EECS 3401 Assignment 3

Deadline March 30, 2016

PROBLEM 1: PLANNING in STRIPS.

A game character, depicted in Fig. 1 as G, is hungry. G is in the kitchen in search for food. There are two appliances in the kitchen: refrigerator (ref) and microwave oven (mo).

Initially, ref is closed and contains two food items: chicken and pizza, that require baking before consumption. It also contains a book. The microwave oven is closed. G is close to mo but not to ref (as depicted in Fig. 1)

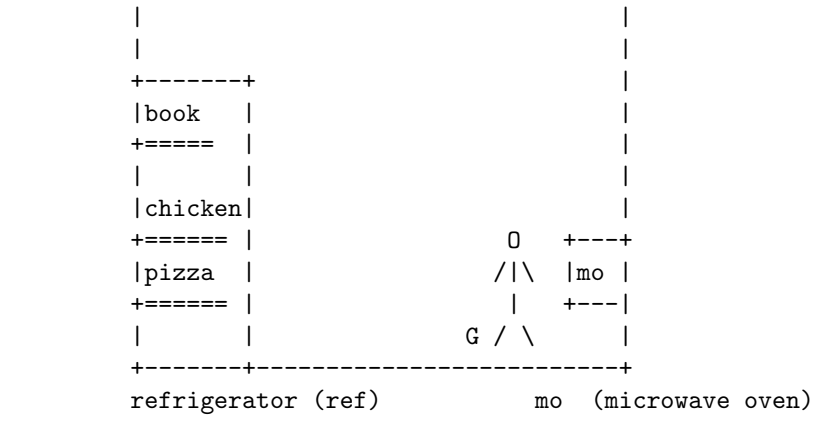

Figure 1: Hungry G scenario

Our character G has to consume some baked food to satisfy hunger.

A. Describe formally the initial state as depicted in Fig. 1. Use the following predicates:

- not_hungry, (true if G is not hungry)
- opened(X), (true if X is opened)
- closed(X), (true if X is closed)
- cl(X), (true if G is close to X)
- in(X,O), (true if X is in O)
- has(X), (true if G has X)
- baked(X), (true if X is baked)

and the following constants:

- ref (refrigerator),
- mo (microwave oven),
- chicken,
- pizza,
- book.

B. Describe formally the list of goals that correspond to G's task.

C. Create a file named, say, 'states' containing the initial state and the goal.

D. Complete the definitions of the following actions available to G and partially specified in the file "eat":

- open(X) (G is to open X)
- close(X) (G is to close X)
- take(X,From) (G is to take X from From)
- putIn(C,In) (G is to place C in In)
- bake(X) (G is to bake X in mo)
- eat(X) (G is to consume X)
- move(To) (G is to move from the current location to To)

To complete the definitions, replace '?' with appropriate expressions.

E. Use STRIPS planner to find a plan for G to satisfy G's hunger by, for instance, entering:

?- consult(eat,planner).

to input the planner code and the definitions of actions and, then executing the goal:

?- see(states),read(S),read(G),seen,plan(S,G,Plan,Final).

to get a plan, where:

- State - is the initial state (see A),
- Goals - is a list of goals to be satisfied in the final state (see B),
- Plan - is a plan to be discovered,
- FinalState - is the resulting state after executing the Plan in the initial situation.

PROBLEM 2: **MARS ESCAPE!**

Path-finding in a known terrain is frequently accomplished using heuristic search algorithms such as A* which always returns a cost-optimal path in a tree-like search space, provided that the heuristic function is admissible.

In this problem A* is used to find several paths on the Martian terrain depicted on image mars_3A.jpg. The paths to be found are: from cell A to cell B, and form cell A to cell C as shown on the image mars_3B.jpg.

