Introduction:
A reinforcement learning process is a process where agents try to come out with the best solutions by interacting with the environment, and the agent in reinforcement learning is *the leaner or decision maker that tries to find the algorithm to solve the task*.

Nowadays, reinforcement learning is widely applied in society. However, to trust RL agents, they need to be explainable, which means humans need to understand how these agents make decisions. If an RL system cannot be fully interpreted, then it becomes less useful because it is harder to judge if the decisions it makes are proper or not. So it is crucial for a RL agent to become explainable, so that its potential can be fully discovered.

Problem Statement and Goal:

There have always been a lot of approaches studied on how to generate interpretation of RL algorithms. This project aims on explaining RL models with causality. Research in cognitive science proposes that humans understand and represent the world through causes and effects. In the process of understanding the world, causal models are built in humans' minds to encode the causality of events, and these models are used to explain why new events happen and by reference to counterfactuals, why those things didn’t happen. By mimicking the way humans view the world, in which the world is treated as models where variables interact with each other according to causal relationships, causal models can also be built to generate explanations for RL agents.

To generate explainable RL agents by causality, current research introduces action influence models. Given actions that an agent takes, an action influence model can be defined as a set of equations which represent the causal relations between variables in the RL environment. With action influence models, explanations for ‘why?’ and ‘why not?’ questions can be generated. Although recent years interest in how to infer action influence models during RL agent training time is increasing, research on it is still significantly less studied. This project aims to fill this gap.

During the process of generating an action influence model, it is essential to provide a directed acyclic graph with its corresponding action matrix. A directed acyclic graph is a graph where each edge in the graph relates to an action taken by the agent and specifies the directed causal relation between variables. And its action matrix specifies the action each graph edge relates to. Currently, during the process of generating an action influence model, the directed acyclic graph with its action matrix has to be provided by hand. So our goal of this project is to generate a directed acyclic graph with its corresponding action matrix automatically by observations from the RL environment.

Proposed Solution

The main idea to solve this problem is to develop a new algorithm from the existing causal discovery algorithm - Greedy Equivalence Search Algorithm. Each edge in the final causal DAG will be generated based on the dataset selected by the action taken by the trained RL agent which has the best performance in the environment.

The proposed algorithm can be spilt into two parts. The first part is *Complete partial DAG* and Action Matrix Generator and the second part is Causal DAG Maker.

The CPDAG in this presentation can be simply considered as partial DAG, which means in the graph, some edges are directed and some are not. Due to time condition, I am not going to introduce in details about the CPDAG, but the formal definition can be found in my report.

Three RL environments where the input datasets are generated will be used. And the results generated will be discussed and evaluated separately.

Background

Reinforcement learning:

As mentioned before, a reinforcement learning process is a process where agents try to find a best solution by interacting with the environment. An Illustration can be seen on the right. In the graph, the
agent performs an action according to the state. Then, the environment responds to its action and pass a new state to let agent observe. Each action will change the total reward, which represents how good the action performed is and this total reward is aimed to be maximized by the agent.

Background - Notions
Action Influence Model:

An action influence model is necessary in generating explanations for a RL agent. Here I will give a graphic explanation of what is action influence model and formal definition can be found in my paper. The figure below is a graphical representation of one action influence model in game StarCraft. In the figure, each edge implies there is a causal effect which can be formally defined by some equation, from one variable to another. And we can see from the graph, each edge is associated with an unique action.

Background - Notions
DAG and AM:

A Directed Acyclic Graph is a graph without cycles and only having
directed edges. In this project, a casual DAG with N variables is represented by a N times N matrix. In the matrix, if element in row i and column j is not zero, then there is a directed edge from variable i to variable j. if element in row i and column j and element in row j and column i are both not zero, then there is an undirected edge from variable i to variable j. A working example is presented on the right.

An action matrix is in the same form of its associated causal matrix, where each non-zero element in the matrix represents the index of the action performed on that edge. A working example is also presented on the right.

Background - Notions
GES algorithm:

The input of ges algorithm is a dataset D with N variables along with an empty graph G that will have the same variables as in D, and output of this algorithm is a CPDAG C with a score assigned to it which indicates how good the CPDAG is at representing the actual casual graphic model.

A very brief example can be seen below. Here each column is a different variable. Each row is a set of observation data. And here is the generated CPDAG and the actual causal DAG.

