# Lab 8

## Part 1

(define (domain lab8)

(:predicates (clear ?X) (on ?A ?B) (smaller ?A ?B))

(:action move

:parameters (?disc ?from ?to)

:precondition (and (smaller ?to ?disc) (on ?disc ?from) (clear ?disc) (clear ?to))

:effect (and (clear ?from) (on ?disc ?to) (not (on ?disc ?from)) (not (clear ?to))))

)

(define (problem lab8)

(:domain lab8)

(:objects t1 t2 t3 d1 d2 d3)

(:init

(smaller t1 d1) (smaller t1 d2) (smaller t1 d3)

(smaller t2 d1) (smaller t2 d2) (smaller t2 d3)

(smaller t3 d1) (smaller t3 d2) (smaller t3 d3)

(smaller d2 d1) (smaller d3 d1) (smaller d3 d2)

(clear t2) (clear t3) (clear d1)

(on d3 t1) (on d2 d3) (on d1 d2))

(:goal (and (on d3 t3) (on d2 d3) (on d1 d2)))

)

## Part 2

Move 1

(:action move  
 :parameters (d1 d2 t3)  
 :precondition  
 (and  
 (smaller t3 d1)  
 (on d1 d2)  
 (clear d1)  
 (clear t3)  
 )  
 :effect  
 (and  
 (clear d2)  
 (on d1 t3)  
 (not  
 (on d1 d2)  
 )  
 (not  
 (clear t3)  
 )  
 )  
 )

Move 2

(:action move  
 :parameters (d2 d3 t2)  
 :precondition  
 (and  
 (smaller t2 d2)  
 (on d2 d3)  
 (clear d2)  
 (clear t2)  
 )  
 :effect  
 (and  
 (clear d3)  
 (on d2 t2)  
 (not  
 (on d2 d3)  
 )  
 (not  
 (clear t2)  
 )  
 )  
 )

Move 3

(:action move  
 :parameters (d1 t3 d2)  
 :precondition  
 (and  
 (smaller d2 d1)  
 (on d1 t3)  
 (clear d1)  
 (clear d2)  
 )  
 :effect  
 (and  
 (clear t3)  
 (on d1 d2)  
 (not  
 (on d1 t3)  
 )  
 (not  
 (clear d2)  
 )  
 )  
 )

Move 4

(:action move  
 :parameters (d3 t1 t3)  
 :precondition  
 (and  
 (smaller t3 d3)  
 (on d3 t1)  
 (clear d3)  
 (clear t3)  
 )  
 :effect  
 (and  
 (clear t1)  
 (on d3 t3)  
 (not  
 (on d3 t1)  
 )  
 (not  
 (clear t3)  
 )  
 )  
 )

Move 5

(:action move  
 :parameters (d1 d2 t1)  
 :precondition  
 (and  
 (smaller t1 d1)  
 (on d1 d2)  
 (clear d1)  
 (clear t1)  
 )  
 :effect  
 (and  
 (clear d2)  
 (on d1 t1)  
 (not  
 (on d1 d2)  
 )  
 (not  
 (clear t1)  
 )  
 )  
 )

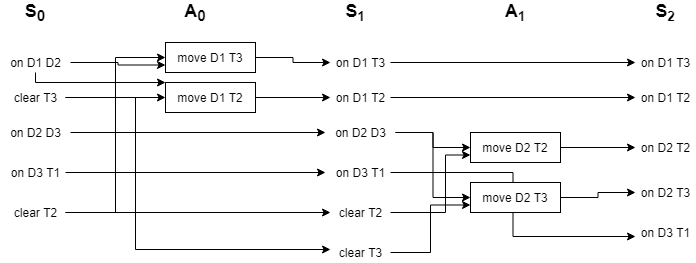
Move 6

(:action move  
 :parameters (d2 t2 d3)  
 :precondition  
 (and  
 (smaller d3 d2)  
 (on d2 t2)  
 (clear d2)  
 (clear d3)  
 )  
 :effect  
 (and  
 (clear t2)  
 (on d2 d3)  
 (not  
 (on d2 t2)  
 )  
 (not  
 (clear d3)  
 )  
 )  
 )

Move 7

(:action move  
 :parameters (d1 t1 d2)  
 :precondition  
 (and  
 (smaller d2 d1)  
 (on d1 t1)  
 (clear d1)  
 (clear d2)  
 )  
 :effect  
 (and  
 (clear t1)  
 (on d1 d2)  
 (not  
 (on d1 t1)  
 )  
 (not  
 (clear d2)  
 )  
 )  
 )

## Part 3



## Part 4

On potential heuristic that could improve the performance of this solution (assuming we are not moving the same piece twice in a row already) is to prefer empty pegs. In the ideal solution, you only ever stack pieces if there are no open pegs or if it’s the last move required for the goal state. By simply preferring a move to an open peg when available, you can avoid unnecessary stacking. Furthermore, if you have two open pegs, always take the furthest one available. That would eliminate the choice in the first action stage, and the previous heuristic would eliminate the choice in the second action state, reducing our potential plan to a direct path through the initial stages.