CS3061 – Artificial Intelligence
2018 Exam Solutions

Question 1

Recall that a goal node connected by arc to Node can be searched by calling frontierSeach([Node]), with the following Prolog clauses:

```
% Base case - Node is goal
frontierSearch([Node|_]) :- goal(Node).

frontierSearch([Node|Rest]) :-
  findall(Next, arc(Node,Next), Children),       % Find all children connected to node by arc
  add2frontier(Children, Rest, NewFrontier),     % Add nodes children to frontier
  frontierSearch(NewFrontier).                   % Search the new frontier
```

*a) Define add2frontier(Children, Rest, NewFrontier) so that frontierSearch([Node]) searches **depth-first** for a goal connected to Node by arc.*

```
% ---------------------------------------------------------------------------
%                   DEPTH FIRST SEARCH - add2frontier (LIFO)
% ---------------------------------------------------------------------------

% Base case, add empty node
add2frontier([], Rest, Rest),

add2frontier([H|T], Rest, [H|TRest]) :-     % Recursively add head of child
  add2frontier(T, Rest, TRest).
```

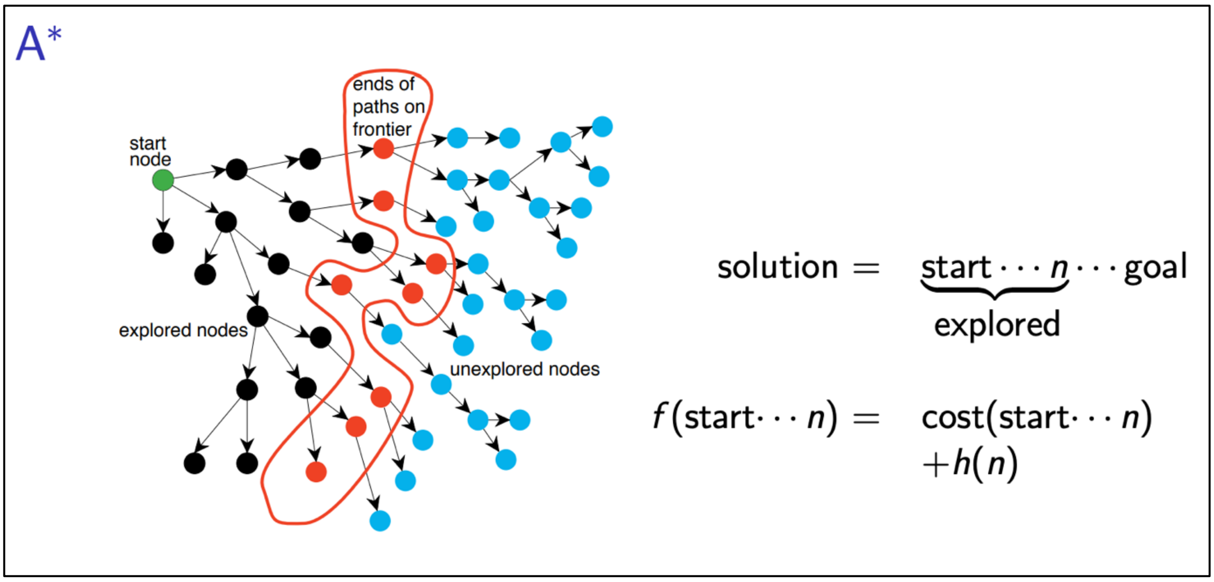
*b) Define add2frontier(Children, Rest, NewFrontier) so that frontierSearch([Node]) searches **breadth-first** for a goal connected to Node by arc.*

```
% ---------------------------------------------------------------------------
%                   BREADTH FIRST SEARCH - add2frontier (LIFO)
% ---------------------------------------------------------------------------

% Base case, add empty node
add2frontier(Children, [], Children).

add2frontier(Children, [H|T], [H|More]) :-    % Reduces Rest of frontier first
  add2frontier(Children, T, More).
```

c) What modifications to add2frontier(Children, Rest, NewFrontier) are required for A-star?

The modifications needed to add2frontier that are required for A-star search is that it must ensure:

$Frontier = [Head|Tail]$ *** where Head has minimal f ***



A* search takes into account a **heuristic value** h which estimates the cost of the cheapest path from the current node n to the target node.

