CS 188 Fall 1997 | Introduction to AI Stuart Russell | **Midterm Solutions**

1. **(18 pts.) True/False**

   (a) (3) True. Although the KB is a memory, the agent could use it just to hold reflex rules (p.202).

   (b) (3) True. It won't return if there is no goal, but completeness doesn't require this—nor can any algorithm guarantee it.

   (c) (3) False. For a single diagonal move, Manhattan distance is 2.

   (d) (3) True. The evaluation function runs minimax to completion.

   (e) (3) True. $P \wedge Q \Rightarrow R$.

   (f) (3) False. The sentence is false whenever P is true sometimes but not always.

2. **(12 pts.) Logical knowledge representation**

   (a) (6) $\forall x\; French(x) \wedge Wine(x) \Rightarrow [\forall y\; Chilean(y) \wedge Wine(y) \Rightarrow Price(x) > Price(y)]$

   (b) (6) $\exists x\; Wine(x) \wedge Chilean(x) \wedge [\forall y\; Chilean(y) \wedge Wine(y) \Rightarrow \neg(Quality(y) > Quality(x))] \wedge [\exists z\; French(z) \wedge Wine(z) \wedge Quality(x) > Quality(z)]$

3. **(18 pts.) Logical Inference**

   (a) (3) Show that a contradiction follows from CNF($\neg\alpha$).

   (b) (4) No. If it's valid, resolution with CNF($\neg\alpha$) terminates. If it's unsatisfiable, resolution with CNF(α) terminates. If it's just satisfiable, neither may terminate.

   (c) (8) $\forall x\; [\forall y\; P(x,y)] \Rightarrow Q(x) = \forall x\; \neg[\forall y\; P(x,y)] \vee Q(x) = \forall x\; [\exists y\; \neg P(x,y)] \vee Q(x)$
   $= \forall x\; \exists y\; \neg P(x,y) \vee Q(x)$
   $= \neg P(x, f(x)) \vee Q(x)$
   and $\forall x\; \exists y\; [P(x,y) \Rightarrow Q(x)] = \forall x\; \exists y\; [\neg P(x,y) \vee Q(x)] = \neg P(x, f(x)) \vee Q(x)$

   (d) (3) $Hates(x,y)$ means person x hates person y. $Q(x)$ means x is a misanthrope.

4. **(10 pts.) Situation calculus**

   The successor-state axiom is for the predicate $Value$: a value gets into a register either if an assignment copied it in, or it was already there and nothing overwrote it with a different value:

$$\begin{aligned}\forall\, a, v, x, s\;\; Value(x, v, Result(a, s)) \;\; \Leftrightarrow &\;\; [(a = Assign(x, y) \wedge Value(y, v, s)) \\ \vee &\;\; (Value(x, v, s) \wedge (\neg\exists\, y, v'\;\; Value(y, v', s) \wedge v \neq v' \wedge a = Assign(x, y)))]\end{aligned}$$

   The effect axiom is much closer to the STRIPS operator:

$$\begin{aligned}&\forall\, x, y, s, v_1, v_2\;\; Value(x, v_1, s) \wedge Value(y, v_2, s) \Rightarrow \\ &Value(x, v_2, Result(Assign(x, y), s)) \wedge \neg Value(x, v_1, Result(Assign(x, y), s))\end{aligned}$$

   but we also need a frame axiom saying that all other values are unchanged:

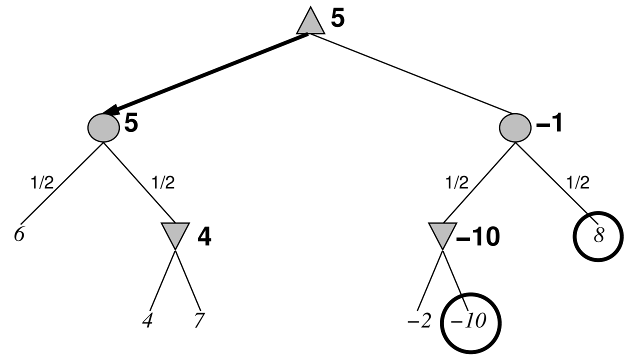
$$\forall\, x, y, z, s, v\;\; Value(z, v, s) \wedge z \neq x \Rightarrow Value(z, v, Result(Assign(x, y), s))$$

5. **(22 pts.) Game-playing**

   (a) (6) This just requires maxing at the max nodes, mining at the min nodes, and averaging at the chance nodes. The MAX player will take the left-hand branch:

(b) (10) Alpha-beta pruning works whenever it is possible to show that the node about to be evaluated or expanded cannot possibly affect the top-level move choice. Without bounds on leaf values, the value of a chance node is also unbounded until all its children have known values, so pruning cannot occur. With bounds, it can be the case that even if the remaining unevaluated leaves have the worst/best possible value, the chance node's value would not be high/low enough to cause a change in move. In the above tree, once the -2 leaf is found, the min node is worth *at most* -2, so the chance node is worth *at most* 4 (because it's right child cannot be worth more than 10), so the top-level move choice cannot be affected by either of the circled leaf values.

(c) (6) There is always a possibility that the unevaluated leaf has value $[\infty, \infty]$—that is, both players want to go there. Finding such a leaf would clearly change the top-level move, since the top-level player would aim to reach that node and can be sure that the othen player will cooperate. SO, no leaves can be pruned.



6. (20 pts.) Planning

(a) (3) The open conditions are those not yet established: $Have(S)$ on *GetGoat* and $Have(C)$ on ASM.

(b) (4) The *GetGoat* step clobbers the causal link for $Have(S)$ from ASM to $Finish$.

(c) (4) *GetGoat* cannot come after $Finish$ so must come before ASM.

(d) (6) $Have(C)$ can be satisfied by a link from $Start$. To satisfy $Have(S)$, we must add another ASM step. The ASM step clobbers nothing, and its precondition is also satisfied by a link from $Start$.

(e) (3) This is just the conditions protected by causal links in force at that point: $Have(S)$ and $Have(C)$