The GES algorithm has three stages: forward, backward and turning, performing add edges remove edge and turn edge operations separately.

More details about this algorithm can be found in my report.

Background
Learning Explanations from Casual Graphic Model

Now I will introduce the process of how to generate ″why″ and ″why not″ explanations after the causal DAG and action matrix is given.

From the flow chart we can see, given a causal graphic DAG specifying causal direction between variables along with its action matrix, during the training phase of a RL agent, Action influence model can be learned. Then explanations of ″why A″ and ″ why not A″ questions can be produced given the AIM above.

The process of explanation generation can be treated as a ″black box″ as it is not the main focus of this project. We will only use the explanation generated to analyse if the causal graphic model and the action matrix we get by the designed method are accurate enough to produce meaningful explanation.

Methodology
CPDAG and Action Matrix Generator

The algorithm is based on GES algorithm. However, instead of running the algorithm directly on the input dataset D, the input data is first grouped by agent’s actions. Then at each step where add edge, remove edge or turn edge operations is performed, rather than calculating the graph score based on the whole input dataset, for each edge, all the datasets grouped by different actions will be gone through one by one, used as the input data to calculate the graph score. The best score among all the scores gotten from datasets will be marked as the final score of applying the operation to that edge.

Methodology
Casual DAG Maker - first stage

About the Causal DAG maker, suppose we have M CPDAGs generated from the CPDAG and Action Matrix Generator. Each CPDAG has its score. The first stage is to add the M CPDAGs to a graph G one by one. For each directed edge with its action in each CPDAG, if the graph G does not contain this edge, we add this edge with its action into G, otherwise we add the score of this edge into the edge score in G. As to undirected edge, we treat it as two direct edges in both directions. After this stage, between two variables in G, there will be multiple edges, each one associated with one unique set of action, direction and score.

Methodology
Casual DAG Maker - second stage

The second stage is to get the final causal DAG by merging all the edges between two variables into one unique directed edge. There is one special parameter: the threshold parameter H in this algorithm. It is used to decide if the edge will be kept in the final DAG produced. If the score of the edge is smaller than H, then it will be removed from the graph. In the environments chosen, we set $H = 0.1$, which is a pretty small value as we want to remove all the impossible edges and keep as many edge as possible for further evaluation at the same time.

Environments and Implementation
Taxi Problem - Environment

The environment of the taxi problem contains a 5×5 grid world. Each square on the grid world is a location. Obstacles are set on the grid and the taxi cannot move across obstacles. There are four special locations, denoted by letter R, G, Y, and B. The taxi is considered as the agent. At the start of each episode, the taxi is initialised at a random square and the passenger is located randomly at one of the four special locations. The taxi's first task is driving to the passenger's location and picking up the passenger. Then it drives the passenger to the destination location, which is another one of the four special locations and the episode terminates.

Environments and Implementation
Taxi Problem - Implementation

First the agent in taxi problem is trained by Q-learning algorithm. Then the game is run for 100 episodes with the trained agent.
After 100 episodes, a data set is gotten with 4 columns: taxi location, passenger location, destination location, action for each. Then the dataset is put into the CPDAG and Action Matrix Generator and the final DAG is made.
The final DAG produced will be a 3 × 3 matrix in the form below.

Environments and Implementation
CartPole Problem - Environment

The environment of the CartPole problem contains a pole attached to a cart moving along a track which is frictionless. In each episode, the pole is placed upright on the cart and the joint between them are unactuated. The force applied can be in left or right direction force. And the task is to keep the pole balanced as long as possible, by applying forces on the cart.

Environments and Implementation
CartPole Problem - Implementation

Here the agent is trained by reinforce algorithm. The dataset gotten has 5 columns: cart velocity, cart position, pole angle, pole angular velocity, action for each. And the final DAG produced will be a 4 × 4 matrix in the form below.

Environments and Implementation
LunarLand Problem - Environment

The environment of the LunarLander problem contains a view window and the lunar lander is considered as the agent. At each episode, the lander starts off the top center of the view window and needs to move towards the landing pad and land safely. This means the lander legs need to touch the moon land.

Environments and Implementation
LunarLand Problem - Implementation

Here the agent is trained by Deep Q-Network algorithm. The dataset gotten has 7 columns: lander xCoor, lander yCoor, lander xV, lander yV, lander angle, lander angularV, action for each. The final

DAG produced will be a 6 × 6 matrix in the form below.

