COMP-424: Sample problems with solution for the midterm

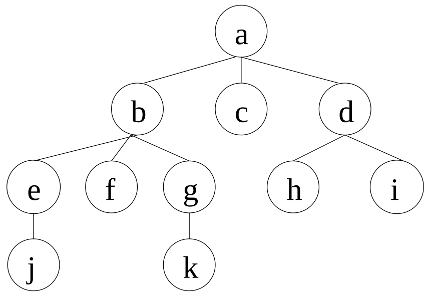
Note: These problems were intended to provide additional practice for the midterm

1. You are the designer of a new route finding system for a car. The driver types the desired destination, and can select a preference, e.g. fastest driving time or shortest distance. The route finder suggests a route. The driver may take the suggested route all the way, or part of the way. In the second case, the driver may ask the system for guidance again, or may go on without asking. The auto makers would like the route finder to be efficient, and to adapt to the preferences of different drivers.

   What search and/or planning methods would be appropriate for this problem? Motivate your answer. What would the states, operators, costs and goals be? Would you use any heuristics?

2. List the order in which nodes are visited in the tree below for each of the following three strategies (choosing leftmost branches first in all cases):

   - Depth-first search.
   - Depth-first iterative-deepening search (increase depth by 1 in each iteration).
   - Breadth-first search.



3. Prove that if h is an admissible heuristic, then $h(n^*) = 0$ for all goal nodes n^*.

   Solution:

   Let $c(n, g)$ denote the cost from node n to the closest goal state g. Since h is admissible, $h(n) \leq c(n, g) \forall n$. So $h(n^*) \leq c(n^*, n^*) = 0$, which means $h(n^*) = 0$.

4. Consider the problem of missionaries and cannibals: 3 missionaries and 3 cannibals are on one side of a river, along with a row boat that is capable of transporting either 1 or 2 occupants. The objective is to find a way in which to move both the missionaries and the cannibals to the opposite side of the river, without ever allowing the number of cannibals on a river bank to exceed the number of missionaries (including the occupants of the boat), because then the cannibals would eat the missionaries. We would like to do this as quickly as possible.

   Formulate this as a search problem. Specify the representation of the states, operators, goal, start state and cost function. Can you find an admissible heuristic?

5. Towers of Hanoi. This is a classical problem, in which you have 3 pegs, A B and C, and a number N of discs with different diameters. A larger disc cannot be placed on top of a smaller disc. The discs are initially placed on peg A, and the goal is to move them all on peg C, in a minimal number of moves.

Formulate this as a search problem. Specify the representation of the states, operators, goal, start state and cost function. Find an admissible heuristic, and show that it is admissible.

6. Russell and Norvig, Problem 6.5. Provide a propositional description of the given text, and write the proofs for the book questions.

   Solution: Our boolean variables will be: $Mythical$, $Magical$, $Mortal$, $Mammal$ and $Horned$. The text of the problem can be translated in propositional logic as follows:

$$\begin{aligned} Mythical &\Rightarrow \neg Mortal & (1)\\ \neg Mythical &\Rightarrow Mortal \wedge Mammal & (2)\\ \neg Mortal \vee Mammal &\Rightarrow Horned & (3)\\ Horned &\Rightarrow Magical & (4)\end{aligned}$$

   Suppose that the unicorn is mythical. Then by applying modus ponens with (1) we can deduce $\neg Mortal$. By further modus ponens with (3) we can deduce $Horned$, and finally by modus ponens with (4) we can deduce $Magical$.

   Similarly, suppose the unicorn is not mythical: $\neg Mythical$. Then by modus ponens with (2) we can deduce $Mammal \wedge Horned$, from which we can deduce $Mammal$. By modus ponens with (3) we deduce $Horned$, and finally, by modus ponens with (4) we deduce $Magical$.

   In both cases we proved that the unicorn is magical and horned. Since neither can led to a contradiction, we cannot wither prove or disprove that the unicorn is mythical.

7. How many Boolean operators (connectors) can there be? Why are some of them not very useful?

   Solution: This question is the same as asking how many truth tables can be defined with n Boolean variables. A truth table of n variables has 2^n entries, and each entry can be either true or false. So the total number of ways in which you can assign values to the entries is 2^{2^n}. That yields A LOT of operators! Of course, half of them are negations of the others. More generally, any Boolean operation can be expressed by using AND and NOT operators (NOT and OR operators), so these are the only ones that you really need.

8. Are the following statements true or false?

   (a) All search algorithms are complete.
   (b) A^* search is complete.
   (c) A^* terminates immediately when a goal state is found.

   Solution:

   False. True. False.

9. [15 points] **Satisfiability:**
   For each of the following statements, state whether it is valid, satisfiable or unsatisfiable. If it is satisfiable, give a satisfying assignment of values to the variables.

   (a) $P \vee Q$
   Solution: Satisfiable; $P = 1$, $Q = 1$ is a satisfying assignment (I am using 1 for true and 0 for false).

(b) $P \implies P$

Solution: $P \implies P$ is the same as $\neg P \vee P = 1$. So the statement is valid.

(c) $(P \implies Q) \wedge (Q \implies R) \wedge (R \implies \neg P)$

Solution: Satisfiable; to do it, remember that false implies anything. So if $P = 0$, $Q = 0$, $R = 0$, then all three implications are true and the whole statement is true. You can check on your own that it is not valid (it is equivalent to $\neg P$).

