9/14/2021

# Assignment 3: Planning

## The instructions—about this Word file template, the evaluation matrix, no more than 4 pages, and appendices, are as before.

## The Example Problem

We will use [this](http://editor.planning.domains/) PDDL system to edit plans. Warren Mansur has implemented a plan for the 4-room problem shown.

Calendar

Description automatically generated

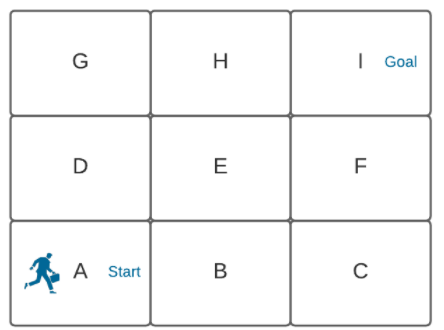
The domain is in Appendix 1. The problem is in Appendix 2. Run this and show the resulting plan via screenshot(s).

Graphical user interface, text, application

Description automatically generated.

## A More Complex Problem

Starting from scratch or using Warren’s code, increase the size of the grid to 3x3 and use pddl to solve the problem below.



**(2.1)** List the PDDL domain.

I saw no need to change Warren’s code from Appendix 1:

(define (domain robot-rooms)

(:predicates

(connected-at-left ?from-room ?to-room)

(connected-at-right ?from-room ?to-room)

(connected-at-top ?from-room ?to-room)

(connected-at-bottom ?from-room ?to-room)

(is-robot ?r)

(in-room ?robot ?room))

(:action move-left

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-left ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room))))

(:action move-right

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-right ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room))))

(:action move-down

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-bottom ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room))))

(:action move-up

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-top ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room)))))

**(2.2)** List the PDDL problem.

(define (problem robot-four-rooms)

(:domain robot-rooms)

(:objects robot room-a room-b room-c room-d room-e room-f room-g room-h room-i)

(:init (is-robot robot)

(connected-at-right room-a room-b)

(connected-at-top room-a room-d)

(connected-at-right room-b room-c)

(connected-at-left room-b room-a)

(connected-at-top room-b room-e)

(connected-at-left room-c room-b)

(connected-at-top room-c room-f)

(connected-at-bottom room-d room-a)

(connected-at-right room-d room-e)

(connected-at-top room-d room-g)

(connected-at-bottom room-e room-b)

(connected-at-right room-e room-f)

(connected-at-top room-e room-h)

(connected-at-left room-e room-d)

(connected-at-left room-f room-e)

(connected-at-top room-f room-i)

(connected-at-bottom room-f room-c)

(connected-at-right room-g room-h)

(connected-at-bottom room-g room-d)

(connected-at-right room-h room-i)

(connected-at-bottom room-h room-e)

(connected-at-left room-h room-g)

(connected-at-left room-i room-h)

(connected-at-bottom room-i room-f)

(in-room robot room-a))

(:goal (in-room robot room-i)))

**(2.3)** Run this and show the resulting plan via screenshot(s).

Graphical user interface, text, application

Description automatically generated.

## A Business Problem

This part of the exercise is more open-ended. Starting from scratch or using Warren’s code, create a suitable business problem in PDDL, and show the solution as below. The input to the problem can be hard-coded. In developing your response, consider starting with a simple problem such as developing a plan to transport goods from point A to point B in a grid with an obstacle, then make it increasingly sophisticated to get the version you submit. Try to be guided by envisioning useful functionality.

**(3.1)** 1-or-2-sentence description of the business problem:

Roomba cleaning robots need to map out rooms and know which areas cannot be access (ie contain obstacles). Therefore, a robot must be able to achieve the optimal path when a perfect grid does not exist.

**(3.2)** PDDL code:

I kept the domain the same, but the problem is as follows (I removed rooms d and f):

(define (problem robot-four-rooms)

(:domain robot-rooms)

(:objects robot room-a room-b room-c room-e room-g room-h room-i)

(:init (is-robot robot)

(connected-at-right room-a room-b)

(connected-at-right room-b room-c)

(connected-at-left room-b room-a)

(connected-at-top room-b room-e)

(connected-at-left room-c room-b)

(connected-at-bottom room-e room-b)

(connected-at-top room-e room-h)

(connected-at-right room-g room-h)

(connected-at-right room-h room-i)

(connected-at-bottom room-h room-e)

(connected-at-left room-h room-g)

(connected-at-left room-i room-h)

(in-room robot room-a))

(:goal (in-room robot room-i)))

**(3.3)** Screenshot of output:

Here we see that the robot must travel through the middle, not along an edge.

Graphical user interface, text, application

Description automatically generated

# Evaluation



# Appendix 1: Domain for 4-room Configuration

(define (domain robot-rooms)

(:predicates

(connected-at-left ?from-room ?to-room)

(connected-at-right ?from-room ?to-room)

(connected-at-top ?from-room ?to-room)

(connected-at-bottom ?from-room ?to-room)

(is-robot ?r)

(in-room ?robot ?room))

(:action move-left

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-left ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room))))

(:action move-right

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-right ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room))))

(:action move-down

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-bottom ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room))))

(:action move-up

:parameters (?robot ?from-room ?to-room)

:precondition (and (is-robot ?robot)

(in-room ?robot ?from-room)

(connected-at-top ?from-room ?to-room))

:effect (and (in-room ?robot ?to-room)

(not (in-room ?robot ?from-room)))))

# Appendix 2: Problem for 4-Room Configuration

(define (problem robot-four-rooms)

(:domain robot-rooms)

(:objects robot room-a room-b room-c room-d)

(:init (is-robot robot)

(connected-at-right room-a room-b)

(connected-at-top room-a room-c)

(connected-at-left room-b room-a)

(connected-at-top room-b room-d)

(connected-at-right room-c room-d)

(connected-at-bottom room-c room-a)

(connected-at-left room-d room-c)

(connected-at-bottom room-d room-b)

(in-room robot room-a))

(:goal (in-room robot room-d)))