*d) What does it mean for A-star to be **admissible**?*

A* is admissible (under cost, h) if it returns a solution of min cost whenever a solution exists i.e A* is fool-proof. A heuristic function is said to be admissible if it never overestimates the cost of reaching the goal i.e the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path.

e) Give three conditions sufficient for A to be admissible. Do these conditions guarantee that A* will terminate? Justify your answer.*

- **Underestimate:** All heuristics are less than the actual cost
- **Termination:** For some $\epsilon > 0$, every arc costs $\geq \epsilon$
- **Finite Branching:** Each node has finite set of arcs

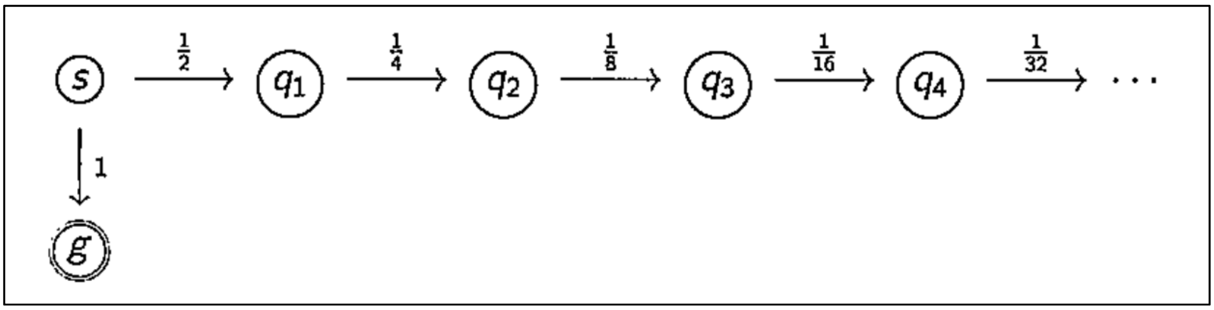
These conditions do guarantee that A* will terminate as if A* does not terminate it does not satisfy condition two and thus, is not admissible. If a solution cannot be found, A* search immediately terminates.

f) True or False: Breadth-first is admissible. Justify your answer

True. Breadth-first search is admissible because the algorithm will always find the shortest path (it might not be the optimal path, if arcs have different costs). Breath-first search will also always terminate and is considered complete.

However, depth-first search is not guaranteed to be admissible.

g) Suppose we were to apply A to the graph below:*



with arcs (s, g), (s, q_1) and (q_n, q_{n+1}) costing 1, $\frac{1}{2}$ and $\frac{1}{2^{n+1}}$ respectively

$$\begin{aligned} \mathrm{cost}(s, g) &= 1 \\ \mathrm{cost}(s, q_1) &= \frac{1}{2} \\ \mathrm{cost}(q_n, q_{n+1}) &= \frac{1}{2^{n+1}} \quad \text{for } n \geq 1 \end{aligned}$$

and heuristics $h(s) = 1$, $h(g) = h(q_n) = 0$. Assuming g is the only goal node, is A-star admissible under this set-up? Justify your answer.

A* search is based off of the cost decision function:

$$f(n) = g(n) + h(n)$$

Starting with start node *s* and goal node *g* we calculate f(n) for both arcs connected to *s*:

$$1)\ f(q_1) = g(q_1) + h(q_1)$$

$$= \frac{1}{5} + 0$$

$$2)\ f(g) = g(g) + h(g)$$

$$= 1 + 0$$

Since $f(q_1) < f(g)$, A* search will choose to explore the route through q_1 in search of g, as the heuristic values down this route are also decreasing and will never exceed 1 so it will continue to explore this infinite route and will not terminate. As a result A-star is not admissible under this set-up.

h)

> Suppose we were to change the cost of an arc (q_n, q_{n+1}) to $\frac{1}{n+2}$
>
> $$\text{cost}(q_n, q_{n+1}) \;=\; \frac{1}{n+2} \quad \text{for } n \geq 1.$$
>
> Would A-star be admissible if all other details of the set-up in part (g) were preserved. Justify your answer.

A* search is based off of the cost decision function:

$$f(n) = g(n) + h(n)$$

With the newly improved cost function it will only take 3 steps away from s in the direction of q for it to exceed the other explored route via g.

