Project 2 Report

Ahmed Mamdouh Khalifa | 40-5216

Semantics Of Predicates

grid_dimensions(Height, Width)

- Height is a number;
- Width is a number

The predicate succeeds if Height and Width are, respectively, the maximum Height position and Width position known on the grid.

---

on_grid(Y, X)

- Y is a number;
- X is a number.

The predicate succeeds if Y, X are within the bounds of the grid that we get from "grid_dimensions".

---

arange(Min, Max, Range)

- Min is the lower range;
- Max is the upper range;
- Range is the list of numbers [Min, Max[.

The predicate is used to create a list of numbers between Min and Max.

---

in(X, L)

- X an element;
- L a list of elements.

The predicate succeeds if the list L contains the element X.

---

agent_at(Y, X)

- Y is a number;
- X is a number.

The predicate succeeds if there is an agent at Y, X, by getting the list of all agents from "members_loc" and using "in".

---

fluent(Y, X, L, C, S)

- Y is the y axis location of Ethan;
- X is the x axis location of Ethan;
- S is the current state;
- C is the carrying capacity at state S;
- L is the list of agents that have been carried at or prior to state S.

The predicate succeeds if S is a valid state, by applying state transitions on the given state.

---

solve(S)

- S is a state.

The predicate succeeds if there is a state S, where

- Ethan's location is the same as the submarine;
- The current capacity is the maximum capacity;
- The list of carried agents is a permutation of the list of all agents,

by calling "fluent(SubY, subX, C, L, S)" where SubX and SubY are the X and Y positions of the submarine respectively, C is the maximum carry capacity, L is the list of all members.

---

ids(S, C)

- S is a state;

- C is the current maximum depth.

The predicate succeeds if there is a goal state S at depth C by calling "call_with_depth_limit" with "solve(S)" and C as its parameters, then calling itself with a new depth limit C+1 if solve fails.

---

goal(S)

- S is a state.

The predicate succeeds if there is a goal state S by calling "ids(S, 0)".

Successor State Axioms

fluent(Y, X, [], C, s0)

This is the initial state of the fluent. It dictates that the location is the same as the starting location from the knowledge base, the carrying capacity is the maximum capacity C and the list of carried members is empty.

---

fluent(Y, X, L, C, result(left, State))

This is the transition axiom for the left action. it calculates the old X position "XOld", makes sure that the position is on the grid using "on_grid", then calls "fluent(Y, XOld, L, C, State)".

---

fluent(Y, X, L, C, result(right, State))

This is the transition axiom for the right action. it calculates the old X position "XOld", makes sure that the position is on the grid using "on_grid", then calls "fluent(Y, XOld, L, C, State)".

---

fluent(Y, X, L, C, result(up, State))

This is the transition axiom for the up action. it calculates the old Y position "YOld", makes sure that the position is on the grid using "on_grid", then calls "fluent(YOld, X, L, C, State)".

---

fluent(Y, X, L, C, result(down, State))

This is the transition axiom for the down action. it calculates the old Y position "YOld", makes sure that the position is on the grid using "on_grid", then calls "fluent(YOld, X, L, C, State)".

---

fluent(Y, X, [[Y, X] | L], C, result(carry, State))

This is the transition axiom for the carry action. It checks if there is an agent at the position Y, X using "agent_at", calculates the previous capacity, checks if the previous capacity "COld" is not bigger than the maximum capacity and the current capacity is not smaller than 0, then calls "fluent(y, X, L, COld, State)".

---

fluent(Y, X, L, Cap, result(drop, State))

This is the transition axiom for the drop action. It checks if the current location is the same as the submarine, calculates the possible values for the old capacity using "arange" then calls "fluent(Y, X, L, COld, State)".

Examples

With members_loc([[1,1],[1,2]])

?- goal(S).

S = result(drop, result(up, result(carry, result(down, result(drop, result(right, result(up, result(carry, result(right, result(down, s0)))))))))) ;
S = result(drop, result(up, result(carry, result(down, result(drop, result(right, result(up, result(carry, result(down, result(right, s0)))))))))) ;
S = result(drop, result(up, result(carry, result(down, result(drop, result(up, result(right, result(carry, result(right, result(down, s0)))))))))) ;
S = result(drop, result(up, result(carry, result(down, result(drop, result(up, result(right, result(carry, result(down, result(right, s0)))))))))) ;

With members_loc([[2,2],[1,2]])

?- goal(S).

S = result(drop, result(up, result(up, result(carry, result(down, result(down, result(drop, result(up, result(carry, result(right, result(right, result(down, s0))))))))))) ;
S = result(drop, result(up, result(up, result(carry, result(down, result(down, result(drop, result(up, result(carry, result(right, result(down, result(right, s0))))))))))) ;
S = result(drop, result(up, result(up, result(carry, result(down, result(down, result(drop, result(up, result(carry, result(down, result(right, result(right, s0))))))))))))