Haohan Xie haohanx@umich.edu

(a) What major functions did you write, what do they do, and why? You can ignore less relevant helper functions. (Write a brief paragraph per function)

I write 7 separate functions to handle the plan Search problem.

(1) Major plan Search function
Use a priority queue to sort the plan by heuristics scores, add possible plan by binding comparison and new actions adding

(2) Deep copy functions
Handle the deep copy problem by python, define a function could deep copy the object of object, the deep copy function implement in the structures.py, as a sub function of each class

(3) Computation functions
Computation the unresolved threats, traverse all links and steps for finding the possible threats and add it into the threats list

(4) Heuristics function
Compute the heuristics scores for sorting the plan in priority queue

(5) Add function
Compare and binding two predicates, get the possible binding for solving the open conditions

(6) Perform function
Replace the variables by binding in steps, links and open_conditions

(7) Creation of potential actions function
Add the new potential actions in to a list. For future binding and insert operation

(b) What search algorithm did you use, and why? If you used heuristics, what were they? (Write at least a paragraph).

I use an easy heuristics function. Since the shorter open_condition means have finished more goals, each time I pop out the plan with shortest open_condition.

(c) Devise your own real-world planning problem and give a full PDDL description. Which planning or search algorithm would you use to solve it?

As a student enrolled in Umich, every semester we face a planning problem. We planed the courses schedule each first 3 weeks.

For the course enrollment planning problem.

A set of courses {c1,c2,c3…}

A set of timesection {t1, t2, t3…}

A courseList cl

Predicates:

hasTimeconflict(c)	c has time conflict with other course
hasTook(c)	has took c course
hasPermit(c)	has Pre-course for course c

hasEnrolled(c) has enrolled c course
moreThanthree(cl) this course list has more than 3 courses

Actions:

Action: Drop(c)
Pre-condition: hasEnrolled(c)
Effect: ~hasEnrolled(c)

Action: Replace(c1,c2)
Pre-condition: hasEnrolled(c1) ∧ ~hasEnrolled(c2) ∧ ~hasTimeconflict(c2) ∧ ~hasTook(c2) ∧ hasPermit(c2)
Effect: ~hasEnrolled(c1) ∧ hasEnrolled(c2)

Action: Enrolled(c)
Pre-condition: ~hasEnrolled(c) ∧ ~hasTimeconflict(c) ∧ ~hasTook(c) ∧ hasPermit(c)
Effect: hasEnrolled(c)

Action: Audit(c) (Could has time conflict)
Pre-condition: ~hasEnrolled(c) ∧ ~hasTook(c) ∧ hasPermit(c)
Effect: hasEnrolled(c)

Algorithm:

Set this problem as a no perform course problem, if I set the goal as:
Goal: moreThanthree(cl) ∧ Enrolled(c4)
For this linear planning problem, I could use the backtrack algorithm. Take c4 course first then check if is not fit the goal, keep enrolled or replace it.