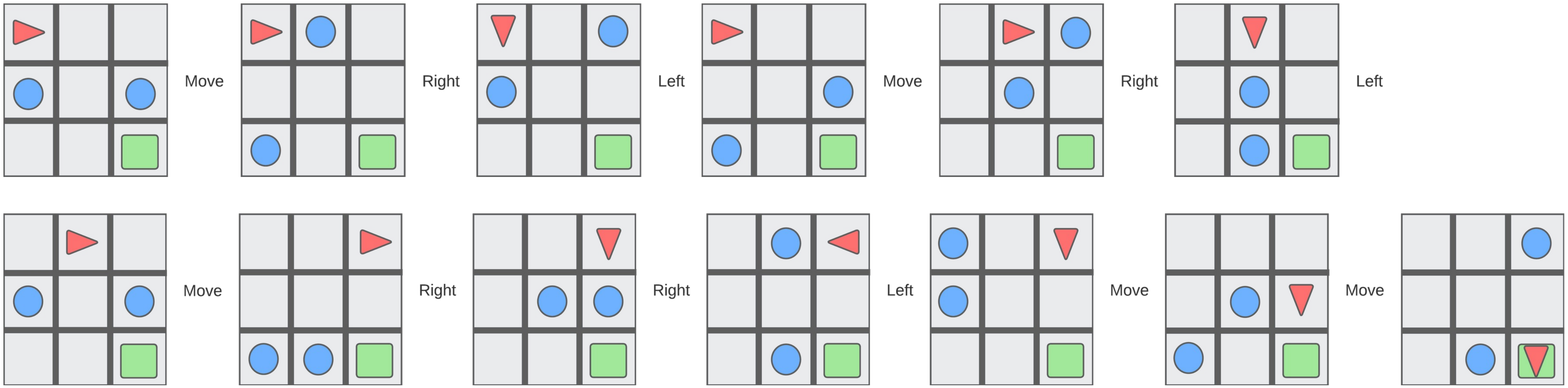
- $1 - 0.9 * t$ if success
- -1 if collision

Algorithm Deep Q Network (DQN)

