Introduction to Artificial Intelligence
CSEN 901
Project 2 : Endgame(Logic Agent)

GenGrid Implementation :

GenGrid method in the Main.Java class was implemented on the idea of splitting the given string on the (;) which gives us 4 different strings with the first being the Height and Width of the grid separated by (,) so we split the string into two strings over the comma. Second String is the iron man position string which is split over (,) into iron man x position and iron man y position. The third string represents thanos position which is split on (,) into x and y positions and fourth string is the positions of the 4 stones which is split over commas into 8 different x and y components for the 4 stones.

After parsing all the values into variables, we write the values in the form of facts line by line. First line we write grid(Height, Width) fact then we add the four facts of the form posS(stone_N, X, Y) denoting the X and Y position of the stone N, then we add posTH(X, Y) fact denoting X and Y positions of thanos, finally we add the fact posIM(X, Y, S0) which denotes the X and Y positions of iron man in the initial situation S0.

Syntax and Semantics of Predicates :

- **stone_collected(X, Y, Stone, result(A,S)):-**

This predicate is used to check if the stone "Stone" in the situation "result(A,S)" where Iron Man is at position "X,Y" is collected or not. This predicate parses a given situation to check if this sequence of actions results in stone "Stone" being collected or not.

```
stone_collected(Stone, X , Y , result(A, S)):-
% Here we check if the last action was one of the movement operators, and then
we update the X,Y accordingly and call the predicate on the rest of the
situation (Without the last action) with the new X,Y.
      (A = left , Y_old #= Y + 1 , stone_collected(Stone, X , Y_old, S) );
      (A = right , Y_old #= Y - 1 , stone_collected(Stone, X , Y_old, S) );
      (A = down , X_old #= X - 1 , stone_collected(Stone, X_old , Y, S) );
      (A = up , X_old #= X + 1 , stone_collected(Stone, X_old , Y, S) );
```

```
% Here we just check if the last action was "snap", if it is we just pass the
rest of the situation to the predicate again with the same X and Y.
      (A = snap, stone_collected(Stone, X, Y, S));

% Here we check if the last action was "collect" and the current position of
Iron Man (X,Y) is the same as the mentioned stone "Stone" , the predicate
returns true.
      (A = collect, posS(X, Y, Stone));

% IF not the predicate will remove the last action and continue checking.
      (A = collect, not(posS(X, Y, Stone)) , stone_collected(Stone, X, Y, S)).
```

Successor state axioms implementation :

We have one main successor state axiom which is posIM(X,Y,result(A,S)) which makes sure that the current X and Y position of iron man holds when applying action A on the previous situation S. The axiom was divided into two parts, effect part where the situation S has not iron man in the same position but an action A caused it to be in X and Y new positions and persistence part where iron man was in the same X and Y positions and after applying the action A, it remains in the same position.

Effect part has the movement action to move iron man into its current position. First action is left where we make sure that Y_new is Y_old - 1 to assure moving one cell to the left. Second action is right where we make sure that Y_new is Y_old + 1 to assure moving one cell to the right. Third action is up where we make sure that X_new is X_old - 1 to assure moving one cell up. Fourth and final action in the effect part is down action where we make sure X_new is X_old + 1 to assure moving one cell down.

Persistence part where we make sure that if iron man was in the same intended position it can only do one action under one condition which is collect action under the condition that there is a stone Stone in the same position which is checked by posS(X_old, Y_old, Stone) and to assure not collecting the stone more than once we use (not(stone_collected(Stone,X_old,Y_old,S)).

Snapped(S) Predicate description :

In the first condition of the predicate it assures S is the result(snap, S1) where S1 satisfies the following conditions :- position of iron man in S1 is the same as the position of thanos where the position of iron man is obtained using successor state axiom posIM described above and the stone_collected predicate which is applied on the stones on the grid to assure that they are collected in the situation S1.

Two Running Examples :

Grid string : 5,5;2,2;5,5;2,3,3,4,4,4,4,5.

Facts Generated :

grid(5, 5).
posS(2, 3, stone_1).
posS(3, 4, stone_2).
posS(4, 4, stone_3).
posS(4, 5, stone_4).
posTH(5, 5).
posIM(2, 2, s0).

Example Query :

?-snapped(S).

S = result(snap, result(down, result(collect, result(right, result(collect, result(down, result(collect, result(down, result(right, result(collect, result(right, s0)))))))))).

Grid string : 5,5;1,2;3,4;1,1,2,1,2,2,3,3.

Facts Generated :

grid(5, 5).
posS(1, 1, stone_1).
posS(2, 1, stone_2).
posS(2, 2, stone_3).
posS(3, 3, stone_4).
posTH(3, 4).

posIM(1, 2, s0).

Example Query :

?-snapped(S).

S = result(snap, result(right, result(collect, result(down, result(right, result(collect, result(right, result(collect, result(down, result(collect, result(left, s0)))))))))).