Experimental Results
Taxi Problem - Casual DAG with AM

Here is the illustration of the generated causal DAG along its action matrix. From the diagram, we can see destination location affects both taxi position and passenger position by drop-off action which logically makes sense. Taxi position has effect on passenger position by performing ″move″ action. This is also meaningful but it is worth noticing that this causal relation is only true after taxi has picked up the passenger, however this condition is not presented in the causal graph generated.

Experimental Results
Taxi Problem - Explanations

The explanations produced by our causal DAG and action matrix above logically make sense, which indicates that the causal DAG and its action matrix generated are rather reliable. However, there is no explanation generated with ″drop off″ action, so we cannot validate that the causal DAG and action matrix generated are fully correct.

Experimental Results
CartPole Problem - Casual DAG with AM

Here is the result of cartPole problem. From the diagram, we can see while most of the causal relations are logically correct, problems still exist. One problem is that, actions ″push cart right″ and ″ push cart left″ should affect all variables in the environment equally. For example, if pole angular velocity is affected by pole angle with action ″push cart right″, it should also be affected by pole angle with action ″push cart left″, which is not shown in the diagram. The edge from pole angle to cart velocity doesn't make sense either because logically, from the description of the environment, it should always be cart that has effect on pole, as the aim of the task is to keep pole upright.

Experimental Results
CartPole Problem - Explanations

The explanations here produced logically make sense, which verifies the causal DAG and it action matrix generated. It is worth noticing that the cart position decides if the pole is balanced or not, so the goal: have better ″cart position″ is reasonable. The causal relation between cart velocity and cart position is not shown in the explanations above, so we still cannot fully verify the DAG and action matrix produced.

Experimental Results
LunarLand Problem - Casual DAG with AM

About LunarLand problem, we can see the diagram here is quite complex with a great number of edges. This graph is partially correct. The main problem of this causal graph is that it interprets correlation as causality. For example, there is correlation between x-coordinate and y-coordinate as while the value of y-coordinate changes, clearly the value of x-coordinate also changes. However, y-coordinate is not the factor that affects the value of x-coordinate, but the velocity of lander in x-axis indeed has effect on the value of x-coordinate.

Experimental Results
LunarLand Problem - Explanations
why perform ″do nothing″ action:

So here are some explanations generated for why perform ″do nothing″ action.

Experimental Results
LunarLand Problem - Explanations
why perform “fire main engine” action:

And Here are some explanations generated for why perform ″fire main engine″ action.

We can tell that some explanations produced in previous two slides are not correct due to the incorrectness of the causal DAG and action matrix generated. For example, because we interpret correlation between x-coordinate and y-coordinate to causal relation, the third explanation of why perform ″do nothing″ action and the second explanation of why perform ″fire main engine″ action do not make any sense logically.

Experimental Results
LunarLand Problem - Explanations
why perform “do nothing” action but not action “fire main engine”:

Here are some explanations generated for why perform ″do nothing″ action but not action “fire main engine”.

Experimental Results
LunarLand Problem - Explanations
why perform “fire main engine” action but not action “do nothing”:

And Here are some explanations generated for why perform “fire main engine” action but not action “do nothing”.

As to ″why not″ explanations generated, most of them are reasonable, which to some extent prove the causal graph along with the action matrix we produced is correct. But still, some of them are logically wrong. For example, in the first explanation of why not perform ″fire main engine ″ action, having more y-coordinate of the lander cannot increase its velocity in a-axis intuitively. Also in the second explanation of why not perform ″do nothing″ action, there is no causality between the lander’s x and y coordinates.

Evaluation
Achievements

So from the experimental results above, it is clear that for a RL agent, by the method designed, the causal DAG along with its action matrix can be generated successfully and meaningful explanations then can be produced as well.

Evaluation
Future Work

However, there are still some problems worth discussing:
First We chose environments with different state spaces. By comparing the results from different environments, we can see the DAG generated of the environment with fewer variables could be more reliable. For example, the casual graph generated in taxi problem is more logically correct than the other two.

Also Our designed algorithm could treat correlation between two variables as causal relation. So further work needs to be studied to explore some way solving this problem.

Finally Although for each environment, explanations are generated and we have evaluated the causal DAG along with its action matrix from logical aspects, a computational evaluation is still needed, so our algorithm can be validated in a more quantitative way.