Predicates *goal, near obstacles, position*

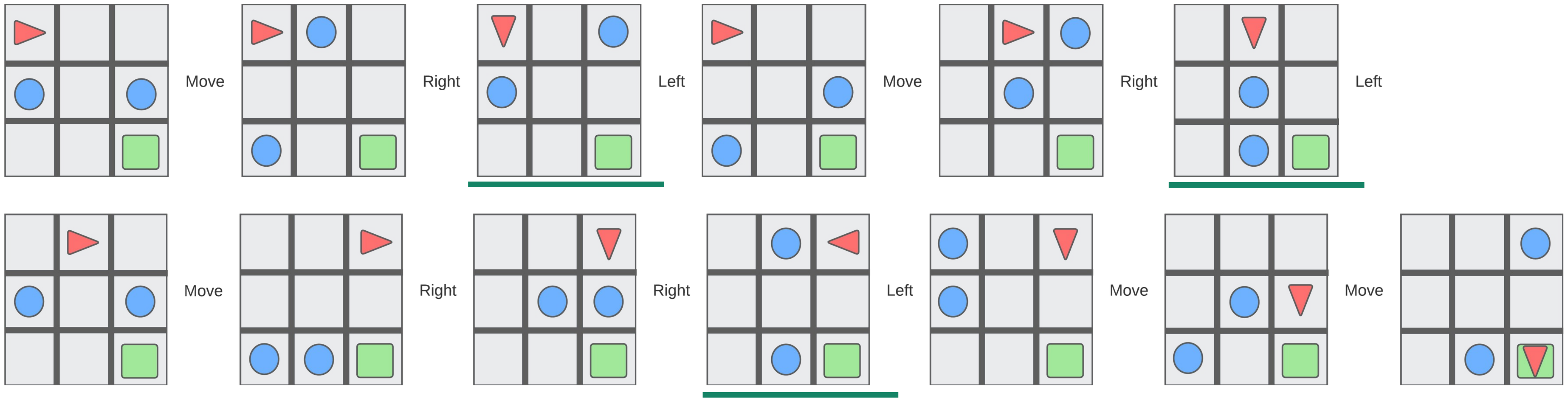
Dynamic Obstacles

History



Predicate: *goal*

Dynamic Obstacles

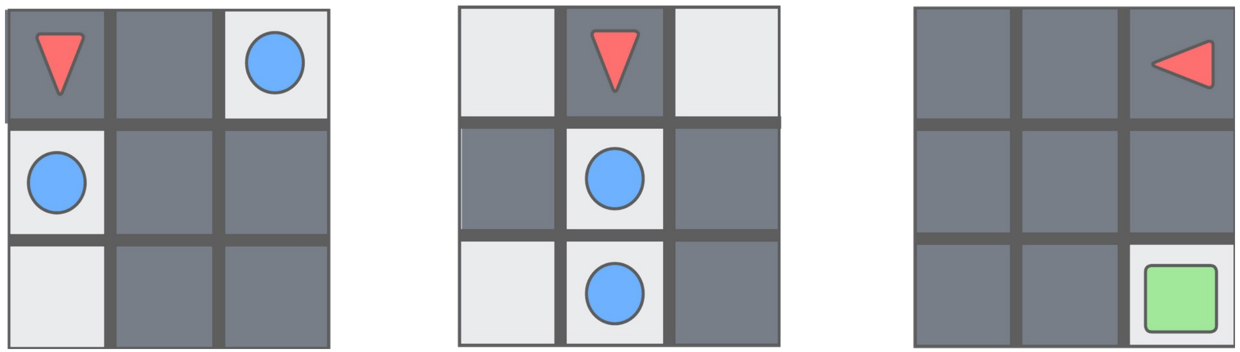


B-HXP (l = 4, δ = 0.9)

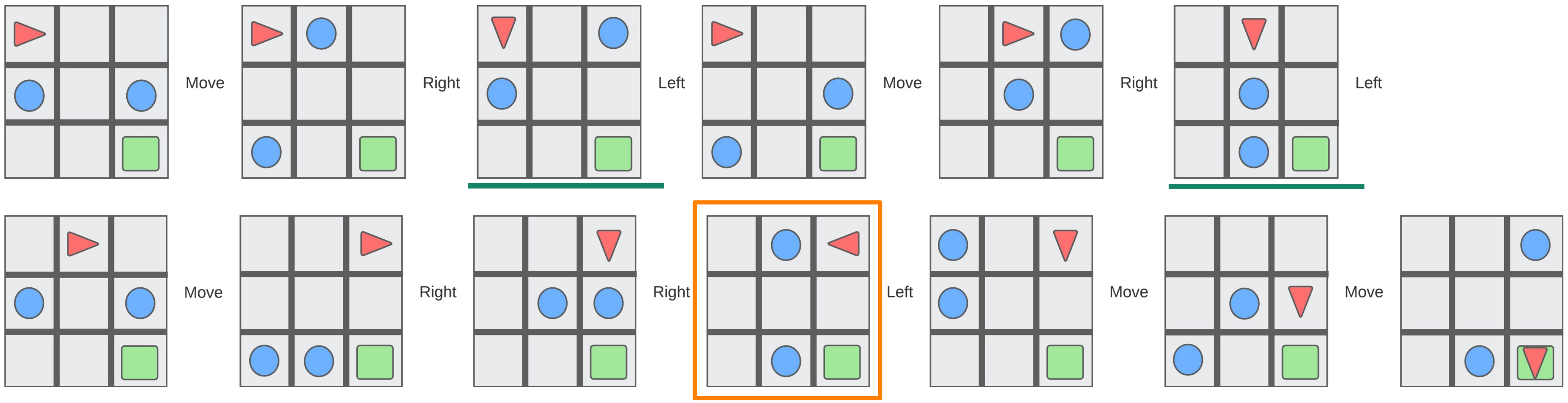
Scores: [-0.009, 0.006 |
0.006, **0.012**, -0.004, 0.009 |
0.139, 0.095, -0.029, 0.079 |
0.273, **0.48**, 0.42, 0.0]

