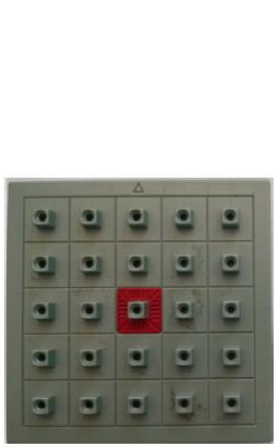
Lab 4

Lunar Lockout Game

Game Pieces

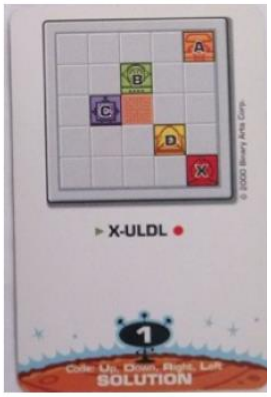
The lunar lockout game has several pieces. First, we have a 5x5 game board with a red square marked in the middle. Next, we have 5 helper spacecraft in various colours and then 1 red spacecraft. We are also given cards that specify initial setup position for some subset of spacecraft. On the back of each card is a solution.


(a) 5x5 Board


(b) Helpers


(c) Red

(d) Puzzle 1



(e) Solution 1

Game Description

The goal of this game it to move the red spacecraft to the centre red square. One can move any spacecraft but they are limited to moving up-down or left-right. Whenever a spacecraft moves, it continues moving until it hits another spacecraft. Note that it is not legal for the spacecraft to move on the board. Therefore, all spacecraft need to cooperate together to stay on the board and help the red spacecraft reach the goal.

Game Play

We present two simple beginner puzzles to show typical game play. Puzzle 1 is the beginner puzzle shown above. Starting from the initial setup, the red spacecraft moves up, left, down, then left to finally end up at the goal position. In this simple puzzle, only the red robot had to move. In the more complex puzzles, we also require other helper spacecraft to move. Puzzle 2 is an example requiring another helper robot to move as well. In this assignment, you will be defining a representation for this puzzle and also implementing a planner to solve them.


(a) Init (b) Move up (c) Move Left (d) Move Down (e) Move Left

Figure 2: Puzzle 1. The solution only requires moving the red spacecraft.

(a) Init (b) P - Move Down (c) P - Move Right (d) R - Move Right (e) R - Move Down

Figure 3: Puzzle 2. The solution requires first moving the purple spacecraft and then the red Spacecraft.

The lunar lockout world (objects, states, actions)

Objects

In the **lunar lockout world** game, we have spacecrafts located in certain positions of the board, and a position can be always defined by a pair (x, y), where the x represents the columns and y represents the row. Therefore, we can represent the world using two types of objects: spacecrafts and coordinates, but since our board is square we can use one type of object called position. If we have a board of 5x5 and 5 spacecrafts the objects will be:

- Spacecrafts: red yellow orange purple green
- Positions: n1 n2 n3 n4 n5

States

Our state will be the position of a spacecraft on the board. This will be represented by a spacecraft and two positions: the first will be the x-coordinate and the second will be the y-coordinate.

Actions

The set of actions will be to move a spacecraft UP, DOWN, LEFT and RIGHT. The parameters of the actions will be the spacecraft to be moved, the spacecraft that will stop the moving spacecraft, and the positions of both. The preconditions to perform the actions will be:

- The position of the stop craft and the position of the moving craft are on the same column or row and the stop craft will ”stop” the moving craft (depending on the type of action: moving up/down or right/left).
- There is no other craft in the middle of the moving craft and the stop craft. To check this precondition we can use different strategies defined in the Domain axioms.

The number of possible states for the suggested representations that you came up with

The final number of possible states follows the formula of the permutations without repetitions, therefore:

$N\_States = (positions\ in\ the\ board)!/(positions\ in\ the\ board - number\ of\ spacecrafts)!$,

In our representation, we have 5 spacecrafts and 25 positions in the board:

$N\_oStates = 25!/(25 - 5)! = 6375600$

The PDDL files required for a planner of your choice to solve this problem (Hint: a state-space search is likely the most simple).

