

Homework III:
Symbolic Planning
DUE: April 19th (Monday) at 11:59PM

Task

Implement a generic symbolic planner. We have provided code for reading an environment description file and generating the corresponding environment object. You are required to write a planner that takes an environment object as an input, and outputs a sequence of actions from the start to the goal states. There is an example environment configuration that is provided.

Undergraduate Students – Write one additional environment description file for the Blocks and Triangles World task.

Graduate Students – Write two additional environment description files for the Blocks and Triangles World and Fire Extinguisher tasks.

Code

The planner should reside in *planner.cpp* file. Your planner must output the entire plan with sequence of all actions that reach the goal. The planner function is as follows:

```
list<GroundedAction> planner(Env* env){ }
```

The current function just returns a list of hardcoded actions that are not parsed from the environment description.

Environment Description



An example environment description file for the Blocks world is given to you in *example.txt*.

In the provided code, we generate an environment object (of the Env class) from the environment description file. The Env class includes the (1) initial conditions, (2) goal conditions, (3) actions,

(4) symbols. An object of the Env class is passed to your planner. Your environment description file should follow the same template as example.txt.

The *Env* class uses the data structures below. You may add more functions to them as needed.

However, DO NOT change the main function!

- *Condition*: this class includes 3 variables: (1) name of the condition, (2) the arguments, (3) if the condition is negated or not.
- *GroundedCondition*: this class includes 3 variables: (1) name of the condition, (2) the values for the arguments, (3) if the condition is negated or not.
- *Action*: this class includes 4 variables: (1) name of the action, (2) action arguments, (3) preconditions, (4) effects.
- *GroundedAction*: this class includes 2 variables: (1) name of the action, (2) values for the arguments.

You are required to write a generic planner that outputs a sequence of steps to go from the initial condition to the goal condition. The output of your planner is a list of *GroundedActions*.

Environments

Note that the parser does not verify if your environment description is valid, so make sure that your description files follow the template (i.e., regular expressions) we have provided. These environment description files are parsed into an environment object that is passed to your planner. The environments are as follows:

1. Blocks and Triangles World

This environment is like the Blocks world problem explained in the class. In addition to the blocks, this environment has triangles that can be moved in the exact same way as blocks with the exception that nothing can be put on top of them. A simple example of this environment with only three objects is shown below. You are required to write a description file for an environment with 5 blocks (B0, B1, B2, B3, B4), 2 triangles (T0, T1) and a Table. The start and goal conditions are:

Start conditions: B0 is on B1, B1 is on B4, B2 is on Table, B3 is on B2, B4 is on Table, T0 is on B0, and T1 is on B3.

Goal conditions: B0 is on B1, B1 is on B3, and T1 is on B0.



2. Fire Extinguisher Environment

In this environment, a pair of robots must put out a fire. This domain has two robots: (1) a quadcopter and (2) a mobile robot. The domain has the following specifications:

- a. The mobile robot can travel between locations.
- b. The quadcopter cannot travel between locations by itself and cannot land on the ground.
- c. The quadcopter can only travel between locations by landing on the mobile robot and having the mobile robot travel to the other location.
- d. The quadcopter can hover around a single location if its battery level is High, but it will not be able to take off if its battery level is Low.
- e. Whenever the quadcopter is on the mobile robot, it can charge its battery by calling the charge action.
- f. The quadcopter has a tank. This tank can be filled with water when the quadcopter is on the mobile robot at location W (where there is water).
- g. The fire is at location F.
- h. The W and F locations are far from each other.
- i. The quadcopter must hover around location F in order to pour water on the fire.
- j. The quadcopter needs to pour water on the fire three times in order to extinguish the fire.
- k. Every time the quadcopter pours water on the fire, its battery level becomes low, and its water tank becomes empty. It would need to go back to W to fill its tank again.

The robots will each start at one of five different locations (A, B, C, D, E), which are far from W and F. The start and goal conditions are:

Start conditions: the quadcopter is hovering around location B. The mobile robot is at location A. The quadcopter's water tank is empty.

Goal conditions: The fire is extinguished.

This task is inspired by the final challenge at the 1st Summer School on Cognitive Robotics at MIT

Execution

To compile the cpp code:

```
>> g++ planner.cpp -o planner.out
```

To execute:

```
>> ./planner.out example.txt
```

Replace *example.txt* with the description file of the environment you want to plan for. Once your planner returns the plan, it will be printed out. It is your responsibility to check whether the plan is valid with respect to the start conditions, actions, and goal conditions.

Submission

You will submit this assignment through Gradescope. You need to upload one ZIP file named *<Andrew ID>.zip*. This should contain:

1. A folder named *code* that contains all code files, including but not limited to, the ones in the homework packet. If there are subfolders, your code should handle relative paths.
2. A PDF file named *<Andrew ID>.pdf*. This should contain a summary of your approach for solving this homework, results, and instructions for how to compile your code.
 - Your planner must be **domain independent**. This means it must be generic and applicable to any environment. **Your planner will be tested with other environments.**
 - You must report and discuss results of applying your generic planner in the given Blocks world environment (example.txt) as well as the additional environments for which you wrote description files.
 - You are required to implement at least one heuristic. You must compare results obtained with and without heuristics.
 - For all environments, discuss the following results with and without heuristics:
 1. Number of steps in the plan.
 2. Number of steps expanded by your search.
 3. Amount of time taken by the planner.

Grading

The grade will depend on:

1. Correctness of planner and additional environment description files.
2. How well-founded is your approach? Can your planner guarantee completeness?
3. Whether your planner is domain independent. Is it implemented as a generic search that can be used to solve a completely different problem?

4. The quality of the plan. Is your plan optimal (minimizes the number of steps)? Can your planner solve problems within 30 seconds?
5. The quality of your write-up. Provide an explanation for all reported statistics. Provide a discussion of the pros and cons of using a heuristic.

Extra credits

For undergraduate students:

1. +10pts for implementing the Fire Extinguisher Environment