Runtime: 370s

Predicates:



Dynamic Obstacles

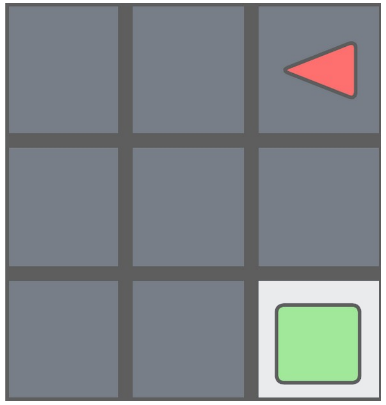


B-HXP (l = 4, δ = 0.9)

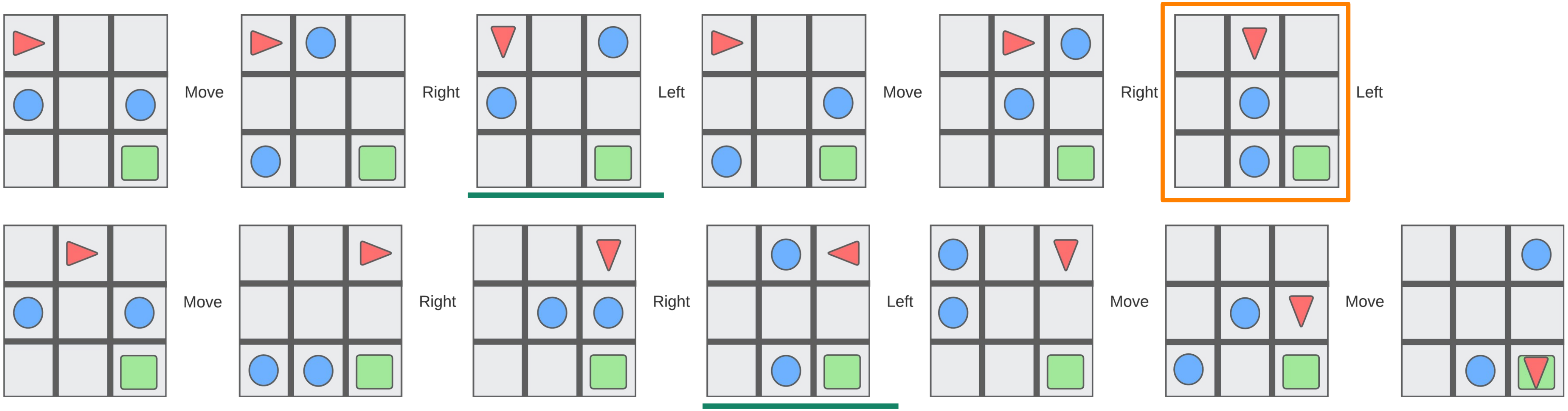
Scores: [-0.009, 0.006 |
0.006, **0.012**, -0.004, 0.009 |
0.139, 0.095, -0.029, 0.079 |
0.273, **0.48**, 0.42, 0.0]

Runtime: 370s

Predicates:



Dynamic Obstacles

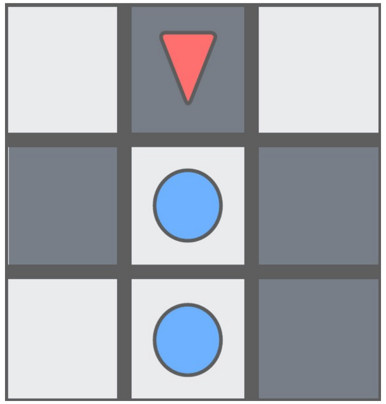


B-HXP ($l = 4$, $\delta = 0.9$)