<u>First Exploration</u>

$$f(q_1) = g(q_1) + h(q_1)$$

$$= \frac{1}{2} + 0$$

$$f(g) = g(g) + h(g)$$

$$= 1 + 0$$

Choose to explore route to goal via q1 since $f(q_1) < f(g)$.

<u>Second Exploration</u>

$$f(q_2) = f(q_1) + g(q_2) + h(q_2)$$

$$= \left(\frac{1}{2}\right) + \frac{1}{3} + 0 = \frac{5}{6}$$

Choose to explore route to goal via q2 since $f(q_2) < f(g)$.

<u>Third Exploration</u>

$$f(q_3) = f(q_2) + g(q_3) + h(q_3)$$

$$= \left(\frac{5}{6}\right) + \frac{1}{4} + 0 = \frac{13}{12}$$

Now, since $f(q_3) > f(g)$, we revert back to our first explored branch $f(g)$ since this will provide us with a route of least cost to the goal node.

As a result of this change to the cost function A* does become admissible as it provides a solution of minimum cost to the the goal given that a route from the start node to the goal node exists.

Question 2

Let $\langle S, A, p, r, \gamma \rangle$ be a Markov decision process.

*a) What is a **policy**? Suppose S consists of three states, and A of two actions, how many possible policies are there?*

A policy is a **decision of what to do** at a given state. This is decided using the Markov decision process based on the probabilities returned from function p.

For

a_1	s_1	s_2	s_3
s_1	.5, 3	.3, 0	.2, −2
s_2	.3, 0	.5, 1	.2, 2
s_3	0, 0	0, 0	1, 1

a_2	s_1	s_2	s_3
s_1	.2, 4	.2, 2	.6, −3
s_2	.1, 1	0, 0	.9, −2
s_3	0, 0	0, 0	1, 0

*b) What is a **γ-optimal policy** and how is it computed from the **γ-discounted value** of a pair (s,a)? How are γ-discounted values computed by value iteration q_o , q_1 , q_2 ...*

A ***γ-optimal policy*** is a policy that chooses the best action to take for each state in the MDP. It is computed from the ***γ-discounted value*** of a pair (s,a) by exploring the best possible action to take by seeking the greatest expected reward. These rewards decay the further away from the current state they become at a decay factor of ***γ*** .

The ***γ-discounted value*** of a pair (s,a) is:

$$\lim_{n \to \infty} q_n(s, a)$$

Where:

$$q_0(s, a) = \sum_{s'} [p(s, a, s') * r(s, a, s')]$$

$$q_{n+1}(s, a) = \sum_{s'} \left[p(s, a, s') \left(r(s, a, s') + \gamma \left(\max_{a'} q_n(s', a') \right) \right) \right]$$

$$q_{n+1}(s, a) = q_0(s, a) + \sum_{s'} \left[p(s, a, s') * \gamma \left(\max_{a'} q_n(s', a') \right) \right]$$

For our given example we have:

$$q_0(s,a) = p(s,a,s_1)r(s,a,s_1) + p(s,a,s_2)r(s,a,s_2) + p(s,a,s_3)r(s,a,s_3)$$

$$V_n(s) = \max\left(q_n(s,a_1), q_n(s,a_2)\right)$$

$$q_{n+1}(s,a) = q_0(s,a) + \gamma[p(s,a,s_1)V_n(s_1) + p(s,a,s_2)V_n(s_2) + p(s,a,s_3)V_n(s_3)]$$

c) Compute $q_2(s_3, a_2)$ for:

$$S = \{s_1, s_2, s_3\}$$

$$A = \{a_1, a_2\}$$

$$\gamma = 0.1$$

$$q_0(s,a) = p(s,a,s_1)r(s,a,s_1) + p(s,a,s_2)r(s,a,s_2) + p(s,a,s_3)r(s,a,s_3)$$

$$V_n(s) = \max\left(q_n(s,a_1), q_n(s,a_2)\right)$$

$$q_{n+1}(s,a) = q_0(s,a) + \gamma[p(s,a,s_1)V_n(s_1) + p(s,a,s_2)V_n(s_2) + p(s,a,s_3)V_n(s_3)]$$

