Artificial Intelligence 1

Past exam questions

Dr Sean B Holden, 2010

1 A short historical note...

There is usually some degree of confusion as to precisely which past exam questions are relevant to this course. Allow me to explain. Prior to 2002 this course was essentially a Prolog programming course, and much of what it now contains was deferred to AI2. The syllabuses changed when I took over the courses in 2002. Consequently, many of the exam questions prior to 2002 are not relevant, although *some* of those for AI2 (which at the time was just called *Artificial Intelligence*) are. The past exam questions mentioned in what follows are the ones that remain relevant to AI1. Some of the older ones *may* mention subjects not at present in the syllabus. Those items can safely be ignored. Finally, the LaTeX files for some of the earlier questions are no longer available so they are not included here. However, copies of the questions can be found in the usual place:

`www.cl.cam.ac.uk/teaching/exams/pastpapers/`

2 Introduction and Agents

There are no past exam questions that continue to be relevant to this part of the course. You *might* at a stretch count the following.

2001, Paper 8, question 8:

Can a computer think? [20 marks]

3 Search

2004, paper 5, question 6:

1. Describe the way in which a problem should be represented in order to allow its solution using a *heuristic search* technique. [5 marks]
2. Define what it means for a search algorithm to be *complete*, and to be *optimal*. [2 marks]

3. Define what it means for a heuristic function to be *admissible*, and to be *monotonic*. [2 marks]
4. Describe the operation of the $A^\star$ heuristic search algorithm. [5 marks]
5. Prove that the $A^\star$ heuristic search algorithm is optimal when applied in conjunction with a monotonic heuristic. State the conditions under which the algorithm is also complete, and explain why this is the case. [6 marks]

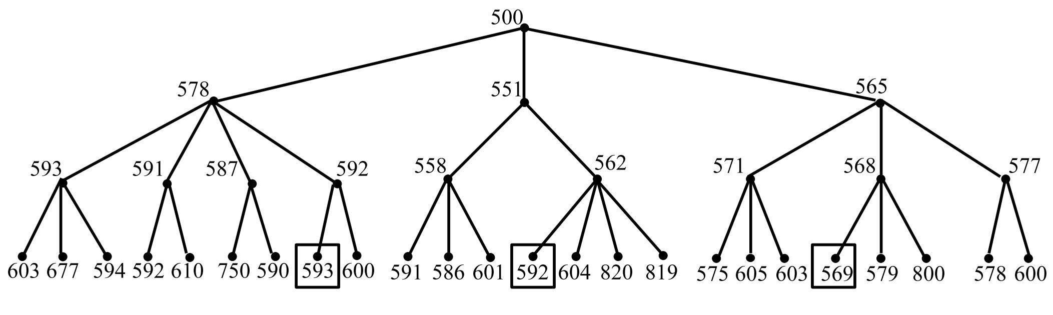
2007, paper 3, question 8:

We have a basic search problem, consisting of a set S of states, a start state s_0, a goal test $G(s)$ which returns True if $s \in S$ is a goal and False otherwise, and a function $\exp(s)$ which returns the set of states obtained by expanding state s.

1. Describe in detail the *Graph Search* algorithm for solving a problem of this type. How does it differ from the related *Tree Search* algorithm? [8 marks]
2. Give a detailed description of the *Recursive Best First* search algorithm, and explain why it might be used in preference to the $A^\star$ algorithm. [12 marks]

2008, paper 3, question 7:

1. Give a general description of the operation of the *Recursive Best-First Search (RBFS)* algorithm. [6 marks]
2. Consider the following search tree.