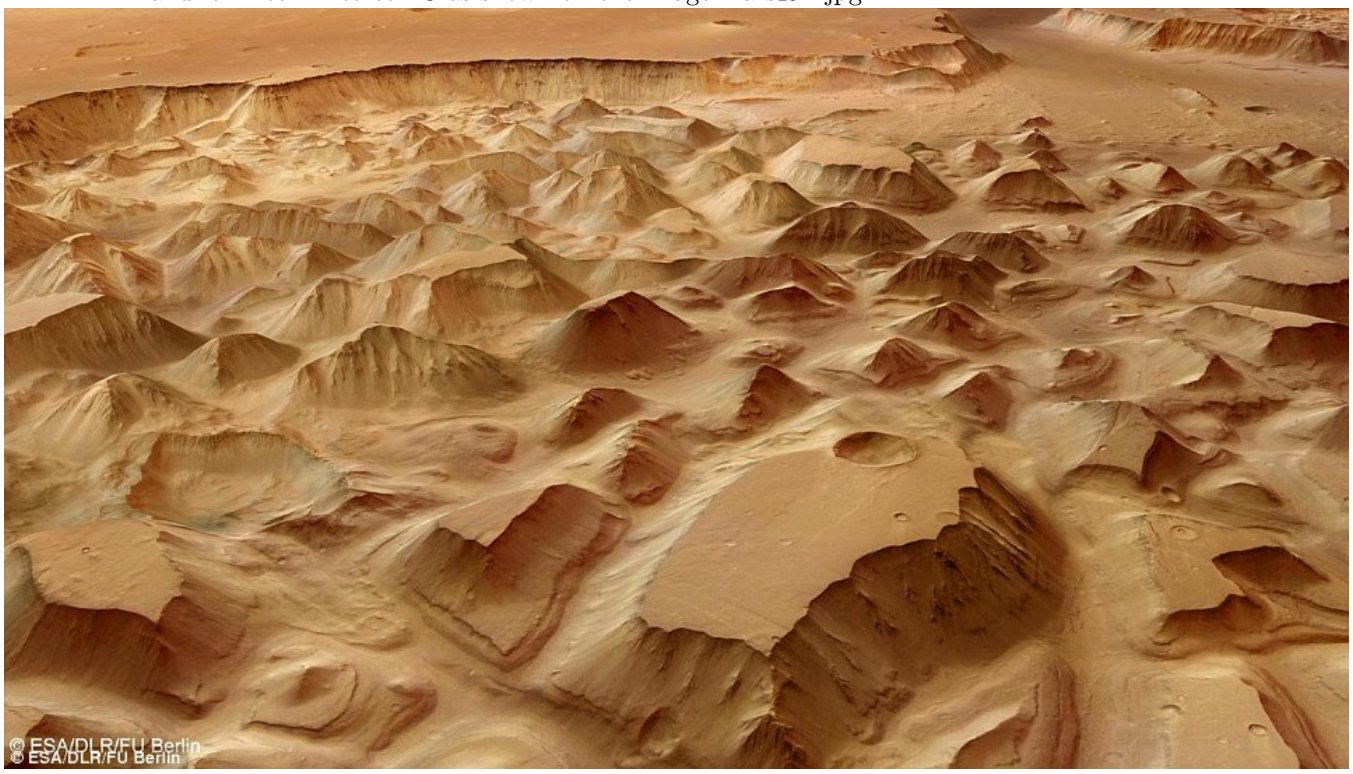

Image 1: image mars3A depicting a sample Martian terrain.

Things to do:

Step A: create an occupancy grid map from image mars3B.

Step B: implement A*.

Next, use the occupancy grid map created in Step A and A* to:

Step 1: find a path from cell A to cell B using an admissible heuristic function;

Step 2: find a path from cell A to cell B using a non-admissible heuristic function;

Step 3: find a path from cell A to cell C using an admissible heuristic function;

Step 4: find a path from cell A to cell C using a non-admissible heuristic function.

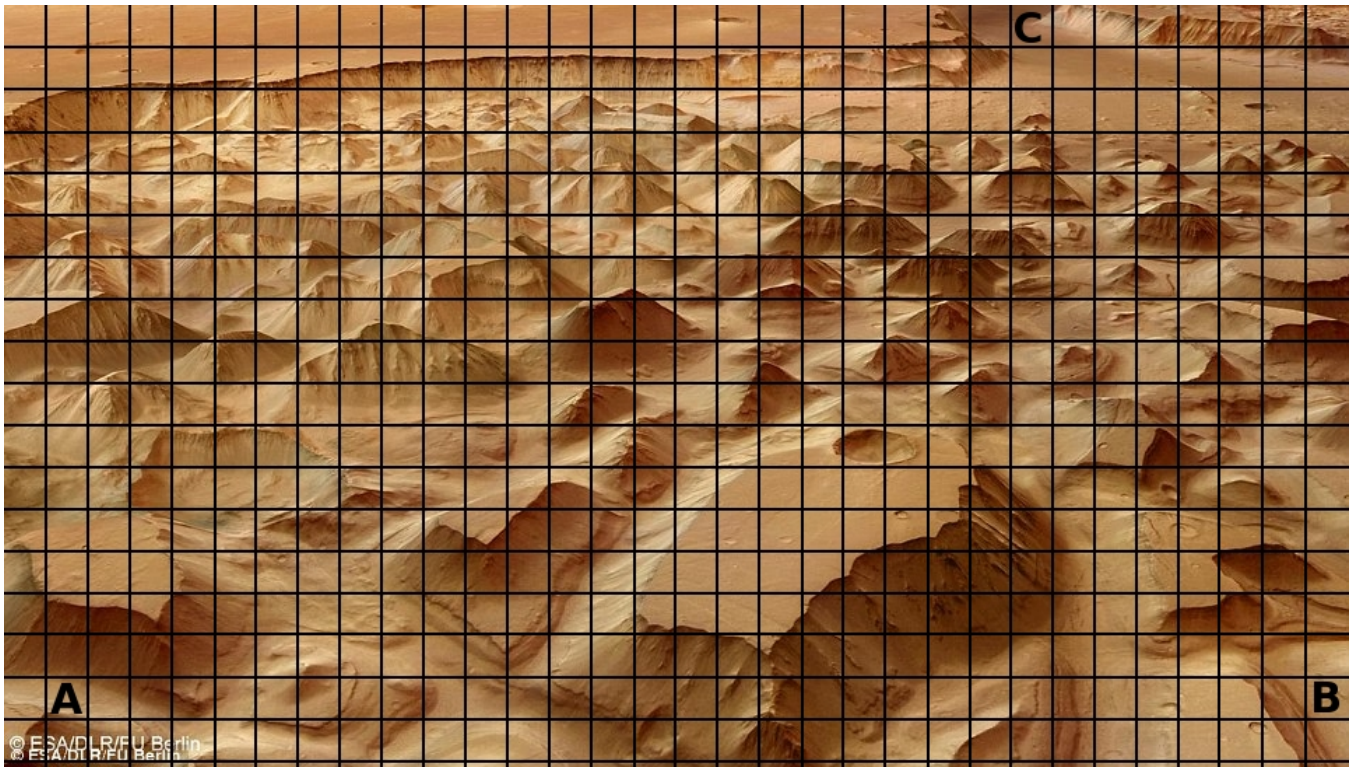

Image 2: mars3B.jpg, occupancy grid map for Problem 2.

Documentation

- define heuristic functions used in Steps 1-4.
- include the grid occupancy map;
- include the grid occupancy maps with solution paths marked on them; discuss the results.

Bonus: for your own implementation of A* in Prolog.