$$q_{1+1}(s_3,a_2) = q_0(s_3,a_2) + \gamma[p(s_3,a_2,s_1)V_1(s_1) + p(s_3,a_2,s_2)V_1(s_2) + p(s_3,a_2,s_2)V_1(s_3)]$$

$$q_0(s_1,a_1) = p(s_1,a_1,s_1)r(s_1,a_1,s_1) + p(s_1,a_1,s_2)r(s_1,a_1,s_2) + p(s_1,a_1,s_3)r(s_1,a_1,s_3)$$

$$= (0.5)(3) + (0.3)(0) + (0.2)(-2)$$

$$= 1.9$$

$$q_0(s_1,a_2) = p(s_1,a_2,s_1)r(s_1,a_2,s_1) + p(s_1,a_2,s_2)r(s_1,a_2,s_2) + p(s_1,a_2,s_3)r(s_1,a_2,s_3)$$

$$= (0.2)(4) + (0.2)(2) + (0.6)(-3)$$

$$= -0.6$$

$$V_0(s_1) = \max\left(q_0(s_1,a_1), q_0(s_1,a_2)\right)$$

$$= \max(1.9, -0.6)$$

$$= 1.9$$

$$q_0(s_2,a_1) = p(s_2,a_1,s_1)r(s_2,a_1,s_1) + \ p(s_2,a_1,s_2)r(s_2,a_1,s_2) + \ \ p(s_2,a_1,s_3)r(s_2,a_1,s_3)$$

$$= (0.3)(0) + (0.5)(1) + (0.2)(2)$$

$$= 0.9$$

$$q_0(s_2,a_2) = p(s_2,a_2,s_1)r(s_2,a_2,s_1) + \ p(s_2,a_2,s_2)r(s_2,a_2,s_2) + \ \ p(s_2,a_2,s_3)r(s_2,a_2,s_3)$$

$$= (0.1)(1) + (0)(0) + (0.9)(-2)$$

$$= -1.7$$

$$V_0(s_2) = \max\,(q_0(s_2,a_1), q_0(s_2,a_2))$$

$$= \max(0.9, -1.7)$$

$$= 0.9$$

$$q_0(s_3,a_1) = p(s_3,a_1,s_1)r(s_3,a_1,s_1) + \ p(s_3,a_1,s_2)r(s_3,a_1,s_2) + \ \ p(s_3,a_1,s_3)r(s_2,a_1,s_3)$$

$$= (0)(0) + (0)(0) + (1)(1)$$

$$= 1$$

$$q_0(s_3,a_2) = p(s_3,a_2,s_1)r(s_3,a_2,s_1) + \ p(s_3,a_2,s_2)r(s_3,a_2,s_2) + \ \ p(s_3,a_2,s_3)r(s_3,a_2,s_3)$$

$$= (0)(0) + (0)(0) + (1)(0)$$

$$= 0$$

$$V_0(s_3) = \max\,(q_0(s_3,a_1), q_0(s_3,a_2))$$

$$= \max(1, 0)$$

$$= 1$$

$$q_1(s_1,a_1) = q_0(s_1,a_1) + \gamma[p(s_1,a_1,s_1)V_0(s_1) + p(s_1,a_1,s_2)V_0(s_2) + p(s_1,a_1,s_3)V_0(s_3)]$$

$$= 1.9 + 0.1[(0.5)(1.9) + (0.3)(0.9) + (0.2)(1)]$$

$$= 2.042$$

$$q_1(s_1,a_2) = q_0(s_1,a_2) + \gamma[p(s_1,a_2,s_1)V_0(s_1) + p(s_1,a_2,s_2)V_0(s_2) + p(s_1,a_2,s_3)V_0(s_3)]$$

$$= -0.6 + 0.1[(0.2)(1.9) + (0.2)(0.9) + (0.6)(1)]$$

$$= -0.484$$

$$V_1(s_1) = \max(q_1(s_1,a_1), q_1(s_1,a_2))$$

$$= \max(2.042, -0.484)$$

$$= 2.042$$

$$q_1(s_2,a_1) = q_0(s_2,a_1) + \gamma[p(s_2,a_1,s_1)V_0(s_1) + p(s_2,a_1,s_2)V_0(s_2) + p(s_2,a_1,s_3)V_0(s_3)]$$