(d) $(P \wedge Q) \vee (P \wedge \neg Q)$

Solution: Satisfiable. For diversity, let's do this by the truth table method:

P	Q	$P \wedge Q$	$P \wedge \neg Q$	$(P \wedge Q) \vee (P \wedge \neg Q)$
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

(e) $(P \vee Q) \wedge (P \vee \neg Q)$

Solution: Satisfiable; if $P = 1$ then obviously the expression is true. If $P = 0$, then it is false. Check that actually, $(P \vee Q) \wedge (P \vee \neg Q) = P$.

10. [15 points] Translate the following sentences into first-order logic:

(a) All students who take AI like to play games.

Solution:

$$\forall s, Student(s) \wedge Takes(s, AI) \rightarrow Likes(s, Games)$$

(b) No students who take AI like to play games.

Solution:

$$\neg(\exists s, Student(s) \wedge Takes(s, AI) \wedge Likes(s, Games))$$

which is equivalent to:

$$\forall s, \neg Student(s) \vee \neg Takes(s, AI) \vee \neg Likes(s, Games)$$

and to:

$$\forall s, (Student(s) \wedge Takes(s, AI)) \rightarrow \neg Likes(s, Games)$$

(c) On Saturday, all students either go to a party or work, but not both,

Solution: Using Sat for Saturday, we have:

$$\forall s, (Work(s, Sat) \wedge \neg Party(s, Sat)) \vee (\neg Work(s, Sat) \wedge Party(s, Sat))$$

(d) All students go to a party on Saturday, except those taking AI.

Solution:

$$\forall s, \neg Takes(s, AI) \rightarrow Party(s, Sat)$$

(e) Exactly two students go to a party.

Solution: I am assuming we are talking about Saturday still.

$$\exists x, \exists y, Party(x, Sat) \wedge Party(y, Sat) \wedge \neg(x = y) \wedge (\forall z, (\neg(z = x) \wedge \neg(z = y)) \rightarrow \neg Party(z, Sat))$$

11. **STRIPS notation:**

    Suppose we want to drive a car from Montreal (M) to Toronto (T). The key must be in the ignition in order to be able to drive the car. Initially we have the key in our pocket and we want the key to be there again at the end of the plan. In order to construct a plan, we have 4 operators: Drive(M) (drive to Montreal), Drive(T), Insert(Key) (put the key in the ignition), Remove(Key). We also have a few propositions to describe possible world characteristics: InPocket(Key), InIgnition(Key), At(Car, M), At(Car,T).

    Describe the four operators above in STRIPS notation. You will need to specify preconditions and postconditions (in terms of add and delete effects) for each operator.

    Solution:

```
Operator: Drive(M)
Preconditions: At(Car, T), InIgnition(Key)
Postconditions:
Add-list: At(Car, M)
Delete-list: At(Car, T)

Operator: Drive(T)
Preconditions: At(Car, M), InIgnition(Key)
Postconditions:
Add-list: At(Car, T)
Delete-list: At(Car, M)

Operator: Insert(Key)
Preconditions: InPocket(Key)
Postconditions:
Add-list: InIgnition(Key)
Delete-list: InPocket(Key)

Operator: Remove(Key)
Preconditions: InIgnition(Key)
Postconditions:
Add-list: InPocket(Key)
Delete-list: InIgnition(Key)
```

12. Russell and Norvig, pg. 412, problem 11.4.

    Solution:

    (a) Initial state description: *At(Monkey, A), At(Bananas, B), At(Box, C), Height(Monkey, Low), Height(Box, Low), Height(Bananas, High)*

    (b) STRIPS descriptions of the action:
    $Go(X, Y)$

    - Preconditions: $At(Monkey, X)$

- Postconditions: $\neg At(Monkey, X)$ (delete list), $At(Monkey, Y)$ (add list)

Note that if you want to be really precise, you can add a predicate Path(A,B) to the precondition (like in the book). That means you can go from A to B if there is a clear path between A and B. You can also add a precondition of $X \neq Y$ to ensure that you do not move to the same place.

$Push(Obj, X, Y)$

- Preconditions: $At(Monkey, X), At(Obj, X)$
- Postconditions: $\neg At(Monkey, X), \neg At(Obj, X)$ (delete list), $At(Monkey, Y), At(Obj, Y)$ (add list).

$Climb(Obj)$

- Preconditions: $At(Monkey, X), At(Obj, X), Height(Obj, Low), Height(Monkey, Low)$
- Postconditions: $\neg Height(Monkey, Low)$ (delete list), $Height(Monkey, High)$.

Note that we could add more preconditions and postconditions, e.g. to say that the monkey is initially on the floor, and after the action it is on top of the object.

$Grasp(Obj)$

- Preconditions: $At(Monkey, X), Height(Monkey, H), At(Obj, X), Height(Obj, H)$
- Postconditions: $Hold(Obj)$ (add list).

(c) The goal description in situation calculus would be:

$$(\exists s)(\exists p)(\exists x) PlanResult(p, s0) = s \wedge Hold(Bananas, s) \wedge At(Box, x, s) \wedge At(Box, x, s0)$$

This goal cannot be achieved using STRIPS-style planning, because STRIPS does not allow us to refer to situations in the past (such as the start state). In STRIPS, we always look at the *current situation* and when an action is taken, the situation is altered.

(d) This illustrates the qualification problem (there are many aspects that we could specify about an action, and our description does not capture all of them).

13. Motion planning

Consider a small square robot which can move in X-Y and a larger triangular obstacle. Draw the configuration-space transform, assuming that the reference point of the robot is in (a) the upper-right corner of the robot; (b) the upper-left corner; (c) the lower-left corner; (d) the lower-right corner; and (e) the center of the robot.