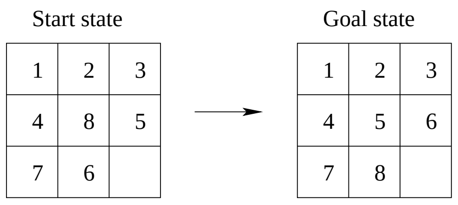


The numbers by the nodes denote the sum of some path cost and heuristic. The boxed nodes are goals. Describe in detail the way in which the RBFS algorithm searches this tree. Your answer should indicate the order in which nodes are expanded, the reason that this order is used, and should state which of the three goals is found and why. Note that smaller numbers represent more desirable nodes. [12 marks]

3. What shortcoming of the A* algorithm does the RBFS algorithm address, and how does it achieve this? [2 marks]

2009, paper 4, question 3:

1. Explain the difference between *uninformed* and *informed* search. List *two* examples of each type of algorithm. [2 marks]
2. In the context of planning, describe what a *heuristic* is and what it means for it to be *admissible*. List *two* examples of typical heuristic functions. [Hint: consider the problem in part (*d*) below.] [2 marks]
3. Explain what A* search is, including the advantages and disadvantages with respect to its theoretical properties. [3 marks]
4. Draw a search tree for the 8-puzzle problem up to depth 4 (start state is depth 0) using the A* algorithm (omit repeated states) with the evaluation function $f(n) = p(n)+h(n)$, where $p(n)$ is the number of steps from the start state (start state is step 0) and $h(n)$ is the number of misplaced tiles. Note that the actions for sliding tiles should be used in this order: right, left, up and down. Write the values of f and of its components p and h under each state. You may use an abbreviated notation indicating only the tiles that change. [10 marks]



5. Briefly explain IDA* search and its advantages and disadvantages. What happens when using IDA* in the search problem in part (*d*) if the IDA* limit is 3? What happens if the limit is 4 (in terms of number of states)? [3 marks]

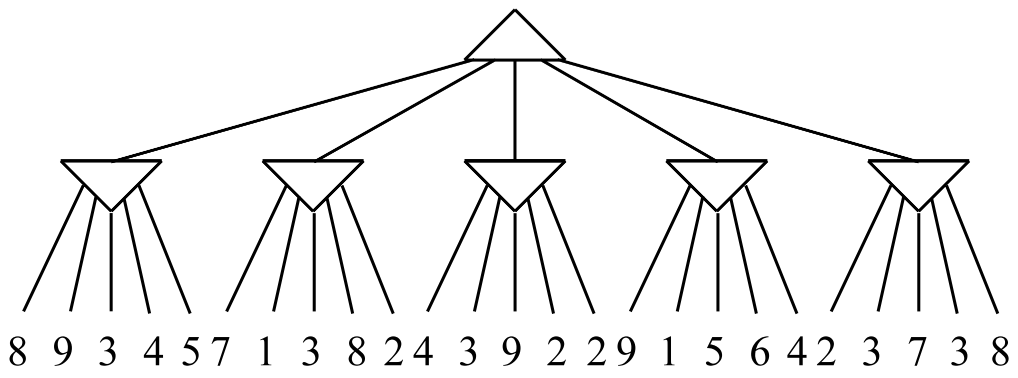
1994, paper 8, question 8:

See web site.

4 Games

2006, paper 3, question 3:

1. Give a detailed description of the *minimax algorithm* for two-player games, illustrating your answer using the following game tree.



[10 marks]

2. Describe the modifications required to the minimax algorithm in order to apply it to realistic games. [5 marks]
3. Give a detailed description of the technique of $\alpha - \beta$ pruning, again illustrating your answer using the game tree above. [5 marks]

2001, paper 9, question 8:

See web site.

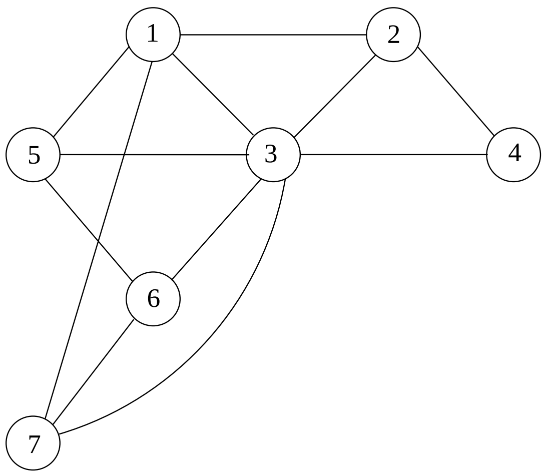
1997, paper 8, question 8:

See web site.