$$= 0.9 + 0.1[(0.3)(1.9) + (0.5)(0.9) + (0.2)(1)]$$

$$= 1.022$$

$$q_1(s_2,a_2) = q_0(s_2,a_2) + \gamma[p(s_2,a_2,s_1)V_0(s_1) + p(s_2,a_2,s_2)V_0(s_2) + p(s_2,a_2,s_3)V_0(s_3)]$$

$$= -1.7 + 0.1[(0.1)(1.9) + (0)(0.9) + (0.9)(1)]$$

$$= -1.591$$

$$V_1(s_2) = \max(q_1(s_2,a_1), q_1(s_2,a_2))$$

$$= \max(1.022, -1.591)$$

$$= 1.022$$

$$q_1(s_3, a_1) = q_0(s_3, a_1) + \gamma[p(s_3, a_1, s_1)V_0(s_1) + p(s_3, a_1, s_2)V_0(s_2) + p(s_3, a_1, s_3)V_0(s_3)]$$

$$= 1 + 0.1[(0)(1.9) + (0)(0.9) + (1)(1)]$$

$$= 1.1$$

$$q_1(s_3, a_2) = q_0(s_3, a_2) + \gamma[p(s_3, a_2, s_1)V_0(s_1) + p(s_3, a_2, s_2)V_0(s_2) + p(s_3, a_2, s_3)V_0(s_3)]$$

$$= 0 + 0.1[(0)(1.9) + (0)(0.9) + (1)(1)]$$

$$= 1$$

$$V_1(s_3) = \max\left(q_1(s_3, a_1), q_1(s_3, a_2)\right)$$

$$= \max(1.1, 1)$$

$$= 1$$

$$q_2(s_3, a_2) = q_0(s_3, a_2) + \gamma[p(s_3, a_2, s_1)V_1(s_1) + p(s_3, a_2, s_2)V_1(s_2) + p(s_3, a_2, s_3)V_1(s_3)]$$

$$= 0 + 0.1[0(2.042) + 0(1.022) + 1(1)]$$

$$= 0.1$$

*d) How can we learn the the **γ-discounted value** when we do not know the probabilities p and immediate rewards r?*

We can learn the ***γ-discounted value*** by reinforcement learning or q-learning. For this purpose it is useful to define another function, which corresponds to taking the action a and then continuing optimally (or according to whatever policy one currently has):

$$Q(s, a) = \sum_{s'} p(s, a, s')\left(r(s, a, s') + \gamma V(s')\right)$$

While this function is also unknown, the experience during learning is based on (s,a) pairs (together with the outcome s'; that is "I was in state s and I tried doing a and s' happened"). Thus, one has an array Q and uses experience to update it directly. This is known as Q-learning.

e) The exploration-exploitation trade-off in c) and how can we adjust the notion of a policy to accommodate the trafe-off.

The **exploration-exploitation** trade-off in c) is that if we need to decide when it is best to explore and when it is best to exploit our knowledge of the environment.

To adjust our policy we can introduce the notion of a **epsilon greedy strategy**. With this strategy we define an exploration rate $\boldsymbol{\varepsilon}$ which we initially set to 1. The exploration rate is the probability that our agent will choose to explore the environment rather than exploit it's knowledge. As the agent learns more about the environment, at the start of each new episode $\boldsymbol{\varepsilon}$ will decay by some rate that we set so that the likelihood of exploration becomes less and less probable as the agent learns more about its environment.

To determine whether the agent will explore or exploit, choose a random number between 0 and 1 and compare it to $\boldsymbol{\varepsilon}$.

Question 3

a) i) What is a definite clause?

A **definite clause** is a Horn clause with **exactly one** positive literal. An example of a definite clause is as follow:

- $\neg p \ \vee q$

ii) What is a Horn clause?

A **Horn clause** is a disjunction of literals with **at most one** positive literal. A horn clause without a positive literal is called a goal clause. There are three different types of horn clauses:

	Disjunction form	Implication form	Read intuitively as
Definite clause	$\neg p \vee \neg q \vee \ldots \vee \neg t \vee u$	$u \leftarrow p \wedge q \wedge \ldots \wedge t$	assume that, if p and q and ... and t all hold, then also u holds
Fact	u	u	assume that u holds
Goal clause	$\neg p \vee \neg q \vee \ldots \vee \neg t$	$false \leftarrow p \wedge q \wedge \ldots \wedge t$	show that p and q and ... and t all hold [note 1]

iii) True or false: Every set of definite clauses is satisfiable. Justify your answer.

