Artificial Intelligence

Planning Systems Labwork
Due: Today, 9pm

Prof. Felipe Meneguzzi*

October 18, 2016

1 HTN Aperture Labs Domain

In this lab, you will formalize the Aperture Robot domain from Section A into an HTN formulation, and write methods/recipes to decompose the problems we give you in Section B. Note that in HTN planners, the goals are tasks to be decomposed using the methods you define, so, if one of your goals was to have the robot at a particular location, you must define a task to move robby to the desired location.

For this assignment you will need to download the JSHOP2 planner, which is available in this course's Moodle. What you must do then is:

- Define tasks and methods to decompose this task (look for the `rover` domain example in JSHOP2);
- Specify at least one problem from the appendix as an HTN problem and run JSHOP2 to solve it;
- When you are finished, you must upload one zip file containing the domain specification as well as the problems you wrote.

When you encode your HTN domain, bear in mind that the PFD procedure from [1, Chapter 11] searches depth-first without any recollection of where the algorithm iterated, so you must add your own control rules to prevent the algorithm from entering an infinite loop. You can achieve this by creating control actions (denoted by the double exclamation mark "`!!`") to mark where you visited.

```
(:operator (!!visit ?l)
()
()
((visited ?l))
0
)

(:operator (!!unvisit ?l)
()
((visited ?l))
()
0
)
```

2 Initial Domain

The goal of this task is to create the methods based on this skeleton domain:

*Thanks to Maurício Magnaguagno for the idea of changing the Robby domain.

```
(defdomain aperture(

  ;------------------------------
  ; Operators
  ;------------------------------

  (:operator (!move ?l1 ?l2)
    (
      (hallway ?l1)
      (hallway ?l2)
      (connected ?l1 ?l2)
      (at ?l1)
    )
    (
      (at ?l1)
    )
    (
      (at ?l2)
    )
  )

  (:operator (!enter ?l1 ?l2)
    (
      (hallway ?l1)
      (room ?l2)
      (connected ?l1 ?l2)
      (at ?l1)
    )
    (
      (at ?l1)
    )
    (
      (at ?l2)
    )
  )

  (:operator (!exit ?l1 ?l2)
    (
      (room ?l1)
      (hallway ?l2)
      (connected ?l1 ?l2)
      (at ?l1)
    )
    (
      (at ?l1)
    )
    (
      (at ?l2)
    )
  )

  (:operator (!pickup ?c ?l)
    (
      (cube ?c)
      (at ?l)
      (in ?c ?l)
```

```
      (unloaded)
    )
    (
      (unloaded)
    )
    (
      (has ?c)
    )
  )

  (:operator (!drop ?c ?l)
    (
      (cube ?c)
      (at ?l)
      (has ?c)
    )
    (
      (has ?c)
    )
    (
      (in ?c ?l)
      (unloaded)
    )
  )

  ;------------------------------
  ; Methods
  ;------------------------------

  (:method (goto ?to)
; TODO This is a skeleton of a method, you must build the methods yourself
    base
    (
      (at ?to)
    )
    ()

    gotoenter
    (
      (at ?from)
      (hallway ?from)
      (room ?to)
      (connected ?from ?to)
    )
    (
      (!enter ?from ?to)
      (goto ?to)
    )

  )

))
```

and problem:

```
(defproblem problem aperture
```

```
  ;------------------------------
  ; Start
  ;------------------------------

  (
    (room room1)
    (room room2)
    (hallway h1)
    (connected h1 room1)
    (connected room1 h1)
    (connected h1 room2)
    (connected room2 h1)
    (cube c1)
    (at h1)
    (in c1 room1)
    (unloaded )
  )

  ;------------------------------
  ; Tasks
  ;------------------------------

  (
    (goto room2)
  )
)
```

References

[1] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Elsevier, Burlington, MA, 2004.

A Aperture Domain Overview

In this assignment you will formalise *ATLAS* (or *P-body* if you are a girl), a research robot that is used by *Aperture* labs to test Portal[1] Technology. ATLAS's job is to navigate through a series of rooms taking **Weighted Companion Cubes** (which we shall call "cubes" from now on) from rooms throughout the environment to a destination area in order to complete the experimentation process.

For this project, we will model a simple environment that ATLAS has to work in. This is essentially a long office hallway that is split into various segments or hallway-locations. These hallway-locations may or may not be connected to various rooms.

