TDT4136 Assignment 6: Planning

This assignment is about planning-graphs. You can work in groups of two. You can choose between question 1 or 2. That is, you are assumed to answer one of these questions. Note that question 2 (programming) is much more difficult and only those who really seek a challenging assignment should attempt to solve it.

The "story" of the assignment:

A gardening agent has to mow the lawn, water the lawn and water the flowers. The lawn is watered using a sprinkler. The flowers can be watered using either the sprinkler or watering can. Once the sprinkler is assigned to a task it cannot be assigned to another task for the purpose of this assignment: we think of the sprinkler as being assigned a single task for the duration of the gardening exercise.

Here are the propositions and actions you will use:

- availSprk – sprinkler is available
- longLawn – lawn is long
- dryLawn – lawn is dry
- dryFlwrs – flowers are dry
- wetLawn – the lawn is wet
- wetFlwrs – the flowers are wet
- shortLawn – the lawn is short

mowLawn – mow the lawn

precond: dryLawn, longLawn
add: shortLawn
delete: longLawn

sprkLawn – water the lawn using the sprinkler

precond: availSprk, dryLawn
add: wetLawn
delete: availSprk, dryLawn

sprkFlwrs – water the flowers using the sprinkler

precond: availSprk, dryFlwrs
add: wetFlwrs
delete: availSprk, dryFlwrs

wcanFlwrs – water the flowers with a watering can

precond: dryFlwrs
add: wetFlwrs
delete: dryFlwrs

The initial conditions and goal are:

Initial cond: availSprkl, longLawn, dryFlwrs, dryLawn
Goal: shortLawn, wetLawn, wetFlwrs

Question 1)

You will construct a planning graph starting from the initial goal. Expand the graph when necessary and extract the plan when the goal is achieved. Draw a graph for each level and show the "add", "delete" (adds negated literals) and "mutex" relations. Use the same numbering notation for the situation and action levels as in Figure 10.10 in the text book. At each level explain in text why do you need (or not) to expand (i.e., a new level).

Find at least one possible plan using the planning graph. Mark this plan by highlighting the constituent nodes and edges in the graph. Write down the plan as a sequence of actions.

Because the planning graph will contain a lot of intersecting edges (add, delete, mutex relations) the drawing may become quite messy, unreadable when drawn by hand. Therefore, it is highly recommended to draw the planning graph on a computer using some vector/diagram editor, e.g. Microsoft Visio and Google Drawing are perfect for the job. Scan of the hand drawing is acceptable, but make sure that your graph is easily readable. Additionally, you may choose to ignore negated literals all together because they are not part of the goal or preconditions for actions.

Deliverables

Deliver the report that includes the following items:

1. Figure with the planning graph and the highlighted plan
2. Explanation for why do you need (or not) to expand at each level
3. A list of actions representing a plan that solves the gardening problem
4. Explanation of how you obtained the plan in 3. from the planning graph

Question 2)

In this question you will implement the graph planner. This is a more challenging task than solving the gardening problem by hand in question 1. It requires strong programming skills and extra time. The algorithm you will implement is *Graphplan* with the following signature:

Graphplan(iCond, goals, actions): returns a plan where

- iCond: initial condition,
- goals: the goal list

- actions: the list of actions.

The pseudo code for the *Graphplan* algorithm is given in the text book (Figure 10.9, page 383). It provides the structure for the algorithm itself but not for the construction of the planning graph (*Expand-Graph* function) and the extraction of a solution from the graph (*Extract-Solution* function).

Expand-Graph adds a new level (literals and actions) to the graph. The procedure for constructing a new level is described on pages 379, 380, 381, 383, 384; several examples are provided there as well.

Extract-Solution searches backwards for solution in a planning graph. The procedure is described on pages 384, 385.

Although detailed explanations for these functions are available in the textbook, no pseudo code is provided, so you will have to figure out some of the details yourself, ask a student assistant or Google.

Deliverables

You must deliver the following items:

- Code with comments
- A brief overview of your implementation
- Trace for the algorithm (literals, actions, literal mutex relations, action mutex relations at each level)
- A plan (sequence of actions) found by you implementation that solves the gardening problem.