Explainable Gridworld Navigation Using ILP

Ayush Salunke
Ayushi Arora

Mentor: Youssef Mahmoud Youssef

Problem Statement

This project focuses on building an explainable Gridworld navigation agent.

- Transparent decision-making is essential in AI systems
- Many learning-based models operate as black boxes
- Gridworld navigation is a simple but representative planning problem

Objective:

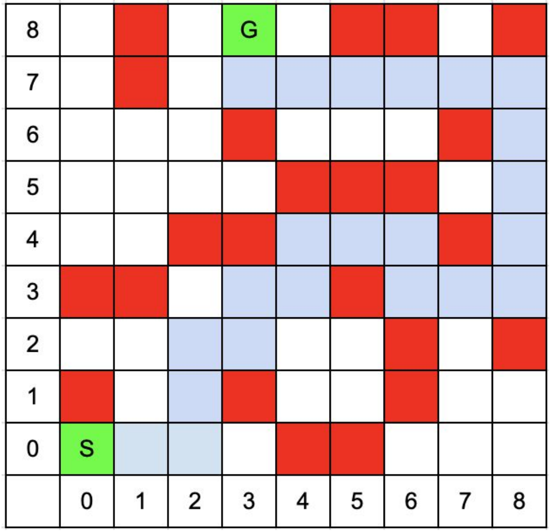
- Learn how an agent should choose actions
- Avoid manually defining navigation rules
- Ensure all decisions remain explainable

Popper for ILP

- Popper is an inductive logic programming system. Popper combines logical reasoning with machine learning to induce rules from examples and background knowledge. [1]
- Instead of manually defining navigation rules, the system employs the Popper ILP framework to induce a symbolic action-selection rule from examples and background knowledge.
- The learned rule is subsequently integrated into a Prolog-based planner to generate paths from a start position to a goal position while avoiding obstacles.
- Popper requires three input files:
    - an examples file
    - a background knowledge (BK) file
    - a bias file

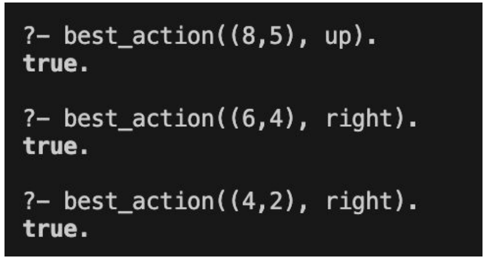
Gridworld environment

- 9 × 9 grid
- Start position: (0,0)
- Goal position: (3,8))
- Obstacles placed manually
- Allowed actions: up, down, left, right



Positive & Negative examples

- Positive examples define correct behavior
- Negative examples define forbidden actions
- An examples file contains positive and negative examples of the relation you want to learn
- For example:

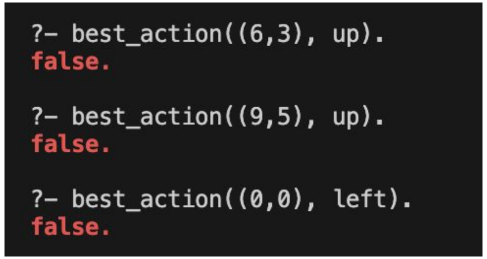


```
?- best_action((8,5), up).
true.

?- best_action((6,4), right).
true.

?- best_action((4,2), right).
true.
```

Positive Example



```
?- best_action((6,3), up).
false.

?- best_action((9,5), up).
false.

?- best_action((0,0), left).
false.
```

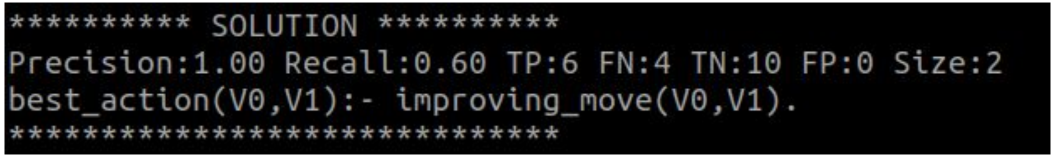
Negative Example

Learning Framework

- Learning performed using Inductive Logic Programming (ILP)
- Implemented with the Popper framework
- Learns symbolic rules instead of numeric parameters
- Target predicate:

  best_action(Position, Action)

- Output is a logical Prolog clause
  - Which gives a link between the best action to take and the actions available.