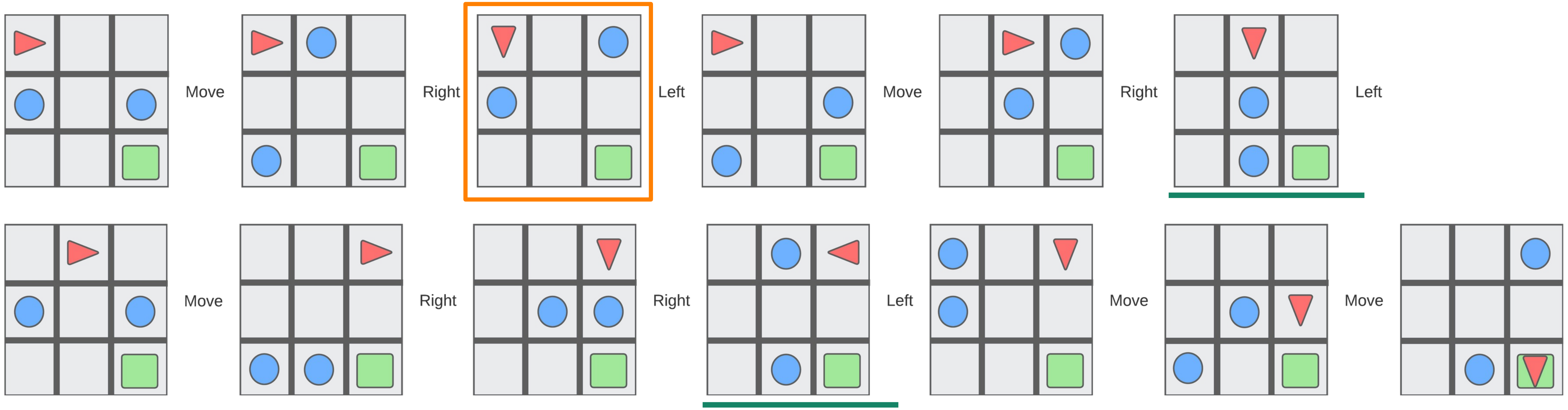
Scores: [-0.009, 0.006 |
0.006, **0.012**, -0.004, 0.009 |
0.139, 0.095, -0.029, 0.079 |
0.273, **0.48**, 0.42, 0.0]

Runtime: 370s

Predicates:



Dynamic Obstacles

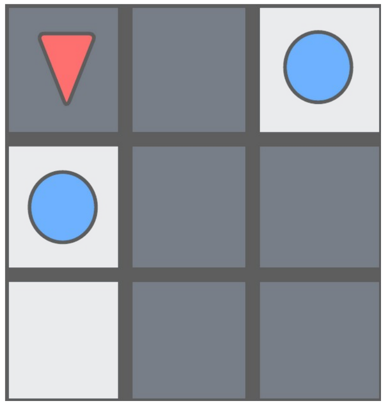


B-HXP (l = 4, δ = 0.9)

Scores: [-0.009, 0.006 |
0.006, **0.012**, -0.004, 0.009 |
0.139, 0.095, -0.029, 0.079 |
0.273, **0.48**, 0.42, 0.0]

Runtime: 370s

Predicates:



Connect4

Transition function Player 2's policy

Actions Column number

State Whole board

Reward function

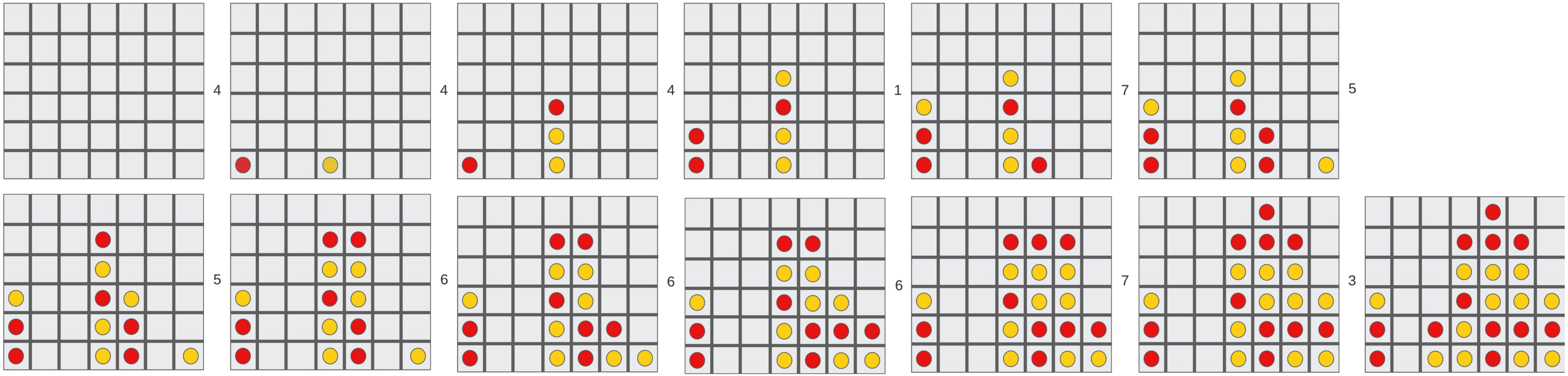
- +1 if win
- -1 if lose
- +0.5 if draw
- +0 otherwise

Algorithm Deep Q Network (DQN)

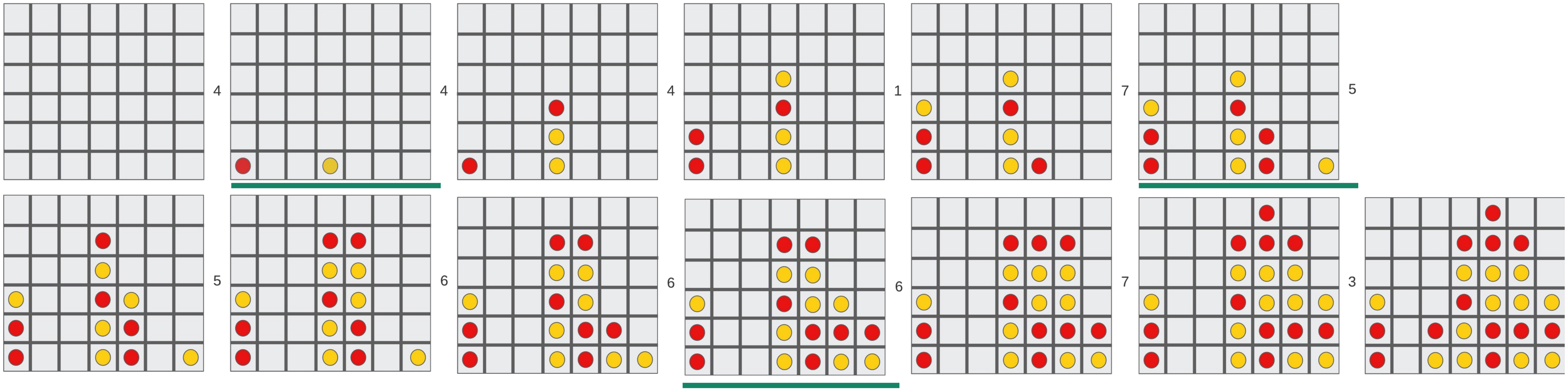
Predicates *win, lose, 3 in a row, avoid 3 in a row, control mid-column*

