CS 188 Spring 2004 — Introduction to AI — Stuart Russell — Midterm Solution

1. (12) Agents and Environments

   (a) (3) True/False There exist task environments (PEAS) in which some pure reflex agents behave rationally.
   TRUE: For assignment A1 you examined a pure reflex vacuum-cleaner agent that was rational for its two-square task environment. In principle, any fully observable environment can be handled by a pure reflex agent.

   (b) (3) True/False There exist task environments (PEAS) in which all pure reflex agents behave irrationally.
   TRUE: See examples on p.48 of AIMA2e.

   (c) (3) True/False The input to an agent program is the same as the input to the corresponding agent function.
   FALSE: the agent program takes the current percept, whereas the agent function takes the percept history.

   (d) (3) True/False Every agent function is implementable by some program/machine combination
   FALSE: In lecture, the example of the halting problem was given.[1] There may also be functions that are computable but cannot be computed in any practical amount of time on any feasible computer, but as yet we have no logically necessary bounds on what "feasible" means.

2. (15) Search
   Consider the problem of moving k knights from k starting squares $s_1, \ldots, s_k$ to k goal squares $g_1, \ldots, g_k$, on an unbounded chessboard, subject to the rule that no two knights can land on the same square at the same time. Each action consists of moving *up to* k knights simultaneously. We would like to complete the maneuver in the smallest number of actions.

   (a) (5) What is the maximum branching factor b in this state space?
   (i) $8k$ (ii) $9k$ (iii) 8^k (iv) 9^k
   (iv). For each of k knights, we have one of 8 moves in addition to the possibility of not moving at all; each action executes one of these 9 choices for each knight (unless some choices conflict by landing on the same square), so there are 9^k actions. We gave partial credit for choosing (ii) or (iii).

   (b) (6) Suppose h_i is an admissible heuristic for the problem of moving knight i to goal g_i by itself. Which of the following heuristics are admissible for the k-knight problem?
   (i) $\min\{h_1, \ldots, h_k\}$ (ii) $\max\{h_1, \ldots, h_k\}$ (iii) $\sum_{i=1}^{k} h_i$
   (i) and (ii). If the h_i were exact, (ii) would be exact for the relaxed problem where knights can land on the same square. The h_i are admissible and hence no larger than the exact values, so (ii) is admissible. (i) is no larger than (ii), so (i) is admissible. (iii) is not admissible. For example, if each g_i is one move from its s_i, then (iii) returns k whereas the optimal solution cost is 1.

   (c) (4) Which of these is the best heuristic?
   (i) $\min\{h_1, \ldots, h_k\}$ (ii) $\max\{h_1, \ldots, h_k\}$ (iii) $\sum_{i=1}^{k} h_i$
   (ii) dominates (i) so it must be as good or better. (iii) is probably of little value since it isn't admissible and completely ignores the capability for parallel moves.

[1] Imagine a lisp function HALT that takes two inputs, a lisp function FUN and the arguments for that lisp function ARGS, and (HALT FUN ARGS) returns T if (FUN ARGS) terminates after some finite time and NIL otherwise. Turing showed that while such a function is easy to describe, there is no way to compute it, that is, no agent program can implement the HALT function.

Here is an outline that gives some intuition as to why HALT can't be computed. Suppose HALT does exist, we can then define a second lisp function (DEFUN PARADOX (S) (IF (HALT S S) (PARADOX S) T)). The function PARADOX returns if (S S) terminates and goes into infinite recursion when (S S) doesn't terminate.

Now we ask ourselves, what does (HALT #'PARADOX #'PARADOX) return, T or NIL? Suppose it returns T, then the IF conditional in PARADOX is true so PARADOX goes into infinite recursion. This means that PARADOX actually doesn't halt when fed itself as input, contradicting our assumption that (HALT #'PARADOX #'PARADOX) returns T. Suppose instead that (HALT #'PARADOX #'PARADOX) returns NIL, then PARADOX called on itself would terminate, returning T, so in fact HALT can't return T either. Since PARADOX is a well defined function, it must be that HALT cannot exist.

3. (25) CSPs and local search
Consider the problem of placing k knights on an $n \times n$ chessboard such that no two knights are attacking each other, where k is given and $k \leq n^2$.

   (a) (5) Choose a CSP formulation. In your formulation, what are the variables?
   Solution A: There is a variable corresponding to each of the n^2 positions on the board.
   Solution B: There is a variable corresponding to each knight.

   (b) (5) What are the values of each variable?
   Solution A: Each variable can take one of two values, {occupied,vacant}
   Solution B: Each variable's domain is the set of squares.

   (c) (5) What sets of variables are constrained, and how?
   Solution A: every pair of squares separated by a knight's move is constrained, such that both cannot be occupied. Furthermore, the entire set of squares is constrained, such that the total number of occupied squares should be k.
   Solution B: every pair of knights is constrained, such that no two knights can be on the same square or on squares separated by a knight's move. Solution B may be preferable because there is no global constraint, although Solution A has the smaller state space when k is large.