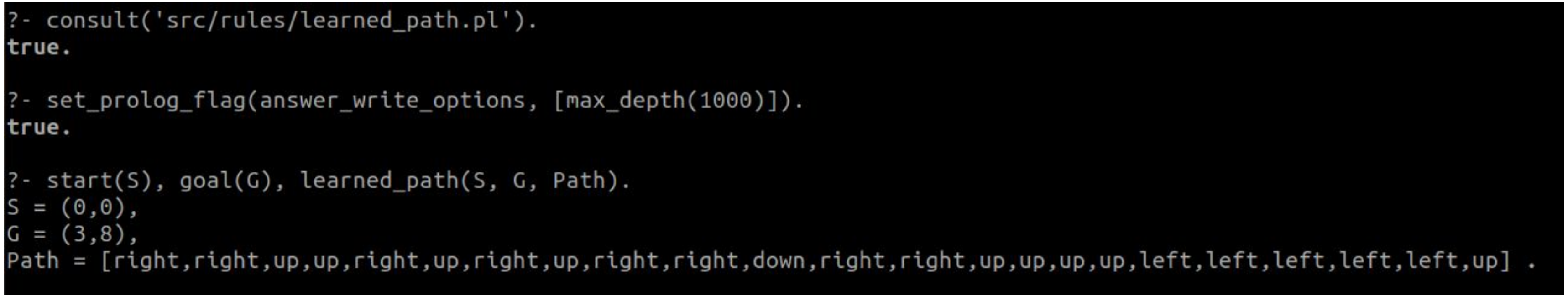
```
********** SOLUTION **********
Precision:1.00 Recall:0.60 TP:6 FN:4 TN:10 FP:0 Size:2
best_action(V0,V1):- improving_move(V0,V1).
******************************
```

Explainable Decision Making

- Decisions represented as logical rules
- Learned model is symbolic and human-readable
- No neural networks or numeric parameters
- Action selection based on distance reduction
- Each action can be logically justified
- Explanation example:

  “This action reduces the Manhattan distance to the goal.”

Path from Start to Goal



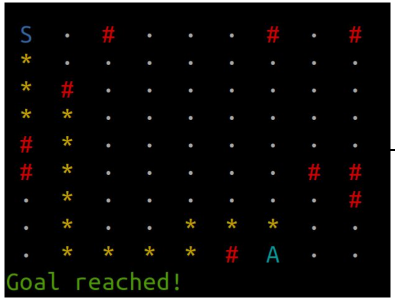
```
?- consult('src/rules/learned_path.pl').
true.

?- set_prolog_flag(answer_write_options, [max_depth(1000)]).
true.

?- start(S), goal(G), learned_path(S, G, Path).
S = (0,0),
G = (3,8),
Path = [right,right,up,up,right,up,right,up,right,right,down,right,right,up,up,up,up,left,left,left,left,left,up] .
```

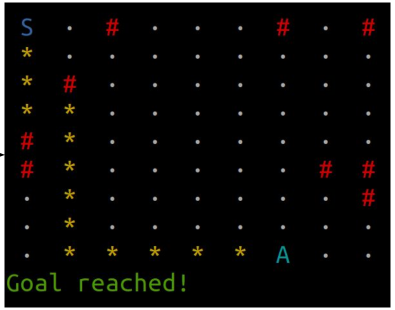
Counterfactual Reasoning

- This is a simple “What-if?” analysis.
- It helps enable us a causal reasoning like, “If a certain fact is changed in the world, how would it affect the output?”
- Supported counterfactuals for our program are:
  - What if we remove an obstacle
  - What if we add an obstacle
  - What if we change the goal
  - What if we change the start position



```
S . # . . . # . #
* . . . . . . . .
* # . . . . . . .
* * . . . . . . .
# * . . . . . . .
# * . . . . . # #
. * . . . . . . #
. * . . * * * . .
. * * * * # A . .
Goal reached!
```

What if we remove the obstacle at (6,9)



```
S . # . . . # . #
* . . . . . . . .
* # . . . . . . .
* * . . . . . . .
# * . . . . . . .
# * . . . . . # #
. * . . . . . . #
. * . . . . . . .
. * * * * * A . .
Goal reached!
```

*Note: for now we have just implemented this in the random gridworld program only for testing purposes

Limitations & Future Work

Limitations:-

- Currently there are no global planning algorithm.
- The current greedy heuristic (manhattan distance) is not an optimal algorithm.
- Popper learning does not work with a randomly generating gridworld.

Future work:-

- We can work on a different search algorithm like A*.
- We can allow diagonal movements for the agent.
- We can also implement a bigger gridsearch or provide a real time UI to track the agents movements.
- Integrate popper ilp with a random start, goal positions along with random obstacles.

Thank You!

Any Questions?