Connect4

History



Predicate: *win*

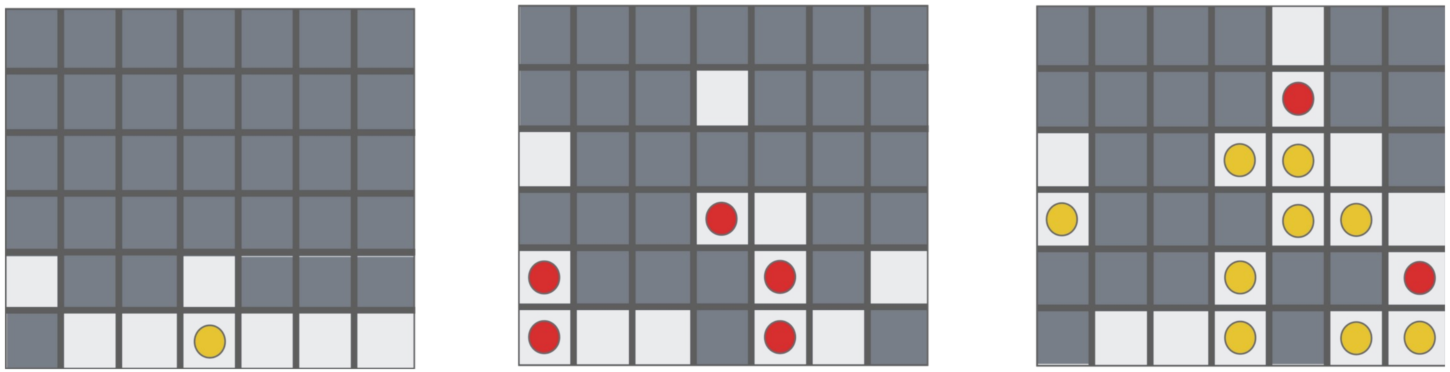


B-HXP (l = 4, δ = 0.9)

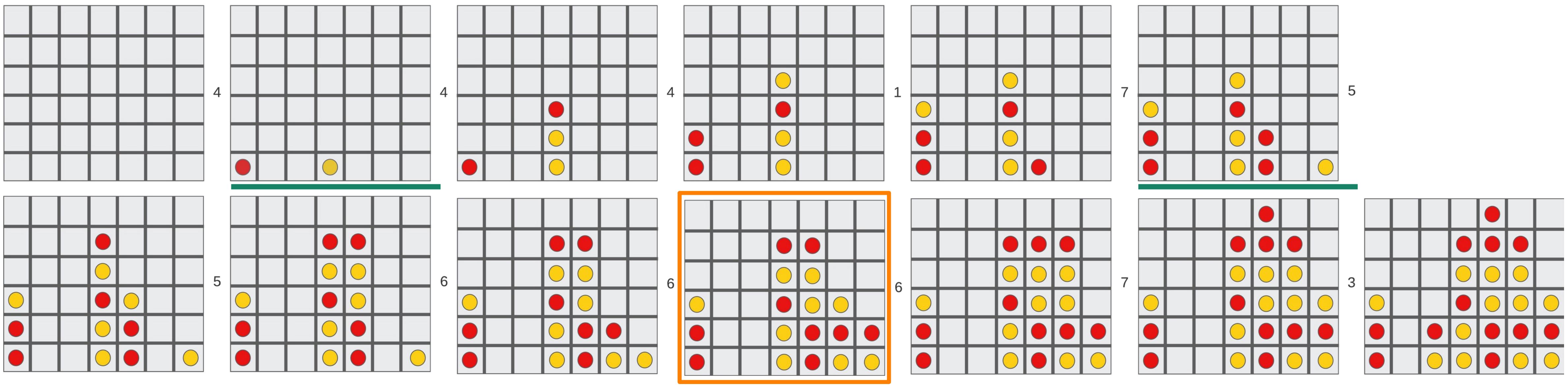
Scores: [0.0 |
0.0004, 0.0, 0.0, 0.0 |
0.0002, 0.0, 0.0, 0.0 |
0.3036, **0.367**, 0.092, 0.08]

Runtime: 112s

Predicates:



Connect4

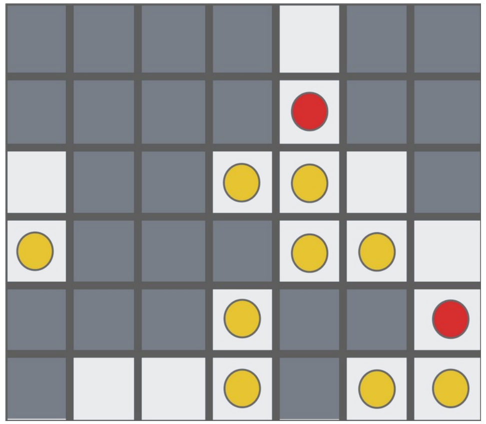


B-HXP (l = 4, δ = 0.9)

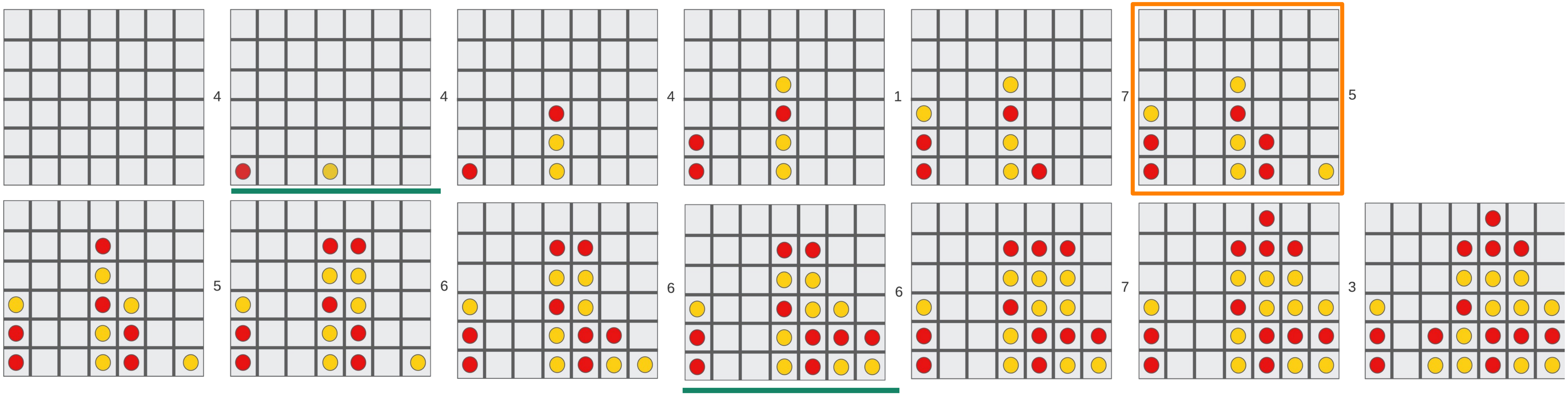
Scores: [0.0 |
0.0004, 0.0, 0.0, 0.0 |
0.0002, 0.0, 0.0, 0.0 |
0.3036, **0.367**, 0.092, 0.08]

Runtime: 112s

Predicates:



Connect4

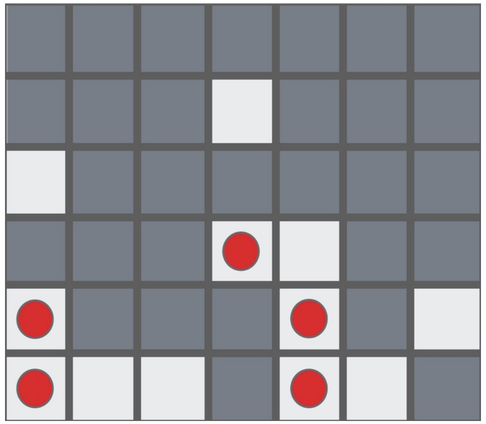


B-HXP (l = 4, δ = 0.9)