   (d) (5) Now consider the problem of putting *as many knights as possible* on the board without any attacks. We will solve this using local search. Briefly describe in English a sensible successor function.
   Any solution must describe a *complete-state* formulation because we are using a local search algorithm. For simulated annealing, the successor function must completely connect the space; for random-restart, the goal state must be reachable by hillclimbing from some initial state. Two basic classes of solutions are:
   Solution C: ensure no attacks at any time. Actions are to remove any knight, add a knight in any unattacked square, or move a knight to any unattacked square.
   Solution D: allow attacks but try to get rid of them. Actions are to remove any knight, add a knight in any square, or move a knight to any square.

   (e) (5) Briefly describe in English a sensible objective function.
   An objective function returns a number describing the desirability of the state. The key requirement is that the objective function must have its global optimum at the optimal solution (here, we are maximizing):
   Solution C: the number of knights placed on the board. Since all states have no attacks, the global optimum of this function is in fact the optimal solution.
   Solution D: Here we need to penalize for attacks. One might suggest maximizing #knights - #attacks, but one must be careful to avoid the possibility that the score can be improved by adding lots more knights at the cost of a few extra attacks. One can show that #knights - 2#attacks works.

4. (12) Propositional logic
Consider a propositional language with four symbols, A, B, C, and D. How many models are there for each of the following sentences?

   (a) (4) $B \lor C$
   12 (3 for B and C, times four for all combinations of A and D values.)

   (b) (4) $\neg A \lor \neg B \lor \neg C \lor \neg D$
   15 (can only be falsified by makin A, B, C, D true)

   (c) (4) $(A \Rightarrow B) \land A \land \neg B \land C \land D$
   0 (the first three conjuncts are inconsistent)

5. (18) Propositional logic
According to political pundits, a person who is radical (R) is electable (E) if he/she is conservative (C), but otherwise is not electable.

(a) (12) Which of the following are correct representations of this assertion?

i. $(R \wedge E) \iff C$
No, a conservative doesn't have to be radical.

ii. $R \Rightarrow (E \iff C)$
Yes, if a person is a radical then they are electable if and only if they are conservative.

iii. $R \Rightarrow ((C \Rightarrow E) \vee \neg E)$
No, this is equivalent to $\neg R \vee \neg C \vee E \vee \neg E$ which is a tautology, true under any assignment.

We decided that this question did not deserve 12 points, so we gave 6 for free and 6 for the answers.

(b) (6) Which of the sentences in (a) can be expressed in Horn form?

(i) Yes

$$\begin{aligned}(R \wedge E) \iff C &\equiv ((R \wedge E) \Rightarrow C) \wedge (C \Rightarrow (R \wedge E))\\ &\equiv ((R \wedge E) \Rightarrow C) \wedge (C \Rightarrow R) \wedge (C \Rightarrow E)\end{aligned}$$

(ii) Yes

$$\begin{aligned}R \Rightarrow (E \iff C) &\equiv R \Rightarrow ((E \Rightarrow C) \wedge (C \Rightarrow E))\\ &\equiv \neg R \vee ((\neg E \vee C) \wedge (\neg C \vee E))\\ &\equiv (\neg R \vee \neg E \vee C) \wedge (\neg R \vee \neg C \vee E))\end{aligned}$$

(iii) Yes, e.g., $True \Rightarrow True$.

6. (18) First-Order Logic

(a) (12) "No two adjacent countries have the same color" can be translated as

i. $\forall x, y\ \neg Country(x) \vee \neg Country(y) \vee \neg Adjacent(x, y) \vee \neg(Color(x) = Color(y))$.

ii. $\forall x, y\ (Country(x) \wedge Country(y) \wedge Adjacent(x, y) \wedge \neg(x = y)) \Rightarrow \neg(Color(x) = Color(y))$.

iii. $\forall x, y\ Country(x) \wedge Country(y) \wedge Adjacent(x, y) \wedge \neg(Color(x) = Color(y))$

iv. $\forall x, y\ (Country(x) \wedge Country(y) \wedge Adjacent(x, y)) \Rightarrow Color(x \neq y)$.

(i) is a correct translation; as stated in class, there is no need to require $\neg(x = y)$ because we know that no country is adjacent to itself.
(ii) is also a correct translation, but the $\neg(x = y)$ is unnecessary.
(iii) is incorrect, it asserts that all x are countries (as well as some other things).
(iv) is syntactically malformed.

(b) (6) Which of the following are valid sentences?

i. $(\exists x\ x = x) \Rightarrow (\forall y\ \exists z\ y = z)$.
Valid. The premise is true for any model in which there is an object. (In lecture, it was stressed that this is *always* the clase for FOL, but we put it in as a premise just in case.) Since there is an object in the model, then for every y we can find a z (namely, the object referred to by y) that is the same as y.

ii. $\forall x\ P(x) \vee \neg P(x)$.
Valid. For any given x, $P(x) \vee \neg P(x)$ is a propositional tautology.

iii. $\forall x\ Smart(x) \vee (x = x)$.
Valid. For every x, $x = x$ is true, so the disjunction is true.