Massachusetts Institute of Technology

6.034 Artificial Intelligence

Fall 2000

Examination Solutions # 1

Problem 1 Enforcing the Rules on Wall Street (30 points)

Part A (12 points)

Worksheet

Initial Database
Joe bought stock in the past two weeks
Joe is a friend of Sam
Sam has insider information
Joe is a relative of Sam

Step	**Rule(s) Triggered**	Bindings	**Rule Fired**	**Assertion(s) Added**
1	R2	?x = Joe	R2	Joe has traded recently
	R4	?x = Joe, ?y = Sam		
	R5	?x = Joe, ?y = Sam		
	R6	?x = Joe		
2	R4	?x = Joe, ?y = Sam	R4	Joe knows Sam
	R5	?x = Joe, ?y = Sam		
	R6	?x = Joe		
3	R3	?x = Joe, ?y = Sam	R3	Joe has insider information
	R6	?x = Joe		
4	R1	?x = Joe	R1	Joe is an insider trader
	R6	?x = Joe		
5	R6	?x = Joe	R6	Joe is an active trader

Part B (12 points)

Rule Name	Antecedent	Consequent
R1	?X has insider information and ?X has traded recently	?X is an insider trader
R2	?X bought stock in the past two weeks	?X has traded recently
R3	?X knows ?Y and?Y has insider information	?X has insider information
R4	?X is a friend of ?Y	?X knows ?Y
R5	?X is a relative of ?Y	?X knows ?Y
R6	?X bought stock in the past two weeks	?X is an active trader.

Initial Database
Joe bought stock in the past two weeks
Joe is a friend of Sam
Sam has insider information
Joe is a relative of Sam

The tree (not required):

```
(Joe is an insider trader)
 |        R1        |
 |                  |
 |                (Joe has traded recently)
 |                                       | R2
(Joe has insider information)            |
 |        R3          | R5               |
 |                    |                (Joe has bought stock in the past two weeks)
 |                    |
(Joe knows ?y)        (Sam has insider information)
 | R4
 |
 |
(Joe is a friend of ?y)
```

(Joe is an insider trader) not in database, R1 deployed, asks for (Joe has insider information). That not in database, R3 deployed, asks for (Joe knows ?y). That not in database, R4 deployed, asks for (Joe is a friend of ?y). That is in the database, binding ?y to Sam. R4 succeeds. R3 succeeds. R3 now checks, with ?y bound, to see if (Sam has insider information, which is in the database. R2 succeeds. Back to second antecedent of R1. Check for (Joe has traded recently). Not in the database, deploy R2, find (Joe bought stock in the past two weeks). Done.

The table:

Step	Pattern matched against the database
1	(Joe is an insider trader)
2	(Joe has insider information)
3	(Joe knows ?y)
4	(Joe is a friend of ?y)
5	(Sam has insider information)
6	(Joe has traded recently)
7	(Joe bought stock in the past two weeks)
8	
9	
10	
11	

Part C (6 points)

Backward chaining systems make all the inferences that follow from the facts in the database. No.

Backward chaining systems can infer something that is not in a goal tree terminating at the question at the top of the tree. No.

Backward chaining systems can instantiate a rule with different bindings for two occurrences of the same pattern variable in that rule. No.

Backward chaining systems can handle a situation in which the same antecedent is used in more than one rule. Yes.

Problem 2 Search in a weird city (38 points)

Part A (6 points, hill-climbing search)

You were to:

- Use hill climbing.
- Use straight-line distance as the heuristic quality-of-node measure.
- With **no** backtracking (also known as no backup).
- With **no** use of a visited or expanded list.

A.1

Search succeeds.

A.2

Search proceeds directy to goal. Nine nodes, not counting goal. Ten accepted as well, because of possible assumption is that the path containing the goal is expanded before goal is tested. Both ways were taught.

Part B (12 points, breadth-first search)

You were to:

- Use breadth-first search.
- With **no** use of a visited or expanded list.

B.1

Sum of nodes in tree of depth d and braching factor b is $b^{d+1} - 1/b - 1 = 2^9 - 1 = 511$ (or 512).

B.2

With visited list, no node expanded twice, so answer is just a count of the nodes, which is 17 (or 18).

Part C (12 points, branch-and-bound and A* search)

You were to:

- Use branch-and-bound search.
- With **no** use of a visited or expanded list.
- With **no** admissible estimate of distance remaining.

C.1

Because of the extremely long length of the final street, all paths even one step short of the goal are relatively short, and require expansion, including paths leading to the extreme left nodes, which do not happen to have any following nodes. Answer is twice previous answer, minus 1 (so as not to double count the start node), hence 1021 (or 1022).

C.2

With expanded list, no node expanded twice, so answer is just a count of the nodes, which is twice what is was before, minus one (so as not to double count the start node), hence 33 (or 34).

C.3

Because of the extremely long length of the final street, admissible estimate does not help. Same as previous answer, 33 (or 34).

C.4

With the winding roads made direct straight lines, the admissible can help, so number of nodes expanded would decrease.

Part D (8 points, branch-and-bound and A^* search)

You were to:

- Consider an arbitrary map, not the maps previously used in this problem.
- Use branch-and-bound search.
- With an expanded list.
- With **no** admissible estimate of distance remaining.

With the problem being to arrive at the destination with the most gas. This would translate to "least gas used" which would mean "least distance travelled," but for the unknown gas station situation. Alas, the gas stations, from the perspective of gas consumption, are like a negative cost, which clobbers the key assumption in branch and bound that costs are all positive. Following illustrates. If expanding on the basis of gas used, would not go beyond I in the search of the top path, but the gas station at G, if providing enough gas, could make the S-G route the lessor route.

```
         10
S *--------------->* I
  |                |
  | 3              |G
  v                |
  *<---------------*
G
```

Problem 3 Doing a constrained search (32 points)

Part A (11 points)

Pure backtracking

```
                   ----------------()---------------
                   |                               |
           -------A1-------                 -------A2------
           |      1        |                |       6       |
       ---B1---        ---B2---         ---B1---        ---B2---
       |  2    |       |  5    |        |   7   |       |       |
     -C1-    -C2-    -C1-    -C2-     -C1-    -C2-    -C1-    -C2-
     | 3 |   | 4 |   |   |   |   |    | 8 |   | 9 |   |   |   |   |
    D1  D2  D1  D2  D1   D2  D1  D2   D1  D2  D1  D2  D1  D2  D1  D2
                                      10  11
```

Part B (11 points)

Backtracking with forward checking

```
               ----------------()---------------
               |                               |
       -------A1-------                 -------A2------
       |      1        |                |      2       |
   ---B1---        ---B2---         ---B1---        ---B2---
   |       |       |       |        |   3   |       |       |
 -C1-    -C2-    -C1-    -C2-     -C1-    -C2-    -C1-    -C2-
 |   |   |   |   |   |   |   |    |   |   | 4 |   |   |   |   |
D1  D2  D1  D2  D1   D2  D1  D2   D1  D2  D1  D2  D1  D2  D1  D2
                                               5
```

Part C (10 points)

The schedule will be a valid, minimal difficulty schedule (in the sense that the most difficult term will not be more difficult than the most difficult term in a nonoptimal schedule).

You can add forward checking, but you will have to keep track of the reduced domain set for every path in the queue, which could involve a great deal of memory.