You will need to model two different types of locations, hallways and rooms. ATLAS can be "at" a particular location at a given point, and only at that location. Two locations can be "connected" to each other, enabling ATLAS to navigate between them (regardless of whether they are rooms or hallways). Note that connections are symmetric, so you need to model the fact that if ATLAS can go from A to B, then going from B to A is also possible. You can also assume that a hallway-location is a whole object - if ATLAS is anywhere in a given hallway-location, then ATLAS can enter any of the rooms connected to that hallway-location, and move from/to any of the other hallway-locations connected to it.

Navigating between hallway-locations and rooms is achieved via "enter" and "exit" actions. The enter action enables moving from a hallway-location to a room, while the exit action enables the opposite - moving from a room to a hallway-location. Remember that the two need to be connected in order to perform an enter or an exit - and you need to model those connections.

To navigate between two connected hallway-locations, ATLAS uses a special "move" action that only works on locations of type hallway. In the domain and problems that you are to model, ATLAS **should not** move within rooms or from one room to another directly, and only enters and exits from hallway-locations.

ATLAS can "pickup" and "drop" cubes located "in" the same room as he is and the result of picking up a cube is that ATLAS "has" a cube. Notice that ATLAS can carry **only one** cube.

The cubes can be in either hallway-locations or rooms (since they both are subtypes of "location"). ATLAS needs to necessarily be "at" a given location in order to pick up the cubes at that location. Once ATLAS picks up the cubes, he must carry the cube to the destination before moving on to the next task at hand (so you may want to model position of cubes as goals). Finally, there is a destination that ATLAS (and the cubes) must end up at - this is another goal that you must model.

As a bonus to this assignment, you must model ATLAS being able to use a *Portal Gun* to generate portals[2] between rooms and use these portals to move the cubes to the destination[3]. Each domain has a particular type of "portal" that specifies two colors (e.g. orange and blue) representing each end of a linked teleportation portal. ATLAS can "teleport" between two locations using a portal if there is an "open" color end of a portal at each room. Portals can only be opened in rooms that contain a "flatwall", limiting the ability to use the portal. If a room contains such a wall, then ATLAS can "shoot" a portal into the room, but only if it is not carrying a cube and is in the room where the portal is being created. In this case, you must change the original movement operators so that all physical connections between rooms and hallways do not allow cubes to pass through them[4]. Thus, you must change the original movement operators so that they **do not** allow cubes to be moved between rooms (i.e. you can only move cubes using teleportation).

Whenever we need to use automated tools to solve problems on our behalf, we must provide a consistent specification of the *transition system* of the underlying problem. If we specify the problem poorly, we jeopardise the planner software ability to generate valid responses, leading to false negatives and unnecessarily long waiting times for the planner at best, or incorrect plans at worse. Thus, specifying the ATLAS domain in PDDL gives you a chance to develop your skills in designing consistent transition systems, helping you avoid bugs in the software you will write in the future to handle all kinds of other processes.

Your assignment is to develop a domain file from the specification above, and then model the situations depicted in the images below as individual problem files. The following hints may be useful, but you are welcome to use your creativity as long as you adhere to the specification mentioned above:

[1] TM®and all that to Valve...

[2] `http://half-life.wikia.com/wiki/Aperture_Science_Handheld_Portal_Device`

[3] `https://www.youtube.com/watch?v=TluRVBhmf8w`

[4] `http://theportalwiki.com/wiki/Material_Emancipation_Grill`

- You need actions for moving between hallway-locations, as well as actions to enable the entering and exiting;
- You need actions to pick up and drop cubes at a specific location, so ATLAS can take the cubes around (There is no sensing involved here; if a cube is declared as being in a particular location, and ATLAS is also at that location, then he can pick up that cube.);
- Remember that when modeling pickup and drop actions, you may need an "unloaded" predicate to indicate you are carrying nothing (otherwise your planner would need a quantified negation, which JavaGP does not have);
- (If you do the bonus task) You need an action to create the portals and another action to teleport between them, notice that it is easier to initialise all problems with the portals created somewhere; and
- Look at the words in quote-marks in the specification above. They may give you a good skeleton to base your domain on.

B Problem Instances

Below are images of the problem instances that you need to model in PDDL, once you are done making your domain file. The legend that accompanies each image should be fairly self-explanatory; remember, you must model connections between hallway-locations for ATLAS to move from one to the other, and you can only enter and exit between a hallway-location and a room that are connected. In the instances shown in Figures 1 through 3, any locations that share an edge can be considered connected (you do not need to explicitly model doors). Locations that share only corners and no edges **are not** connected.