Domain PDDL file:

```
(define (domain lunar)
   (:requirements :strips :typing)
   (:types spacecraft - object
         number - object)

   (:predicates
      (less ?n1 - number ?n2 - number)
      (one-less ?n1 - number ?n2 - number)
      (pos ?s - spacecraft ?x - number ?y - number)
   )

   (:action move_up
      :parameters (?s1 ?s2 - spacecraft
                ?sx ?s1y ?s2y ?new_s1y - number)

      :precondition   (and
                     (pos ?s1 ?sx ?s1y)
                     (pos ?s2 ?sx ?s2y)
                     (one-less ?new_s1y ?s2y)
                     (less ?s1y ?s2y)
                     (forall (?s - spacecraft ?y - number)
                         (not (and (pos ?s ?sx ?y) (less ?y ?s2y) (less ?s1y ?y)))
                     )
                  )

      :effect (and
               (not (pos ?s1 ?sx ?s1y))
               (pos ?s1 ?sx ?new_s1y)
            )
   )

   (:action move_down
      :parameters (?s1 ?s2 - spacecraft
                 ?sx ?s1y ?s2y ?new_s1y - number)

      :precondition   (and
                     (pos ?s1 ?sx ?s1y)
                     (pos ?s2 ?sx ?s2y)
                     (one-less ?s2y ?new_s1y)
                     (less ?s2y ?s1y)
                     (forall (?s - spacecraft ?y - number)
                         (not (and (pos ?s ?sx ?y) (less ?y ?s1y) (less ?s2y ?y)))
                     )
                  )

      :effect (and
               (not (pos ?s1 ?sx ?s1y))
               (pos ?s1 ?sx ?new_s1y)
            )
   )

   (:action move_left
      :parameters (?s1 ?s2 - spacecraft
```

```
            ?sy ?s1x ?s2x ?new_s1x - number)

    :precondition   (and
                (pos ?s1 ?s1x ?sy)
                (pos ?s2 ?s2x ?sy)
                (one-less ?s2x ?new_s1x)
                (less ?s2x ?s1x)
                (forall (?s - spacecraft ?x - number)
                    (not (and (pos ?s ?x ?sy) (less ?s2x ?x) (less ?x ?s1x)))
                )
              )

    :effect (and
            (not (pos ?s1 ?s1x ?sy))
            (pos ?s1 ?new_s1x ?sy)
          )
  )

  (:action move_right
    :parameters (?s1 ?s2 - spacecraft
             ?sy ?s1x ?s2x ?new_s1x - number)

    :precondition   (and
                (pos ?s1 ?s1x ?sy)
                (pos ?s2 ?s2x ?sy)
                (one-less ?new_s1x ?s2x)
                (less ?s1x ?s2x)
                (forall (?s - spacecraft ?x - number)
                    (not (and (pos ?s ?x ?sy) (less ?x ?s2x) (less ?s1x ?x)))
                )
              )

    :effect (and
            (not (pos ?s1 ?s1x ?sy))
            (pos ?s1 ?new_s1x ?sy)
          )
  )

)
```

Problem PDDL file:

```
(define (problem easy_less_1)
   (:domain lunar)

   (:objects   red yellow orange purple green - spacecraft
            n1 n2 n3 n4 n5 - number)

   (:init
      (less n1 n2)
      (less n1 n3)
      (less n1 n4)
      (less n1 n5)
      (less n2 n3)
      (less n2 n4)
      (less n2 n5)
      (less n3 n4)
      (less n3 n5)
      (less n4 n5)

      (one-less n1 n2)
      (one-less n2 n3)
      (one-less n3 n4)
      (one-less n4 n5)

      (pos red n3 n1)
      (pos yellow n3 n4)
      (pos orange n2 n2)
      (pos purple n1 n1)
      (pos green n5 n5)
   )


   (:goal
      (and
         (pos red n3 n3)
      )
   )
)
```

Exercises

1. **Write a problem PDDL file for the puzzle 2 shown in figure 3?**

2. **Is there any difference in the size of the states of puzzle 1 and puzzle 2?**