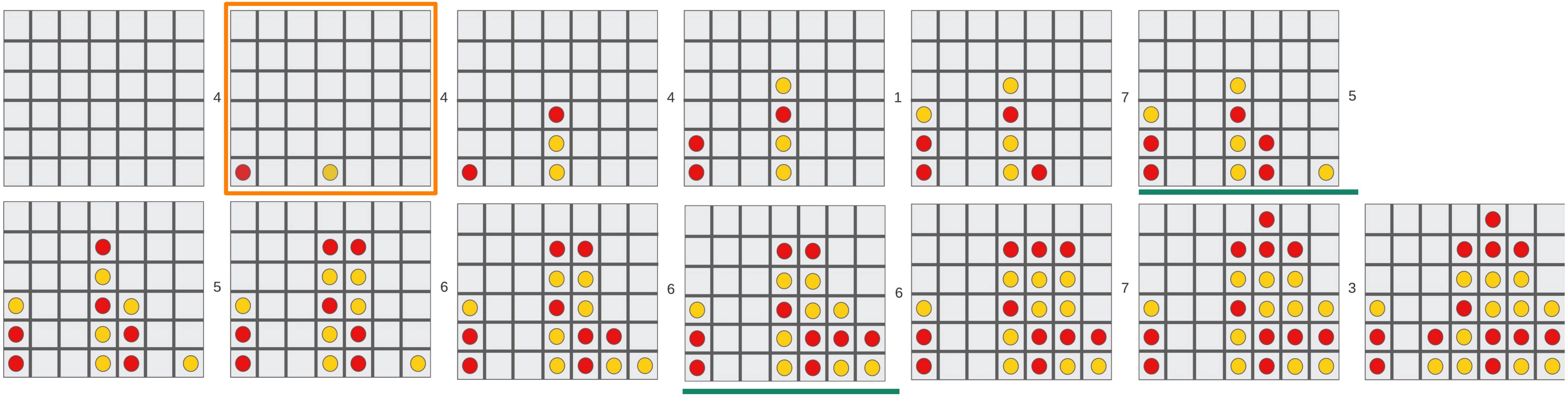
Scores: [0.0 |
0.0004, 0.0, 0.0, 0.0 |
0.0002, 0.0, 0.0, 0.0 |
0.3036, **0.367**, 0.092, 0.08]

Runtime: 112s

Predicates:



Connect4

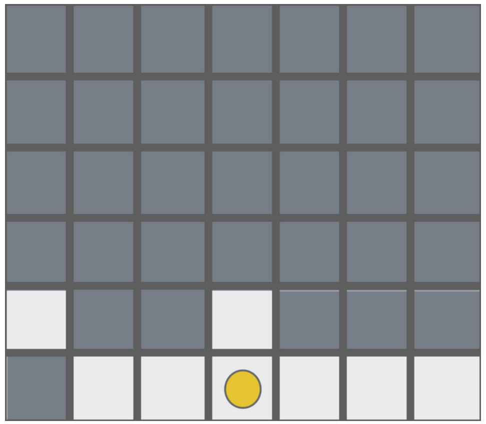


B-HXP (l = 4, δ = 0.9)

Scores: [0.0 |
0.0004, 0.0, 0.0, 0.0 |
0.0002, 0.0, 0.0, 0.0 |
0.3036, **0.367**, 0.092, 0.08]

Runtime: 112s

Predicates:



Conclusion

HXP:

- Analyse past agent’s interactions with the environment:
  - Predicate-based approach
  - Action importance evaluation
- Approximate HXP to reduce computation time

Conclusion

Backward HXP:

- Analyse past agent’s interactions with the environment:
  - Predicate-based approach
  - Action importance evaluation
- Plain HXP
- Approximate computation of PAXp
- Provide to the user important actions and predicates

Conclusion

Backward HXP:

- Analyse past agent’s interactions with the environment:
  - Predicate-based approach
  - Action importance evaluation
- Plain HXP
- Approximate computation of PAXp
- Provide to the user important actions and predicates

Limits:

- Transition function must be known
- Approximate PAXp
- <u>Complexity:</u> importance score and search space for PAXp computation

Future works:

- Feature ordering heuristics to produce insightful predicates
- Additional experiments