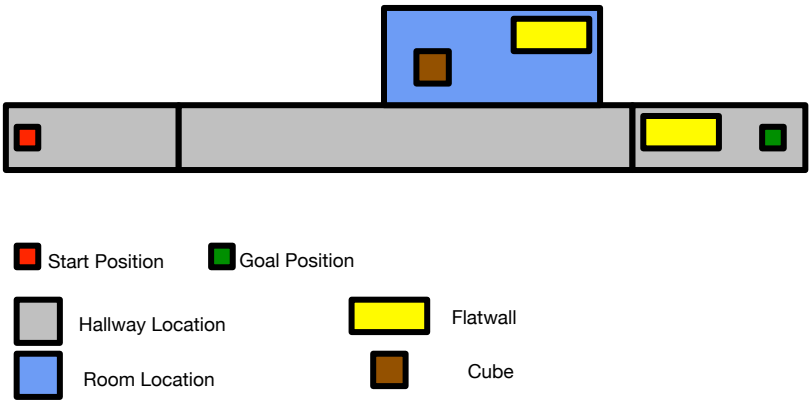

Figure 1: Problem 1

Notice that the above pictures give you an idea of the initial state of the world (which you must encode in your PDDL problem file). They also tell you what the goals are - ATLAS's final location (in green), and the various cubes that must be collected on the way. If you look at the syntax of example PDDL problem files, you will see that these are the three main parts of a problem file - (1) the objects, (2) the initial state, and (3) the goals.

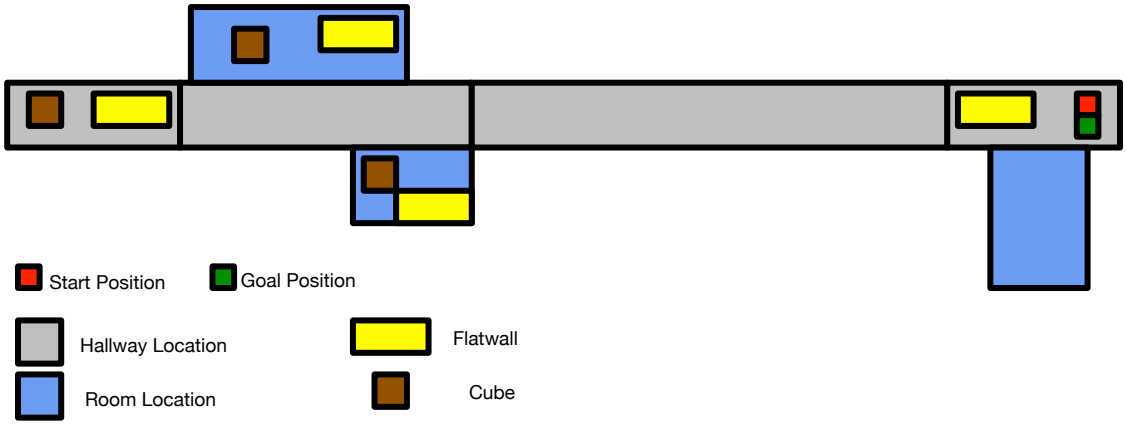

Figure 2: Problem 2

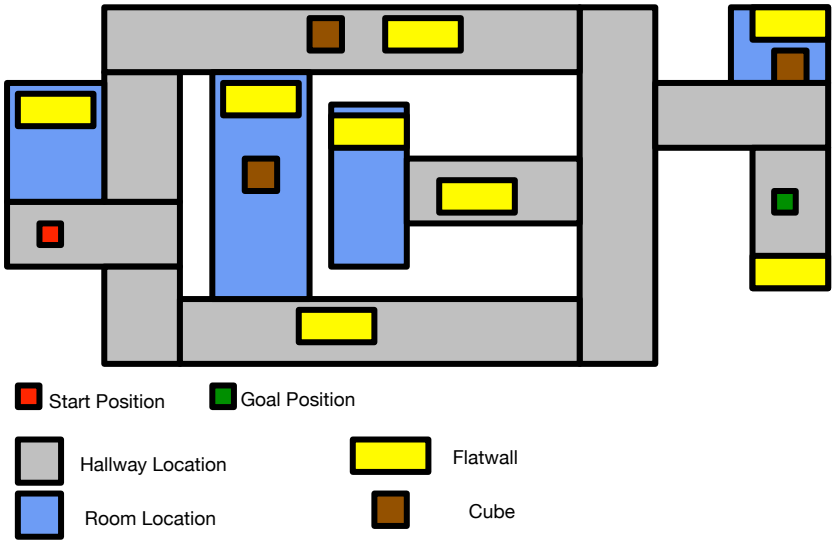

Figure 3: Problem 3