5 Constraint satisfaction problems

2010, paper 4, question 1:

This question addresses the problem of colouring the following graph using a *constraint satisfaction* approach.



The colours available are amber, black and crimson which we will denote by A, B and C respectively. We want to assign a colour to each node in the graph in such a way that if there is an edge (n_1, n_2) between any pair of nodes then n_1 and n_2 have different colours.

1. Explain how this problem can be represented as a constraint satisfaction problem. [2 marks]
2. Explain how we can apply *forward checking* in the process of solving a constraint satisfaction problem. Illustrate your answer using the above graph colouring problem where the initial steps are, in order, $1 = A$, $2 = B$, $6 = B$, $5 = C$. In particular, you should show how the process can *reduce branching* and *induce backtracking*. [8] marks]
3. Explain how we can apply *constraint propagation* using *arc consistency* in the process of solving a constraint satisfaction problem. Illustrate your answer using the same initial steps in the same order. Determine whether or not backtracking occurs earlier in this case and explain why. [10 marks]

2005, paper 3, question 3:

1. What are the advantages and disadvantages of *constraint satisfaction problems* (CSPs) compared with search algorithms such as $A^\star$ search *etc.* [3 marks]
2. Give a general definition of a CSP. Define the way in which a solution is represented and what it means for a solution to be *consistent* and *complete*. [5 marks]
3. Assuming discrete binary constraints and finite domains explain how breadth-first-search might be used to find a solution and why this is an undesirable approach. [3 marks]
4. Give a brief description of the basic *backtracking algorithm* for finding a solution. [4 marks]

5. Describe the *minimum remaining values heuristic*, the *degree heuristic* and the *least constraining value heuristic*. [5 marks]

6 Knowledge representation and reasoning

2010, paper 4, question 2:

Evil Robot's creator has sent him on a mission. He must go to the Secret Mountain Hideout, put on an orange boiler suit so that he blends in, then pick up two items called Component 1 and Component 2 and join them together. Finally, he has to press the BIG RED BUTTON (which only works when the two components are joined together) in order to cause something horrible to happen. Evil Robot's internal systems have been constructed using the *situation calculus* and a theorem prover.

1. Give a brief outline of the *situation calculus*, concentrating on the fundamental elements that you'd expect to see independently of any specific problem. [4 marks]
2. Suggest three logical formulae that might appear in Evil Robot's knowledge base in order to describe the initial state for the above problem. [3 marks]
3. Give an example of a *unique names axiom* that might appear in the knowledge base. Why might such axioms be required? [2 marks]
4. Give an example of a *unique actions axiom* that might appear in the knowledge base. [1 marks]
5. Give two examples of a *possibility axiom* that might appear in the knowledge base. [4 marks]
6. Give two examples of a *successor-state axiom* that might appear in the knowledge base. One of these should in addition address the *ramification problem*. Explain how it does this. [6 marks]

2006, paper 4, question 4:

1. Give a brief description of the way in which *if-then rules* can be used as a basis for knowledge representation and reasoning. What essential elements would you expect to be included in such a system? [3 marks]
2. In the context of such a system, give detailed descriptions, illustrating your answers with specific examples, of the following concepts:
   - Pattern matching. [2 marks]
   - Reason maintenance. [2 marks]

- Forward chaining. [4 marks]
- Conflict resolution strategies. [4 marks]
- Backward chaining with backtracking. [5 marks]

2003, paper 9, question 8:

1. Explain what the terms *ontological commitment* and *epistemological commitment* mean in the context of a language for knowledge representation and reasoning. What are the ontological and epistemological commitments made by propositional logic and by first order logic? [4 marks]
2. You wish to construct a robotic pet cat for the purposes of entertainment. One purpose of the cat is to scratch valuable objects when the owner is not present. Give a brief general description of *situation calculus* and describe how it might be used for knowledge representation by the robot. Include in your answer one example each of a *frame axiom*, an *effect axiom*, and a *successor-state axiom*, along with example definitions of suitable predicates and functions. [12 marks]
3. Give a brief description of the *representational frame problem*, the *inferential frame problem*, the *qualification problem*, and the *ramification problem*. [4 marks]

7 Planning

2009, paper 4, question 4:

1. Describe the following in the context of planning:
   - STRIPS operators; [1 marks]
   - situation space *versus* plan space (point out the differences); [2 marks]
   - partial order planning (briefly explain key features). [2 marks]
2. Consider the following partial order planning problem for creating a picture of an aquarium. The goal is to have painted_background and also drawn(Fish), drawn(Crab) and drawn(Seahorse). The start state is empty(Picture). You can use the following:
   - paint_background with precondition empty(Picture) and with effects painted_background and ¬empty(Picture)
   - draw(x) with no preconditions and with effects drawn(x) and ¬empty(Picture)

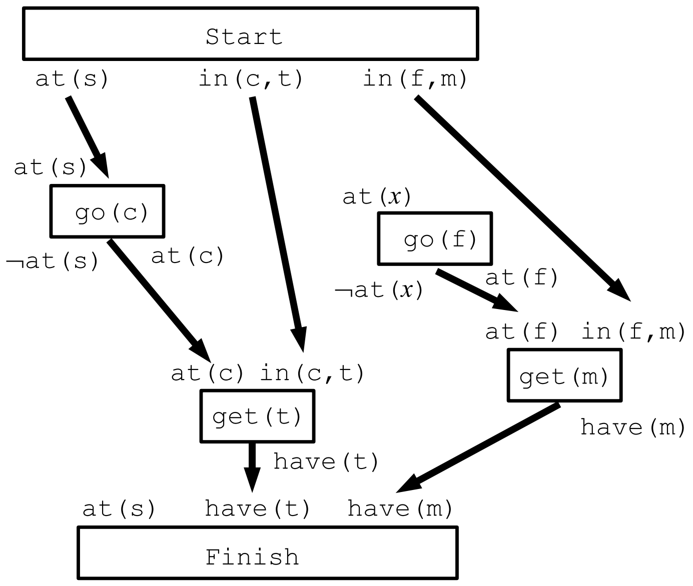
   (a) Write the *start* state and the *finish* state. Draw a partial order plan with preconditions above the operators and effects below the operators. Draw the causal links. [8 marks]

(b) Define what threats are. Comment on whether there are any in your partial order plan above, and how you would solve them. Add temporal links to your partial plan as dotted arcs. [4 marks]

(c) How many solution plans are there in your partial order plan? List all solution plans with total ordering of steps implied by your partial order plan. [3 marks]

2008, paper 4, question 6:

A brilliant student has finished his exams and is making a well-deserved cup of tea. He is confused however, and is trying to use the *partial order planning* algorithm to solve part of the problem. Using the abbreviations f for 'fridge', c for 'cupboard', s for 'sink', m for 'milk' and t for 'tea', his start state is $\{\texttt{at}(\texttt{s}), \texttt{in}(\texttt{c}, \texttt{t}), \texttt{in}(\texttt{f}, \texttt{m})\}$. Using x and y to denote variables, he has two actions. The first action is $\texttt{get}(y)$ having preconditions $\texttt{at}(x)$ and $\texttt{in}(x, y)$, and effect $\texttt{have}(y)$. The second action is $\texttt{go}(y)$ having precondition $\texttt{at}(x)$ and effects $\neg\texttt{at}(x)$ and $\texttt{at}(y)$. His goal is $\{\texttt{at}(\texttt{s}), \texttt{have}(\texttt{t}), \texttt{have}(\texttt{m})\}$. So far he has made the following attempt at finding a plan:



In this diagram, arrows denote causal links.

1. Can the $\texttt{at}(x)$ precondition on $\texttt{go}(\texttt{f})$ be achieved by adding an ordering constraint and causal link from Start to $\texttt{go}(\texttt{f})$, and perhaps one or more further ordering constraints, in such a way that the plan remains valid? Explain your answer. [4 marks]

2. Describe a method, different to any suggested in part *(a)*, by which the $\texttt{at}(x)$ precondition on $\texttt{go}(\texttt{f})$ can be achieved in such a way that the plan remains valid. [8 marks]

3. Describe a way in which the plan can be completed after making the addition you have described in part *(b)*. [8 marks]

2003, paper 8, question 8:

1. Describe the *STRIPS* language for representing states, goals, and operators within a planning system. [5 marks]
2. Give a definition of a *plan*, a *consistent plan*, and a *complete plan*. [5 marks]
3. Describe the *initial plan* used as a starting point by the partial-order planning algorithm. [5 marks]
4. Outline the way in which the partial-order planning algorithm constructs a plan beginning with the initial plan. Include in your answer a description of a *threat* along with an explanation of how the algorithm can attempt to remove threats by promotion or demotion. [5 marks]

1998, paper 7, question 6:

See web site.