It is always possible to find a model for a set of definite clauses. The interpretation with all atoms true is a model of any set of definite clauses. Thus, **a definite-clause knowledge base is always satisfiable**. However, a set of Horn clauses can be unsatisfiable.

Example 5.18: The set of clauses {a, false←a} is unsatisfiable. There is no interpretation that satisfies both clauses. Both a and false←a cannot be true in any interpretation.

iv) Outline an algorithm to determine whether a set of Horn clauses is satisfiable.

Horn satisfiability (HORNSAT) is a P-complete problem meaning that it can be solved in linear time. A **horn clause** is a clause with at most one positive literal and any number of negative literals.

```
begin
  let S = {C1, ..., Cm}, where A = C1 ^ ... ^ Cm
  consistent = true;
  change = true;

  for each propositional letter P in A do
    V(P) = false;
  endfor;

  for each P such that (P) is a basic Horn formula in A do
    V(P) = true
  endfor;

  while change and consistent do
    change = false;

    for each basic Horn formula C in S and consistent do
      // This condition finds false
      if C is of the form ¬P1 v ... v ¬Pm and V(P1) = ... = V(Pm) = true then
        consistent = false;
      // This condition verifies truth
      else if C is of the form ¬P1 v ... v ¬Pm v P and V(P1) = ... = V(Pm) = true and V(P) = false then
        V(P) = true;
        change = true;
        // Remove horn formula from set
        S = S-{C}
    endfor;
  endwhile;
```

b) True or false: A set of KB clauses is satisfiable if and only if the atom false is a logical consequence of KB. Justify your answer, stating what it means for a cause to be a logical consequence of KB.

A query against a KB is a **logical consequence** if the answer to the query is "yes". A query against a KB is not a logical consequence if the answer to the query is "no".

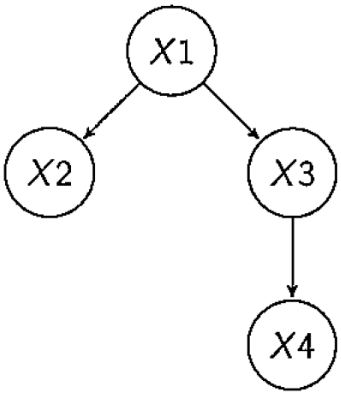
Therefore, in order for a set of KB clauses to be satisfiable the atom false **can be a logical consequence** of KB however, **must not be a logical consequence** of the set of KB clauses in question.

c)

(c) Given the Bayes net and probabilities below for the Boolean variables $X1, X2, X3, X4$ (with negations $\overline{X1}, \overline{X2}, \overline{X3}, \overline{X4}$), calculate the probabilities in (i), (ii) and (iii).

$P(X1) = 0.3$

$P(X2|X1) = 0.7$
$P(X2|\overline{X1}) = 0.5$



$P(X3|X1) = 0.2$
$P(X3|\overline{X1}) = 0.6$

$P(X4|X3) = 0.6$
$P(X4|\overline{X3}) = 0.6$

i) Calculate P(X1|X2):

Using Bayes rule this can be solved as:

$$P(X1|X2) = \frac{P(X2|X1)P(X1)}{P(X2)}$$

$$P(X1|X2) = \frac{0.7 * 0.3}{P(X2)}$$

We can then solve P(X2) using **marginalisation:**

$$P(X2) = P(X2|X1)P(X1) + P(X2|!X1)P(!X1)$$

$$P(X2) = (0.7)(0.3) + (0.5)(1 - 0.3)$$

$$P(X2) = 0.56$$

And then, plugging this back in we get:

$$P(X1|X2) = \frac{0.7 * 0.3}{0.56}$$

$$P(X1|X2) = 0.375$$

ii) Calculate P(X3|X2):

Since X3 and X2 are not connected within the network they can be considered independent events and thus the probability of X3 given X2 is 0 as they hold no influence over one another.

iii) Calculate P(X3|X4):