8 Learning

2007, paper 4, question 7:

A very simple neural network designed to solve a two-class classification problem where the classes are labelled as 0 and 1 takes input vectors $\mathbf{x}^T = (\begin{array}{cccc} x_1 & x_2 & \cdots & x_n \end{array})$ and has a weight vector $\mathbf{w}^T = (\begin{array}{cccc} w_1 & w_2 & \cdots & w_n \end{array})$, both with real-valued elements. It computes the function

$$f(\mathbf{w};\mathbf{x}) = \text{sgn}\left(\mathbf{w}^T\mathbf{x}\right) = \text{sgn}\left(\sum_{i=1}^{n} w_i x_i\right)$$

where the function sgn is defined as

$$\text{sgn}(z) = \frac{1}{1+e^{-z}}.$$

There exists a training sequence for the network containing m labelled examples

$$((\mathbf{x}_1, o_1), (\mathbf{x}_2, o_2), \ldots, (\mathbf{x}_m, o_m))$$

where the o_i denote desired outputs and take values in $\{0, 1\}$.

1. For the given training sequence, the error of the network when the weights are set to $\mathbf{w}$ is to be defined by the function

$$E(\mathbf{w}) = \lambda||\mathbf{w}|| + \sum_{i=1}^{m} \left(o_i \log \frac{1}{f(\mathbf{w};\mathbf{x}_i)} + (1-o_i)\log\frac{1}{1-f(\mathbf{w};\mathbf{x}_i)}\right)$$

where λ is a fixed, real-valued parameter, we use natural logarithms, and $||\mathbf{w}|| = \sum_{i=1}^{n} w_i^2$. Derive an algorithm that can be used to train this neural network by attempting to find a weight vector minimizing $E(\mathbf{w})$. [17 marks]

2. Describe the way in which your algorithm might be affected by applying it using different values for the parameter λ, in particular very large or very small values. [3 marks]

2005, paper 4, question 4:

This question is *no longer relevant.* It covers material no longer in the syllabus.

2004, paper 6, question 7:

In the following, N is a feedforward neural network architecture taking a vector

$$\mathbf{x}^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$$

of n inputs. The complete collection of weights for the network is denoted $\mathbf{w}$ and the output produced by the network when applied to input $\mathbf{x}$ using weights $\mathbf{w}$ is denoted $N(\mathbf{w}, \mathbf{x})$. The number of outputs is arbitrary. We have a sequence $\mathbf{s}$ of m labelled training examples

$$\mathbf{s} = ((\mathbf{x}_1, \mathbf{l}_1), (\mathbf{x}_2, \mathbf{l}_2), \ldots, (\mathbf{x}_m, \mathbf{l}_m))$$

where the $\mathbf{l}_i$ denote vectors of desired outputs. Let $E(\mathbf{w}; (\mathbf{x}_i, \mathbf{l}_i))$ denote some measure of the error that N makes when applied to the ith labelled training example. Assuming that each node in the network computes a weighted summation of its inputs, followed by an activation function, such that the node j in the network computes a function

$$g\left(w_0^{(j)} + \sum_{i=1}^{k} w_i^{(j)} \text{input}(i)\right)$$

of its k inputs, where g is some activation function, derive in full the backpropagation algorithm for calculating the gradient

$$\frac{\partial E}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial E}{\partial w_1} & \frac{\partial E}{\partial w_2} & \cdots & \frac{\partial E}{\partial w_W} \end{pmatrix}^T$$

for the ith labelled example, where $w_1, \ldots, w_W$ denotes the complete collection of W weights in the network. [20 marks]